EXPLANATION-DRIVEN LEARNING-BASED MODELS FOR VISUAL RECOGNITION TASKS

by

ZACHARY ALAN DANIELS

A dissertation submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Doctor of Philosophy

Graduate Program in Computer Science

Written under the direction of

Dimitris N. Metaxas

and approved by

_____

_____

_____

_____

New Brunswick, New Jersey

October, 2020

ABSTRACT OF THE DISSERTATION

Explanation-Driven Learning-Based Models for Visual Recognition Tasks

By Zachary Alan Daniels

Dissertation Director:

Dimitris N. Metaxas

Safety-critical applications (e.g., autonomous vehicles, human-machine teaming, and automated medical diagnosis) often require the use of computational agents that are capable of understanding and reasoning about the high-level content of real-world scene images in order to make rational and grounded decisions that can be trusted by humans. Many of these agents rely on machine learning-based models which are increasingly being treated as black-boxes. One way to increase model interpretability is to make explainability a core principle of the model, e.g., by forcing deep neural networks to explicitly learn grounded and interpretable features. In this thesis, I provide a high-level overview of the field of explainable/interpretable machine learning and review some existing approaches for interpreting neural networks used for computer vision tasks. I also introduce four novel approaches for making convolutional neural networks (CNNs) more interpretable by utilizing explainability as a guiding principle when designing the model architecture. Finally, I discuss some possible future research directions involving explanation-driven machine learning.

Acknowledgements

I owe the success of my PhD to many people. I would like to express my sincere gratitude to my advisor, Prof. Dimitris N. Metaxas, for his support and guidance throughout the course of my PhD. I would also like to thank the other members of my PhD committee, Profs. Konstantinos Michmizos, George Moustakides, and Fuxin Li, for reviewing my thesis and providing useful feedback that improved my research and presentation capabilities. Additionally, I would like to thank Prof. Apostolos Gerasoulis who served as the fourth member of my qualifying exam committee in addition to Profs. Metaxas, Michmizos, and Moustakides. I'd like to thank the Air Force Research Laboratory (AFRL) and Wright State University's Autonomy Technology Research Center (ATRC) where I interned for two summers. In particular, I'd like to thank my mentors at the AFRL and ATRC, Dr. Donald Venable, Mr. Christopher Menart, Dr. Pascal Hitzler, and Dr. Michael Raymer as well as all of the other people who made the internships possible. I'd especially like to thank the other interns who I worked closely with: Dylan Bowald, Logan Frank, Ashwin Kanhere, and Charlie Veal. I'd also like to thank all of the students at the Center for Computational Biomedicine, Imaging, and Modeling (CBIM), many of whom I've had the pleasure of collaborating with. I'd like to thank all of my other collaborators, teachers, and mentors who I've had the pleasure of learning from throughout my PhD studies. I also want to thank the administrative and support staff at Rutgers and especially, those in the computer science department. Finally, I'd like to thank all of the members of my family who have encouraged and supported me over the course of this PhD.

Funding: Parts of this work were supported by the Dynamic Data-Driven Applications Systems (DDDAS) program of the Air Force Office of Scientific Research (AFOSR) and by the National Science Foundation Graduate Research Fellowship Program (NSF GRFP) under Grant No. DGE-1433187.

Dedication

To my family, especially my wife, parents, and siblings who encouraged and supported me over the duration of my PhD.

Table of Contents

Abstract ii

Acknowledgements iii

Dedication iv

List of Tables xiii

List of Figures xv

1. Introduction 1

1.1. Motivation 1

1.1.1. A Brief Overview of Existing Interpretation Methods for Convolutional Neural Networks 3

1.1.2. Issues with Post-Hoc Explanations 4

1.1.3. Towards Grounded Explanation-Driven Models for Visual Recognition Tasks 5

1.1.4. A Simplified Example Application: Scene Classification Using Object-Based Representations 8

1.2. Contributions of the Dissertation 9

1.2.1. *Augmenting Visual Concepts:* Incorporating Knowledge into Deep Neural Networks Using External Knowledge Graphs 10

1.2.2. *Deriving New Visual Concepts:* Discovering a Novel Representation for Explanation-Driven Visual Recognition 10

1.2.3. *Deriving New Visual Concepts from Auxiliary Data Sources:* Jointly Learning Topic Models and Visual Classifiers 12

1.2.4. *Adapting Visual Concepts:* Utilizing Scenario-Based Models in Dynamic Settings 12
1.3. Outline of the Dissertation 13

2. Background: Interpretable Models in Machine Learning and Computer Vision 15
2.1. Why is Model Important? 16
2.2. Defining Model Interpretability 20
2.2.1. Lipton's Definition: Transparency and Post-Hoc Interpretability 21
2.2.2. Murdoch et al.'s Definition: The Predictive, Descriptive, Relevant Framework for Model Interpretability 23
2.2.3. Doran et al.'s Definition: Opaque, Interpretable, and Comprehensible Systems 25
2.2.4. Doshi-Velez and Kim's Definition: Data-Driven Operational Definitions of Interpretability 26
2.2.5. Ras et al.'s Definition: Users, Laws, Explanations, and Algorithms as Considerations of Interpretable Models 28
2.2.6. Other Discussions on Interpretable Machine Learning and Explainable Artificial Intelligence 30
2.3. Existing Approaches for Interpreting Deep Neural Networks for Computer Vision-Related Tasks 30
2.3.1. Understanding the Internal Components and Mechanisms of the Model 31
Visualizing Learned Filters 31
Understanding the Feature Space via Clustering, Nearest Neighbors, and Dimensionality Reduction 32
Visualizing Activation Maps 33
Discovering Maximally-Activating Image Patches 34
Relating Pixels and Neurons 34

Activation Maximization and Extensions for Visualizing the Preferred Input of a Neuron . . . 36
Perturbation-Based Approaches . . . 37
Inverting Learned Representations . . . 37
Ascribing Human-Understandable Concepts to Specific Neurons 38
Other Approaches for Understanding the Behavior of Neurons in Deep Neural Networks . . . 39
Improving Transparency via Model/Knowledge Distillation . . . 40
2.3.2. Understanding the Decisions/Predictions Output by the Model . 40
Methods Related to Work Discussed in Section 2.3.1 . . . 41
Sliding Window- and Sliding Occlusion-Based Approaches . . . 42
Other Methods Based on Perturbing the Input Image and/or Intermediate Representations . . . 43
Class Activation Mapping and Extensions . . . 43
Other Attention-Based Approaches . . . 45
Other Approaches Capable of Generating Visual Explanations . 45
Combining Deep Learning with Prototype Learning . . . 45
Counterfactual and Contrastive Explanations . . . 46
Grounding Learning-Based Models to Visual Concepts and Visual Attributes . . . 47
Other Models that Learn to Explain Decisions Beyond Pixels . . 50
2.4. Acknowledgment of Additional Resources . . . 50

3. Augmenting Visual Concepts: Incorporating Knowledge into Deep Neural Networks Using External Knowledge Graphs . . . 52
3.1. Introduction . . . 52
3.2. Related Work . . . 56
3.2.1. Combining Knowledge Graphs and Deep Neural Networks for Computer Vision Tasks . . . 56

3.2.2. Exploiting Object Information for Visual Recognition Tasks . . . 57
3.3. Problem . . . 58
3.4. Data . . . 59
3.4.1. ADE20K . . . 59
3.4.2. WordNet . . . 60
3.5. Methodology . . . 60
3.5.1. Summary of our Approach . . . 60
3.5.2. Classifying Scene Images Using an Object-Based Model . . . 61
3.5.3. Aligning ADE20K to WordNet . . . 62
3.5.4. Calibrating Object Recognition Scores . . . 65
3.5.5. Exploiting the Hierarchical Structure of the Knowledge Graph to Refine Object Predictions . . . 66
3.5.6. Training the Scene Classification Model . . . 67
3.6. Experimental Results and Analysis . . . 67
3.6.1. The Importance of Utilizing Grounded, Semantic Information . . 67
3.6.2. Understanding the Limitations and Impact of Noisy Object Recognition . . . 69
3.6.3. Improving Performance by Utilizing Knowledge Graphs . . . 70
3.6.4. Understanding the Effects of Object Prediction Score Calibration 72
3.6.5. Refining Object Predictions by Exploiting the Known Structure of the Knowledge Graph . . . 73
3.6.6. Qualitative Results . . . 74

4. Deriving New Visual Concepts: Discovering a Novel Representation for Explanation-Driven Visual Recognition . . . 79
4.1. Introduction . . . 79
4.2. Related Work . . . 83
4.2.1. Learning Meaningful Groups of Objects . . . 84
4.3. Problem . . . 85

4.4. Data 85
4.5. Methodology 86
4.5.1. Identifying Scenarios from Data: Pseudo-Boolean Matrix Factorization 86
Formulation of Pseudo-Boolean Matrix Factorization 87
Selecting the Number of Scenarios 91
Initializing the Dictionary and Encoding Matrices 93
Solving the Pseudo-Boolean Matrix Factorization Optimization Problem (without Visual Feedback) 94
4.5.2. ScenarioNet: Identifying and Recognizing Scenarios from Visual Data 95
Training ScenarioNet 97
Generating Explanations: Interpreting the Output of ScenarioNet 97
4.6. Experimental Results and Analysis 101
4.6.1. Examples of Learned Scenarios 101
4.6.2. Content-Based Comparison 102
4.6.3. Identifying Some Failure Cases of ScenarioNet 103
4.6.4. Reconstruction Error of Pseudo-Boolean Matrix Factorization . . 106
4.6.5. Comparison to a Model that Bottlenecks Through Object Predictions 108
4.6.6. Comparisons to Other Methods for Scene Classification 109
Comparison to Baseline CNNs 110
Comparison to Other Object-Based Representations 111
Comparison to Visual Attribute-Based Representations 111
4.6.7. Evaluating ScenarioNet's Explanations via Human Subject Experiments 112

5. Deriving New Visual Concepts from Auxiliary Data Sources: Jointly Learning Topic Models and Visual Classifiers 116

5.1. Introduction 116
5.2. Related Work 118
5.3. Problem 119
5.4. Data 120
5.5. Methodology 120
5.5.1. Pre-Training the Feature Extraction Neural Network 121
5.5.2. Extracting Key Terms from Natural Language Text 122
5.5.3. Learning Topic Models using Matrix Factorization 124
5.5.4. Incorporating Topic Modeling into Convolutional Neural Networks 126
5.6. Experimental Results and Analysis 128
5.6.1. Evaluation Metrics 128
5.6.2. Text-Based Experiments 128
5.6.3. Imaging-Based Experiments 129
5.6.4. Analysis of Quantitative Results 130
5.6.5. Qualitative Results 133

6. Adapting Visual Concepts: Utilizing Scenario-Based Models in Dynamic Settings 135
6.1. Introduction 135
6.2. Related Work 138
6.2.1. Active Vision and Dynamic Data-Driven Applications Systems . 138
6.2.2. High-Level Information Fusion 139
6.2.3. Open Set Recognition 139
6.3. Problem 139
6.4. Data 140
6.5. Methodology 141
6.5.1. Neural Network Training Procedure 142
6.5.2. Sensing: Understanding the Input Data 143

6.5.3. Processing: From Pixels to Human-Understandable Representations . . . 143
Scenarios as Grounded and Interpretable Representations 143
Mapping from Scene Views to Scenarios 146
Fusing View-Level Scenarios into Scene-Level Scenarios 147
6.5.4. Decision Making/Predicting: Classifying Scenes 148
The Weibull-Calibrated Support Vector Machine for Open Set Classification 148
6.5.5. Updating: Adapting the Models and Adjusting the Sensors . . . 150
Updating the Scenario Representation 151
Updating ScenarioNet 153
Updating the Scene Classification Model 154
Formulating a Policy for Exploring the Scene 154
6.6. Experimental Results and Analysis 155
6.6.1. Understanding the Importance of Object-Based Representations 155
6.6.2. Understanding the Limitations of Using Object Presence as Features 156
6.6.3. Justifying Scenarios as Discriminative, Human-Understandable Features 157
6.6.4. Evaluating the W-SVM for Identifying Unknown Scene Categories 159
6.6.5. Evaluating the Dynamic-Variant of Pseudo-Boolean Matrix Factorization and the Branching Scenario-Recognition Neural Network 161
6.6.6. Understanding the Necessity of Exploration 162
6.6.7. Evaluating the Exploration Component of our Proposed Framework 163
6.6.8. Qualitative Results 164

7. Conclusion and Future Work 177
7.1. Conclusion 177
7.2. Future Work 182

7.2.1. Expanding Scenarios Beyond Co-Occurrence Relations 182
7.2.2. Outputting Richer, Easier-to-Interpret Explanations 183
7.2.3. Exploring Prototype-Based Approaches 184
7.2.4. Utilizing Interpretable Non-Linear Classifiers 184
References 186

List of Tables

3.1. Accuracy on the scene classification task for several baseline models. We compare visual features to ground truth semantic features. 68

3.2. We evaluate the performance of various approaches for scene classification including an unconstrained ResNet-18 model and two-stage models involving object recognition (OR) (using object labels from the initial object set and expanded object set) followed by a logistic regression (LR) classifier. 71

3.3. We evaluate how object recognition and scene classification are affected by calibrating the object prediction scores. Specifically, we look at the three stage model consisting object recognition (OR), followed by object score calibration, and finally, followed by logistic regression (LR) for scene classification. In all cases, we consider the expanded object set. . . 73

3.4. We evaluate the effect of the object refinement strategy on scene classification accuracy. Specifically, we look at the four stage model consisting object recognition (OR), followed by object score calibration, followed by object refinement using the mined object hierarchy, and finally, followed by logistic regression (LR) for scene classification. In all cases, we consider the expanded object set. 74

4.1. We evaluate how easily objects and scenarios can be recognized from visual data. We also evaluate the predictive power of the object- and scenario-based features are for the downstream scene classification task. 109

4.2. We evaluate scene classification accuracy on a number of different features for the SUN-RGBD and ADE20K datasets. 110

5.1. We evaluate the predictive power of various features extracted from natural language medical reports for diagnosing chest-related illnesses. . . . 128
5.2. We evaluate the performance of learned visual classifiers. The top table looks at the overall predictive performance of the model while the bottom table looks at predictive performance on individual diagnostic labels (bottom). We compare a standard ResNet-22 architecture with our modified architecture that bottlenecks through the topic modeling layer. 130
6.1. Object-based representations for scene classification are very discriminative, especially when compared to purely visual features (using a fine-tuned ResNet-18). It is also apparent that using all views of a scene results in significant performance gains compared to using just a single view. 155
6.2. Classification performance diminishes when noisy predicted object presence features are used for scene classification. 156
6.3. We measure the performance of logistic regression and W-SVM (using 30 predicted scenario probabilities as features) when all classes are known. 160
6.4. We measure the performance of combining scenarios with various popular classifiers for open set recognition task using 30 scenarios and all views with 7 known classes and 7 unknown classes. Results are averaged over 10 random trials. 160
6.5. We compare the quality of a dictionary learned by Dynamic PBMF to one learned with regular PBMF. 161
6.6. Understanding the the performance of the branching convolutional neural network model for scenario recognitionin combination with different classifiers. 161
6.7. We explore how well the learned policy performs on the task of active explanation-driven classification of indoor scenes. We show the trade-offs between maximizing predictive performance and minimizing unnecessary exploration. 163

List of Figures

1.1. An example of an insufficient and incorrect explanation of a decision made by a deep neural network. The network focuses on the meaningful parts of the image when predicting the correct label "dog" (e.g., it focuses on the mouth). This would make one think that the net generates reasonable explanations. However, the network uses the similar parts of the image as evidence when predicting the completely unrelated "snake" class, suggesting that the explanations might not be trustworthy. 4

1.2. Examples of different types of visual attributes (a type of visual concept): 1) group-level, hand-labeled attributes where every instance of the same class shares the same attributes, 2) hand-labeled attributes for individual instances, and 3) instance-level attributes extracted from natural language descriptions. 6

1.3. Comparing the traditional visual recognition pipeline to a pipeline constructed around using visual concepts as interpretable features in combination with interpretable classifiers. 7

2.1. A visualization of some of the learned filters in the first convolutional layer of AlexNet trained on the ImageNet dataset. 31

2.2. Visualizing the activation maps (and positive component of the activation maps) of the learned filters at multiple layers in a simple convolutional neural network. 33

2.3. Left: An input image of a snake; Center: A visualization of the importance of each pixel for predicting the scene category "snake" based on vanilla backpropagation; Right: A visualization of the importance of each pixel for predicting the scene category "snake" based on guided backpropagation. 35

2.4. The reconstructed image (right) of a tree (left) based on inverting the feature representation learned by a deep neural network. 38

2.5. Left: Grad-CAM applied to an image of a cat and dog, conditioned on the "dog" class. Using Grad-CAM, one can see that the net learns to attend to features related to the dog's face and paws. Right: Guided Grad-CAM (which involves combining Grad-CAM with Guided Backpropagation) provides a more fine-grained explanation. 44

3.1. An overview of the scene classification task, which involves using a model to map from an image to the image's scene category. Image from [1]. . . 58

3.2. Left: A histogram relating each object to the number of times it appears in the subset of the ADE20K dataset used in our experiments. Right: The scene class distribution used in our experiments. Image from [1]. . . 59

3.3. Here we show a small subgraph of the aligned knowledge graph between ADE20K and WordNet after extensive pruning. Note how much additional semantic information is captured by even this small portion of the full knowledge graph. Image from [1]. 64

3.4. We evaluate how well a deep neural network can perform on the multi-object recognition task for objects that appear in at least 25 training instances in the subset of the ADE20K dataset used in our experiments. Image from [1]. 70

3.5. We attempt to quantify the effect of sample size on multi-object recognition. Left: We see how many objects are selected as we set different thresholds for the minimum number of times an object must appear in the dataset to not be pruned. Right: We see how object recognition performance is positively impacted as the minimum number of times an object appears in the dataset increases. Image from [1]. 71

3.6. We quantify how scene classification performance is affected by the object set size and quality. We evaluate scene classification performance as we set different thresholds for the minimum number of times an object must appear in the dataset to not be pruned. Left: We train and test models using the *ground truth* object data. Right: We train and test models when using *predicted* object recognition scores. Image from [1]. 72

3.7. We show an example annotated explanation of the output of our knowledge graph-based approach applied to a dining room scene instance. We show the scene class prediction, the model's confidence in its scene class prediction, and the top-10 strongest pieces of evidence in favor of the prediction. Correct concept predictions are highlighted in green, and incorrect concept predictions are highlighted in red. 76

3.8. We show an example annotated explanation of the output of our knowledge graph-based approach applied to a street scene instance. We show the scene class prediction, the model's confidence in its scene class prediction, and the top-10 strongest pieces of evidence in favor of the prediction. Correct concept predictions are highlighted in green, and incorrect concept predictions are highlighted in red. 76

3.9. We show an example annotated explanation of the output of our knowledge graph-based approach applied to a living room scene instance. We show the scene class prediction, the model's confidence in its scene class prediction, and the top-10 strongest pieces of evidence in favor of the prediction. Correct concept predictions are highlighted in green, and incorrect concept predictions are highlighted in red. 77

3.10. We show an example annotated explanation of the output of our knowledge graph-based approach applied to a bathroom scene instance. We show the scene class prediction, the model's confidence in its scene class prediction, and the top-10 strongest pieces of evidence in favor of the prediction. Correct concept predictions are highlighted in green, and incorrect concept predictions are highlighted in red. 77

3.11. We show an example annotated explanation of the output of our knowledge graph-based approach applied to a bedroom scene instance. We show the scene class prediction, the model's confidence in its scene class prediction, and the top-10 strongest pieces of evidence in favor of the prediction. Correct concept predictions are highlighted in green, and incorrect concept predictions are highlighted in red. One interesting thing to note with this instance is that the concept "cushion" is missing whereas the parent class "cushion.n.03" is present. This is because "cushion" specifically refers to the cushion of a sofa whereas "cushion.n.03" is an ancestor of the"pillow" concept. 78

3.12. We show an example annotated explanation of the output of our knowledge graph-based approach applied to a kitchen scene instance. We show the scene class prediction, the model's confidence in its scene class prediction, and the top-10 strongest pieces of evidence in favor of the prediction. Correct concept predictions are highlighted in green, and incorrect concept predictions are highlighted in red. 78

4.1. A visual representation of the pseudo-Boolean matrix factorization, which takes a binary matrix representing which visual concepts appear in each scene instance and decomposes it into a dictionary matrix, which assigns visual concepts to scenarios, and an encoding matrix, which assigns scenarios to each scene instance. 81

4.2. An overview of the ScenarioNet architecture. The key contribution of the architecture is the scenario block, which replaces the final fully connected layers of a standard convolutional neural network. The scenario block consists of three parts: 1) a set of layers that predict the presence of each scenario in a given image and are compatible with the class activation mapping technique [2], enabling the network to identify which parts of an image ScenarioNet attends to when recognizing whether or not a scenario is present in a given image, 2) layers that use a pseudo-Boolean matrix factorization-based loss function to fine-tune the dictionary of scenarios and provide feedback to the scenario recognition layers, and 3) layers equivalent to a multinomial logistic regression classifier that use scenarios as low-dimensional, interpretable features for the downstream classification task. 82

4.3. Above, we show the difference between traditional matrix multiplication and Boolean matrix multiplication. Boolean multiplication replaces the addition operator with a union operator. It is important that the matrix factorization for identifying scenarios uses Boolean matrix multiplication instead of traditional matrix multiplication because when reconstructing the objects-scene instances matrix, we only care about object presence and not object counts. 88

4.4. We show an example annotated explanation of the output of ScenarioNet applied to an outdoor highway scene instance. We show the scene class prediction and top-3 strongest predicted scenarios with the highest influence scores for the downstream classification task along with the corresponding activation maps. 98

4.5. We show example explanations output by ScenarioNet when ScenarioNet is applied to two dining room scenes. We show the scene class prediction and the top-3 strongest predicted scenarios with the highest influence scores for the downstream classification task along with the corresponding activation maps. Top image from [3]. 99

4.6. We show an example explanation output by ScenarioNet when ScenarioNet is applied to an art gallery scene. We show the scene class prediction and the top-3 strongest predicted scenarios with the highest influence scores for the downstream classification task along with the corresponding activation maps. 100
4.7. We show an example explanation output by ScenarioNet when ScenarioNet is applied to a bathroom scene. We show the scene class prediction and the top-2 strongest predicted scenarios with the highest influence scores for the downstream classification task along with the corresponding activation maps. 100
4.8. We show several example scenarios learned by ScenarioNet on the ADE20K dataset. 102
4.9. ScenarioNet can be used to compare two images based on high-level similarities and differences. Image from [3] 103
4.10. We show an example where ScenarioNet doesn't perform as expected. ScenarioNet is highly confident that the {book, bookcase} is present in the above image, but when looking at the parts of the image that the network attends to when making this decision, we see that it exploits contextual clues using objects outside the scenario. It completely ignores focusing on the actual bookcase and instead attends to the desk object. 104
4.11. We evaluate the reconstruction error between a recovered and ground truth matrix as the dimensionality of the reduced representation is varied using a variety of methods on the SUN-RGBD dataset. 105
4.12. We evaluate the reconstruction error between a recovered and ground truth matrix as the dimensionality of the reduced representation is varied using a variety of methods on the ADE20K dataset. 106
4.13. We show an example question for evaluating whether a human participant believes a scenario is meaningful. 112

4.14. We show an example question for evaluating whether a human participant can predict the scene class of a given scene instance based only on detected scenarios that the network deems important for making the true classification decision. 114

5.1. We show an example ground-truth medical report for the chest x-ray analysis problem. Image from [4]. 117

5.2. We show the pipeline for extracting a set of key terms from a natural language medical report. The pipeline consists of four stages: negative scope detection, basic pre-processing, key term extraction using SGRank, and vectorizing the key terms in a bag-of-key terms representation. . . . 123

5.3. We present the structure of the proposed neural network model for combining topic modeling with convolutional networks. Image from [4]. . . . 127

5.4. We show an example of an x-ray and some of its highly-ranked topics. . 132

5.5. We show an example of the net attending to a topic relating to enlarged hearts. 133

5.6. We show several examples of some of the more interesting topics learned by our approach on the OpenI dataset. Many of the learned topics make sense from a visual diagnosis perspective. However, we also see that the model sometimes extracts topics that make sense from a natural language processing perspective but not from a diagnostic or visual perspective. For example, our model learns to group different terms related to the spine, but this group doesn't express any information about the visual appearance of the spine. Similarly, it groups terms related to view of an x-ray, but this topic is visually and diagnostically meaningless. Lastly, we see it group terms related to examinations (in particular "prior examinations"), which once again, doesn't offer any meaningful information for the diagnosis, and it also picks on the keyword "prior" and incorrectly groups the term "prior granulomatous disease" with other terms about "prior examinations". 134

6.1. We show a visual overview of the problem of active explanation-driven classification of indoor scenes. In this problem, an agent is placed in the center of an indoor room and with few sensor adjustments, must assign a label to the scene or identify that the scene is atypical and and update its internal models and knowledge base. 140

6.2. A visual representation of the max-pooling operator of view-level scenario predictions to determine scene-level scenario predictions. 147

6.3. A visual representation of the continuous learning variant of PBMF where when a new class is encountered, the existing scenario dictionary is frozen, and we learn a new dictionary of scenarios specific to the new class that gets appended to the existing scenario dictionary. 151

6.4. As new class-specific scenarios are discovered via the dynamic-variant of pseudo-Boolean matrix factorization, the scenario-predicting neural network is updated by learning new branches consisting of layers designed to recognize the new class-specific scenarios. 153

6.5. Results show that the majority of objects in our extended SUN360 dataset cannot accurately recognized from scene images. 156

6.6. Left: We attempt to measure how scenario recognition performance is affected by the number of scenarios. We also see that refining the dictionary based on visual feedback is useful for improving scenario recognition. Right: We train a model to learn to recognize 30 scenarios. Results show that scenarios can be relatively accurately recognized from scene images, especially when compared to predicting individual object classes (Figure 6.5). 157

6.7. We attempt to understand how well a scenario-based model works for scene classification when using only single-views and when using all available views of a scene. 158

6.8. We show an example of a bathroom scene decomposed into eight views in order to highlight both the explanatory power of our proposed approach and the utility of using information from multiple views for scene understanding and classification. For each view, we show the predicted scenarios. This example presents an interesting case. If an agent were to start with the first view, it wouldn't be able to predict a bathroom scene because it only detects the "closet/door" scenario. However, if the agent looks at views 3, 4, 5, 6, and 8, then there is strong evidence that the scene is a bathroom. Another interesting property of this example is that it includes an uninformative view, specifically view 7 which is essentially a blank wall. In this case, our neural network doesn't detect any useful semantic information and as a result, doesn't output any scenario prediction with high confidence. 168

6.9. We show an example of a church scene decomposed into eight views in order to highlight both the explanatory power of our proposed approach and the utility of using information from multiple views for scene understanding and classification. For each view, we show the predicted scenarios. 172

6.10. We show an example of an office scene decomposed into eight views in order to highlight both the explanatory power of our proposed approach and the utility of using information from multiple views for scene understanding and classification. For each view, we show the predicted scenarios. In this case, we see that our method is good but not perfect. The proposed model detects some scenarios that aren't present in the actual image, e.g., the computer scenario in view 2. Similarly, in views 6 and 7, we see that the model predicts a scenario related to kitchen appliances because it sees a cabinet, but most of the other concepts in the scenario are missing. 176

Chapter 1

Introduction

1.1 Motivation

In recent years, machine learning has played an increasingly important role in modern society, powering many technologies that are used every day by millions of people (e.g., web search, machine translation, image/video editing, and recommender systems for e-commerce and media). Machine learning is used when it is impossible or extremely difficult to code explicit instructions for some given task. Instead, machine learning is concerned with teaching machines (computers) how to perform a given task based on past experiences and observations (often, a large collection of data samples). For example, nobody has figured out how to write an explicit program capable of accurately recognizing hundreds of objects in images because there are too many variations of appearance (e.g., differences in viewpoint, illumination, deformation, occlusion, background clutter, and intraclass variation) to conceptualize and subsequently, hard-code all of the relevant information about a given object class. In contrast, one can create machine learning-based approaches capable of performing object recognition with near human-level accuracy. Unlike traditional software development, machine learning involves specifying the general form of a mathematical model and adjusting the components of the model form (e.g., its parameters) by minimizing some error function on a collection of "training" data. In the object recognition task, this means fitting the parameters of a classification model based on large collections of images where each image contains a single object and a corresponding label for the object. After the model is "trained", during deployment/execution, a new image gets fed as input to the trained model, which then outputs a predicted label.

In particular, the growing importance of machine learning-based technologies in

recent years has been and continues to be driven mainly by the rapidly advancing subfield of machine learning known as deep learning. The deep neural network (DNN), the most common implementation of a deep learning architecture, is currently the gold standard for solving challenging problems involving mapping some input data consisting of complex sensor data (e.g., images and video) to one or more output labels. Deep neural networks perform so well at this task because they learn how to transform the raw input data into a form that can be exploited by the classification/regression model while simultaneously learning the classification/regression model. This is in contrast to most classic machine learning models (e.g., support vector machines, decision trees, linear regression), which typically rely on transforming the raw input data to some useful hand-engineered "features".

Deep neural networks often learn features that are much better suited for a given problem than traditional hand-engineered features, which are frequently suboptimal because they are primarily created via human instinct and (incomplete) domain knowledge. Thus, these deep learned features are often more discriminative than hand-engineered features for a given target task. However, this typically comes at the cost of less interpretability. Hand-engineered features are designed to extract some meaningful, human-understandable properties from raw input signals. In contrast, deep learned features are extracted via a complex series of computations and often take the form of unconstrained numerical vectors where the meaning of these vectors is not immediately apparent. As such, many deep learning models are treated as black-boxes: models that take in data and output decisions without the user understanding the inner mechanisms of how the decisions were reached and without providing evidence to support predictions, i.e., why certain decisions were made.

In this thesis, we consider general problems related to visual recognition, the task of automatically assigning some label to an image or video. Visual recognition problems often appear in safety-critical applications such as autonomous driving, human-machine teaming, and medical image analysis. Such safety-critical applications often require *interpretable* models, and thus, are incompatible with black-box models like deep neural networks. Furthermore, such applications require the use of computational agents that

are capable of understanding and reasoning about the high-level content of real-world scene images in order to make rational and grounded decisions that can be trusted by humans. Thus, for these applications, one would want to modify deep neural networks to increase their interpretability while maintaining their high predictive power. One way to increase model interpretability and encourage the model to make rational, human-like decisions is to make explainability a core principle of the model, e.g., by forcing deep neural networks to explicitly learn grounded and interpretable features. In this thesis, we introduce several novel approaches for making convolutional neural networks (CNNs), a type of deep neural network that is particularly well-suited for analyzing visual data, more interpretable by utilizing explainability as a guiding principle when designing the model architecture.

1.1.1 A Brief Overview of Existing Interpretation Methods for Convolutional Neural Networks

Before we introduce the specific methods proposed in this thesis, we first consider the most popular methods for interpreting the decision-making process of convolutional neural networks. There are many ways to define model interpretability, many of which will be discussed in the next chapter. For now, consider the definition provided by Lipton [5]. Lipton defines model interpretability based on two properties: post-hoc explainability and model transparency. Post-hoc explainability focuses on generating explanations from trained models. Transparency focuses on constructing models where humans can understand each component of the model, including the inputs, parameters, learning process, and shape of the decision boundary.

The most popular methods for understanding the predictions of deep convolutional neural networks are based on post-hoc explainability. For example, recent work has focused on looking at local linear approximations of the model's behavior [6], generating visual explanations of deep neural network features and attention maps (e.g., [2, 7, 8]), and training auxiliary models that generate natural language explanations for the decisions made by another model [9, 10]. Post-hoc explanation is a powerful tool because it can be applied to off-the-shelf models with minimal additional training and little to

no modification of the model structure/architecture. For example, Grad-CAM [8] can be used to generate attention maps for any convolutional neural network (CNN) architecture. This method has an additional benefit: because the structure and parameters of the model remain unchanged, there is no loss in accuracy.

1.1.2 Issues with Post-Hoc Explanations

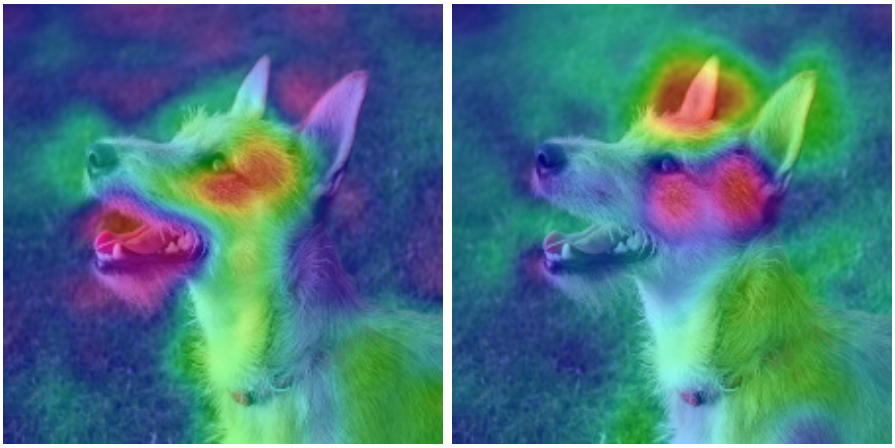


Figure 1.1: An example of an insufficient and incorrect explanation of a decision made by a deep neural network. The network focuses on the meaningful parts of the image when predicting the correct label "dog" (e.g., it focuses on the mouth). This would make one think that the net generates reasonable explanations. However, the network uses the similar parts of the image as evidence when predicting the completely unrelated "snake" class, suggesting that the explanations might not be trustworthy.

However, post-hoc explainability is not without its limitations. Rudin [11] offers a discussion about why post-hoc explainability is often ill-suited for high-stakes applications. Methods for post-hoc explainability sometime provide explanations that are not faithful to what the original model computes. For example, Hendricks et al. [9] try to generate natural language descriptions that explain the output of a trained visual recognition model by training a separate recurrent neural network (RNN) to map the features learned by the recognition model to natural language explanations. However, sometimes Hendricks' method will generate explanations that express false information about the contents of an image (e.g., it might output "the bird is a cardinal because of its red head" when, in reality, the image contains a cardinal with a brown head). Another issue with post-hoc explainability brought up by Rudin is that explanations often

do not make sense or do not provide enough detail to understand what the black box model is doing. For example, when generating attention maps, sometimes the network focuses on parts of an image that seem irrelevant (to a human) when making a specific classification decision. Figure 1.1 which shows an example where a neural network attends to the similar parts of an image when predicting the correct class ("dog") and an incorrect class ("snake"). Likewise, sometimes the visual explanations make sense to a human but require human interpretation. For example, suppose a neural network is trained for scene classification. The neural network predicts that a scene belongs to the category "bathroom" and attends to a "toilet" object when making this decision. This seems like a reasonable explanation; however, a human cannot understand why the net chooses to attend to this region. Is it because the net understands the concept of "toilet"? Is it because the net is focusing on the "porcelain" texture? Is it because the net is focusing on other information such as shape and color? Finally, Rudin argues that post-hoc explainable methods often lead to overly complicated explanations that lead to human error in subsequent decision making; e.g., the biases of the model are often misunderstood (which leads to "unfair" models), and errors in the model can be challenging to debug. Instead, Rudin argues that models must be designed from the ground-up with explainability as a guiding principle (i.e., transparency should be prioritized over post-hoc explainability).

Recent work has shown that others agree with this philosophy, and many works have been proposed that attempt to ground the decisions of complex models to human-understandable concepts (e.g., see [1, 3, 9, 12–16]). Learning grounded explanation-driven models for visual recognition tasks is the primary focus of this dissertation.

1.1.3 Towards Grounded Explanation-Driven Models for Visual Recognition Tasks

One way to make explainability a core principle of the model is by forcing the model to explicitly learn grounded and interpretable features. Visual concepts make up one class of human-understandable features. We define visual concepts as semantic properties that are shared across different categories (e.g., objects, scenes, etc.) and are able to

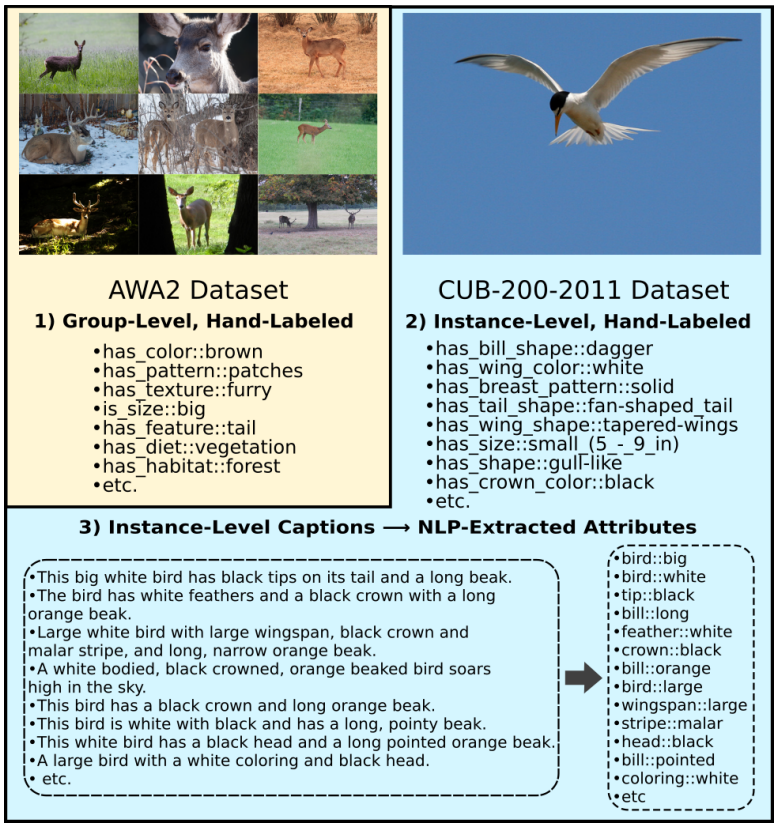


Figure 1.2: Examples of different types of visual attributes (a type of visual concept): 1) group-level, hand-labeled attributes where every instance of the same class shares the same attributes, 2) hand-labeled attributes for individual instances, and 3) instance-level attributes extracted from natural language descriptions.

be recognized from visual data. Visual concepts can include properties such as the presence of specific objects or events in a scene or visual attributes [17–20] such as those appearing in Figure 1.2.

The most basic way to construct transparent models using visual concepts is to first learn a mapping between visual input to a set of visual concepts (e.g., using a DNN) and then train simple, easy-to-interpret classification models (e.g., linear models, decision trees, rule sets) that use these visual concepts as features for some target task. This approach forms the foundation of the methods proposed in this dissertation. A visual representation of this approach appears in Figure 1.3. However, the aforementioned approach is not without its flaws.

Existing visual concept-based models face two significant issues. *First,* errors can be made during visual concept extraction/recognition phase due to 1) limited training

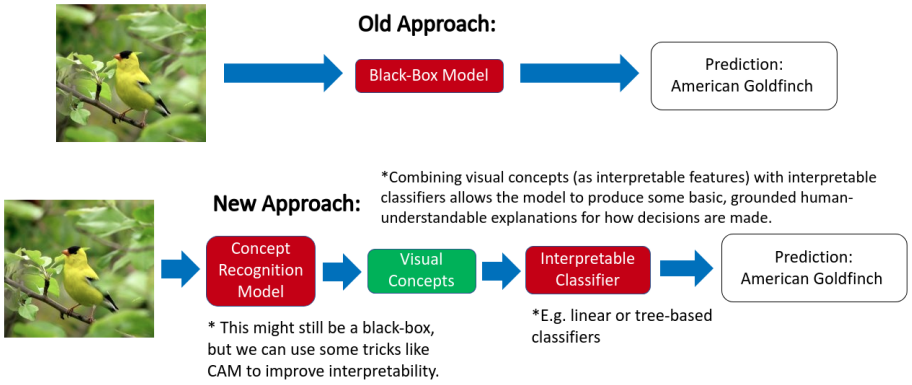


Figure 1.3: Comparing the traditional visual recognition pipeline to a pipeline constructed around using visual concepts as interpretable features in combination with interpretable classifiers.

data that cannot capture all variations of appearance needed to accurately recognize a wide range of classes, 2) incomplete, ambiguous, and imperfect label information that makes the training process noisy and imperfect, and 3) visual subtlety of certain concepts that prevents these concepts from being learned without injecting additional human knowledge into the system. To extract visual concepts from images, one must train a classifier to recognize each visual concept from visual data, and occasionally, the classifier will make mistakes (e.g., recognizing an image of a bird as "furry" or failing to recognize an image of a dog as "furry"). Some visual concepts will have particularly low "ease-of-recognizability" (i.e., are frequently incorrectly predicted), and this information is typically not known before the concept recognition models are trained. A lack of confidence in the predictions for hard-to-recognize visual concepts is a big problem in terms of model transparency because models that use these visual concepts as features can no longer be trusted, which, in turn, limits ease of interpretability. *Second,* many existing visual concept-based models consider large sets of concepts, many of which are redundant or uninformative with respect to some target task, and it is rarely known a priori how discriminative a concept will be with respect to the target task. Uninformative and redundant attributes add unnecessary complexity to the model, making it more difficult to conceptualize the model's decision-making process. In this dissertation, we propose several methods to overcome these limitations.

1.1.4 A Simplified Example Application: Scene Classification Using Object-Based Representations

Before we introduce the contributions of this thesis, we want to briefly discuss the particular application used to evaluate the effectiveness of most (but not all) of the work in this thesis. In the majority of the work introduced in this thesis, we restrict the visual recognition task to scene classification (assigning a label to an image whose contents consist of one or more objects, e.g., classifying the content of images into categories such as "bathroom", "kitchen", "park", and "city street"). Likewise, we generally restrict the set of visual concepts to include only 1) the presence or absence of a known set of objects or 2) some representation derived from the presence or absence of a known set of objects. Restricting the application and visual concepts is done for several reasons. First, humans have an intuitive understanding of how specific objects relate to specific scenes. For example, if a human is told that a scene contains a shower, sink, and bathtub, then the human is likely to hypothesize that the scene is a bathroom. Because humans understand how objects and scenes relate to one another, scene classification is an excellent task for evaluating the explanations output by machine learning models that use object-based representations as the building blocks for the explanations. Second, there are many large-scale, well-curated, and publicly-available datasets for scene classification, and there are several publicly-available ontologies and knowledge graphs that relate objects to one another. Thus, from a logistics perspective, scene classification is a problem that is easy to evaluate and aligns well with our proposed methods. Third, scene classification is more challenging of a problem than object recognition (e.g., due to clutter, wider diversity in appearance between images of the same scene class, etc.) and thus, is useful for better understanding some of the strengths and limitations of the methods introduced in this thesis.

It should be noted that while we focus on scene classification using object-based representations as the motivating example application for most of this thesis, the methods introduced in this thesis are much more general. For example, these same methods can be easily adapted for and applied to different applications and can easily utilize

different visual concepts (e.g., visual attributes instead of object presence information). For example, in Chapter 5, we take the method introduced in Chapter 4 for scene classification using object-based representations and apply it to a completely different domain. Specifically, the method is applied to the classification of biomedical images using visual concepts extracted from natural language medical reports.

1.2 Contributions of the Dissertation

In this dissertation, we introduce several approaches for making convolutional neural networks more interpretable by utilizing explainability as a guiding principle when designing the model architecture. The purpose of these neural networks is to address tasks involving a visual recognition component. Specifically, "explainability" is imposed by explicitly forcing these neural networks to learn features that are grounded to semantic concepts that are well-understood by humans. The methods introduced in this thesis all rely on a pipeline consisting of two parts. The deep neural network learns two components: 1) a mapping from visual input to a set of visual concepts and 2) a classification model mapping the visual concepts to class assignments. However, as was previously mentioned, the visual concepts used in the introduced methods must satisfy two properties to maximize interpretability and trust; the set of visual attributes must be 1) able to be easily recognized from visual data given some dataset, and 2) discriminative for the given target task.

Our research is guided by the following questions:

1. How can we identify or derive a small subset of visual concepts that remain easy-to-recognize from visual data and discriminative for some target task?
2. How can we use existing human knowledge (in the form of knowledge graphs) to augment an initial set of visual concepts and eliminate noise in the visual concepts caused by imperfect, incomplete, and ambiguous labeling?
3. How can we exploit relationships between visual concepts to learn more robust yet still interpretable representations which can be used for higher-level visual recognition tasks?

4. How can we minimize human labeling effort by automatically extracting visual concepts from auxillary data sources?
5. How can we adapt visual concepts for use in dynamic settings?

In the following subsections, we offer brief overviews for each of the proposed methods that address the aforementioned set of question.

1.2.1 *Augmenting Visual Concepts:* Incorporating Knowledge into Deep Neural Networks Using External Knowledge Graphs

Large quantities of meaningful, formalized knowledge are available in the form of public knowledge graphs. At present, the information captured in these knowledge graphs is inaccessible to most deep neural networks, which can only exploit patterns in the signals they are given to classify. In this research direction, we explore how combining deep neural networks with external knowledge graphs can result in more robust and explainable models. Specifically, we align the atomic visual concepts present in the ADE20K dataset [21] (i.e., objects) to WordNet [22], a hierarchically-organized lexical database. Using this knowledge graph, we expand the set of visual concepts which can be identified in ADE20K and subsequently, exploit the hierarchical relationships between these concepts to improve the recognition of the visual concepts. A convolutional neural network is trained to recognize the expanded set of visual concepts, and the hierarchical organization of the concept labels is used to create a graphical model that can identify and correct inconsistencies in the visual concept predictions. The expanded set of predicted visual concepts is then used as features in combination with a linear classifier in order to perform scene classification.

1.2.2 *Deriving New Visual Concepts:* Discovering a Novel Representation for Explanation-Driven Visual Recognition

In the second research direction, we introduce "scenarios" as a new way of representing the semantic content of images. The scenario is an interpretable, low-dimensional, data-driven representation consisting of sets of frequently co-occurring visual concepts.

Consider the example task of scene classification where our visual concepts will consist of the presence of a known set of objects. Scenarios should satisfy a few key properties:

1. Scenarios are composed of one or more visual concepts.
2. The same visual concept can appear in multiple scenarios, and this should reflect the context in which the visual concept appears, e.g., {keyboard, screen, mouse} and {remote control, screen, cable box} both contain the "screen" object, but in the first scenario, the screen is a computer monitor, and in the second scenario, it is a television screen.
3. The semantic content of images can be decomposed as combinations (i.e., the union) of scenarios, e.g., a bathroom scene instance might decompose into: {shower, bathtub, shampoo} $\cup$ {mirror, sink, toothbrush, toothpaste} $\cup$ {toilet, toilet paper}.
4. Scenarios are flexible and robust to missing visual concepts. A scenario can be present in an image without all of its constituent visual concepts being present. For example to determine that a "computer" scenario (e.g., {monitor, keyboard, mouse, cpu tower}) is present in a scene image, it is sufficient if only a few of the objects in this set are present (e.g., maybe the mouse is missing).

Scenarios are learned from data using a novel matrix factorization method which is integrated into a new neural network architecture, the ScenarioNet. Using ScenarioNet, we can recover semantic information about real world images at three levels of granularity: 1) the image class, 2) the scenarios, and 3) the atomic visual concepts. Because ScenarioNet uses scenarios as human-interpretable features in combination with a linear classifier, it can generate explanations for all decisions made, and furthermore, ScenarioNet can be combined with post-hoc interpretation methods such as class activation mapping (CAM) [2] to further improving the interpretability of the model.

1.2.3 *Deriving New Visual Concepts from Auxiliary Data Sources:* Jointly Learning Topic Models and Visual Classifiers

The previous two research directions rely on humans manually labeling the presence or absence of a set of visual concepts in an image for every image in the training dataset. Furthermore, human experts must explicitly specify this set of visual concepts. This labeling process is incredibly time-consuming. Instead, in certain domains, images commonly come paired with natural language descriptions highlighting key characteristics of the image. For example, in medical domains, doctors frequently summarize (as a natural language report) their impressions and findings, which they derive from medical images. In this third research direction, we explore how visual concepts can be automatically extracted from paired images and text with minimal human supervision. Specifically, we extend the ScenarioNet architecture, so it can jointly learn topic models and visual classifiers from data sources consisting of images paired with natural language text.

1.2.4 *Adapting Visual Concepts:* Utilizing Scenario-Based Models in Dynamic Settings

In the previous research directions, we assumed that once a representation is learned or specified, it cannot be updated. This assumption is ill-suited for many real world problems where things change over time, and the model must have the capability to adapt with these changes. The final research direction that we explore in this dissertation focuses on how to learn adaptable representations that remain human-understandable. Specifically, we focus on a particular use case in scene understanding: the active explanation-driven classification of indoor scenes. This task involves placing an agent in some environment, and based on its sensory input, the agent must assign a label to the perceived scene and generate a human-understandable explanation for this decision. The agent can adjust its sensor(s) to capture more details about the scene, but there is a cost associated with manipulating the sensor(s), and when the agent encounters unknown scene categories, it must be capable of refusing to assign a label to

the scene, requesting aid from a human, and updating its underlying knowledge base and machine learning models. We qualitatively and empirically justify why exploration is important for scene classification. We explore the strengths and weaknesses of object-based representations and thoroughly investigate using the "scenario" representation for explanation-driven scene classification. We show how scenarios can be combined with existing methods for open set recognition and propose methods for efficiently updating scenario-based models as new scene categories are encountered. Lastly, we show how to frame active explanation-driven scene classification as a reinforcement learning problem.

1.3 Outline of the Dissertation

The remainder of this thesis is laid out as follows:

- In Chapter 2, we offer a thorough introduction to the field of interpretable machine learning. This chapter focuses on three components: 1) defining the importance of interpretability in machine learning, 2) defining the meaning of interpretabilty with respect to machine learning, and 3) highlighting the existing approaches for interpreting the internal behavior of deep neural networks and explaining the decisions made by deep neural networks, specifically for visual recognition tasks.
- In Chapter 3, we introduce a framework for integrating deep neural networks with external knowledge graphs in order to augment some initial set of visual concepts.
- In Chapter 4, we introduce the "scenario" representation, a data-driven representation for finding meaningful groups of visual concepts, and we introduce the ScenarioNet architecture which embeds scenarios into deep neural networks.
- In Chapter 5, we extend the ScenarioNet architecture to enable it to jointly learn topic models and visual classifiers from data sources consisting of images paired with natural language text.
- In Chapter 6, we extend scenarios and ScenarioNet to dynamic settings involving active exploration and out-of-distribution data. In such settings, the model must

be periodically updated in an efficient manner to account for novel data and shifting distributions.

- In Chapter 7, we offer brief concluding remarks and discuss potential future research directions.

Chapter 2

Background: Interpretable Models in Machine Learning and Computer Vision

In the last decade, machine learning, and in particular, deep learning, has played an increasingly important role in modern society. Machine learning algorithms form the backbone for many technologies that are used every day by millions of people. For example, machine learning plays a vital role in technologies such as web search, recommender systems (e.g., for news, social media, music, and e-commerce applications), machine translation, speech transcription, image and video editing, and medical image analysis. However, many modern learning-based models (especially neural network-based approaches) are treated as **black-box models**: models that take in data and output decisions without the user understanding the inner mechanisms of how the decisions were reached and without providing evidence to support predictions, i.e., why certain decisions are made. In contrast, safety-critical applications (e.g., self-driving cars, human-machine teaming, automated medical diagnosis) often require *interpretable models* for decision-making. Understanding the decision-making process of machine learning models is an incredibly important problem that has seen a lot of interest in the machine learning research community and in related communities that rely heavily on machine learning such as the computer vision, natural language processing, bioinformatics, and biomedical imaging research communities [23–32]. The focus of this thesis is on designing interpretable models for visual recognition tasks. In this chapter, we examine the existing literature related to interpretable machine learning, specifically in the context of complex models sucInterpretabilityh as deep neural networks and applied to visual recognition tasks.

2.1 Why is Model Important?

Before discussing the technical details of model interpretability, we start by highlighting the importance of model interpretability. In "The Mythos of Model Interpretability" [33], Lipton discusses several use cases where model interpretability is especially important. These use cases highlight six important properties of interpretable models:

- **Trust:** Many applications require a human to trust the decisions made by a machine learning model. For example, in the autonomous driving setting, human participants must have faith that the models that control the vehicle will operate in safe and reliable ways. In such applications, a human must have confidence that a model will perform well when it encounters novel real-world scenarios. This means the model must exhibit high accuracy on the given target task both at training and test time, and subsequently, the model must exhibit high generalizability and robustness with respect to unseen data. Another way of thinking about whether or not a model is trustworthy is by determining if a human is confident enough in a machine learning model that he or she feels safe relinquishing control to the model. In this case, it is important to consider whether the model makes the same mistakes or ideally, fewer/less severe mistakes compared to a human, i.e. one must consider whether there is any cost associated with replacing a human with a machine learning model. Typically, trust in the model is strongly correlated with how well humans understand the inner workings of the model.
- **Causality:** Models are typically optimized to make associations, but researchers often use them in hopes of inferring novel properties or generating hypotheses about some given domain or task. Some classes of interpretable models have the potential to illuminate such strong associations and by doing so, could be useful for guiding humans in formulating hypotheses about causal effects between variables present in some dataset. Consider a simple example in the medical domain. Suppose a model is trained to predict the likelihood that a patient will develop lung cancer in the future. If the model has the capability of highlighting which

features are strongly indicative that a person will develop lung cancer, then a human could examine the identified set of highly-relevant features, see that smoking is strongly correlated with developing lung cancer, and finally hypothesize that smoking might cause lung cancer. This hypothesis could then be validated via additional experimentation, data collection, and statistical testing.

- **Transferability:** Humans exhibit a rich capacity for generalizing and transferring learned skills to unfamiliar situations. For example, humans can quickly learn to grasp objects that they've never seen before based on very limited interactions with these objects, and similarly, humans can learn to categorize novel breeds of dogs after seeing just a few examples. Humans are capable of performing these tasks because they have the ability to generalize their previous experiences and knowledge to new situations. In contrast, machine learning models often struggle with efficiently adapting to new domains and tasks. This is because machine learning models are trained on datasets designed for very specific tasks and tested using data sampled from the same distribution. Grounded and interpretable models are useful in understanding how a model will generalize/adapt to new and changing environments/settings. By understanding the computational procedure of a model and how the model represents abstract concepts and features, one can begin to understand how robust a model will be to small (or even large) changes in the underlying distribution of the training data. This property of *transferability* is also necessary for understanding the robustness of specific models to adversarial attacks. It is important to understand how specific inputs affect the decision-making process of a model. Otherwise, one can manipulate the data to force the model to output specific decisions. This is an especially big problem when machine learning models are very complex (i.e., as is the case with deep learning models which have large numbers of parameters and are composed of many small computational units) [34–38]. For example, much research has been done which shows that changing the pixels of an image in imperceptible ways can cause deep neural networks to output completely nonsensical predictions with high confidence. The more interpretable a model is, the easier it is for the

human model architect to know how to modify the model to correct for these vulnerabilities and prevent future vulnerabilities.

- **Informativeness:** In general, a machine learning model's objective is to minimize some sort of error, and this is frequently done at the expense of the model's interpretability. For example, deep neural networks typically achieve very high predictive accuracy but are difficult to interpret, whereas linear models, decision trees, and graphical models are often less powerful in terms of predictive performance, but provide humans with additional insight about the problem being addressed. In many applications, the model interpretability is more important than a few extra percentage points of accuracy. Machine learning models are often used in intelligent decision support systems where machine learning models are designed to augment and aid humans in performing a given task rather than replace them. In these intelligent decision support systems, machine learning models provide additional information as a means of helping humans in the decision-making process. For example, in computer-assisted medical diagnosis, machine learning models must provide specific evidence in favor or against a specific diagnosis, so a human can make the final decision about the diagnosis with as much useful information as possible. Similarly, this information is important not only for making a decision about which label to assign (e.g., a diagnosis) but also for helping the human decide which action should be taken in response to the decision (e.g., which treatment plan should be taken given the diagnosis and context). For example, the same disease can manifest via different symptoms (e.g., the common cold can manifest through sneezing and stuffy noses but also via coughing), and the proper treatment is dependent on both the diagnosis as well as the key symptoms (e.g., using a decongestant if the symptoms are localized in the nose or an expectorant if the symptoms are localized in the lungs). In essence, intelligent decision support systems necessitate the use of machine learning models capable of generating explanations specifying why specific decisions were made.

- **Fair and Ethical Decision-Making:** Machine learning models are often used

for applications which require fair and ethical decision-making, e.g., credit scoring, job application filtering, criminal justice sentencing and bail determination, and news and social media curation. In such applications, it is very important to verify that these machine learning algorithms do not capture and utilize illegal or unethical biases present in the training data. For example, when using machine learning models for deciding who is worthy of receiving a loan and determining the interest rate on the loan, it is illegal to consider (and discriminate based on) a person's gender, race, and disabilities. In other cases, the model might capture biases that are not as obvious to the designer of the machine learning algorithm. For example, researchers have shown that highly accurate face matching algorithms can be biased for different demographic groups [39] because the training data was unintentionally sampled from a very specific population.

- **Ability to Debug and Improve the Model:** Model interpretability is also necessary in order to be able to debug and improve the model. For example, if a self-driving car makes a fatal error, it is important to understand why the error was made, so the model can be corrected. Debuggability via interpretability is also related to the properties of transferability and fair and ethical decision-making because by understanding a model and its flaws (e.g., lack of generalizability and unethical biases), one can more efficiently design new models or update the existing model in order to fix such flaws.

Doshi-Velez and Kim [40] offer another perspective on why interpretability is important in machine learning. They argue that interpretable machine learning models are necessary for domains where **incompleteness** in the problem formulation is a key characteristic. Consider several example domains marked by "incompleteness", and as an aside, note how these domains often require models satisfying the properties outlined by Lipton. In scientific understanding via machine learning models, humans try to gain some sort of knowledge about a domain or phenomena from some set of data by examining the patterns captured by the model. Because it isn't necessarily clear what constitutes "knowledge" and because there is no optimal formulation for how knowledge

should be extracted from data, knowledge is typically derived using the explanations output by the model (e.g., in the same way, interpretable models can be used to hypothesize about causal relations as discussed above). In safety-driven applications, it is generally infeasible or impossible to create a complete list of scenarios in which a machine learning model may fail because one cannot test every fringe case. Instead, one needs to rely on explanations output by the model to verify that the model is trustworthy and performing as expected. When constructing machine learning models that must consider ethical constraints, it is impossible to encode a perfect notion of fairness into a model because "fairness" is too abstract of a concept. Instead, explanations are necessary to identify if the model is capturing unethical biases, so the model can subsequently be updated in order to make it more fair. Interpretable models are also useful for validating that the objective used to train a model is the same as what is necessary to solve a given problem. Often times proxy objective functions do not completely capture the complexity of the desired true problem. Interpretable models enable humans to understand how far the model's objective strays from the desired objective. Finally, machine learning models often must balance multiple objectives, and it isn't always clear how to encode the proper balance. In this case, interpretable models enable humans to evaluate the trade-offs being made by the model.

2.2 Defining Model Interpretability

It is clear that interpretable models are useful in many situations. However, it is not clear what exactly an "interpretable machine learning model" is. In fact, there is no mathematical or universally agreed upon definition of what interpretability means as it relates to machine learning models. Instead, every researcher has a slightly different definition of interpretability in the context of machine learning. In this section, we review several of the more popular definitions.

2.2.1 Lipton's Definition: Transparency and Post-Hoc Interpretability

Lipton [33] defines an interpretable model based on two properties: **transparency** and **post-hoc interpretability**. Transparency involves understanding how a machine learning model works at the level of 1) the entire model, 2) the individual components of the model (e.g., features, parameters, etc.), and 3) the learning algorithm. Transparency can be further defined by three additional properties: **simulatability**, **decomposability**, and **algorithmic transparency**. Simulatability is concerned with answering the question: "Can a person contemplate the entire model at once?" A model exhibiting simulatability means a human can take the input data and the parameters of the model and in a reasonable amount of time, step through every calculation required to produce a prediction. Examples of traditional machine learning models that exhibit simulatability include low-dimensional and sparse linear models, shallow decision trees, and small decision sets/ rule-based classifiers. Simulatable models are the models that are best understood by humans; however, this property comes at a cost: only very simple models are simulatable, which generally limits predictive power, and even for simple models, working through the model computations can still be very time-consuming. Decomposability is less strict of a property. A decomposable model requires that each part of the model (the inputs, parameters, and calculation) admits an intuitive explanation. For example, each node in a decision tree can be easily expressed by a plain-text description, and each weight in a linear model correlates with the importance/influence of the corresponding feature. Linear models, decision trees, rule-based classifiers, and decision sets are all examples of models exhibiting decomposability. However, decomposable models still have their disadvantages. Decomposable models can sometimes be misleading in terms of interpretability. For example, linear models, which are considered to be very decomposable, rely on linear weights that can be very fragile to small perturbations, which can sometimes make them a bit difficult to interpret. As with simulatable models, decomposable models often require sacrificing some predictive power. Finally, algorithmic transparency, as the name suggests, involves understanding the learning algorithm.

This might involve understanding the shape of the decision boundary, understanding whether the model will provably converge to a unique solution or local/global optimum, and understanding theoretical bounds on generalizability. Once again, models that are algorithmically transparent have their disadvantages. For certain applications, it isn't sufficient enough to just know properties of the model; for example, it might be more important to know how each feature effects the overall decision (e.g. credit scoring). Once again, predictive power might be limited with algorithmically transparent models; e.g., deep neural networks are very powerful models, but the theoretic properties of the learning algorithms used to train them (typically backpropagation) are often not well-understood.

Post-hoc explainability is very different than transparency. Post-hoc explainability deals with justifying the decisions of already trained models without truly understanding the inner mechanisms of the machine learning model. Examples of post-hoc explainability includes visualizing feature importance, looking at the local behavior of trained models, generating natural language explanations using pre-learned features, and explaining by example (including counter-factual explanations). Post-hoc interpretation methods have an advantage over transparent interpretation methods: post-hoc explainable methods can be used off-the-shelf with already trained models without sacrificing predictive performance. However, post-hoc explanations are not without their flaws. According to Rudin [11], methods for post-hoc explainability sometime provide explanations that are not faithful to what the original model computes; explanations often do not make sense or do not provide enough detail to understand what the black-box model is doing; and post-hoc explainable methods often lead to overly complicated explanations that lead to human error in subsequent decision-making [11].

Relation to the proposed work: The methods presented in this thesis utilize both transparent methods (specifically, decomposability by grounding features to human-understandable visual concepts and combining these features with human-understandable classifiers like linear models) and post-hoc explainable methods (specifically, by visualizing which parts of an image our models attend to when recognizing visual concepts).

2.2.2 Murdoch et al.'s Definition: The Predictive, Descriptive, Relevant Framework for Model Interpretability

Murdoch et al. [41] provide an alternative definition of interpretability in machine learning based on a desiderata consisting of three properties: **predictive accuracy**, **descriptive accuracy**, and **relevancy**. Predictive accuracy captures how well the model approximates the underlying relationships present in the data and is usually measured by examining how well the model fits the data given some type of supervision. However, unlike in standard supervised learning, it is important that interpretable models have high predictive accuracy for all classes; otherwise, the model can't be trusted. In contrast, descriptive accuracy quantitatively measures how well the *interpretation method* captures the relationships learned by machine learning models; i.e., descriptive accuracy measures how well the interpretation method actually explains the decision-making process of the model. This is often an issue that must be considered by post-hoc methods of interpretability, some of which rely on auxiliary models that try to explain the model of interest. In such cases, one must carefully verify that the auxiliary model is actually capturing the decision-making process of the model of interest and not generating plausible but inaccurate explanations. Accuracy is not the only important characteristic when determining which interpretation method to use for a machine learning model of interest. Relevancy is another important factor. An interpretation method is said to be relevant if the method provides some useful insight for a specific audience for a given problem. For example, patients, doctors, biologists, and statisticians may all be interested in understanding a model that makes predictions about a given genome, but these actor might prioritize different (yet still consistent) interpretations given the same model. It should be noted that there are often trade-offs between predictive accuracy, descriptive accuracy, and relevancy.

Like Lipton, Murdoch et al. further analyze interpretability by examining model-based approaches (transparent methods in Lipton's definition) and post-hoc approaches. In terms of model-based interpretability, Murdoch et al. also stresses the importance of

simulatability and modularity (Lipton's decomposability) but also introduce the concepts of **sparsity**, **domain-based feature engineering**, and **model-based feature engineering**. Sparsity suggests that the explanations produced by machine learning models should be as simple as possible and involve as few components (e.g., features) of the model as is absolutely necessary for a given decision. For example, an explanation that relies on just ten features is typically much easier for humans to conceptualize and understand than an explanation based on millions of features. Domain-based feature engineering is another important concept related to interpretable models. If the features of a model are well-understood, then these features can act as the atomic elements used in the explanations generated by the model. Likewise, using expert-crafted features enables humans to test specific hypotheses when the purpose of the machine learning model is to discover novel knowledge. In contrast, model-based feature engineering is also feasible (e.g., deep neural networks achieve such high predictive performance in part because they automatically perform feature engineering). However, model-based feature engineering needs to be carefully implemented in order to maintain the interpretability of the model.

In terms of post-hoc explainable methods, Murdoch et al. stress the importance of considering both **dataset-level interpretability** and **prediction-level interpretabilityl**. Dataset-level interpretability tries to understand how patterns learned by a machine learning model relate to a particular response or sub-population of the dataset (e.g., understanding which features are correlated with positively or negatively predicting a specific class). Prediction-level interpretability deals with understanding how certain relationships learned by the model affect a specific prediction (e.g., understanding how a prediction would change as some feature is manipulated).

Relation to the proposed work: The work introduced this thesis tries to maintain high predictive accuracy while constructing interpretable models with high descriptive accuracy. The models proposed in this thesis rely on a combination of domain-based feature engineering (by grounding features to human-understandable visual concepts) and model-based feature engineering (by deriving novel data-driven human-understandable representation). Likewise, the explanations produced by the proposed

models tend to incorporate elements of sparsity. Often the proposed models operate over very low-dimensional, sparse, and human-understandable feature vectors. The proposed models focus on prediction-level interpretability but are also compatible with dataset-level interpretability.

2.2.3 Doran et al.'s Definition: Opaque, Interpretable, and Comprehensible Systems

Doran et al. [42] also defines interpretability by categorizing machine learning models. These categories are very similar to Lipton's categorizations, but there are some differences. These categorizations include **opaque systems** where machine learning systems do not provide any insights into their algorithmic mechanisms, **interpretable systems** which enable users to mathematically analyze the algorithmic mechanisms of the system, and **comprehensible systems** which output symbolic explanations of how a decision is made by the model. Doran also stresses that interpretable models should exhibit several important properties similar to those discussed in Section 2.1. Specifically interpretable models should instill "confidence" and "trust" in a user, and interpretable models should embody "safety", "ethicality", and "fairness". Finally, Doran et al. argue that truly explainable artificial intelligence (which supersedes interpretable machine learning) should have a reasoning component so that models truly explain their decisions instead of leaving it up to humans to figure out how to interpret a model's decision.

Relation to the proposed work: By Doran et al.'s definition, most of the proposed models presented in this thesis would fall under the category of "comprehensible systems". Our proposed models rely on recognizing intermediate representations that are human-understandable and symbolic. These symbolic representations are then used as atomic elements for composing explanations which explain how a machine learning model arrives at some decision.

2.2.4 Doshi-Velez and Kim's Definition: Data-Driven Operational Definitions of Interpretability

Doshi-Velez and Kim [40] attempt to provide a more principled way of defining and measuring the interpretability of machine learning models. Their definition of model interpretability relies on the idea that model interpretability is intrinsically tied to the application that the model is being used for and as such, can be measured via evaluation metrics that are tied to that application. These evaluation procedures can be further broken down into a taxonomy consisting of **application-grounded evaluation**, **human-grounded evaluation**, and **functionally-grounded evaluation**.

Application-grounded evaluation is primarily concerned with conducting human experiments embedded within the context of a real application. For example, for medical decision support systems, this involves having a machine learning model make suggestions about diagnoses and generate explanations for why these diagnoses were made. A doctor can then evaluate this information and use it in a downstream task (i.e., making a final decision and creating a treatment plan). In this setting, it is important to compare the quality of the decisions and explanations output by the machine learning model with those created by human experts. In the medical diagnosis example, a doctor can evaluate both the output of a machine learning model as well as medical reports generated by other doctors. A machine learning model would be considered useful and interpretable if the "user" doctor has no preference between the machine learning explanations and the human expert-produced explanations or in the best case, prefers the machine learning-based decisions and explanations.

Human-grounded evaluation involves conducting simpler human-subject experiments which still capture the essence of the target application. Often, these experiments can be completed by lay people, which allows for a larger subject pool and is less expensive than experiments requiring domain experts. These experiments might involve binary forced choice problems (where a human must choose the better of two explanations), forward simulation and prediction problems (where a human must use the explanations

produced by the model to recreate the prediction/output of the model), and counterfactual simulation where a user must take an explanation, the input, and the output and determine what needs to be changed in order to change the model's prediction to some desired output.

Finally, functionally-grounded evaluation minimizes human involvement in the evaluation procedure. In this case, some formal definition of interpretability is used as a proxy for explanation quality. Such evaluation methods are good for evaluating initial prototype systems but should usually be followed up with human-subject experiments.

In addition to defining how to evaluate the interpretability of machine learning models, Doshi-Velez and Kim also discuss some potential task-related and method-related latent dimensions of interpretability which should be considered when evaluating interpretable machine learning models. **Task-related latent dimensions of interpretability** consider the following questions:

- *Does the explanation offer a global or local interpretation of the task?* Global interpretability means the model can identify meaningful patterns that are present in general for a given task; i.e., the model identifies information that is useful for extracting general knowledge about the application domain, which is useful for tasks such as improving the scientific understanding of a given domain. In contrast, local interpretability means models can produce explanations which offer justifications/reasons for why specific decisions were made by the model.
- *In what ways is a problem formulation incomplete? What is the severity of the incompleteness?*
- *How much time can a user devote to understanding the explanation?* Some applications are time-critical and require humans to immediately understand the output of a model whereas others allow humans a plenitude of time to thoroughly study and validate the explanations output by a model. If an explanation takes too long to understand for a given task then the model has limited intepretability and utility.

- *What level of expertise does the user need with respect to a given task?* Interpretability is dependent in part on who is using the system and how familiar the user is with the given domain/task.

Method-related latent dimensions of interpretability deal with understanding what the basic unit of the explanation is (i.e., is it raw features, prototypes, or some derived semantic concepts?), how many of these basic units need to be combined in order to generate an explanation, how these units are combined/structured in relation to one another in order to produce an explanation, and how to express the uncertainty and stochasticity of the explanation.

Relation to the proposed work: Proper evaluation is a core component of the proposed work. The experiments in this thesis utilize both human-grounded evaluation and functionally-grounded evaluation. The proposed models also consider some of the latent dimensions of interpretability described by Doshi-Velez and Kim. The models typically generate local intepretations, are simple and can be understood in a short amount of time, and do not require much domain expertise by the user. In terms of method-related latent dimensions of interpretability, the key component of the proposed research is defining the basic units of the explanation (e.g., we consider visual concepts/attributes, objects, scenarios, and topics) and how to combine these units.

2.2.5 Ras et al.'s Definition: Users, Laws, Explanations, and Algorithms as Considerations of Interpretable Models

Like Doshi-Velez and Kim, Ras et al. [43] argue that the interpretability of a model is largely dependent on who is using the model, what application the model is addressing, and how the model is being used. Specifically, Ras et al. contend that explainable artificial intelligence involves four components: users, laws and regulations, explanations, and algorithms. Each of these components is necessary to provide a context where the adequacy of various explanation methods can be evaluated. We briefly define what each component means:

- **Users:** The first consideration (according to Ras et al.) when determining how

interpretable a model is focuses on how much technical knowledge a user must have in order to understand the model for their use cases. An expert user might include a machine learning engineer who understands the model at a deep level, including how the algorithm works and the intricacies of training the model. On the opposite end of the spectrum is a lay user who does not have any knowledge of how the model was implemented and doesn't care about underlying mathematical principles or how the model integrates with other software components. The lay user only cares about how to use the model for practical applications.

- **Laws and Regulations:** The interpretability of a model is also guided by existing laws and regulations. For example, on one hand, the model cannot reveal too much personal information captured by the model during training. On the other hand, for many applications (e.g., loan approval/rejection), the model legally needs to be able to generate concise and easy-to-understand explanations, and these explanations must be accessible to the lay person.

- **Explanations**: The interpretability of the model is also dependent on the type of explanation that the model produces and the applications in which these explanations will be used. The type of explanation then influences the algorithm used to generate the explanation. For example, when evaluating a convolutional neural network for image classification, if all a user cares about is that the model is training properly, it might be enough for the model to just highlight which parts of the image are important for making a given prediction, but in other applications, the user might require much more fine-grained explanations that explain both *where* the net is looking and *why* it looks at these regions.

- **Algorithms:** In all cases, Ras et al. state that explainable models should have several important properties: **high fidelity** (the faithfulness of the explanation method with respect to the actual decision-making process of the model, i.e., high descriptive accuracy in terms of Murdoch et al.'s definition of interpretability), **high clarity** (the explanation is unambiguous), **high parsimony** (the explanation should be as simple as possible), **high generalizability** (the explanation

method should be useful for a wide range of models/algorithms), and **high explanatory power** (the explanations should be able to capture and express all relevant and necessary information). Furthermore, Ras et al. provide a taxonomy of explanation methods for complex machine learning models. These include rule-based explanations, attribution-based explanations which highlight the influence of each feature, and intrinsic explanations which are specific to a given architecture.

Relation to the proposed work: The proposed methods discussed in this thesis consider many of these properties, e.g., who is using the system and what is the appropriate explanation for some given task. The proposed models are designed keeping in mind trade-offs between high fidelity, clarity, parsimony, generalizability, and explanatory power.

2.2.6 Other Discussions on Interpretable Machine Learning and Explainable Artificial Intelligence

The aforementioned definitions of interpretability are only a sampling of all possible definitions. It should be noted there are many other works that consider issues related to defining the meaning of interpretability in the context of machine learning, understanding the importance and use cases of interpretable machine learning and explainable artificial intelligence, clarifying the limitations of existing approaches for interpretable machine learning, and proposing general guidelines for evaluating interpretability. Other work in this direction includes [11, 29, 44–61].

2.3 Existing Approaches for Interpreting Deep Neural Networks for Computer Vision-Related Tasks

In the previous sections, we discussed the importance of interpretable machine learning models and how to define interpretability based on desirable properties of the machine learning model or interpretation method. In this section, we highlight some existing

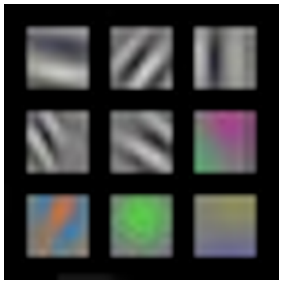


Figure 2.1: A visualization of some of the learned filters in the first convolutional layer of AlexNet trained on the ImageNet dataset.

approaches for interpreting complex machine learning models, specifically in the context of analyzing models designed to perform computer vision-related tasks. Since the majority of modern approaches in computer vision rely on deep neural networks, the approaches discussed in this section are primarily concerned with 1) interpreting the internal behavior of deep neural networks and 2) interpreting the decisions and predictions output by deep neural networks.

2.3.1 Understanding the Internal Components and Mechanisms of the Model

Deep neural networks are incredibly complex models. Many of these models have hundreds of layers with millions of parameters. As such, it is incredibly difficult for humans to understand what patterns and information these parameters are capturing. In this section, we explore some existing methods for probing the network in order to understand the internal components and mechanisms of the model. In general, these discussions will focus on convolutional neural networks [62,63]. It should be noted that the methods proposed later in this thesis do not directly address this type of model interpretability; however, understanding the internal components and mechanisms of the model is an extremely important related topic and thus, is worth briefly discussing.

Visualizing Learned Filters

The simplest method for understanding the learned parameters of a convolutional neural network are to directly visualize the filters (a.k.a. kernels) learned by the model.

In Figure 2.1, some of the learned filters of the first convolutional layer of AlexNet trained on the ImageNet dataset are visualized. In general, the filters learned in the first layers of a convolutional neural network consist of edge and spot detectors, and offer some interpretability about the types of features being learned by the neural network. Likewise, visualizing the early layers of a neural network can provide a good sanity check to verify that the network is training appropriately. However, as one looks at the filters learned by deeper and deeper layers of a neural network, these filters become less understandable by humans and often appear meaningless. As such, directly visualizing the learned features of a neural network often offers limited interpretability, and better methods are needed to understand the information captured by filters (and more generally, other types of "neurons") that appear deeper in the network.

Understanding the Feature Space via Clustering, Nearest Neighbors, and Dimensionality Reduction

Visualizing the learned filters provides a useful tool for understanding the first layer of a convolutional neural network. Simple methods also exist for understanding some of the information captured by the final layer of the feature extraction portion of the neural network that directly precedes layers of the network responsible for classification or regression. The output of this layer is often a high-dimensional (hundreds to thousands of dimensions) feature vector. One can extract these feature vectors from the training set, and then construct a reference database mapping the learned features to individual images. Then, one can apply a clustering algorithm to see if images belonging to the same class also belong to the same cluster. One can also query the nearest neighbors of a specific image to see if similar images are close in the feature space. It should be noted, however, that there is no guarantee that the learned features are well-suited to a particular distance metric (e.g., Euclidean distance), so interpretation via clustering and nearest neighbors might be misleading. A related and common technique in the machine learning community is to apply a dimensionality reduction technique such as T-SNE [64] and see if classes form natural clusters and exhibit some meaningful structure in the low-dimensional space. The aforementioned methods are useful for verifying that

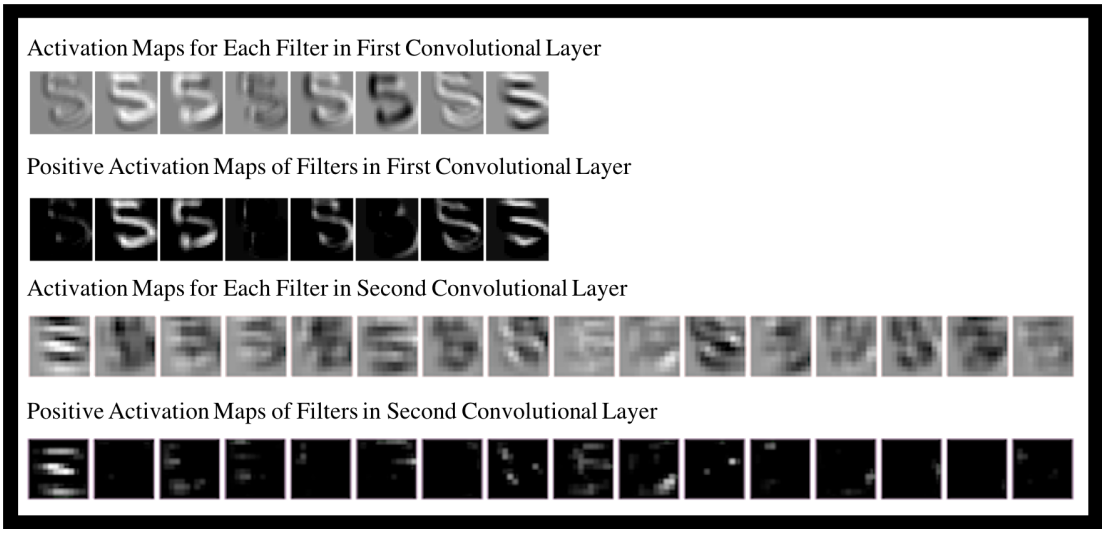


Figure 2.2: Visualizing the activation maps (and positive component of the activation maps) of the learned filters at multiple layers in a simple convolutional neural network.

the neural network learns meaningful feature spaces, but most of the parameters of the model are still not well-understood.

Visualizing Activation Maps

Each filter in a convolution layer outputs an activation map showing how well the filter matches an input image at each location in the image. Figure 2.2 shows an example of the activation maps of a simple convolutional neural network trained to classify handwritten digits. By visualizing the activation maps of a given filter, one can begin to understand what types of features activate a particular filter (e.g., [65] shows how visualizing the activation maps throughout a neural network for a given image can be useful in understanding the information captured by the neural network). Unlike the previously discussed methods, these activation maps can be generated for any filter in the network, and the activation maps remain equally interpretable throughout the network. However, it is still hard to quantify exactly what information is being captured based on filter responses on a single image. Instead, it is more important to understand what types of image features generally activate each the filter/neuron.

Discovering Maximally-Activating Image Patches

One way to understand what types of image features generally activate each neuron is by finding the set of image patches from the training set that maximally activates a specific neuron [66]. To find these patches, one first identifies a neuron of interest, then each image in the training set is run through the network, the maximum response (both value and location) for the neuron is recorded, and the top-N image patches with the highest response for the given neuron are stored and presented to the user. Often times, these neurons will activate for some specific visual concept. For example, a net trained to perform face recognition might have neurons that activate for specific parts of the face (i.e., one neuron might activate for eyes, another for noses, and another for ears). Several problems exist with this method. First, it is very computationally expensive because the neural network must be run over a large subset of the training images. Second, it is not known a priori which neurons will activate for meaningful (i.e., human-understandable) concepts and thus, this procedure is time-consuming because it requires examining many neurons in order to discover useful information about what the network is learning.

Relating Pixels and Neurons

An alternative approach to understanding what types of data activates a given neuron is by studying the sensitivity of a given neuron to changes in specific pixels of an input image. In particular, this requires understanding which pixels in the input image most affect the response of a given neuron. This can be done by looking at the gradient of the neuron with respect to a given input image. This can be computed by performing a forward pass through the network and then computing the gradient of the neuron of interest using backpropagation. However, often the "images" produced by this method look very noisy because the gradient captures information about how pixels affect the neuron in both positive (via activation) and negative (via suppression) ways. In practice, users often care more about understanding which pixels are responsible

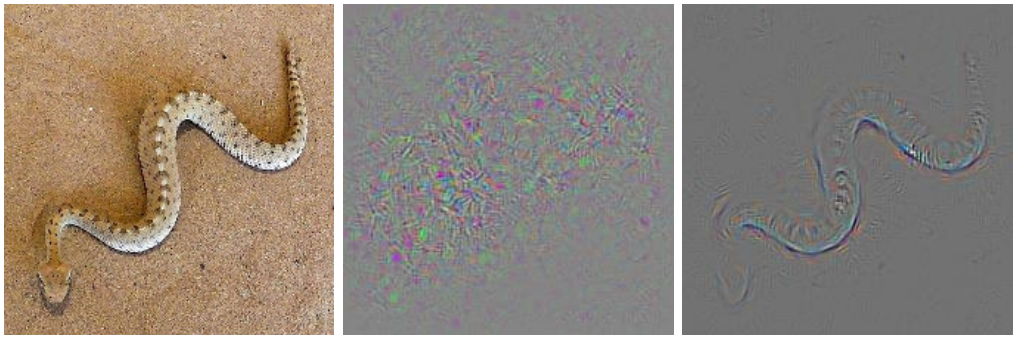


Figure 2.3: Left: An input image of a snake; Center: A visualization of the importance of each pixel for predicting the scene category "snake" based on vanilla backpropagation; Right: A visualization of the importance of each pixel for predicting the scene category "snake" based on guided backpropagation.

for activating a given neuron. This idea led to the creation of **guided backpropagation** [66]. In guided backpropagation, when backpropagating the gradient, all of the negative components of the gradient are set to zero. This has the effect of producing "images" which sharply highlight which parts of an image are responsible for activating a given neuron. Figure 2.3 shows the visualizations using vanilla backpropagation and guided backpropagation when analyzing a neuron that activates when predicting the "snake" category from an image.

An alternative approach is based on the idea of systematically undoing the operations performed by the neural network to reach a specified neuron. In this approach, one desires to map the responses of neurons back to the input image space (as was done with guided backpropagation) in order to visualize the important input patterns that trigger a response by the specified neuron. The **Deconvolutional Network** [67, 68] is a special convolutional neural network architecture that performs the operations (e.g., pooling, convolution, etc.) of the original neural network of interest in reverse (i.e., using a topologically-reversed ordering of the layers and through the use unpooling and transposed convolution), to map activations to pixels instead of mapping pixels to activations.

Activation Maximization and Extensions for Visualizing the Preferred Input of a Neuron

Activation maximization [65,69–72] is another way of evaluating what types of image features are most likely to activate some target neuron. Activation maximization works by finding the synthetic input image I that maximizes the activation of a specific neuron j (subject to some type of regularization on the input image $R(I)$) by utilizing the fact that the activation of neuron j, $A_j(I)$, is a function of the input image I (α denotes the strength of the regularization):

$$\arg\max_I A_j(I) - \alpha R(I)$$

The synthetic image is then found via gradient ascent. One would hope that some of the important semantic features that are being captured by the neuron might become apparent in the generated image, which provides useful information to the human user of the model.

Several extensions exist for activation maximization. Nguyen et al. [73] realized that the same neuron often activates for multiple types of features, but traditional activation maximization methods don't account for the *multifaceted* nature of neurons. In response to this limitation, Nguyen et al. propose an algorithm that generates multiple synthetic images, each of which is designed to explicitly uncover a different type of image that activates the target neuron. Cadena et al. [74] are interested in a similar problem: discovering invariances in the responses of a given neuron. They propose an approach that searches for a batch of images that strongly activate a target neuron where the images in the batch are maximally distinct from each other. By doing so, they hope to find whether the neuron is invariant to some specific transformations in the image space. Another limitation of activation maximization is that it doesn't assume any knowledge of what an input image is supposed to look like, and as such, can often produce very unusual visualizations. This realization led to several works that try to impose priors on the image generating process, typically using Generative Adversarial Neural Networks [75]. Examples of methods that use generator networks to produce the preferred input of a neuron include [76, 77]. Another extension was proposed by Godi

et al. [78] where activation maximization-like methods were used for finding a mask for a given image that highlights the parts of the image which can maximally activate a target neuron. It should be noted that similar approaches for visualizing the preferred input of some neuron exist, e.g., the popular inceptionism/DeepDream [79] method.

Perturbation-Based Approaches

Instead of finding the input that maximizes the activation of a specific neuron, one can make small perturbations in the input space to see how these neurons change or similarly, understand how the performance of the network is affected on the target task (i.e., understanding the robustness of the network). These perturbations can involve making changes to individual pixels, transforming the input image geometrically (e.g., by affine transformations such as rotation, skewing, stretching, shearing, translation, scaling, etc.) or in the color space (e.g., by manipulating contrast, hue, saturation, etc.), and masking different parts of the image. Work in this direction includes [80–83].

Inverting Learned Representations

Another way one can evaluate what type of information is being captured by a deep neural network is to figure out how much information about the original image is preserved by the representations learned at various layers in the model (including the final feature representation preceding the classification layers). **Inversion** [84] is one approach for addressing this problem, and it involves learning how to invert representations by finding a mapping from some representation to image space. One way of understanding how much information is preserved and compressed by the neural network is to take the representation output by every layer in the neural network and see how well the original image can be recovered from each of these representations. Figure 2.4 shows an example of reconstructing an image of a tree based on inverting the feature representation learned by a deep neural network. In general, the representations that appear deeper in the neural network capture more invariant and abstract information. There are still a lot of open questions related to inversion, e.g., determining the best architecture/optimization problem for performing the inversion [85–88], using inversion

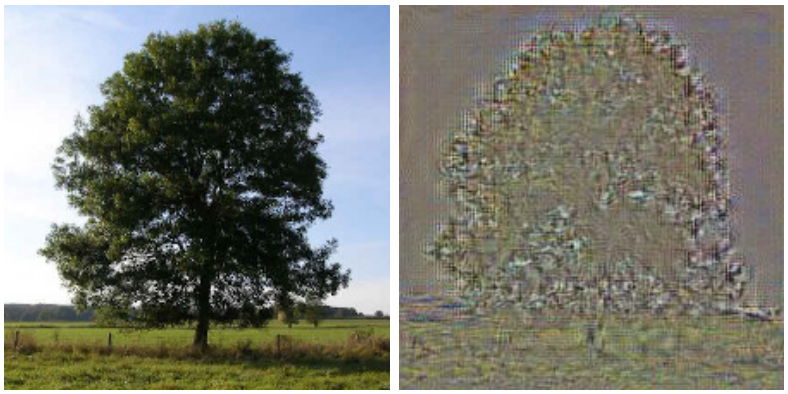


Figure 2.4: The reconstructed image (right) of a tree (left) based on inverting the feature representation learned by a deep neural network.

in clever ways to better understand the behavior of the model (e.g., understanding the intra-class knowledge captured by deep neural networks [89]), and using inversion for specific applications (e.g., face analysis [90]).

Ascribing Human-Understandable Concepts to Specific Neurons

So far, methods have been introduced for evaluating what neurons are learning by primarily looking at the types of visual input that activate specific neurons. In this section, we discuss some methods for assigning meaningful human-understandable concepts to specific neurons. Some basic work has been done studying whether specific neurons encode specific classes of representations for neural networks trained on specific problem domains. For example, some research has shown that deep networks trained for scene classification naturally encode information about objects [91], and deep networks for object recognition naturally capture parts-based detectors [92]. Other works propose more general methods for ascribing human-understandable concepts to specific neurons.

Alain and Bengio [93] offer one framework for understanding the information captured by individual neurons based on **linear probes of the hidden layers**. In this work, a linear classifier is trained after every layer in a pre-trained network for the original problem domain. By examining each linear classifier, one can see which neurons are important for making a decision, and which classes can be accurately predicted as one moves deeper and deeper into the network.

Network Dissection is an alternative framework for evaluating how well individual neurons align with a set of pre-specified semantic concepts [94–97]. Network dissection consists of three parts: 1) identifying a large set of human-understandable visual concepts, 2) recording the neurons' responses to each of these concepts, and 3) quantifying how well each neuron's response aligns with each concept. The goal of network dissection is to see how well deep neural networks disentangle meaningful representations (i.e., can semantic concepts be captured by single neurons or is this information distributed across multiple neurons?). Other work has extended network dissection for interpreting special cases of deep learning architectures such as GANs [98].

Fong and Vedaldi [99] propose yet another approach for understanding how the neurons in a deep neural network align with a set of known semantic concepts. Their method, **Net2Vec** [99], maps semantic concepts to vector embeddings by discovering relationships with filter activations. Unlike network dissection, Net2Vec isn't concerned with finding how individual neurons correlate with semantic concepts, but instead, it focuses on understanding what information is captured by *combinations* of neural network filters.

Several other works exist along the direction of characterizing the neurons of deep neural networks by aligning them to semantic concepts. In particular, [100, 101] will be examined in later sections.

Other Approaches for Understanding the Behavior of Neurons in Deep Neural Networks

Other methods exist for characterizing the meaning and importance of specific neurons/layers in deep learning architectures. In this section, we highlight several of these approaches. One example of such an approach is the **Singular Vector Canonical Correlation Analysis** [102] which can be used to measure the intrinsic dimensionality of layers in a neural network, to probe the learning dynamics of a network as it trains, and for showing where class-specific information forms in the network. Saini and Papalexakis [103] propose a factorization-based approach for understanding how deep neural networks operate. Their approach provides a tool for understanding how well a

network has been trained, and can provide information about how different high-level patterns are transformed as the these patterns progress through the hidden layers of the network. Dhamdhere et al. propose a method for [104] understanding the importance of specific neurons in a deep neural network based on **conductance**, the flow of **Integrated Gradients'** [105] attribution via this hidden unit.

Improving Transparency via Model/Knowledge Distillation

In this section, we discuss a very different approach for improving the model transparency of complex machine learning models. This approach is based on the idea of taking a complex model like a deep neural network and distilling it into a simpler, more interpretable model. **Model distillation** (also known as **knowledge distillation**) is based on the idea that one can take a very complex model (e.g., a very deep neural network) and train a much simpler model to mimic the larger model's behavior/output (e.g., by training the simpler model to output the same value as the larger network) [106–108]. In early work, model distillation was used to compress a large neural network into a smaller neural network for improved computational efficiency with little loss of predictive accuracy [108]. However, recent work has built off the idea of model distillation and applied it in order to improve the interpretability of complex machine learning models instead of (or in addition to) improving computational efficiency. These models take deep neural networks and distill them into simpler, better understood, more interpretale models such as decision trees [109–111] and other classes of simpler models [112, 113].

2.3.2 Understanding the Decisions/Predictions Output by the Model

The second way one can interpret a machine learning model is by trying to figure out how a given model arrives at a specific decision. For example, consider a model trained to predict whether a CT image of the lungs contains cancer or not. If the model is a deep neural network, while understanding which neurons activate for the concept of "cancer" is interesting and useful for verifying the utility and reliability of the model, it doesn't provide the user with too much interesting additional information about the

problem being addressed by the model and only promotes limited trust in the model for nontechnical users who are unfamiliar with how neural networks work. Instead, it is more useful for the model to output a prediction (that the image contains cancer) *and* an explanation for that decision (e.g., by highlighting which parts of an image contain the cancerous mass and what properties of the image make the model think that the highlighted region is tumorous; e.g., is it the color, shape, texture, etc.?). The work proposed in this thesis focuses on this problem of explaining the decisions output by a deep convolutional neural network in ways that are able to be easily understood by humans. In this section, we highlight some of the existing and state-of-the-art methods for explaining specific decisions and predictions output by complex machine learning models, typically convolutional neural networks.

Methods Related to Work Discussed in Section 2.3.1

It should be noted that many of the works discussed in the previous section (Section 2.3.1: Understanding the Internal Components and Mechanisms of the Model) can be easily adapted for explaining specific decisions output by the neural network. In fact, in practice, many of these methods are more frequently used for understanding specific decisions instead of specific internal neurons. Examples of methods that can be used to understand both decisions and hidden neurons include Guided Backpropagation and Deconvolutional Networks. Generally, in these methods, the only technical difference between understanding a specific neuron and a specific decision is conditioning the explanation on a neuron in a hidden layer or an output neuron in the classification layer, respectively. On the opposite end of the spectrum, several techniques introduced in the following sections can be adapted to help understand specific internal neurons, and so it should be noted that while a careful effort was made to thoroughly categorize various methods for interpreting complex machine learning models, this chapter provides just one of many potential taxonomies for categorizing these methods.

Sliding Window- and Sliding Occlusion-Based Approaches

Perhaps the simplest and most intuitive method for understanding the decisions output by deep neural networks is by understanding which individual pixels (or regions of pixels) in the input image are most influential with respect to a given decision [114]. There are two naïve ways of discovering this information. Oquab et al. [115,116] propose an approach based on **sliding windows**. A window is slid across an image. The image is cropped to each window, and these individual patches of images are passed through a pre-trained neural network classifier. Patches with high classification scores for the target class are important for making the target decision, and oppositely, patches with low classification scores for the target class are unimportant for making the target decision. By recording this information for the center pixel as the window slides over the entire image, one can generate a heatmap showing which parts of the image are most informative for the output decision. A similar approach was proposed by Zeiller and Fergus [68]. Instead of using a sliding window, they use a **sliding occlusion**. Their method perturbs the input image by sliding a grey box across the image, and then classifying the entire image. If the classification score for the target class decreases when some region is occluded, the occluded region is important for making the decision. As before, by recording this information for the center pixel as the occlusion slides over the entire image, one can generate a heatmap showing which parts of the image are most informative for the output decision. While these methods typically do a good job of discovering which parts of the image are important for making specific decisions about assigning some target class, they are very computationally expensive because the neural network must perform a forward pass for each perturbed image. **Prediction Difference Analysis (PDA)** [117] is a similar approach to [68] because PDA also involves examining how perturbations of small patches of an image affect a given model's output, but PDA is much more rigorous in how it removes information from the image.

Other Methods Based on Perturbing the Input Image and/or Intermediate Representations

The sliding occlusions approach is one simple way of perturbing the image in order to understand how changes in the input affects the output decision. The **RISE (Randomized Input Sampling for Explanation of Black-box Models)** method [118] sequentially applies many random occlusion masks on the image and then learns how to weigh each of these masks to generate a heatmap of the important parts of the image. [119] expands on RISE by iteratively, adaptively, and intelligently sampling image masks. Several other perturbation approaches exist which involve using more complex perturbations such as blurring, masking, and in-painting the important regions of an image [120–125]. Furthermore, these methods also learn to identify the important regions of an image without relying on sliding windows. Several methods also explore how perturbing intermediate representations in the hidden layers affect the output as well [126, 127].

Class Activation Mapping and Extensions

Oquab et al. [7] and concurrently, Zhou et al. [2] realized that it's possible to identify the parts of an image a neural network attends to when making a decision using just a single forward pass through the network. These methods are capable of providing localized explanations without expensive, post-processing ad hoc steps. **Class Activation Mapping (CAM)** exploits the fact that the convolutional layers in convolutional neural networks preserve (some) spatial information in their activation maps. In particular, the last convolutional layers should have a good compromise between high-level semantics and detailed spatial information. Global pooling compresses the information present in each of the final activation maps and expresses this information as a single value per map. A linear classifier is trained on the compressed feature vector output by the global pooling. The weights of the linear classifier are then used to perform a weighted summation of the original activation maps, and this weighted combination of activation maps (after re-scaling back to the size of the original image) highlights

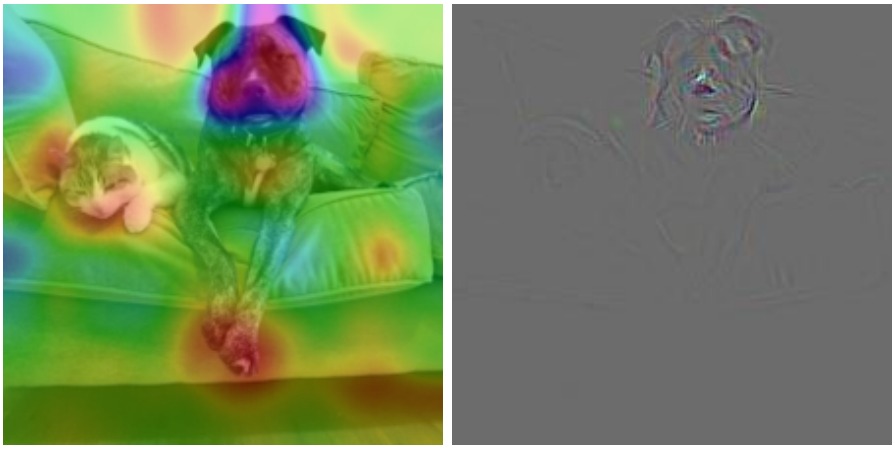


Figure 2.5: Left: Grad-CAM applied to an image of a cat and dog, conditioned on the "dog" class. Using Grad-CAM, one can see that the net learns to attend to features related to the dog's face and paws. Right: Guided Grad-CAM (which involves combining Grad-CAM with Guided Backpropagation) provides a more fine-grained explanation.

which portions of an image the network is attending to when making a specific decision/prediction. If global max pooling [7] is used to compress the activation maps, then CAM highlights the most discriminative areas of an image with respect to some decision. If global average pooling [2] is used, then CAM highlights a larger portion of the relevant areas of a image with respect to some decision. As an example to show the difference between these two pooling operators, global max pooling might focus on the ears and teeth of a dog whereas global average pooling might focus on the entire body of a dog. Figure 2.5 shows an attention-based explanation. CAM is not without its disadvantages; namely, it requires using a specific class of network architectures. Selvaraju et al. [8] proposed an extension to CAM (**Grad-CAM**) which used the gradients of the network during backpropagation to discover how to weigh the activation maps instead of requiring the use of a linear classifier to find the weights. As such, Grad-CAM is more general than CAM (and Grad-CAM has actually explicitly been shown to be a generalization of CAM), and can be applied to general convolutional neural network architectures. Subsequently, other extensions have been proposed to further improve upon CAM and Grad-CAM, e.g., Grad-CAM++ [128], Smooth Grad-CAM++ [129], and others [130–134].

Other Attention-Based Approaches

Perturbation-based approaches and CAM-related approaches are not the only methods for generating heatmap-like visualizations of which parts of an image is important for making a specific decision. Alternative methods exist that incorporate attention into the network training/decision process in order to improve performance on the target task, and as a result, the learned attention masks capture different types of information than those generated using post-hoc methods like CAM. Examples of such approaches include [135–137].

Other Approaches Capable of Generating Visual Explanations

It should be noted that there are many other techniques that attempt to produce explanations in the input space (e.g., by quantifying the importance of specific pixels or regions of pixels) beyond those that have already been discussed. Examples of such techniques include **Layerwise-Relevance Propagation (LRP)** [138, 139] and extensions [140, 141], **Local Interpretable Model-Agnostic Explanations (LIME)** [6], **Integrated Gradients** [105] (which are utilized and extended by several methods previously discussed), **DeepLIFT** [142], **Excitation Backpropagation** [143], **Anchors** [144], **Shapley Additive Explanations (SHAP)** [145], and others [146–148].

Combining Deep Learning with Prototype Learning

Up to this point, discussions have almost exclusively focused on visual explanations for understanding the decisions and predictions output by the deep neural network; i.e., the explanations highlight which pixels or regions of pixels are most important for making specific decisions. While these methods are useful and as it currently stands, are the most popular methods used in practiced, they are not without their flaws [11, 149–151]. One particular flaw with most of these visual explanations is that they highlight *where* a network is attending without explaining *why* the network cares about these regions. One way to overcome this flaw is by tightly integrating the explanation process with the model training process and making explainability and human-understandability of

the explanation a priority of the model. Several methods have been proposed along this direction. In this section, we introduce one such method: combining deep learning with **prototype learning**.

One way to integrate explanations into the learning process is by integrating deep neural networks with prototype learning [14,15]. These methods involve training a neural network that learns to either *identify* "prototypical" images/image patches from the training set or to *generate* prototypical images/image patches via some encoder-decoder mechanism. Prototypical images are defined as the images that are most representative of a given class, and in practice, one learns to identify or generate such images by learning how to cluster the training images in some feature space satisfying some distance metric. Once these prototypical images/image patches are learned, an input image can be encoded as a feature vector by computing similarities between the input image and all other prototypes, and then classification layers can be used to map from the similarity-based feature vectors to the decision/prediction. [14] dealt with learning prototypical images whereas [15] proposed a similar framework capable of learning prototypical image patches. By considering prototypical patches, Chen et al. created a neural network architecture capable of justifying its decisions by identifying the parts of the input image that were visually similar to prototypical parts of other images in the training set (e.g., the wing of the bird in the input image is similar to the prototypical wing of a cardinal in the training set, so the input image likely contains a cardinal).

Counterfactual and Contrastive Explanations

Other approaches rely on explaining machine learning models using **counterfactual** [152–156] and **contrastive** [157–161] **explanations**. Counterfactual explanations aim to identify how a given input image can be changed in order to change the class prediction to some other desired class prediction. Contrastive explanations are very closely related to counterfactual explanations and attempt to answer the question: "which properties of the input caused the model to classify the input as X and not Y?" Counterfactual and contrastive explanations are useful because they can take the form: "This is an instance of X because it has property A *and doesn't have property B*." Unlike most

of the other explanation methods that have been discussed so far, counterfactual and contrastive explanations rely on providing evidence based on both the presence *and absence* of certain properties when generating explanations. Similarly, such types of explanations are more useful for verifying whether a model is working properly because they provide more fine-grained explanations and enable humans to ground the explanations to reasoning mechanisms that humans are already comfortable with (i.e., comparisons are easier to understand and validate than purely "additive" explanations which only provide evidence in favor of a given class based on the presence of certain properties).

Grounding Learning-Based Models to Visual Concepts and Visual Attributes

The previous two classes of approach improved the interpretability of the learning-based models by grounding the generated explanations to examples in the dataset (or derived from the training set). In particular, prototype-based models ground explanations to other images (or patches of images) that are representative of a given class, and methods based on counterfactual/contrastive explanations ground the explanations to comparisons with other classes. In this section, we discuss some methods that ground explanations explicitly to human-understandable concepts (much like what was done in Section 2.3.1). It should be noted that the methods discussed in this section are the most similar to the methods that are the focus of this dissertation.

Recently, a number of approaches have been introduced that are designed to make neural networks more transparent by grounding the decisions of the networks to some source of semantic knowledge. **Interpretable Basis Decomposition (IBD)** [162] is a technique that expands upon class activation mapping by decomposing the attention maps of a neural network for some target task as a weighted combination of attention maps focusing on individual semantically-meaningful visual concepts (e.g., for scene classification, this involves decomposing the scene-level attention map into a set of attention maps, each of which focuses on a different object). In order to discover this decomposition, first, separate classifiers are trained for each individual concept in a set of known concepts. Then, a completely separate classifier is trained using the

same features for the target task. In an iterative process, the concept classifier that explains most of the direction of the target classifier is greedily selected and a residual is computed. Next, the concept classifier that explains most of the direction of the residual is selected, once again in a greedy manner, and a new residual is computed. This process repeats until the residual cannot be decreased anymore by any of the remaining concept classifiers.

Kim et al.'s approach, **Quantitative Testing with Concept Activation Vector (TCAV)** [100] is a similar approach capable of providing an interpretation of a neural net's internal state in terms of human-friendly concepts. TCAV involves using directional derivatives to quantify the degree to which a user-defined concept is important to a desired classification result. For example, TCAV might highlight the fact that the prediction of the "zebra" class is sensitive to the presence of stripes.

Ghorbani et al. propose the **Automated Concept-based Explanation (ACE)** [101] method to to identify and extract higher-level human-understandable concepts by aggregating related local image segments across a diverse dataset. ACE learns concepts that satisfy three properties important for interpretability: meaningfulness (humans can assign some meaning to the concepts learned by the model), coherency (instances of the same concept should appear perceptually similar while instances of different concepts should appear perceptually differenct), and importance (the object should play some role in the decisions output by the network; i.e., ignoring the concepts would meaningfully impact model performance).

Hendricks et al. [9] proposes an approach for improving model transparency by learning an auxiliary model that uses the features of a trained convolutional neural network as input to a language-based model (e.g., a recurrent neural network) that **generates natural language explanations** for the decisions made by the visual recognition model. This approach suffers a significant flaw: it is not known how well the explanations output by the model match the features that are truly being used by the visual classification model. It could be the case that the explanation model finds explanations that seem plausible, but are not truly explaining what the base network is doing; e.g., the generated explanations might point to evidence that isn't even present

in the input image. In follow up work [13], Hendricks et al. corrected for this flaw by grounding the natural language explanations to visual cues. This is done by selecting explanations that are both image- and class-relevant.

A lot of other work has been done in the direction of grounding explanations to human-understandable concepts. For example, Sarker et al. [12] propose grounding the decisions of neural networks to knowledge graphs, and Xie et al. [163] discuss a method for grounding neural networks to pre-specified, human-understandable input concepts. Qi et al. [164] propose an explanation module which embeds high-dimensional information captured by a deep neural network layer into a low-dimensional explanation space. The predictions output by the network can then be constructed from just the few concepts extracted by the explanation module, and these concepts can be visualized, showing which high-level concepts the neural network model uses when making decisions.

in general, many methods in the computer vision literature (including many of the methods discussed in this section) ground models to some sort of **visual concept** or **visual attributes** [17]. Visual attributes consist of semantic properties that are shared across various categories (e.g., objects, scenes, etc.) that should be easily recognized from visual data [18–20, 165]. Visual attributes can include visual characteristics, affordances/functionality, sentimental or emotional state, and the presence of objects and parts-of-objects. Attributes can be binary (is_wooden:true), discrete (wing_color:red), continuous (number_of_legs:2), or even relative (dogs:fuzzier_than:turtles) [166]. Visual attributes have been applied to many domains including object recognition [167], scene classification [168, 169], animal identification [20, 170], fashion analysis [171], and face recognition [172]. They have also been applied to a wide range of tasks including visual recognition [173], zero-shot learning [165, 174], image retrieval [175], active annotation [176], image editing [177], and conditional image generation [178]. Attributes can be identified by explicit human input and crowd-sourcing [167], by mining web data [179, 180], and by unsupervised vision methods [181]. One easy way to improve model interpretability is by predicting which visual attributes are present in an image and then making decisions based on these human-understandable visual attributes.

The work presented in this thesis is very closely related to a number of the aforementioned methods [1,3,4]. For example, one of our proposed approaches [1] focuses on using knowledge graphs to improve deep learning explainability like Sarker et al. [12]. Another of our approaches is similar to interpretable basis decomposition, but instead of learning the concept classifiers disjointly from the target classifiers, we train a single model that first identifies and predicts meaningful sets of objects (termed "scenarios") and then use these scenarios as interpretable features that are fed into a linear classification model for the target task. In general, all of our proposed methods rely on the idea that a machine learning model must make decisions grounded to some human-understandable intermediate representation (e.g., visual attributes). The methods introduced in this thesis focus on some of the issues caused by this approach, and how these issues can be overcome in order to make learning-based models for visual recognition tasks easier-to-understand and more trustworthy.

Other Models that Learn to Explain Decisions Beyond Pixels

Other models exist that attempt to learn how to explain the decisions of complex machine learning models for visual recognition tasks beyond just pixel-based explanations and using more complex reasoning. For example, [182] presents an unsupervised method to learn an **explainer neural network** which tries to explain the decisions/predictions of some other pre-trained convolutional neural network by disentangling the target network's feature maps into object-part features and then uses these object-part features to reconstruct features in the higher layers of the network. Another example includes the **Self-Explaining Neural Network** [183] where explanations are intrinsic to the model and satisfy three desiderata for explanations: explicitness, faithfulness, and stability.

2.4 Acknowledgment of Additional Resources

The structure and content of this chapter was informed in part by a number of valuable online resources that highlight the state-of-the-art with regards to interpretability and

interpretation methods in the context of machine learning, deep learning, and computer vision. These resources include [184–189]. Several images were generated with code/software provided by [190, 191].

Chapter 3

Augmenting Visual Concepts: Incorporating Knowledge into Deep Neural Networks Using External Knowledge Graphs

Note: This chapter consists of joint work with Logan Frank (Ohio State University), Christopher Menart (Air Force Research Laboratory, Sensors Directorate), Dr. Michael Raymer (Wright State University), and Dr. Pascal Hitzler (Kansas State University). This chapter is based on work originally published as "A Framework for Explainable Deep Neural Models Using External Knowledge Graphs" [1].

3.1 Introduction

The focus of this thesis is developing several approaches for grounding deep neural networks to human-understandable visual concepts. However, several issues exist when utilizing a neural network that bottlenecks through a visual concept recognition layer. Namely, some of these visual concepts cannot be recognized with high accuracy from images, and even those concepts that can be recognized with high accuracy might not be discriminative for the target classification task. In order to overcome the two issues mentioned above, this thesis presents several major methodological topics for identifying improved representations based on a given initial set of visual concepts. Specifically, the thesis consists of four parts: augmenting an existing set of visual concepts using knowledge graphs, deriving new representations from an existing set of visual concepts, deriving new representations from auxiliary data sources consisting of images paired with textual descriptions of the images, and adapting the learned representations as new data is encountered. In this chapter, we focus on the first topic: augmenting an existing set of visual concepts using knowledge graphs.

Over the last decade, deep neural networks (DNNs) have become the standard class of algorithms for solving challenging supervised machine learning problems, especially when the input consists of complex sensor data (e.g., images and video). As has been discussed in previous chapters, one of the prominent differences between DNNs and older machine learning algorithms is that DNNs are significantly more opaque in their decision-making. Prior work [192] has shown that deep convolutional neural networks used for visual recognition tasks frequently capture and rely on patterns in the data that are very different from those utilized by the human visual system. Subsequently, DNNs are often (incorrectly) overly confident when they classify images. In this chapter, we propose integrating deep neural networks with external sources of human-generated semantic knowledge via external knowledge graphs. We hypothesize and experimentally validate that incorporating background knowledge from an external knowledge graph can improve a neural model's explainability and robustness.

At present, DNNs are only capable of capturing semantic information by identifying statistical patterns from large sets of observational data. While many DNNs demonstrate impressive capabilities when it comes to learning complex tasks, such as image recognition, they rarely learn to perform these tasks in ways that are very different from human reasoning and decision-making. Thus, DNNs often learn obfusicated forms of basic knowledge of the world rather than directly learning semantic concepts. Even state-of-the-art neural network-based models for visual recognition tasks rely on discovering features capturing lower-level patterns (e.g., texture and color information) rather than the high-level knowledge (e.g., information about shape, structure, semantics, and relationships) which humans use to make similar decisions [193].

By aligning neural networks with external semantic knowledge, we hope to help constrain the DNNs to behave in a human-like manner when making decisions and predictions. Enforcing this human-like decision-making process, we hope to alleviate many of the notable flaws previously discussed. Specifically, by grounding neural networks to human knowledge, we enable them to generate high-level explanations that are comprehensible to human beings.

In this chapter, we explore the effectiveness of aligning and integrating neural networks with external sources of knowledge (in the form of knowledge graphs). Experiments are conducted on the ADE20K dataset [21], focusing on the task of scene classification. The ADE20K dataset consists of images of general indoor and outdoor scenes captured at ground-level using standard RGB cameras. We choose this dataset because it provides both 1) scene category labels for each image and 2) information about which objects (and their parts) are present in each image. This object information can be used as an additional form of supervision which ties the sensor data (images) to external semantic symbols.

We propose a framework that integrates external knowledge (from knowledge graphs) with a deep convolutional neural network. Furthermore, we investigate how our proposed approach affects explainability and robustness compared to traditional DNN architectures. We use WordNet, a hierarchically-organized lexical database, as our source of external knowledge [22]. We align the 1,268 object types (not including object parts) labeled in the ADE20K [21] dataset with their corresponding terms (synsets) in the WordNet ontology. Once this alignment is complete, we construct a hierarchy of objects and their ancestor categories based on subclass-superclass relationships. This hierarchy enables us to model information about the objects in ADE20K at different levels of granularity. By modeling hierarchical relations between object labels, we can derive an expanded label set for each image, which provides much more information than the flat set of object labels that come with ADE20K. For example, if we know that a "silver bird statue" is present in an image, then our hierarchy would also tell us that there are also instances of "statue", "decoration", and "artwork" present in the image because each of these new labels is an ancestor class of "silver bird statue". We can then train an object recognizer on this expanded object set, and use the predicted object probabilities as features that subsequently feed into a logistic regression model that performs scene classification. Because the features (object probabilities) are interpretable and the logistic regression model is a simple linear classifier, the model can generate easy-to-understandable explanations. We also explore how to exploit the structure of the hierarchy to improve object recognition accuracy. To do so, we propose

a structured prediction model designed to correct mistakes between object predictions involving parent-child relationships.

One might question why it is necessary to use WordNet to expand the set of objects beyond just those provided with the dataset. The rationale for this expansion is that object labels included in many datasets are often noisy and incomplete. Because these datasets are often created via crowdsourcing, labels can sometimes be redundant and ambiguous, e.g., objects that are semantically equivalent might be labeled differently because humans have different names for the same objects. Sometimes labels are overly specific, e.g., an object might be labeled as a "wooden statue" while not being labeled using higher-level categories such as "statue" and "artwork". Humans also occasionally make mistakes, so they occasionally incorrectly label objects that do not exist in a scene and vice versa. Some mistakes are not due to human error but are instead due to limited training data. For example, the majority of objects in the ADE20K dataset appear fewer than ten times, and thus, there is often not enough training data for the model to learn how to recognize these sparsely-appearing objects because deep neural networks are incredibly data-hungry. Similarly, some objects appear so small in the images (i.e., only hundreds of pixels in area), that DNNs cannot learn the fine-grained visual patterns necessary to accurately them.

By using a knowledge graph to expand the object set, some of the aforementioned issues can be addressed:

- Ambiguous labels are merged.
- Object information is captured at multiple levels of granularity.
- Ancestral categories appear more frequently than some of their children, so while there may not be enough training data to learn to recognize some object, there might be enough training data to recognize one of its ancestors.

Thus, the proposed method for augmenting visual concepts is useful in many ways for reducing noise in the visual concept recognition stage of a semantically-grounded model.

3.2 Related Work

We present our work in the context of existing efforts to combine human knowledge with neural networks. Some of these efforts exploit knowledge graphs directly; other work utilizes prior knowledge imposed directly by the creator of the model; and other work learns how to automatically discover knowledge graphs directly from the data. We do not discuss existing methods for making deep learning models more interpretable as such methods are already thoroughly discussed in Chapter 2.

3.2.1 Combining Knowledge Graphs and Deep Neural Networks for Computer Vision Tasks

Some recent efforts have attempted to combine knowledge graphs with neural networks for visual recognition tasks. Marino et al. [194] train a neural network to predict every node in a given knowledge graph and then propagate information between nodes to refine predictions. Goo et al. [195] relate objects to one another based on taxonomies consisting of subclass-superclass relations in order to learn better features that are useful for distinguishing between classes that share the same superclass but are often confused with one another. Guo et al. [196] learn a hierarchical classifier consisting of a convolutional neural network for feature extraction and a recurrent neural network that improves predictions by exploiting relationships between the predicted classes. Yan et al. [197] is another approach that utilizes DNNs for hierarchical classification. Srivastava et al. [198], Fan et al. [199], Kuang et al. [200], and Zhang et al. [201] propose different tree-structured concept ontologies that organize large numbers of concept classes (often objects) using coarse-to-fine labels, and some of these approaches automatically discover inter-related learning tasks. Roy et al. [202] is another approach that learns fine-to-coarse tree-structured DNN classifiers but learns these structures incrementally, enabling the addition of new classes without retraining the entire network. Deng et al. [203] introduce Hierarchy and Exclusion (HEX) graphs. HEX graphs are capable of capturing semantic relations by examining mutual exclusion, overlap, and subsumption between two labels applied to the same object. Other approaches [204, 205] attempt to

learn semantic embeddings driven by the existing information present in hierarchies and knowledge graphs. Finally, some approaches attempt to exploit ontologies to improve the explainability and interpretability of deep neural networks [12], and improve deep learning-based image interpretation (extracting structured semantic descriptions from images) [206].

While not as related to our approach as the previously mentioned methods, it is worth mentioning that there are a number of approaches that try to learn to predict or exploit instance-level "scene graphs", e.g., see references 207–213. Likewise, another interesting direction that involves fusing neural networks with knowledge is designing deep networks capable of performing neural-symbolic reasoning over very simple scenes, e.g., see references 214–217. Finally, it is also worth noting that there has been some interest in constructing ontologies that are useful for visual recognition tasks. ImageNet [218] is a very popular ontology that is based on WordNet, and it is very useful for training hierarchical classification models for general object recognition. SceneNet [219] is an ontology for relating different scene categories based on perceptual similarity.

3.2.2 Exploiting Object Information for Visual Recognition Tasks

As has been mentioned previously in Section 1.1.4, we use scene classification using object-based features as our motivating application. Incorporating object-based information into visual recognition pipelines is a well-studied problem in the computer vision community. One common approach for exploiting object information for scene understanding-related tasks is to model contextual information about scenes based on object relations using a probabilistic graphical model [220–226]. These graphical model-based approaches generally follow one of two paradigms. The first paradigm involves exploiting information about multiple, related tasks (e.g., scene classification, object recognition, semantic segmentation) as a means for improving performance on each of the individual tasks. The second paradigm utilizes information about relationships between objects and the scene category to improve performance on a target task.

There are other ways of using relationships between objects in order to improve scene understanding. One way to do so is by learning how to organize objects into

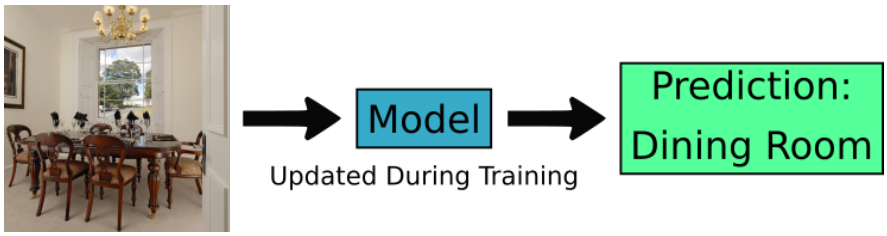


Figure 3.1: An overview of the scene classification task, which involves using a model to map from an image to the image's scene category. Image from [1].

hierarchies and taxonomies, and then employing the learned hierarchies in order to improve performance on the target scene understanding task [227–229]. Similarly, one can learn how objects naturally assemble into tree structures (which do not necessarily consist of the same types of hierarchies as discussed above) and sets, e.g., see 230–232. In particular, tree-based context models have shown promise [225, 230, 231] for refining object predictions and detecting out-of-context objects. Other methods exploit hierarchies of concepts for more specific and niche applications such as content-based image retrieval [233–235].

ObjectBank [236, 237] is yet another way of utilizing object-based information for scene understanding tasks. ObjectBank involves using the output of generic object detectors as feature extractors for scene classification tasks.

3.3 Problem

In our experiments, we consider the problem of scene classification, a standard visual recognition task where the goal is to learn a model capable of mapping from an input image to the image's scene category. Please see Section 1.1.4 for more details. Figure 3.1 shows the typical pipeline for scene classification where an image is fed into a model, and the model makes a prediction about the identity of the scene category (e.g., dining room, kitchen, park, street, etc.).

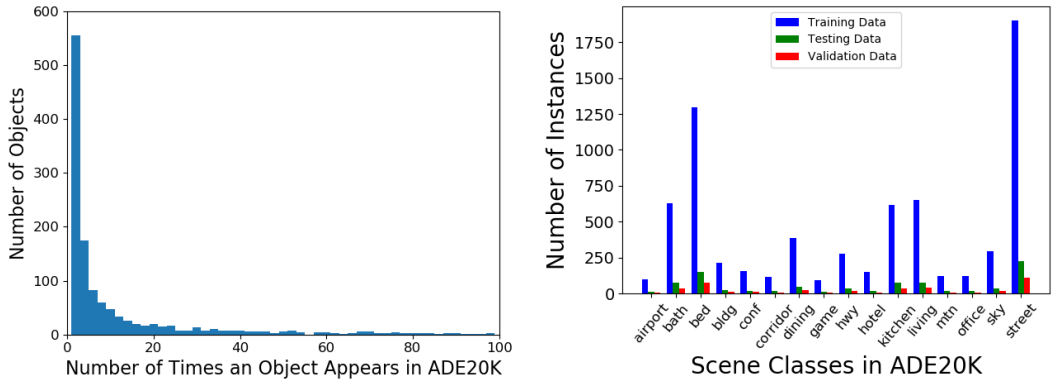


Figure 3.2: Left: A histogram relating each object to the number of times it appears in the subset of the ADE20K dataset used in our experiments. Right: The scene class distribution used in our experiments. Image from [1].

3.4 Data

3.4.1 ADE20K

The ADE20K dataset [21] consists of images of indoor and outdoor scenes captured at ground-level using a standard RGB camera. Each image in the dataset is paired with a scene label (e.g., bedroom, bathroom, park), the set of objects and their parts present in the image, and pixel-level segmentations for these objects and parts. We utilize information associated with all 1,268 unique first-level objects provided by the dataset and exclude parts-based information. We ignore the pixel-level segmentation information. In Figure 3.2, we show how often each object appears in the dataset.

For our experiments, we use the subset of scene classes that have at least 100 images in the combined training, validation, and test set. This filtering results in 16 scene classes and 8,446 total images. 7,131 images are used for training, 876 for testing, and 439 for validation. Figure 3.2 lists the scene classes used in the majority of our experiments and shows how images are distributed between these classes.

We artificially inflate the size of our training data using data augmentation. When an image is sampled from the dataset, we pass it through a data augmentation pipeline consisting of the following steps. First, the image is randomly cropped in such a way that the area of the cropped image is 80% of the original image. Next, the image is

flipped with 50% probability. The brightness, contrast, and saturation are randomly jittered. Each image is resized to 224 pixels-by-224 pixels and normalized using the mean pixel value and standard deviation derived from the training dataset.

3.4.2 WordNet

WordNet [22], a hierarchically-organized lexical database, is used as our source of external knowledge. WordNet groups nouns, verbs, adjectives, and adverbs into sets of "cognitive synonyms" called synsets. Subsequently, these synsets are connected into a hierarchical structure using hypernym-hyponym (a.k.a. superclass-subclass) relations. We align the objects in the ADE20K dataset to their corresponding synsets in WordNet. This alignment enables us to 1) expand the set of object labels to include all parent objects, resulting in more complete, less noisy, and richer semantic information, and 2) hierarchically organize the objects, and exploit the structure of the hierarchy to improve the capabilities of the proposed model.

3.5 Methodology

3.5.1 Summary of our Approach

Our approach consists of several stages, which can be summarized as:

1. Align the objects in the ADE20K dataset to synsets in the WordNet lexical database.
2. Generate and prune an object hierarchy.
3. Using the mined hierarchy, generate an expanded object label set.
4. Train an object recognition CNN to predict the expanded object label set.
5. Calibrate the object prediction scores.
6. Refine the object prediction scores to fix violations in constraints imposed by the mined object hierarchy.
7. Using the refined object prediction scores as features, train a linear multinomial logistic regression model for the scene classification task.

In the following sections, we introduce the specific technical details of our approach.

3.5.2 Classifying Scene Images Using an Object-Based Model

The first stage of our proposed method involves learning a model capable of recognizing the objects that are present in each scene image. This can be done using deep convolutional neural networks (CNNs) [62, 238]. In this work, we employ the ResNet-18 [239] architecture, a popular off-the-shelf CNN architecture, as our object recognition model. A CNN can be trained to predict the scene class directly from pixel-level data, but this limits the model's explainability. Instead, the approach introduced in this chapter decomposes the classification problem into multiple steps. First, the neural network predicts all of the objects (and their parent classes) present in a scene (discussed in this section and the next). Second, a separate model calibrates the scores output by the object recognition model to account for issues with imbalanced and limited data (discussed in Section 3.5.4). Third, the object recognition results are refined using a structured prediction model that exploits the structure of the hierarchy derived from WordNet (discussed in Section 3.5.5). Finally, the calibrated and refined predictions are used as features that are input to a (linear) multinomial logistic regression model that performs the actual scene classification (discussed in Section 3.5.6). Note that each of the aforementioned models needs to be trained seperately. Otherwise, we've found that the neural network tends to learn to circumvent our desired representations, leading to intermediate features that might encode hidden, uninterpretable, and brittle information.

By learning a model that bottlenecks through object predictions, we arrive at intermediate "features" that are discriminative for the scene classification task while still being understandable to humans. Since a simple linear model is used as the classifier, the classification model is also interpretable. Thus, we have interpretable features, each of which represents some measure of how likely an object is to be present in a given image, and an interpretable classifier that specifies the importance of each object for making specific decisions.

Traditional object recognition involves predicting a single object that is typically

the focal point of an image. It is assumed that the object is centered within the frame of the image, and the object makes up the majority of the image's pixels. In contrast, our proposed method requires learning a model capable of simultaneously predicting the presence of all objects in a scene (multi-object recognition). Likewise, these objects vary widely in size and placement within the image. Thus, the task of multi-object recognition in scene images is a much more challenging problem than traditional object recognition, and experimental results in later sections will show that there are many challenges with training a standard deep neural network to solve this problem. That being said, for now we treat the multi-object recognition problem as a binary multi-label classification problem, and we optimize our network by trying to minimize the multi-label binary cross entropy loss:

$$\text{Goal: } \min_{\vec{o}} loss_{bmce},\ loss_{bmce} = -\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(o_i^{(j)} log(\hat{o}_i^{(j)}) + (1-o_i^{(j)}) log(1-\hat{o}_i^{(j)})\right) \tag{3.1}$$

where $o_i^{(j)} \in \{0,1\}$ is the true label for an object i in a given scene instance j, $\hat{o}_i^{(j)} \in [0,1]$ is the probability output by the neural network that object i is present in scene instance j, M is the total number of objects, and N is the total number of training examples in a mini-batch. We train the network using a learning rate of 1.0e-3, a weight decay value of 1.0e-5, and a batch size of 16. ADAM is used as our optimization algorithm. We train the net until convergence is achieved on a held out validation set.

3.5.3 Aligning ADE20K to WordNet

The basic approach discussed in the previous section is not without its flaws. The object labels included in the ADE20K dataset are often noisy and incomplete. Labels are often ambiguous. As an example, "bowl" and "bowls" are treated as separate labels despite being semantically equivalent. Another potential issue is that some labels are overly specific. For example, "silver bird statue" is a label in the dataset, whereas there is limited practicality to having such fine-grained labels. Issues with imperfect annotation are another challenge. Since the dataset is annotated by humans who occasionally make mistakes, sometimes, errors are made by incorrectly labeling objects that do not exist

in a scene or vice versa. There are also challenges that arise due to limited training data. For example, some objects are challenging to learn to recognize from visual data because they appear infrequently. In fact, the majority of objects in the dataset appear fewer than ten times. Deep neural networks are incredibly data-hungry and as such, cannot capture all of the variations of appearance for data-limited object classes. Similarly, some objects are so small in the images, that the neural network cannot learn the fine-grained visual patterns necessary to accurately these objects.

Many of the aforementioned issues can be alleviated by aligning the object labels in ADE20K to the WordNet database. To do this alignment, we begin by semi-automatically mapping (with manual corrections) each of the 1,268 objects in the ADE20K dataset to their corresponding synsets in WordNet. If the object cannot be mapped directly to a synset, it is instead mapped to its closest matching ancestor synset (hypernym). After the mapping is complete, an object hierarchy is constructed by recursively traversing the direct hypernyms of each term in WordNet until the root node for WordNet is reached. Then, the hierarchy is pruned to remove redundant nodes and edges. Specifically, we prune parent nodes when the instances of parent node exactly match the instances of the child node. For example, if "wall" is the only example of "partition" in the dataset because of how the dataset was initially labeled, and the set of instances for"partition" in the dataset exactly matches the set of instances for "wall", we can remove "partition" and connect "wall" directly to the parent class of "partition": "structure". Furthermore, we can perform additional pruning to remove nodes if their corresponding object appears fewer than k times in the subset of the ADE20K dataset used in our experiments because it is often necessary to have some minimum number of examples in order to train an accurate object recognition model. In Figure 3.3, we show a small subgraph of the aligned knowledge graph to highlight how much semantic information can be gained by aligning the objects in the ADE20K dataset to WordNet.

An expanded object set can be generated from the final pruned object hierarchy. Unlike the original object set, which consisted largely of the leaf nodes of the object

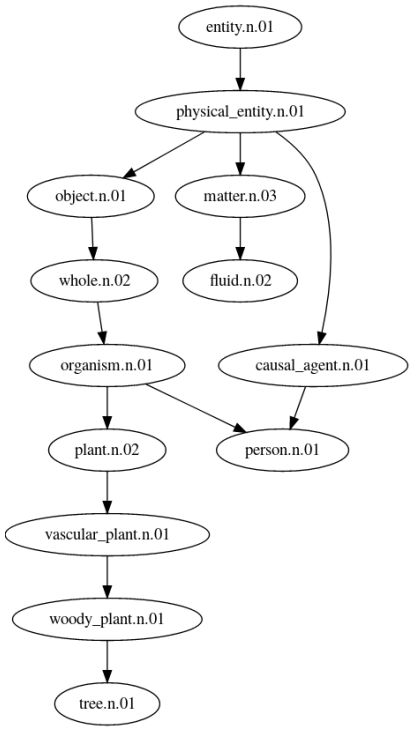


Figure 3.3: Here we show a small subgraph of the aligned knowledge graph between ADE20K and WordNet after extensive pruning. Note how much additional semantic information is captured by even this small portion of the full knowledge graph. Image from [1].

hierarchy, the expanded object set treats every node in the hierarchy as its own object/label. An object recognition model can then be trained to predict the expanded object set. By expanding the graph and predicting the expanded object set, some of the previously mentioned issues can be addressed:

- Ambiguous labels are merged, e.g., “bowl” and “bowls” are mapped to the same synset in WordNet.
- Object information is captured at multiple levels of granularity. Instead of relying on predicting only extremely fine-grained labels such as “silver bird statue”, more general parent classes such as “statue” and “art” can be predicted as well.
- Ancestral categories appear more frequently than some of their children, so while there may not be enough training data to learn to recognize some object, there might be enough training data to recognize one of its ancestors. This preserves

some information about the object that may have otherwise been lost when pruning for rare objects.

3.5.4 Calibrating Object Recognition Scores

Deep neural networks are powerful tools for making predictions and decisions, but since they have a considerable number of parameters and are highly nonlinear, they have issues overfitting to the training data resulting in overconfidence in their predictions. Many real world applications, especially safety-critical applications, require learning-based models to be highly accurate while also being capable of outputting realistic measures of confidence for each prediction (i.e., a calibrated confidence [240]). Using calibrated confidences is exceptionally important from an interpretability viewpoint. If a model says an object is present with a probability larger than 0.5, it should mean that the net believes the object is more likely present in the image than not. Thus, it is important to calibrate the object prediction scores output by our object recognition model, and in this section, we discuss one approach for doing so.

First, the neural network is trained for object recognition on the training data. The parameters of the trained neural network are frozen. Then, the logits (the values output by the network before the sigmoid function) are extracted for each object for each *validation* instance. Our goal is to learn scaling a and shift b parameters for a sigmoid function that maps the logits l_i to a calibrated score $\hat{o}'_i$ for each object i.:

$$\hat{o}'_i = \frac{1}{1 + e^{-a_i(l_i - b_i)}} \tag{3.2}$$

Traditional confidence calibration methods like Platt scaling [241] and temperature scaling [240] use the negative log-likelihood as the supervisory single (i.e., they perform maximum likelihood estimation). In contrast, we calibrate by minimizing a continuous approximation [242] of the F1-measure:

$$\text{Goal: } \min_{\vec{a},\vec{b}} loss_{f1},\ loss_{f1} = -\frac{1}{M}\sum_{i=1}^{M}\left(\frac{2 * \sum_{j=1}^{N} \hat{o}_i'^{(j)} o_i^{(j)}}{\sum_{j=1}^{N} \hat{o}_i'^{(j)} + \sum_{j=1}^{N} o_i^{(j)}}\right) \tag{3.3}$$

where M is the total number of objects, N is the number of validation instances, $o_i^{(j)} \in \{0, 1\}$ is the true label for object i in instance j, and $\hat{o}_i'^{(j)}$ is the calibrated score

for object i in instance j. A continuous approximation of the F1-measure is used as the loss function for the calibration model because the F1-measure is better suited for problems involving class imbalance than the negative log-likelihood, and our object labels tend to be very imbalanced (i.e., many objects appear very infrequently). The f1-measure considers the trade-off between precision and recall, which makes it less aggressive than maximum likelihood estimation, which often pushes probabilities closer to zero for very rare classes.

3.5.5 Exploiting the Hierarchical Structure of the Knowledge Graph to Refine Object Predictions

The mined object hierarchy is useful beyond providing a mechanism for generating the expanded object set. The hierarchy is also useful for helping humans understand when the network makes specific mistakes. Using the object hierarchy, we can treat the problem as a structured prediction problem, and we can refine the object predictions output by the neural network. In this section, we propose a structured prediction model that exploits the structure of the object hierarchy to correct object predictions.

The object hierarchy tells us that a child object should never be predicted as being present when its parent class is predicted as being absent. Up to this point, we have modeled the prediction of the expanded object set as a *flat* classification problem, i.e., the network does not directly utilize the information about how different objects in our expanded object set relate to one another. Since the network is not constrained to perform hierarchical classification, it occasionally (in $\sim 1\%$ of predictions) makes the error mentioned above: predicting a child object when some of its ancestral classes are not present. To correct these easily identifiable mistaken predictions, we formulate the following optimization problem:

$$\begin{aligned} &\text{Goal: } \min_{\vec{\hat{o}}''} loss_{refine}, \\ &loss_{refine} = \sum_{j=1}^{N} \left(\sqrt{\sum_{i=1}^{M} (\hat{o}_i^{\prime\prime(j)} - \hat{o}_i^{\prime(j)})^2} + \sum_{(q,r)\in(child,parent)} max(\hat{o}_q^{\prime\prime(j)} - \hat{o}_r^{\prime\prime(j)}, 0) \right) \end{aligned} \tag{3.4}$$

where M is the number of objects, N is the number of instances, $\hat{o}_i^{\prime\prime(j)}$ is the refined

prediction score for object i in instance j, $\hat{o}_i^{\prime(j)}$ is the calibrated prediction score for object i in instance j, and $(child, parent)$ is the set of all (child, parent) object relations. The idea behind this optimization problem is to minimally change the scores of the predictions (enforced by the first term of the optimization objective) while correcting cases where the score for the child is larger than the score for the parent (enforced by the second term of the optimization objective). Essentially, the model is trying to figure out how the predictions can be changed to satisfy the knowledge graph-based constraints in a minimally-disruptive manner.

3.5.6 Training the Scene Classification Model

Once the multi-object recognition, calibration, and refinement models are trained/encoded, we can train a separate linear multinomial logistic regression model to perform scene classification using the refined object recognition probabilities as input features. To train this scene classification model, we can use the standard multi-class variant of the cross entropy loss:

$$\text{Goal: } \min_{\vec{s}} loss_{ce},\ loss_{ce} = -\frac{1}{N}\sum_{j=1}^{N} log\left(\hat{s}_{true}^{(j)}\right) \tag{3.5}$$

where $\hat{s}_{true}^{(j)} \in [0, 1]$ is the score output by the model for the true scene class for a given scene instance j, and once again, N is the total number of training examples in a mini-batch.

3.6 Experimental Results and Analysis

In the sections that follow, we quantitatively investigate the effectiveness of each component of the proposed approach.

3.6.1 The Importance of Utilizing Grounded, Semantic Information

In this first set of experiments, we aim to see if any benefit can be gained in terms of predictive accuracy by using features grounded to semantic information (e.g., object

Approach	Scene Classification Accuracy
Unmodified ResNet	0.817
Ground Truth Objects (Initial Set) + Logistic Regression	0.910
Ground Truth Objects (Expanded Set) + Logistic Regression	0.910

Table 3.1: Accuracy on the scene classification task for several baseline models. We compare visual features to ground truth semantic features.

presence) instead of using features purely discovered by a neural network. An unmodified ResNet-18 CNN trained for end-to-end scene classification serves as our baseline. In order to evaluate the importance of semantic information, we also train a linear multinomial logistic regression model whose features consist of the *ground truth* object labels for each scene image. Initially, the model only considers the objects present in the ADE20K dataset, but we also want to see how much additional utility can be gained by using the expanded object set derived from WordNet. Thus, we train an additional logistic regression model for scene classification that uses the expanded object set as features. Results appear in Table 3.1.

We can glean some useful insight from this experiment. First, while the unmodified ResNet model performs well, it notably under-performs (by about 10%) the linear model that has access to *perfect* information about the objects present in a scene. Second, we see that if we have perfect knowledge of the initial set of objects, then little is gained (with respect to scene classification) by using the expanded object set derived from WordNet. A plausible explanation for this result is that when the machine learning model has access to perfect information, the model might plausibly discover and, subsequently, exploit the relevant relationships that exist between objects.

Note that, in practice, the model does not have perfect knowledge of the objects in a scene. Instead, this information must be recovered from some sensor (in this case, a camera) via object recognition or detection. This requires a separate learning-based model (in our case, we will use another ResNet-18 neural network). In the next several experiments, we will make the following observations: 1) multi-object recognition in scene images is extremely error-prone, and 2) by integrating information from the mined object hierarchy into the proposed framework, we can learn models that are more reliable, encourage greater trust in the model, and achieve higher predictive

performance.

3.6.2 Understanding the Limitations and Impact of Noisy Object Recognition

Multi-object recognition from scene images can be highly prone to error for several reasons. First, the object recognition model is trained using imperfect and occasionally ambiguous labels. Second, in order to recognize that an object exists in an image, the model must be capable of either approximately localizing the object in the image or detecting surrounding context clues. Third, some objects make up a very small portion of the image (i.e., sometimes only hundreds of pixels). If the object is too small, then there might not be enough details to learn the fine-grained patterns needed to distinguish between similar objects. Fourth, and perhaps most importantly, there is often not enough training data for many of the objects. Without a sufficient amount of data, the network cannot learn features capable of capturing all of the variations of appearance for a given object.

To get a basic understanding of how well a deep neural network can learn to perform multi-object recognition from scene images for our data, we train a ResNet-18 model to predict all objects that appear at least 25 times in the training data. Once the model is trained, we evaluate it by computing the average precision (a summary statistic of the area under the precision-recall curve) for each object on the test dataset. Results appear in Figure 3.4.

We see from these experimental results (Figure 3.4) that most of the objects in the dataset (even after heavily pruning to at least 25 appearances per object) are recognized with poor accuracy. Unsurprisingly, we've also observed that the quality of the object recognition model tends to correlate with the number of training instances for each object class. In Figure 3.5, as we prune the object set to include only those objects that appear most frequently, the object recognition mean average precision (mAP) value increases.

Next, we attempt to understand how pruning the object set based on the minimum number of appearances affects performance on the scene classification task. The left

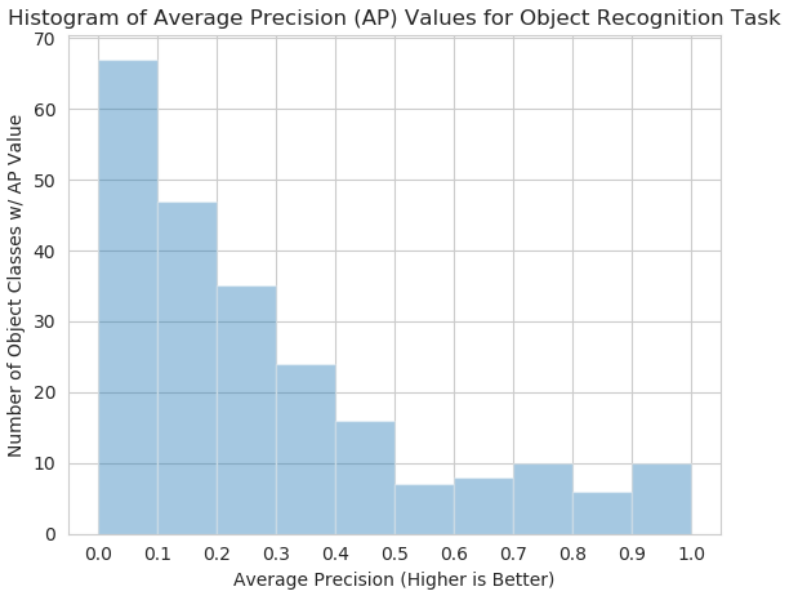


Figure 3.4: We evaluate how well a deep neural network can perform on the multi-object recognition task for objects that appear in at least 25 training instances in the subset of the ADE20K dataset used in our experiments. Image from [1].

graph in Figure 3.6 shows the degradation of classification quality as objects are pruned when using the *ground truth* object data as features, and the right graph this degradation as objects are pruned when using the *predicted* object probabilities as features.

3.6.3 Improving Performance by Utilizing Knowledge Graphs

In Section 3.5.3, we discussed how aligning the objects in the ADE20K dataset to WordNet is beneficial for improving the robustness and explainability of the model. Namely, ambiguous labels are merged; object information is captured at multiple levels of granularity; and while there might not be enough training data to learn how to recognize some objects, there might be enough data to learn how to recognize their ancestors, preserving some information that might have otherwise been lost. We can experimentally validate some of these claims.

Figure 3.5 shows that as objects are pruned based on the number of times they appear in the training dataset, the expanded object set generally achieves equal or slightly higher object recognition results despite working with a larger set of selected

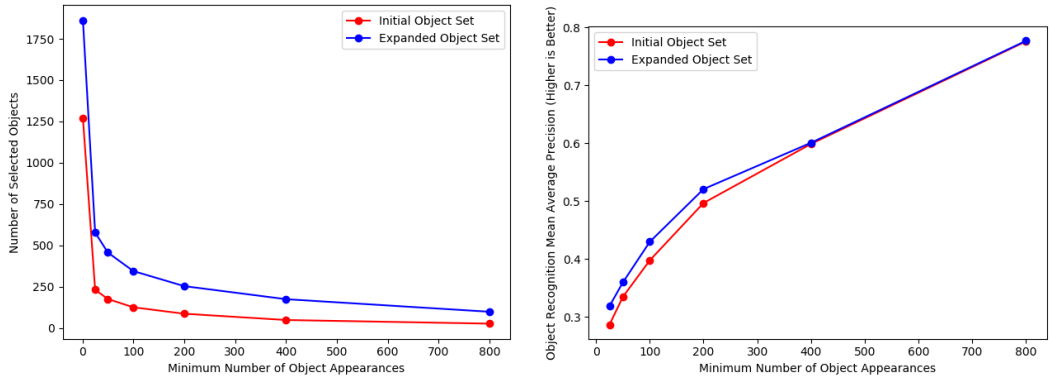


Figure 3.5: We attempt to quantify the effect of sample size on multi-object recognition. Left: We see how many objects are selected as we set different thresholds for the minimum number of times an object must appear in the dataset to not be pruned. Right: We see how object recognition performance is positively impacted as the minimum number of times an object appears in the dataset increases. Image from [1].

Approach	**Min. Obj. Appears.**	**Obj. Rec. mAP**	**Scene Class. Acc.**
Unmodified ResNet	N/A	N/A	0.817
OR + LR: Initial	400	0.599	0.751
OR + LR: Expanded	400	0.601	0.788
OR + LR: Initial	800	0.776	0.752
OR + LR: Expanded	800	0.777	0.764

Table 3.2: We evaluate the performance of various approaches for scene classification including an unconstrained ResNet-18 model and two-stage models involving object recognition (OR) (using object labels from the initial object set and expanded object set) followed by a logistic regression (LR) classifier.

objects. Figure 3.6 shows that while performance on the scene classification task still degrades as objects are pruned, the effect is significantly less severe when employing the expanded object set. In Table 3.2, we summarize how the initial object set-based model (OR + LR: Initial), expanded object set-based model (OR + LR: Expanded), and unmodified ResNet-18 DNN all compare for the scene classification task. While the unmodified ResNet-18 model outperforms the object-based models, it only outperforms the best object-based model by a few percentage points. Additionally, the object-based models remain significantly more interpretable than the unmodified ResNet model.

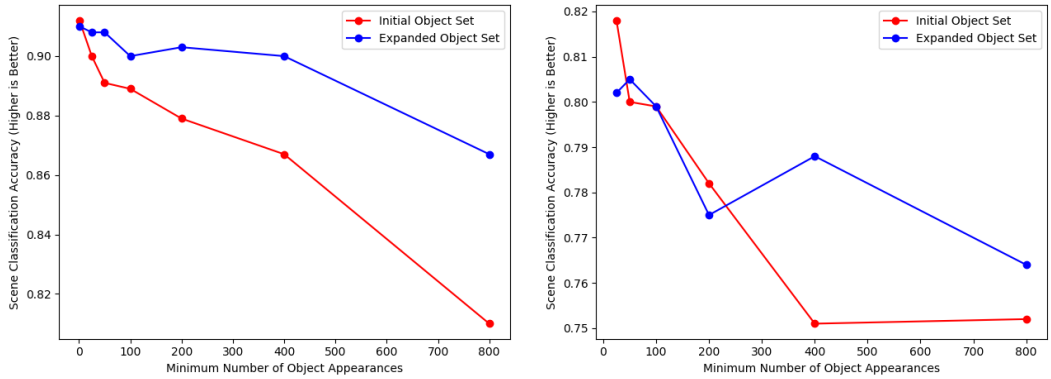


Figure 3.6: We quantify how scene classification performance is affected by the object set size and quality. We evaluate scene classification performance as we set different thresholds for the minimum number of times an object must appear in the dataset to not be pruned. Left: We train and test models using the *ground truth* object data. Right: We train and test models when using *predicted* object recognition scores. Image from [1].

3.6.4 Understanding the Effects of Object Prediction Score Calibration

Up to this point, the experiments have assumed that each model outputs uncalibrated probability estimates. In this set of experiments, we validate the effectiveness of the calibration method proposed in Section 3.5.4. We also want to understand the effect of employing calibrated object prediction scores on the scene classification task. In this set of experiments, we use the macro-f1-measure over all object classes as our evaluation metric because we want to know how the object recognition model performs when the prediction scores are thresholded at 0.5. We're interested in this quantity because it aligns well with human expectations; most humans intuitively assume a score higher than 0.5 means the object is present in an image, and a score lower than 0.5 means the object is absent from an image. Table 3.3 shows that calibration on the validation data improves the f1-measure by several percentage points in each tested case, suggesting that the calibration is effective. However, the calibration method does come with drawbacks; namely, since the calibration method manipulates the parameters of a sigmoid function, some information is lost near the asymptotes of the sigmoid. This loss

Approach	Min. Obj. Appears.	Calib.?	Obj. Rec. Macro-F1	Scene Class. Acc.
OR + LR: Expanded	400	No	0.557	0.788
OR + LR: Expanded	400	Yes	0.587	0.773
OR + LR: Expanded	800	No	0.723	0.764
OR + LR: Expanded	800	Yes	0.741	0.754

Table 3.3: We evaluate how object recognition and scene classification are affected by calibrating the object prediction scores. Specifically, we look at the three stage model consisting object recognition (OR), followed by object score calibration, and finally, followed by logistic regression (LR) for scene classification. In all cases, we consider the expanded object set.

of information is a possible explanation for why there is a minor (about 1%) decrease in scene classification performance.

3.6.5 Refining Object Predictions by Exploiting the Known Structure of the Knowledge Graph

In our final set of experiments for this chapter, we evaluate the effectiveness of the object prediction refinement strategy introduced in Section 3.5.5.

We begin by considering the calibrated model for the object recognition neural network trained to predict the expanded set of objects with at least 400 examples in the training data. On the test dataset, the model makes 150,325 total predictions about the presence of objects. Of these predictions, we identify 784 violations of the constraints imposed by the known knowledge graph (i.e., a child is predicted present whereas its parent is predicted absent). 1,547 object predictions are involved in these 784 violations. Initially, 730 of these object predictions are correct, and 817 of these object predictions are incorrect. After refinement, 828 of the 1,547 violating object predictions are correct, and 719 of the violating object predictions are incorrect.

We can also consider the calibrated model for the object recognition neural network trained to predict the expanded set of objects with at least 800 examples in the training data. On the test dataset, the model makes 85,041 total predictions about the presence of objects. Of these predictions, we identify 441 violations of the constraints imposed by the known knowledge graph. 876 object predictions are involved in these 441 violations. Initially, 418 of these object predictions are correct, and 458 of these object predictions are incorrect. After refinement, 484 of the violating object predictions are correct, and

Approach	Min. Obj. Appears.	Refined?	Scene Class. Acc.
OR + LR: Expanded	400	No	0.773
OR + LR: Expanded	400	Yes	0.774
OR + LR: Expanded	800	No	0.754
OR + LR: Expanded	800	Yes	0.758

Table 3.4: We evaluate the effect of the object refinement strategy on scene classification accuracy. Specifically, we look at the four stage model consisting object recognition (OR), followed by object score calibration, followed by object refinement using the mined object hierarchy, and finally, followed by logistic regression (LR) for scene classification. In all cases, we consider the expanded object set.

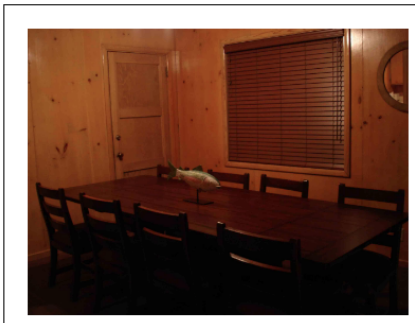
392 of the violating object predictions are incorrect.

We can make several noteworthy observations. First, the object recognition models very rarely make predictions (about 1-2% of object predictions) that violate the knowledge graph constraints despite having no prior knowledge of the relationships that exist between objects. Second, the proposed refinement strategy seems to be capable of correcting some of the incorrect object predictions. 98 of 817 mistakes are fixed in the first experiment, and 66 of 458 mistakes are fixed in the second experiment), providing additional evidence that it is useful to combine knowledge graphs with learning-based models for visual recognition tasks. However, it should be noted that the refinement strategy corrects such a small proportion of the total number of object recognition mistakes that the refinement doesn't significantly affect the downstream scene classification task (see Table 3.4 for empirical validation of this).

3.6.6 Qualitative Results

Finally, we want to show some qualitative results highlighting the explanatory power of the proposed approach. We show several examples in Figures 3.7, 3.8, 3.9, 3.10, 3.11, and 3.12. In each example, we state the predicted class, the model's confidence in its prediction, and a list of the top ten concepts that serve as the strongest evidence in favor of the model's decision. We report a score for each concept (object) which is the score representing the model's belief that the concept exists in the image after calibration and refinement multiplied by the corresponding weight in the multinomial logistic regression relating the given concept to the predicted class. Correct concept

predictions are highlighted in green, and incorrect concept predictions are highlighted in red. While the model isn't perfect, for most instances, when the model predicts the correct class, it does so based on valid evidence. Concepts with the suffix ".n.xx" are derived from the WordNet lexical database. Other concepts without this suffix are taken directly from the ADE20K initial set of concepts. Note that in some cases, we see repeated objects. For example, in Figure 3.7, see both "chair" and "chair.n.01". This is because "chair.n.01" is the ancestor class of all chairs whereas "chair" corresponds to the specific subset of chairs labeled as "chair" in the ADE20K dataset, and the mapping is not one-to-one (e.g., because ADE20K might treat "chair" and "armchair" as different objects, whereas "chair.n.01" includes "chair", "armchair", and all other types of chairs annotated in ADE20K). Note that in some cases, the label in the ADE20K initial set of objects and the parent class from WordNet are one-to-one. In such cases, we discard the WordNet label, and only keep the ADE20K label; details about this pruning are discussed in earlier sections of this chapter.



The network predicts this scene is an instance of a **dining room** with confidence **0.803** because the network thinks it sees the following evidence:

- **Concept:** chair, **Score:** 0.844
- **Concept:** table, **Score:** 0.683
- **Concept:** chair.n.01, **Score:** 0.58
- **Concept:** seat.n.03, **Score:** 0.32
- **Concept:** table.n.02, **Score:** 0.303
- **Concept:** windowpane.n.01, **Score:** 0.206
- **Concept:** painting.n.01, **Score:** 0.197
- **Concept:** windowpane, window, **Score:** 0.175
- **Concept:** pane.n.01, **Score:** 0.113
- **Concept:** art.n.01, **Score:** 0.097

Figure 3.7: We show an example annotated explanation of the output of our knowledge graph-based approach applied to a dining room scene instance. We show the scene class prediction, the model's confidence in its scene class prediction, and the top-10 strongest pieces of evidence in favor of the prediction. Correct concept predictions are highlighted in green, and incorrect concept predictions are highlighted in red.

The network predicts this scene is an instance of a **street** with confidence **0.998** because the network thinks it sees the following evidence:

- **Concept:** sidewalk, pavement, **Score:** 1.31
- **Concept:** plant, flora, plant life, **Score:** 0.496
- **Concept:** building.n.01, **Score:** 0.413
- **Concept:** building, edifice, **Score:** 0.353
- **Concept:** container.n.01, **Score:** 0.321
- **Concept:** road.n.01, **Score:** 0.312
- **Concept:** way.n.06, **Score:** 0.31
- **Concept:** person, individual, someone, somebody, mortal, soul, **Score:** 0.307
- **Concept:** person.n.01, **Score:** 0.291
- **Concept:** causal_agent.n.01, **Score:** 0.281

Figure 3.8: We show an example annotated explanation of the output of our knowledge graph-based approach applied to a street scene instance. We show the scene class prediction, the model's confidence in its scene class prediction, and the top-10 strongest pieces of evidence in favor of the prediction. Correct concept predictions are highlighted in green, and incorrect concept predictions are highlighted in red.

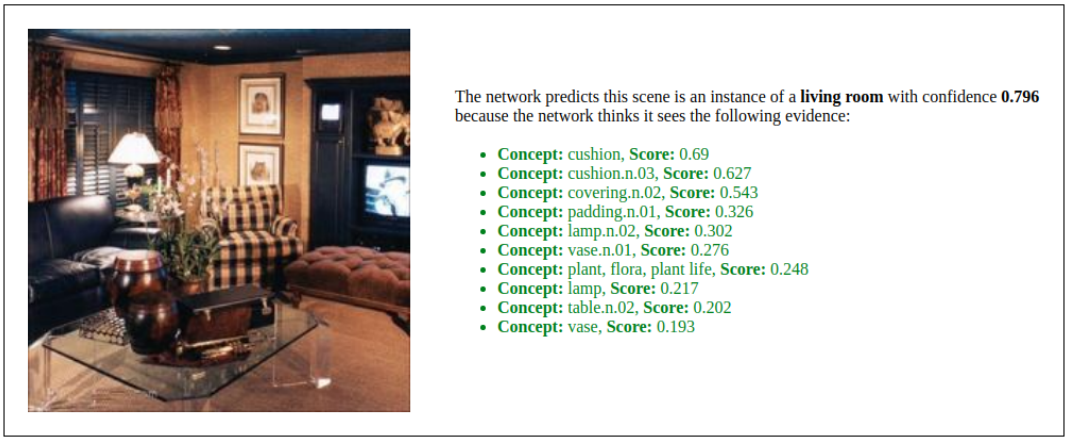


Figure 3.9: We show an example annotated explanation of the output of our knowledge graph-based approach applied to a living room scene instance. We show the scene class prediction, the model's confidence in its scene class prediction, and the top-10 strongest pieces of evidence in favor of the prediction. Correct concept predictions are highlighted in green, and incorrect concept predictions are highlighted in red.

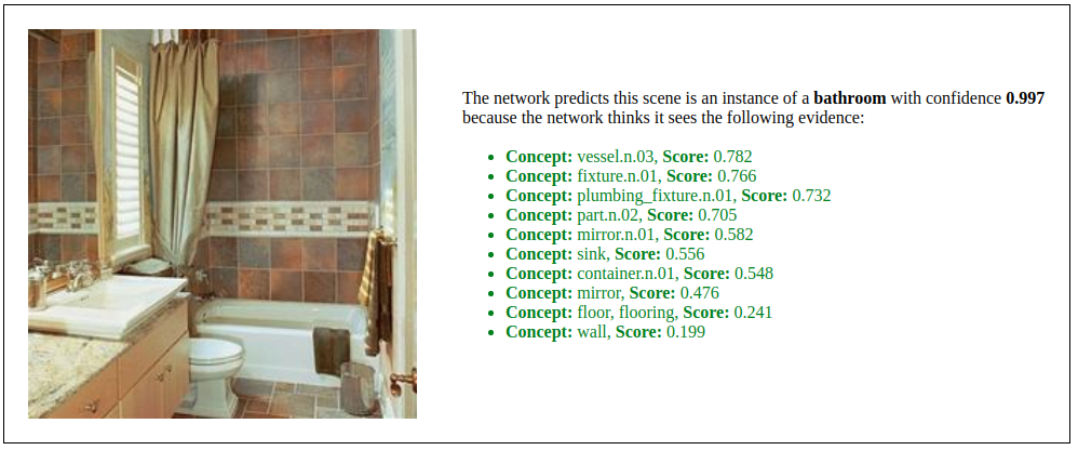


Figure 3.10: We show an example annotated explanation of the output of our knowledge graph-based approach applied to a bathroom scene instance. We show the scene class prediction, the model's confidence in its scene class prediction, and the top-10 strongest pieces of evidence in favor of the prediction. Correct concept predictions are highlighted in green, and incorrect concept predictions are highlighted in red.

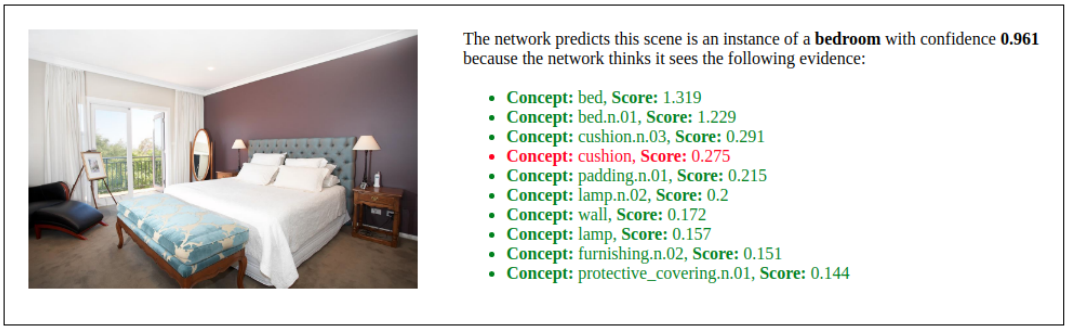


Figure 3.11: We show an example annotated explanation of the output of our knowledge graph-based approach applied to a bedroom scene instance. We show the scene class prediction, the model's confidence in its scene class prediction, and the top-10 strongest pieces of evidence in favor of the prediction. Correct concept predictions are highlighted in green, and incorrect concept predictions are highlighted in red. One interesting thing to note with this instance is that the concept "cushion" is missing whereas the parent class "cushion.n.03" is present. This is because "cushion" specifically refers to the cushion of a sofa whereas "cushion.n.03" is an ancestor of the"pillow" concept.

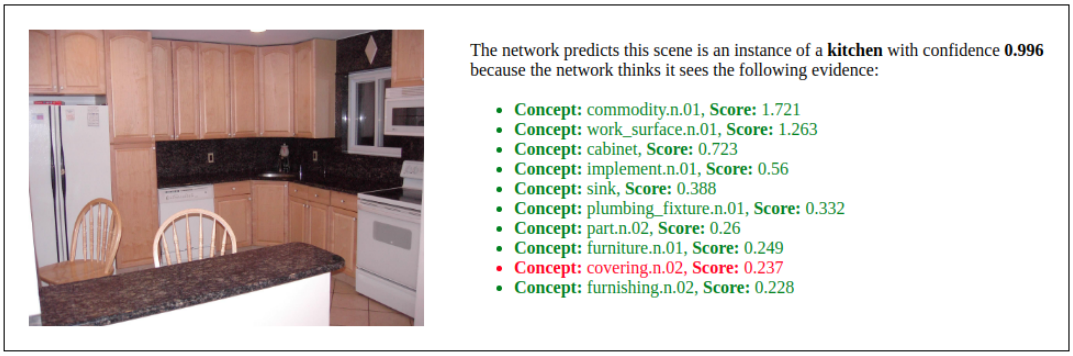


Figure 3.12: We show an example annotated explanation of the output of our knowledge graph-based approach applied to a kitchen scene instance. We show the scene class prediction, the model's confidence in its scene class prediction, and the top-10 strongest pieces of evidence in favor of the prediction. Correct concept predictions are highlighted in green, and incorrect concept predictions are highlighted in red.

Chapter 4

Deriving New Visual Concepts: Discovering a Novel Representation for Explanation-Driven Visual Recognition

Note: This chapter is based on work originally published as "ScenarioNet: An Interpretable Data-Driven Model for Scene Understanding" [3].

4.1 Introduction

Recall once again that the focus of this thesis is to present several approaches for grounding deep neural networks to human-understandable visual concepts. Also, recall that several issues exist when employing a neural network that bottlenecks through a visual concept recognition layer, and as a result, we propose several major methodological extensions for improving visual concept-based neural network models. In the last chapter, we focused on the first extension: augmenting an existing set of visual concepts using knowledge graphs. In this chapter, we focus on the second extension: deriving new representations from an existing set of visual concepts. Specifically, the new derived representation should satisfy several properties: the representation should be easily understood when presented to human participants; the representation should be extracted from visual input data with reasonable accuracy, and the representation should be discriminating for the downstream target task. In this chapter, we present a novel interpretable data-driven representation that satisfies these properties.

As a rule of thumb, explainable machine learning models should satisfy two properties: 1) the input features should be low-dimensional and able to be easily understood by humans, and 2) the model (e.g., a classifier) should be transparent, i.e., structurally and computationally simple (e.g., with few parameters), easy for a human to inspect every component, and operate in a principled manner. The method proposed in this

chapter satisfies these two requirements. We introduce a new feature representation, the scenario, which satisfies the first property because scenarios are low-dimensional and grounded to human-understandable semantic concepts. We also discuss how to integrate scenarios with convolutional neural networks (CNNs) to improve their transparency, thus satisfying the second property.

We introduce **scenarios**, an interpretable, data-driven representation derived from an initial set of visual concepts. Scenarios are based on *sets of frequently co-occurring visual concepts*. Scenarios should satisfy a few key properties:

1. Scenarios are composed of one or more visual concepts.
2. The same visual concept can appear in multiple scenarios, and this should reflect the context in which the concept appears. For example, if our visual concepts consist of objects as in the previous chapter, then {keyboard, screen, mouse} and {remote control, screen, cable box} might make up two distinct scenarios. Notice that both of these example scenarios contain the "screen" object, but in the first scenario, the screen is a computer monitor, and in the second scenario, it is a television screen.
3. Scenes can be decomposed as combinations of scenarios. If we once again use objects as the base visual concepts, a bathroom scene instance might decompose as the union of {shower, bathtub, shampoo}, {mirror, sink, toothbrush, toothpaste}, and {toilet, toilet paper}.
4. Scenarios are flexible and robust to missing objects. A scenario can be present in a scene instance without requiring all of its constituent objects to also be present.

Scenarios can be identified from training data via our novel pseudo-Boolean matrix factorization (PBMF). PBMF takes a binary matrix relating visual concepts to individual scene instances and decomposes it into two matrices: a dictionary matrix and an encoding matrix. Each basis vector of the dictionary represents one scenario. The encoding matrix assigns scenarios to each individual scene instance. Figure 4.1 presents a visual representation of the decomposition. PBMF is integrated into the proposed

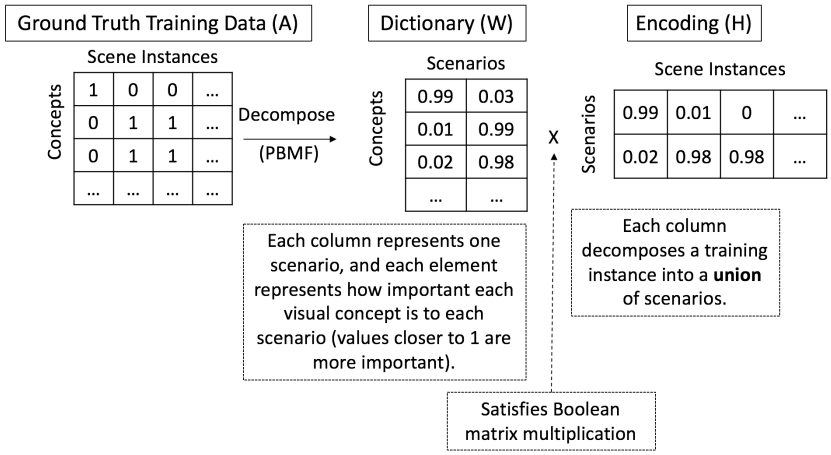


Figure 4.1: A visual representation of the pseudo-Boolean matrix factorization, which takes a binary matrix representing which visual concepts appear in each scene instance and decomposes it into a dictionary matrix, which assigns visual concepts to scenarios, and an encoding matrix, which assigns scenarios to each scene instance.

ScenarioNet model, a special convolutional neural network (CNN) architecture that refines the scenario dictionary based on visual feedback and predicts scenarios from visual data.

ScenarioNet replaces the final convolutional layers in standard CNNs with the scenario block (see Figure 4.2). The scenario block consists of three parts. The first set of layers in the scenario block is responsible for predicting which scenarios are present in an image and additionally, are compatible with the Class Activation Mapping technique [2] introduced in Chapter 2, so the model can identify which parts of an image are attended to by ScenarioNet when recognizing the presence of each scenario. The second set of layers in the scenario block use a PBMF-based loss function to guide the learning/refinement of the dictionary of scenarios and provide feedback to the scenario recognition layers, enabling the model to predict the presence and strength of each scenario in a given image. The final set of layers in the scenario block are equivalent to a multinomial logistic regression model that takes scenarios predictions as input features and outputs predictions for the downstream classification task.

It should be noted that during training, ScenarioNet only requires information about the *presence* (not location) of visual concepts in an image. For the downstream classification task, ground truth class labels are needed during training. During testing, only

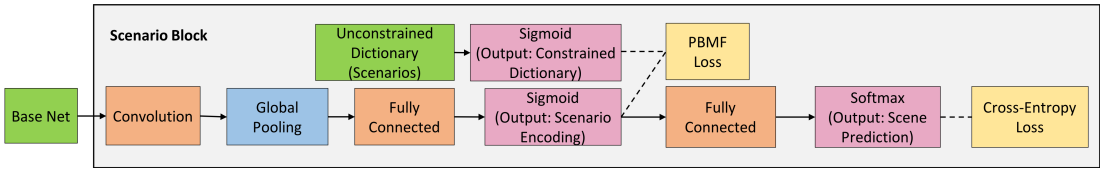


Figure 4.2: An overview of the ScenarioNet architecture. The key contribution of the architecture is the scenario block, which replaces the final fully connected layers of a standard convolutional neural network. The scenario block consists of three parts: 1) a set of layers that predict the presence of each scenario in a given image and are compatible with the class activation mapping technique [2], enabling the network to identify which parts of an image ScenarioNet attends to when recognizing whether or not a scenario is present in a given image, 2) layers that use a pseudo-Boolean matrix factorization-based loss function to fine-tune the dictionary of scenarios and provide feedback to the scenario recognition layers, and 3) layers equivalent to a multinomial logistic regression classifier that use scenarios as low-dimensional, interpretable features for the downstream classification task.

images are provided.

As in the previous chapter, we once again use scene classification as our motivating application. Furthermore, we again use object presence as our base visual concepts. However, it should be noted that the framework can be applied to other applications and utilize other types of visual concepts, and the extension provided in the next chapter will highlight a very different application.

ScenarioNet has some advantages over other convolutional neural network architectures. Specifically, it's more interpretable than traditional CNNs because it is capable of producing semantically- and visually-grounded evidence when making decisions and predictions. For example, consider the scene classification task. By bottlenecking through a scenario recognition layer, the network can use the predicted scenarios as low-dimensional semantic features for the downstream scene classification task. These scenarios are generally easy-to-interpret by humans, and subsequently, humans can verify the existence of each predicted scenario in a given image by inspecting the scenario-localizing attention maps output by the model. Finally, the model provides the level of influence each scenario exerts when assigning a class, so humans can understand how important each of the predicted scenarios is for making the downstream classification. All of these tools provided by ScenarioNet help humans to better understand how the model arrives at each of its decisions.

One might question why it is necessary to bottleneck through the scenario representation instead of using the base visual concept predictions. As was seen in the previous chapter, many of the base visual concepts are often hard to recognize from visual data. In contrast, we will show that scenarios can be recognized from visual data with much higher accuracy; thus, decisions based on scenarios can be trusted with much higher confidence than using the predicted base visual concepts. Another advantage of the scenario representation is that it is much lower-dimensional than a representation derived from predicting the entire initial set of visual concepts. Likewise, the scenario representation groups visual concepts that co-occur with high frequency, and the scenarios during the learning process are constrained to be approximately orthogonal to one another, so the scenario representation has very little redundancy in the information it encodes compared to the representation consisting of predictions over the initial set of visual concepts. Thus, it is much easier for a human to evaluate decisions based on tens of scenarios compared to hundreds or thousands of the initial visual concepts. Similarly, by the formulation of ScenarioNet, in contrast to purely human-specified visual concepts, scenarios are naturally discriminative for the downstream target task. To summarize, recall the major flaws with visual concept-based models mentioned several times throughout this thesis: visual concepts can be difficult to recognize from visual data, and many visual concepts are often redundant because the set of visual concepts might be very high-dimensional, and a significant portion of the visual concepts will not be discriminative for the downstream task. Scenarios suffer from neither of these issues.

4.2 Related Work

In this section, we present our work in the context of existing efforts to learn meaningful groups of objects for scene understanding tasks since this is very similar to the example use case discussed in this chapter. We do not discuss existing approaches for improving the interpretability of machine learning models nor do we discuss methods related to exploiting visual concepts as these methods have already been thoroughly discussed in Chapter 2.

4.2.1 Learning Meaningful Groups of Objects

Our proposed method is not the first method for discovering meaningful groups of objects and exploiting context between objects. The simplest class of object-based representations are those that utilize *pairwise* co-occurrence relationships between objects (e.g., [222]) in order to model context between objects. Scenarios can capture information beyond pairwise relationships by efficiently learning groups of objects that vary in size.

Many other works focus on modeling hierarchical relationships between objects. Feng and Bhanu [243] propose a method for constructing tree-based hierarchies of concepts based on object co-occurrence graphs. In their method, objects sharing an ancestor node make up scene concepts, which are similar to our scenarios but different in key ways. Specifically, to compute explicit scenarios from these scene concepts, one must identify where the object tree should be cut. Moreover, while it is possible to assign individual concepts to multiple scene concepts by cutting the tree at different ancestor nodes, it is challenging, if not impossible, to assign an object to multiple scene concepts where the object serves different functions within each scene concept. For example, recall how a "screen" object can take on different meanings when paired with a keyboard and mouse versus a cable box and remote. A tree structure cannot capture this type of context, but our scenarios can easily handle this situation. Furthermore, our method provides information beyond clustering objects, e.g., PBMF (and by extension, ScenarioNet) can tell us how important each object is to a given scenario and how to decompose scene instances as the union of scenarios.

There are a number of other tree-based and hierarchical models for scene understanding applications. Choi et al. [231] introduces a method for organizing objects into a tree structure where the edges of the tree express positive and negative correlations between objects and latent variables. Fan et al. [234] construct ontologies for content-based image retrieval using visual concept-based hierarchies. Lan et al. [229] explore modeling context at three levels: parts of objects, objects, and visual composites.

Past work has also explored how to use sets of objects to improve object detection.

Li et al. [244] propose a method for finding groups of objects of arbitrary size and detecting these groups within images by employing deformable parts model. Cinbis and Sclaroff [232] construct classifiers that exploit object-object and object-scene relations within sets of objects to re-score and remove noisy detections.

4.3 Problem

As has been mentioned previously, in this chapter, we will use scene classification with object-based representations as our motivating example application. Please see Sections 1.1.4 and 3.3 for more details. Scene classification with object-based representations is an especially good application for evaluating the interpretability of scenarios. This is because humans have an intuitive understanding of how objects relate to one another and to specific scene categories. For example, humans naturally understand that showers, sinks, and toilets frequently co-occur, and if a swimming pool were to be added to this set of objects, humans would likely think that something is unusual/anomalous about the grouping. Similarly, if humans know that a scene contains a shower, sink, and toilet, then they can quickly hypothesize that the scene is a bathroom. In other applications and with other types of visual concepts, humans might not have such a natural understanding of the domain, and thus, it would be more difficult to run human validation experiments.

It should be noted that further discussions in this chapter will be targeted more specifically in terms of objects instead of visual concepts and scene classification instead of general classification tasks.

4.4 Data

We conduct experiments on the SUN-RGBD [245] and ADE20K [21] datasets, both of which are benchmark datasets designed for evaluating scene classification models. Each dataset is divided into separate training and test sets using the recommended splits for the SUN-RGBD and a random split for the ADE20K dataset. Furthermore, we only consider objects that appear in at least fifty instances for both datasets. This

pruning results in using 118 objects for the SUN-RGBD dataset and 339 objects for the ADE20K dataset. To augment our training data, we apply random cropping and horizontal mirroring. On the SUN-RGBD dataset, we restrict our class labels to the 15 most frequently occurring scene classes, reserving 100 samples per class for test data, and generating 1000 samples per class (via applying data augmentation to the training instances) for the training data. For the ADE20K dataset, we restrict our class labels to the 31 most frequently occurring scene classes, reserving 25 samples per class for test data, and generating 500 samples per class (via applying data augmentation to the training instances) for training data. This careful stratified sampling of training and test instances ensures that we can evaluate the performance of our proposed model without having to worthy about issues of class imbalance. Furthermore, for both datasets, we use twenty percent of the training instances as validation data for tuning hyperparameters. We use the ResNet-18 architecture [239] as our backbone CNN architecture and replace the final fully-connected layers with the scenario block.

4.5 Methodology

4.5.1 Identifying Scenarios from Data: Pseudo-Boolean Matrix Factorization

In this section, we consider the problem of how to form scenarios from the base visual concepts; i.e., we propose a method for clustering objects into scenarios based on co-occurrence statistics.

Assume that there exists a training set of scene instances and a finite set of pre-specified objects. For each scene instance in the training set, ground-truth annotations for the presence (or absence) of every object in the object set are provided via human labeling. For each of the training instances, we encode object presence information in the form of a binary vector. In this vector, an element is one if the corresponding object is present in the scene instance; otherwise, if the corresponding object is absent in the image, the element is zero. The vectors for all training instances are concatenated to form a matrix A. Each row of A corresponds to a specific object, and each column

corresponds to a specific training instance.

Next, we specify the number of desired scenarios k. Details of how to estimate this value from the data are discussed in Section 4.5.1. Then, A is decomposed into two *approximately* binary matrices. The first matrix resulting from this decomposition is a dictionary matrix W which assigns objects to scenarios. The second matrix resulting from this decomposition is an encoding matrix H that tells us how each scene instance can be represented as a combination of the learned scenarios. Each column of W represents a single scenario, and each row of W represents a specific object. If element W_{ij} is zero (or very small), object i does not belong in scenario j. The closer W_{ij} is to one, the stronger influence object i exerts on scenario j. Each column of H represents a specific scene instance, and each row of H represents a specific scenario. If element H_{ij} is zero (or very small), then scenario i is not a component of scene instance j. The closer H_{ij} is to one, the more influence scenario i exerts on scene instance j.

Formulation of Pseudo-Boolean Matrix Factorization

We propose a novel matrix factorization-based approach for identifying scenarios. Our approach approximates Boolean matrix factorization (BMF) [246] in a differentiable framework. In BMF, A, W, and H are binary matrices and the matrix multiplication is Boolean (denoted as $\circ$), i.e., instead of using summation ($1 + 1 = 2$), we take the union instead ($1 \cup 1 = 1$). The rationale behind this modeling decision is presented in Figure 4.3. To summarize the figure, Boolean matrix multiplication is important for our problems because we only care about recovering the absence or presence of each object in a scene during the reconstruction and we do not care about object counts. Taking the union of all of the objects in all of the present scenarios has the effect of assigning an object to a scene instance only once, instead of counting every time it appears in a constituent scenario of the given scene instance. Thus, Boolean matrix multiplication prevents the problem of overcounting objects when all we care about is object presence and not object count. However, Boolean matrix factorization is not ideal for our specific use case; namely, we want to integrate it into a neural network architecture which requires an optimization formulation that is continuous,

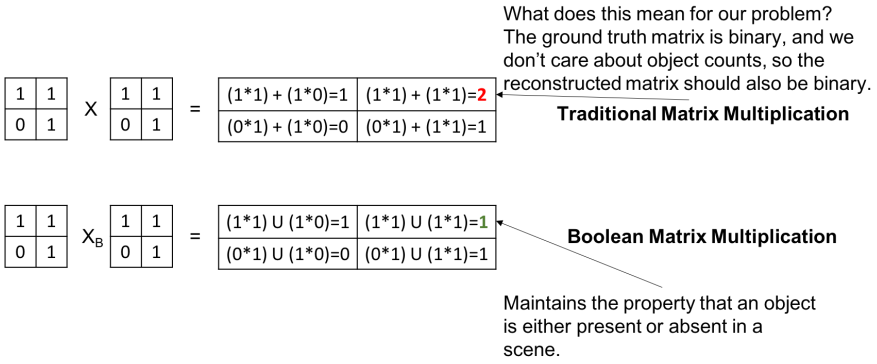


Figure 4.3: Above, we show the difference between traditional matrix multiplication and Boolean matrix multiplication. Boolean multiplication replaces the addition operator with a union operator. It is important that the matrix factorization for identifying scenarios uses Boolean matrix multiplication instead of traditional matrix multiplication because when reconstructing the objects-scene instances matrix, we only care about object presence and not object counts.

and we want the learned dictionary to satisfy several other properties specific to our problem of interest. As such, we need to make several approximations to Boolean matrix factorization. In this section, we start with the basic formulation of BMF and step-by-step transform it into a novel matrix factorization that is better suited for our specific use case.

Suppose we have m objects, n scene instances, and k scenarios. The basic formulation of Boolean matrix factorization is as follows:

$$\min_{W,H} ||(A - W \circ H)||_1 \; s.t. \; W \in \{0,1\}^{m \times k}, H \in \{0,1\}^{k \times n} \tag{4.1}$$

Beyond modeling the problem more realistically than a traditional unconstrained or non-negative matrix factorization, BMF is well-suited for identifying scenarios from data for several reasons. First, it efficiently compresses and preserves semantic information about the scene contents into a low-dimensional representation. Second, the basis vectors can be easily understood by humans. Third, it discovers meaningful interactions between objects and thus, enables us to systematically identify powerful contextual relationships that exist between specific sets of objects. Fourth, the encoding vectors tend to be sparse, meaning each scene instance is expressed as the union of a small subset of scenarios.

Ultimately, we want to integrate the matrix factorization with a neural network.

Neural networks require differentiable functions. Thus, it is important that the optimization problem for the matrix factorization is differentiable and can be solved by gradient descent. The formulation in Equation 4.1 is not continuous, and thus, cannot be solved via gradient descent. Instead, we need to approximate Boolean matrix multiplication as a continuous function, and in the optimization formulation, we need to relax the values of W and H to lie in $[0, 1]$ instead of being perfectly binary. The obvious way of converting from Boolean matrix multiplication to a continuous approximation of Boolean matrix multiplication is to replace the logical operators with those from infinite-valued/fuzzy logic [247, 248]. In infinite-valued logic, the conjunction operator is replaced by the weak conjunction operator $x \cap y \approx min\{x, y\}$, and the disjunction operator is replaced by the weak disjunction operator $x \cup y \approx max\{x, y\}$. When x and y are binary, then these operators are identical to their Boolean logic counterparts, and when x and y instead belong to the interval $[0, 1]$, then the output of these operators remain in the interval $[0, 1]$. However, using these operators is difficult in practice. W and H are very large matrices, and so there are many operations that need to be replaced by the aforementioned infinite-valued logic operators. Because the max and min operators are not easy to optimize over (requiring the use of subgradients), and because these operators are used many times in the optimization formulation, the optimization landscape can become very jagged, which is difficult to optimize over, and the computational graph needed to perform backpropagation becomes very large, making the optimization resource-heavy in terms of memory and computation. Also, the infinite-valued logic operators can only be used when the H matrix is binary. To clarify, if we know that scenario a and scenario b are present in a given scene, then for each object, we can take the weak disjunction of their real-valued scores for scenario a and b, so if object x has a score of 0.85 in scenario a and a score of 0.75 in scenario b, then $W_{x,a} \cup W_{x,b} \approx max\{W_{x,a}, W_{x,b}\} = max\{0.85, 0.75\} = 0.85$. However, if H is not binary, then there is no way to say that "scenario a and scenario b are present in a given scene"; instead, we must say scenario a is present in a scene with score H_a, and scenario b is present in a scene with score H_b. There is no obvious way to incorporate these scores in the matrix factorization. Instead, we would likely need to threshold the H

matrix which further complicates the optimization problem, making the optimization landscape much more difficult to optimize over when using gradient descent, and once again, increasing the size of the computation graph needed for backpropagation.

Instead, we can approximate Boolean matrix multiplication in such a way that the approximation is still continuous, W and H can both be real-valued in the interval $[0, 1]$, and the matrix multiplication (and backpropagation over the matrix multiplication) can be computed much more efficiently by vectorizing many operations. We can approximate Boolean matrix multiplication (in an ad hoc way) as $W \circ H \approx \min(WH, 1)$. When W and H are binary, this is equivalent to Boolen matrix multiplication; however, it can still handle cases where W and H lie in $[0, 1]$. Yet, this approximation is not without its flaws. We highlight one major flaw with a simple example. Suppose we have three scenarios present in a scene with encoding coefficients close to one, and suppose that object x in scenario a takes on a value of 0.3, object x in scenario b takes on a value of 0.35, and object x in scenario c takes on a value of 0.25. Using the weak disjunction from the infinite-valued logic operators, we would find that $max(0.3, 0.35, 0.25) = 0.35$ which says the object isn't likely present in the image, and this agrees with common sense. Using the new formulation, we would find: $min(0.3 + 0.35 + 0.25, 1) = 0.9$, which suggests that the object is present in the scene, and this is disagrees with our expectations. Thus, our formulation is flawed when many scenarios contribute small amounts of evidence in favor of a specific object. However, we found that, in practice, this problem is relatively uncommon, and later we discuss how to impose additional constraints on W and H to push values close to zero and one, helping to further alleviate this problem. Another issue with using $\min(WH, 1)$ to approximate Boolean matrix multiplication is that it results in cases where the gradient vanishes. Instead, we adjust the approximation even further to avoid the vanishing gradient problem: $\min(WH, 1) \approx \min(WH, 1 + 0.01WH)$. Our basic **pseudo-Boolean matrix factorization (PBMF)** formulation is as follows:

$$\min_{W,H} ||(A - \min(WH, 1 + 0.01WH))||_F^2 \; s.t. \; W \in [0, 1]^{m \times k}, H \in [0, 1]^{k \times n} \tag{4.2}$$

Equation 4.2 can be further improved, specifically for finding better scenarios. We

modify the basic formulation of pseudo-Boolean matrix factorization to include three additional terms: an orthogonality penalty that encourages diversity between scenarios, a penalty that forces the dictionary matrix to be as close to binary as possible, and a penalty that forces the encoding matrix to be as close to binary as possible. It should be noted that the penalties for binarizing the dictionary and encoding matrices can also be replaced by penalties encouraging sparsity (e.g., the $\ell 1$-norm) in the dictionary and encoding matrices which accomplishes similar purposes while traditionally being more friendly to optimize over since sparse penalty functions be written as convex functions. We also introduce a weight matrix Ω, which can be determined via an inverse document frequency weight scheme, that decreases the importance of common objects and increases the importance of rare objects during the factorization.

$$\begin{aligned} &\min_{W,H} ||\Omega \bullet (A - \min(WH, 1 + 0.01WH))||_F^2 \\ &+ \alpha_1 ||W^\intercal W - diag(W^\intercal W)||_F^2 + \alpha_2 ||W - W^2||_F^2 + \alpha_3 ||H - H^2||_F^2 \\ &s.t.\ W \in [0,1]^{m \times k}, H \in [0,1]^{k \times n}, \\ &\Omega_{ij} = \max\left(A_{ij} * \left(1 + \log\left(\frac{n}{f_n(i)}\right)\right), \beta\right) \end{aligned} \tag{4.3}$$

$\bullet$ denotes element-wise matrix multiplication. The αs represent trade-off hyperparameters that can be determined via hyperparameter-tuning methods (we use sequential model-based optimization [249], a.k.a. Bayesian optimization, to set these values). $f_n(i)$ is a function that recovers the number of scene instances in which object i is present in the training dataset. β is a constant value representing the lower bound of the inverse document frequency set as a hyperparameter. We set $\beta = 0.5$ in our experiments.

There are several questions we need to address with regard to how to formulate and solve the PBMF optimization problem. These include how to select the number of scenarios, how to initialize the dictionary and encoding matrices, and how to computationally solve the PBMF optimization problem.

Selecting the Number of Scenarios

The formulation of scenarios presented in this chapter requires the number of scenarios k to be pre-specified. There are several methods available that can aid in determining the

proper number of scenarios. However, it should be noted that there is not necessarily a correct number of scenarios. By varying the number of scenarios, the model can capture information about a scene at different levels of granularity. For example, consider using objects as our base visual concepts. If the number of scenarios is set to two, one scenario might represent indoor scenes while the second might represent outdoor scenes. If the number of scenarios is set to the number of classes in the dataset, then the learned scenarios will likely be class-correlated scenarios where each scenario captures the objects most common to a specific class. If the number of scenarios is set to much larger than the number of classes, the model will capture more information about local context; e.g., it might capture a computer scenario consisting of a keyboard, monitor, mouse, computer tower, and computer screen. That being said, some methods exist for identifying the number of scenarios by considering the trade-off between finding a low-dimensional representation while preserving as much semantic information as possible.

The simplest approach for estimating the number of scenarios is to generate scenarios using different values of k and have a human manually inspect the output scenarios. The human can then pick the value of k where the identified scenarios are most consistent with his or her expectations (i.e., where the objects don't appear to be under- or over-clustered). A more principled and quantitative approach is to generate scenarios at different values of k, plot the optimum of the objective function of the PBMF optimization problem (or alternatively, just the reconstruction error component of the objective function) at evenly-spaced intervals of k, and identify the elbow in the graph where the values begin to plateau. A third approach makes use of the orthogonal non-negative matrix tri-factorization [250]. This is the non-negative counterpart to the singular value decomposition: $A \approx USV^\intercal,\ s.t.\ U \succeq 0, S \succeq 0, V \succeq, UU^\intercal = I, V^\intercal V = I, S$ is diagonal. Then, k can be determined by finding the number of elements in S greater than or equal to some threshold. Fourth, we can implement scenario selection directly into PBMF by imposing an $\ell 2, 1$-norm on W and/or H.

In this work, we take a fifth approach that is specifically useful for finding k when PBMF is integrated into a convolutional neural network. We treat k as a hyperparameter, and we use sequential model-based optimization (a.k.a. Bayesian optimization)

using Tree-Structured Parzen Estimators [249] to efficiently find "good" values of k. Specifically, we find the value of k such that the (weighted) sum of the evaluation metrics for scenario recognition and scene classification are maximized on a held-out validation set.

Initializing the Dictionary and Encoding Matrices

The PBMF optimization problem is highly non-convex. Thus, the solution achieved via gradient descent is sensitive to the initial choice of W and H. While it is possible to initialize W and H randomly, we have found that the method converges faster and to better solutions if we employ a more careful initialization scheme. We present one such scheme in this section.

Non-negative matrix factorization (NMF) finds basis vectors that decompose scenes into their parts by finding clusters of objects. NMF is an easier optimization problem than PBMF, and many libraries exist for efficiently performing NMF. However, NMF is ill-suited for our problem for two reasons. First, it uses traditional matrix multiplication instead of Boolean matrix multiplication, and as discussed previously, this type of matrix multiplication is not ideal for our applications. Second, the dictionary and encoding matrices output by NMF are only constrained to be non-negative, meaning elements of the learned basis can take arbitrarily large values, which makes interpreting the dictionary and encoding matrices more challenging than the matrices output by PBMF. However, the bases produced by NMF with some re-scaling of the output matrices can provide a good initialization for pseudo-Boolean matrix factorization, which can then subsequently be used to refine the dictionary and encodings output by NMF to meet the constraints of our problem while continuing to lower the reconstruction error.

To rescale the matrices output by NMF, we can repurpose the diagonal scaling trick introduced by Zhang et al. [251]. Zhang et al. showed that if A is binary and can be factored exactly as $A = WH$ where W and H are non-negative, then there exists W^* and H^* such that every element of W^* and H^* are in the (inclusive) interval between zero and one, and $A = W^*H^*$. Furthermore, there exist linear transformations from

W to W^* and from H to H^* using a diagonal matrix D: $W^* = WD$ and $H^* = D^{-1}H$. We now show how to apply this to initialize W and H for our pseudo-Boolean matrix factorization.

We begin by using standard non-negative matrix factorization to find W_{nmf} and H_{nmf}. We then rescale W_{nmf} and H_{nmf} using Zhang et al.'s method.

$$\begin{aligned} W_{initial} &= W_{nmf}D = W_{nmf}D_W^{-\frac{1}{2}}D_H^{\frac{1}{2}}, \\ H_{initial} &= D^{-1}H_{nmf} = D_H^{-\frac{1}{2}}D_W^{\frac{1}{2}}H_{nmf}, \\ D_W &= diag([max(w_{nmf\_1}), ..., max(w_{nmf\_k})]), \\ D_H &= diag([max(h_{nmf\_1}), ..., max(h_{nmf\_k})]), \end{aligned} \tag{4.4}$$

where $max(w_{nmf\_i})$ is the maximum value of column i of W, and $max(h_{nmf\_j})$ is the maximum value of column j of H.

It should be noted that if A cannot be factored exactly as $A = W_{nmf}H_{nmf}$, which is common in practice, then the elements of $W_{initial}$ and $H_{initial}$ are not guaranteed to fall within the interval $[0, 1]$. However, the rescaling still pushes the elements of $W_{initial}$ and $H_{initial}$ much closer to the desired interval than the arbitrary solution output by non-negative matrix factorization. This leads to much faster convergence since the variable matrices are properly scaled. It should also be noted that this initialization procedure is computationally expensive because it requires solving an additional matrix factorization problem, but we've found that it leads to qualitatively better solutions with faster convergence than randomly initializing W and H and directly solving for the pseudo-Boolean matrix factorization.

Solving the Pseudo-Boolean Matrix Factorization Optimization Problem (without Visual Feedback)

We've talked about how to select the number of scenarios and how to initialize the dictionary and encoding matrices. In this section, we consider how the PBMF problem can be solved without any visual feedback, i.e., outside of the neural network. In this case, we're interested in decomposing the ground truth A matrix (for the training data) into W and H. In later sections, we discuss how to integrate the PBMF into a neural

network in order to map visual data to scenarios.

Since we ensured that all of the components of PBMF are differentiable, the PBMF optimization problem can be solved with methods based on gradient descent. Specifically, the problem can be solved in two ways. The first way it can be solved is via alternating optimization, where we solve for W and H separately in an alternating fashion until convergence, as is done with traditional non-negative matrix factorization. Alternatively, W and H can be jointly solved for by using standard gradient descent. We've verified that both methods work and produce similar results, but the alternating optimization approach is generally a bit more stable during the optimization and tends to be more robust to sensitivity in the hyperparameters of the optimization algorithm.

Likewise, to enforce the boundary constraints on W and H (i.e., all elements of W and H must fall in the interval between zero and one, inclusive), we can employ either projected gradient descent or explicitly enforce the boundaries using sigmoids: $W = sigmoid(W_{unconstrained})$ and $H = sigmoid(H_{unconstrained})$. Once again, we have observed that, in practice, both approaches produce similar results. However, the sigmoid-based approach enables us to more easily integrate PBMF into a neural network (further details will be discussed in later sections).

4.5.2 ScenarioNet: Identifying and Recognizing Scenarios from Visual Data

To this point, we have assumed that we have perfect knowledge of all ground-truth object data. This assumption enables us to directly solve for the encoding vector when presented with a previously unseen scene instance. To do so, we just need to hold the scenario matrix constant and directly solve for the encoding vector (or matrix if we're recovering the ground truth scenarios for multiple scene instances).

In practice, ground truth object data is never given at test time. Instead, the scenario encoding for a specific scene instance must be recovered entirely from visual data. To perform this recovery, we integrate pseudo-Boolean matrix factorization with

convolutional neural networks. We introduce the **ScenarioNet** architecture, a neural network-based model that jointly learns to *identify* and *recognize* scenarios from real-world visual data while using the predicted scenario encoding as features for the downstream scene classification task. During training, ScenarioNet learns how to map pixels to an estimated scenario encoding matrix $\hat{H}$ while fine-tuning the dictionary W to adapt to the imperfect $\hat{H}$. Furthermore, W also changes based on feedback from the scene classification task in order to improve the discriminating power of the learned scenarios. The key architectural difference between ScenarioNet and traditional convolutional neural networks is the **scenario block** (see Figure 4.2).

The scenario block replaces the final fully-connected classification layers in traditional CNNs. Instead of passing the output of the final convolutional layers of a neural network directly into the classification layers, we pass them through the scenario block. The first set of layers in the scenario block is responsible for recognizing scenarios from visual data. This set of layers consists of a global average pooling (GAP) layer, followed by a fully-connected layer that feeds into a sigmoid layer. This combination of layers makes the network compatible with the class activation mapping technique [2], which enables humans to identify which parts of an image ScenarioNet attends to when deciding whether or not each scenario is present in the image. The output of the sigmoid layer is the scenario encoding vector (or matrix, when multiple images are passed through the net simultaneously): $\hat{H}$. The sigmoid forces each element of the vector to be between zero and one, enabling the net to perform constrained optimization over $\hat{H}$. This vector estimates which scenarios are likely to be in a given image. The output of the scenario encoding layers then feeds into a PBMF loss layer. The PBMF loss layer is the second component of the scenario block, and it is responsible for fine-tuning the scenario dictionary (once again constrained via a sigmoid function) and providing supervisory feedback to the network to adjust the feature extraction portion of the network and the scenario recognition layers. The output of the scenario encoding layers simultaneously feeds into the third component of the scenario block: a sequence of layers equivalent to a multinomial logistic regression model. This final set of layers performs scene classification by using the scenarios as low-dimensional, human-understandable

features in combination with an easy-to-interpret linear classifier.

Training ScenarioNet

Now that we've defined the ScenarioNet model, we discuss how to train the model. During training, ScenarioNet is provided with images paired with ground truth information about the presence (not location) of objects in each image and the class labels of each image. During testing, ScenarioNet only requires images (and not object information). The training process has several stages. First, an initial scenario dictionary is learned without any visual feedback using the ground-truth object presence data. Next, the net is trained to predict the scenario encodings while the dictionary is fine-tuned, but we do not yet incorporate any feedback from the scene classification task. Then, we freeze the network, and we train a softmax classifier for scene classification on top of the frozen network. Finally, we jointly fine-tune the net to perform scenario recognition and scene classification while also continuing to fine-tune the dictionary. Each step of the training process takes between 10 and 20 epochs.

In our experiments, instead of learning the entire network from scratch, we fine-tune a modified ResNet-18 model initially trained on ImageNet [218]. To train our network, we use the ADAM optimizer [252]. We use a learning rate of $1.0e-4$, a weight decay of $1.0e-5$, and a mini-batch size of 16.

Generating Explanations: Interpreting the Output of ScenarioNet

In this section, we focus on how to interpret the output of ScenarioNet and understand how the network arrives at a given prediction.

Given an input image, ScenarioNet outputs several useful pieces of information. First, it provides a probabilistic scene class assignment. Second, it provides a vector of scenario encoding coefficients, which tells us a list of potential scenarios that are likely to be present in an image. Third, the network outputs the dictionary of scenarios, which tells us which objects belong to which scenarios as well as how influential each object is to each of its "parent" scenarios. Fourth, since the network is compatible with the Class Activation Mapping technique, we can extract attention maps that can localize

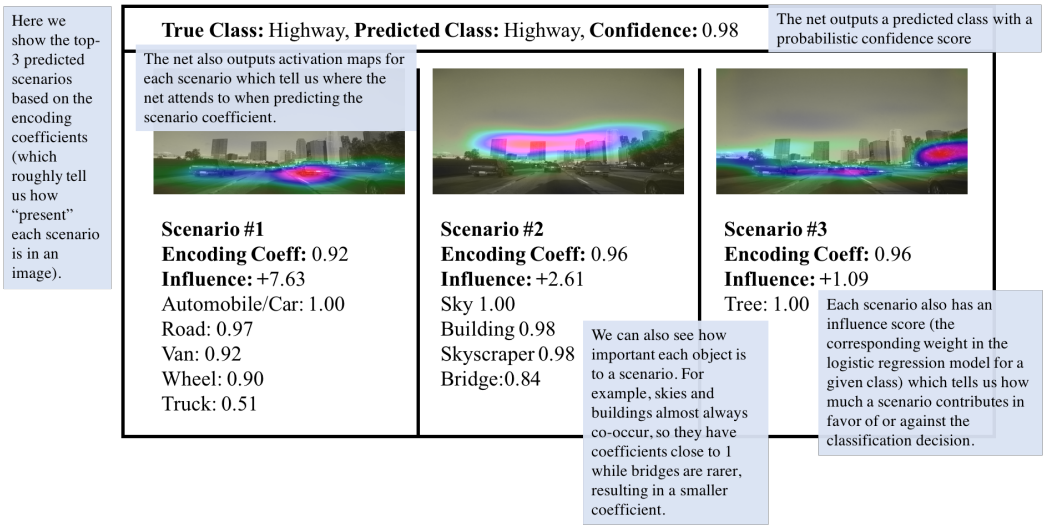


Figure 4.4: We show an example annotated explanation of the output of ScenarioNet applied to an outdoor highway scene instance. We show the scene class prediction and top-3 strongest predicted scenarios with the highest influence scores for the downstream classification task along with the corresponding activation maps.

the parts of each scenario important for predicting the presence of the scenario.

In Figures 4.4, 4.5, 4.6, and 4.7 we show several examples of the explanations output by ScenarioNet. In each case, we show a scene instance decomposed into the strongest detected scenarios that are most influential for the downstream classification, as determined by ScenarioNet. Let us examine the top graphic in Figure 4.5 in more detail. ScenarioNet correctly predicts that the scene category is "dining room" and does so with high confidence. The top-3 scenarios provide evidence to support this conclusion. The first scenario focuses on dining areas; the second scenario focuses on kitchen appliances; and the third scenario focuses on decorative flowers. Furthermore, we can look at the encoding coefficients for each scenario to get an idea of how strongly the net predicts a given scenario. Note that all of the encoding coefficients for the dining room example are close to one because we are specifically examining only the strongest detected scenarios. Recall that ScenarioNet uses scenarios as features for the downstream scene classification task. We define scenario$i$'s *influence score* for a specific class j to be the weight in the multinomial logistic regression model relating scenario i with class j. If the value of the influence score is a large positive number, the scenario provides strong evidence in favor of predicting the specified class. If the influence score

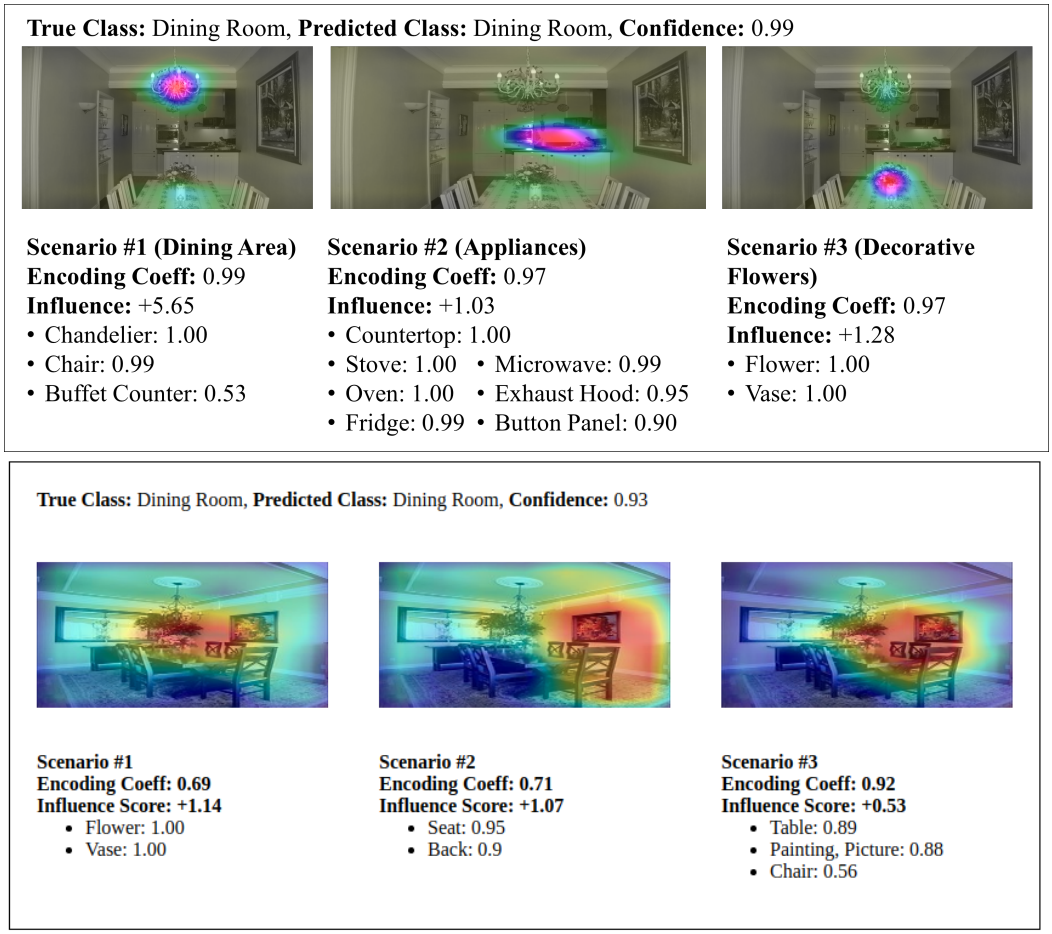


Figure 4.5: We show example explanations output by ScenarioNet when ScenarioNet is applied to two dining room scenes. We show the scene class prediction and the top-3 strongest predicted scenarios with the highest influence scores for the downstream classification task along with the corresponding activation maps. Top image from [3].

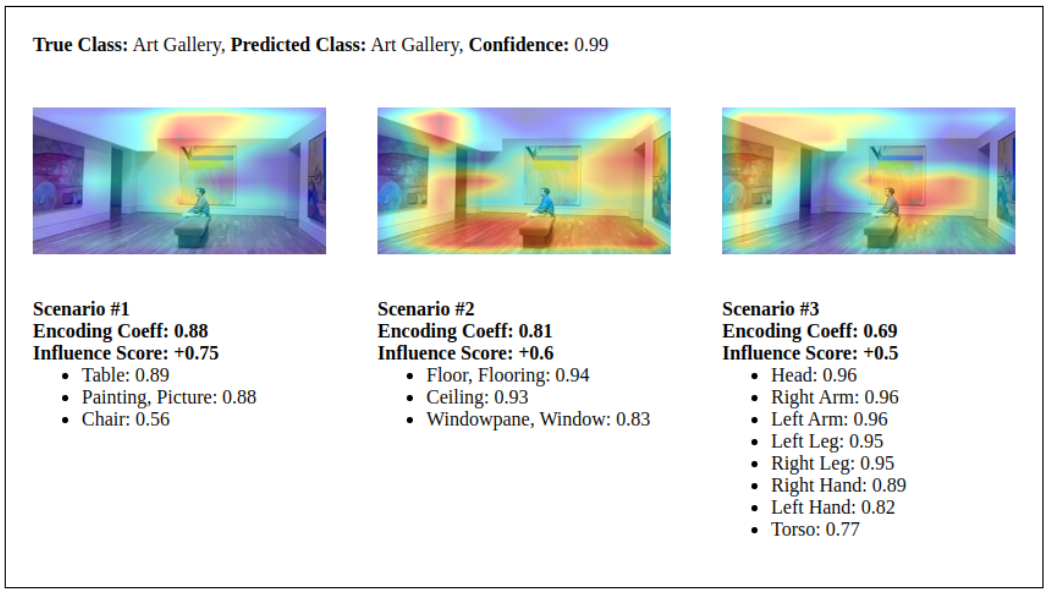


Figure 4.6: We show an example explanation output by ScenarioNet when ScenarioNet is applied to an art gallery scene. We show the scene class prediction and the top-3 strongest predicted scenarios with the highest influence scores for the downstream classification task along with the corresponding activation maps.

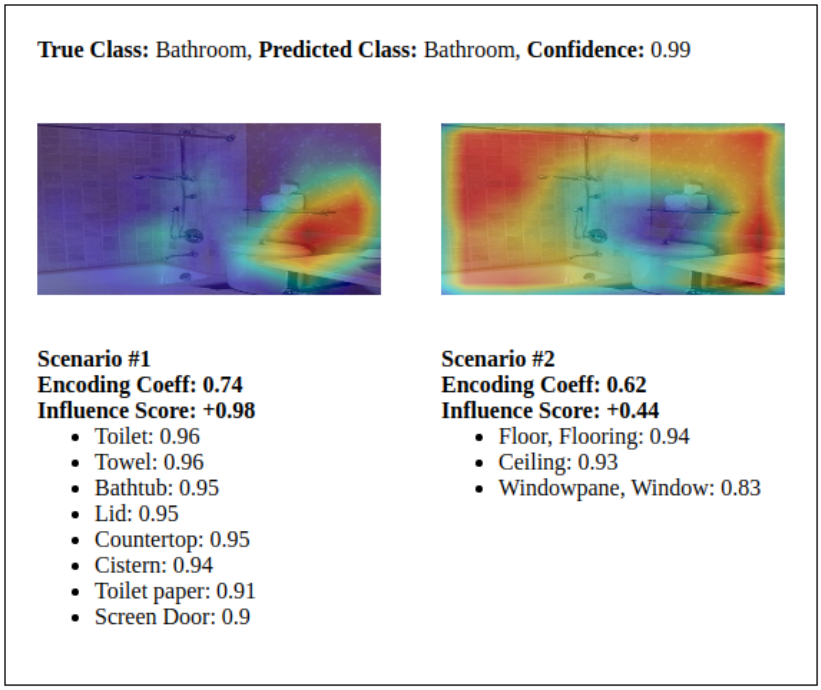


Figure 4.7: We show an example explanation output by ScenarioNet when ScenarioNet is applied to a bathroom scene. We show the scene class prediction and the top-2 strongest predicted scenarios with the highest influence scores for the downstream classification task along with the corresponding activation maps.

is a large negative number, the scenario provides strong evidence against predicting a specific class. For the dining room image example, scenario 1 provides strong support for the "dining room" scene class, whereas scenarios 2 and 3 provide weaker support. We can also examine the influence each object exerts on each scenario. In scenario 1, the "chandelier" and "chair" objects play a larger role in defining the scenario than the "buffet counter" object. Finally, we can verify that the net is truly learning to recognize scenarios and does not rely entirely on contextual clues by examining the scenario attention maps. We see that each of the predicted scenarios is indeed present in the image. Furthermore, the regions that the net attends to do contain objects that partially constitute the scenarios.

4.6 Experimental Results and Analysis

In this section, we show some qualitative and quantitative results demonstrating the utility of pseudo-Boolean matrix factorization and ScenarioNet. We begin by highlighting some example scenarios identified via ScenarioNet. Next, we show an example of how scenarios can be used for comparing images at a semantic level. Then, we explore some cases where ScenarioNet doesn't perform as expected that we qualitatively observed as we were analyzing our model and results. We then evaluate how well the input matrix can be reconstructed from PBMF compared to other matrix factorization methods. Next, we show that ScenarioNet is competitive with existing methods for the scene classification task. Finally, we present some human validation studies to show the interpretability of the scenario representation and plausibility of the explanations output by the network.

4.6.1 Examples of Learned Scenarios

Up to this point, we've discussed scenarios mostly in a conceptual manner while only presenting a few examples of actual scenarios learned via PBMF or ScenarioNet. In this section, we show some examples of scenarios learned using ScenarioNet on the ADE20K dataset (see Figure 4.8). By doing so, we hope to qualitatively highlight the diversity,

Some Example Scenarios:

- {Ceiling, Floor, Wall}
- {Fog Bank, Mountain Pass, River, Snow}
- {Grass, Bench, Path}
- {Bicycle, Manhole, Railing, Sidewalk/Pavement, Stairs}
- {Apparel/Clothes, Bag/Handbag/Pocketbook/Purse, Closet Hangers, Hat, Jacket, Shirt, Shoes, Sweater, Trousers/Pants, Chest of Drawers}
- {Computer Case, Computer, Desk, Keyboard, Monitor, Mouse, Mousepad, Printer, Screen, Speaker}
- {Head, Left Arm, Left Foot, Left Hand, Left Leg, Neck, Person, Right Arm, Right Foot, Right Hand, Right Leg, Torso}
- {Bathtub, Cistern, Countertop, Faucet, Lid, Screen Door, Shower, Sink, Soap Dispenser, Tap, Toilet Paper, Toilet, Towel, Towel Rack}

Figure 4.8: We show several example scenarios learned by ScenarioNet on the ADE20K dataset.

meaningfulness, and ease-of-interpretation of the learned scenario representation.

4.6.2 Content-Based Comparison

Because ScenarioNet bottlenecks through features explicitly grounded in human-understandable concepts, it is useful for quickly providing a high-level overview of the similarities and differences between two scene images. Furthermore, this can be done without making comparisons based on individual objects, which, as we have shown in previous chapters and will show in later this chapter, can be difficult to detect from visual data accurately. Figure 4.9 shows an example of comparing two images based on their semantic content. ScenarioNet correctly identifies that both images are park scenes containing people and grassy areas. It also correctly picks up on the fact that the humans in the top image appear in much more detail (i.e., it expresses humans as a combination of their body parts), whereas the humans in the second image appear further away, and thus, it only detects the concept of "person" without making a note of the individual body parts. Similarly, the network picks up on the fact that the second image contains some man-made outdoor features such as buildings and pathways in contrast to the first image.

Query: How are the Images Similar?
Result:
-Both images are "park" scenes.
-They share the following scenarios: (person) and (grass, path, earth, tree)

Query: How do the Images Differ?
Result:
-Image 1 contains the scenario (head, arms, legs, bag)
-Image 2 contains the scenario (building, lamp, road, sidewalk, window)

Figure 4.9: ScenarioNet can be used to compare two images based on high-level similarities and differences. Image from [3]

4.6.3 Identifying Some Failure Cases of ScenarioNet

We feel that it is important to highlight a few cases that we've observed when analyzing our model and results where ScenarioNet fails to perform as expected.

The first flaw that we've observed is that ScenarioNet sometimes is overly aggressive in creating single-object scenarios. While the number of single-object scenarios increases as the number of scenarios increases (i.e., as k grows larger) as would be expected, it should be noted that even for low values of k, PBMF and ScenarioNet still find a significant number of single-object scenarios. While single-object scenarios are valid with respect to our formulation of PBMF, if the network produces too many single-object scenarios, it defeats the purpose and minimizes the advantages of using scenarios for generating explanations.

The second flaw is the opposite: sometimes PBMF/ScenarioNet combines what would logically be two scenarios into one scenario. For example, on one dataset, PBMF finds the scenario: {bed, coffee table, end table, footrest, lamp, nightstand, pillow, remote, blanket, smoke detector, sofa}. In practice, this scenario should likely be split into: {bed, lamp, nightstand, pillow, blanket} and {coffee table, end table, footrest, lamp, pillow, remote, blanket, sofa} and smoke detector should likely not be assigned to either scenario.

Figure 4.10: We show an example where ScenarioNet doesn't perform as expected. ScenarioNet is highly confident that the {book, bookcase} is present in the above image, but when looking at the parts of the image that the network attends to when making this decision, we see that it exploits contextual clues using objects outside the scenario. It completely ignores focusing on the actual bookcase and instead attends to the desk object.

The third flaw is that ScenarioNet is still far from perfect in its predictions. As such, it makes mistakes in two ways: 1) by predicting scenarios as being present when they are actually absent in the image and 2) by predicting scenarios as being absent when they are actually present in the image.

The fourth flaw is that when ScenarioNet predicts the scenarios present in an image, it sometimes focuses on context clues using objects that exist outside of the scenario. By looking at the attention maps output by the network, we can see that sometimes the network ignores the objects that are in a scenario, and attends to context clues instead. For example, in Figure 4.10, ScenarioNet is highly confident that the {book, bookcase} is present in a scene, but when looking at the parts of the image that the network attends to when making this decision, we see that it completely ignores focusing on the actual bookcase and instead attends to the desk object.

The fifth common flaw that we encountered was that when correctly predicting the presence of a scenario, the network will often attend to limited parts of the scene. To clarify, it might attend to parts of objects (e.g., in Figure 4.7, the network attends to only a portion of the toilet and not the entire toilet). Similarly, if there are multiple objects indicative of a scenario, the network might attend to just one of the objects

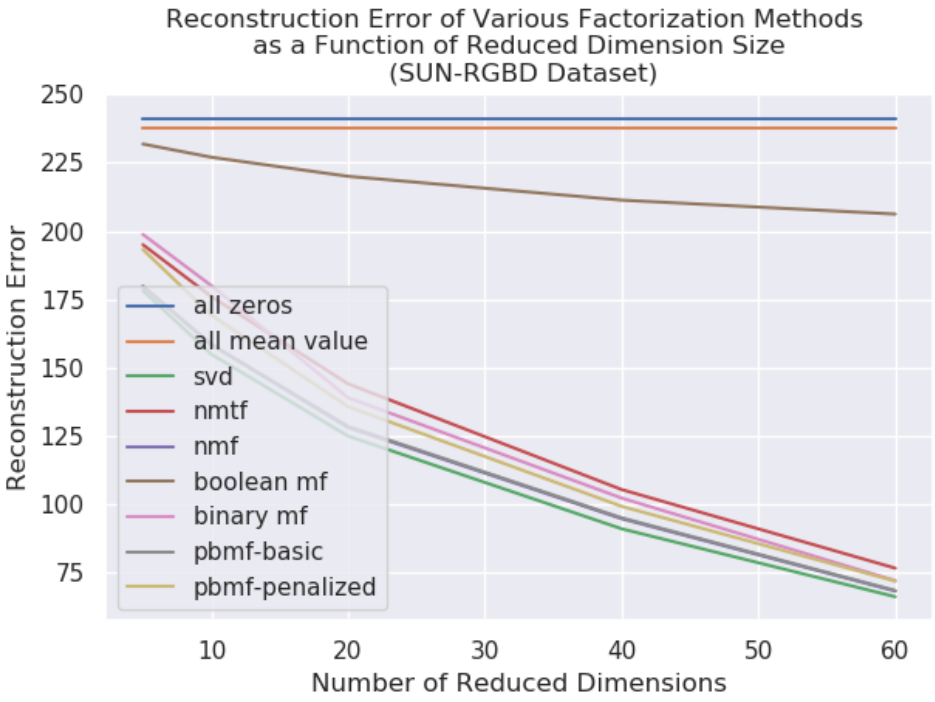


Figure 4.11: We evaluate the reconstruction error between a recovered and ground truth matrix as the dimensionality of the reduced representation is varied using a variety of methods on the SUN-RGBD dataset.

(e.g., in the top graphic of Figure 4.5, when predicting the dining area scenario, it focuses mostly on the chandelier and not the chairs despite both objects being in the scenario and both objects being in the image).

Finally, the sixth flaw encountered is that the network struggles with detecting "stuff" (i.e., non-rigid, ill-defined objects) versus "things" (i.e., rigid, well-defined objects). For example, in Figure 4.6 despite being a better examples of where the network can approximately localize the {floor, ceiling, window} scenario, the network still struggles with localizing the structural elements such as floors and walls. Similarly, we've found that the network struggles with attending to scenarios involving visually tricky objects such as mirrors and windows.

These are a few select flaws that we encountered as we analyzed our model and results; however, we acknowledge that this list is not exhaustive. We hope to improve the model and overcome some of these flaws in future work.

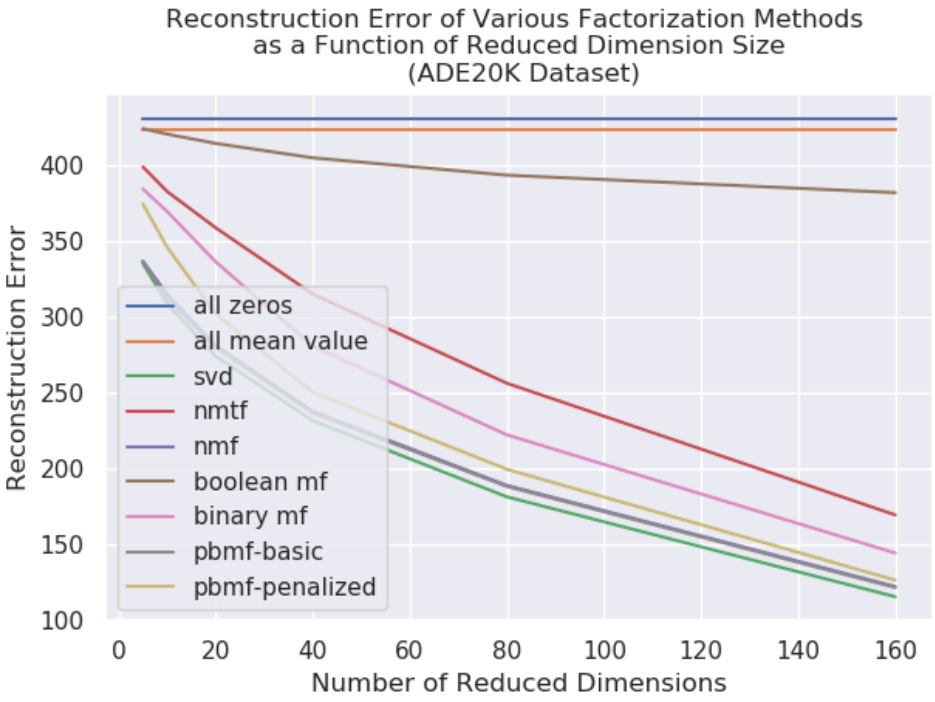


Figure 4.12: We evaluate the reconstruction error between a recovered and ground truth matrix as the dimensionality of the reduced representation is varied using a variety of methods on the ADE20K dataset.

4.6.4 Reconstruction Error of Pseudo-Boolean Matrix Factorization

In our first quantitative experiment, we look at how well pseudo-Boolean matrix factorization can reconstruct the input object-scene matrix from the learned dictionary and encoding matrices. PBMF is a lossy decomposition, meaning the original matrix typically cannot be exactly recovered from the learned dictionary and encoding matrices. The purpose of this experiment is to measure the information loss (with respect to recovering the ground truth object presence matrix) as a result of the decomposition. We assume the input consists of the perfect, ground-truth object presence matrix A, and we reconstruct A using the reconstruction term of the desired matrix factorization optimization problem, e.g., for the basic PBMF formulation (Equation 4.2), this would be $A \approx min(WH, 1 + 0.01WH)$.

We consider three cases of PBMF: PBMF-Basic (Equation 4.2), PBMF-Full (Equation 4.3) with uniform weighting instead of Ω, and PBMF-Full using the proposed

inverse document frequency-based weight matrix. We compare PBMF to several other matrix factorization methods that are either commonly used in practice or also work with binary matrices. These methods include the singular value decomposition (SVD), non-negative matrix tri-factorization (a.k.a. non-negative singular value decomposition, or NNSVD) [250], nonnegative matrix factorization (NMF) [253], greedy Boolean matrix factorization [246], and binary matrix factorization [251]. We also see how PBMF compares to "reconstructed" matrices consisting of all-zeros and all-mean values as baselines.

We initialize the basis and encoding matrices using the method described in Section 4.5.1. Results are plotted in Figure 4.11 for the SUN-RGBD dataset and Figure 4.12 for the ADE20K dataset. On both of these datasets, the reconstruction resulting from PBMF-Basic exhibits low reconstruction error. In general, it only performs slightly less well than the much less constrained SVD, and often performs very slightly better than NMF (i.e., the lines essentially overlap in the Figure).

Reconstruction error is not the only metric we should consider. If the model were to only focus on minimizing reconstruction error, it would give too much importance to common objects and often leads to learning basis vectors that lack diversity. Instead, we need to add orthogonality constraints and reweigh the importance of all of the object classes (so rare objects gain more importance during when reconstructing the input matrix, and common objects lose some importance when reconstructing the input matrix). Figures 4.11 and 4.12 demonstrate how adding orthogonality constraints and reweighing rare classes impacts the reconstruction error. Namely, as constraints and penalties are added to the PBMF formulation, the model sacrifices some reconstruction accuracy but remains competitive with other, similar matrix factorization approaches like greedy Boolean matrix factorization and binary matrix factorization. In practice, we found that dictionaries learned using the PBMF-Full formulation are better suited for higher-level tasks such as scene classification and image retrieval than the dictionaries learned via PBMF.

4.6.5 Comparison to a Model that Bottlenecks Through Object Predictions

We now conduct a set of experiments to justify why bottlenecking through a scenario recognition layer should be preferred to directly bottlenecking through the object recognition layer (e.g., as was done in the previous chapter). We train a multi-object recognition model using ResNet-18 as our backbone network, and the predictions from the object recognition model are fed into a linear multinomial logistic regression model to perform scene classification. The multi-object recognition predicts all 118 objects for the SUN-RGBD dataset and all 339 objects from the ADE20K dataset. We consider 15 scene classes for the SUN-RGBD dataset and 31 scene classes for the ADE20K dataset (see Section 4.4 for more details). We measure the mean average precision for evaluating the multi-object recognition task and the accuracy for evaluating the scene classification task. It should be noted that there is not a true "ground truth" scenario encoding for the test data. In order to approximate a "ground truth" scenario encoding matrix for the test data, we learn the dictionary on the training data using ScenarioNet and solve the PBMF encoding on the test data, holding the dictionary constant, but this approximation is not perfect because it does not account for any visual feedback on the test data, and the trade-off parameters for the PBMF formulation need to be adapted for differences in size (i.e., number of elements) between the input matrices $A_{training}$ and A_{test}. We set the number of scenarios based on sequential model-based optimization (see Section 4.5.1 for more details). We find that using 11 scenarios for the SUN-RGBD dataset and 36 for ADE20K dataset results in the best trade-off between scenario recognition and scene classification. However, this means there is approximately one scenario associated with every scene class, which is not incredibly useful to consider in terms of the generated explanations because class-correlated scenarios do not allow for highlighting the differences in semantic content for instances of the same class; thus, the explanation is too coarse. Thus, we also constrain the number of scenarios to be at least one and a half times the number of scene classes and re-run the sequential model-based optimization to find a more realistic number of scenarios

	SUN-RGBD		*ADE20K*	
Concept Type	**Concept Recog. mAP**	**Scene Class. Acc.**	**Concept Recog. mAP**	**Scene Class. Acc.**
Objects (SUN: 118, ADE: 339)	0.233	0.551	0.333	0.724
Scenarios (SUN: 11, ADE: 36)	0.511	0.517	0.642	0.802
Scenarios (SUN: 23, ADE: 67)	0.488	0.555	0.537	0.760

Table 4.1: We evaluate how easily objects and scenarios can be recognized from visual data. We also evaluate the predictive power of the object- and scenario-based features are for the downstream scene classification task.

for practical applications. With this constraint, we find that using 23 scenarios for the SUN-RGBD dataset and 67 for ADE20K dataset results in the best trade-off between scenario recognition and scene classification. Results appear in Table 4.1.

As in the previous chapter, we see that the average object cannot be recognized with high accuracy from visual data. In contrast, the average scenario can be recognized with much higher accuracy; albeit, there is still much room for improvement. Thus, decisions and predictions made based on scenarios can be trusted much more than those based on object predictions. Similarly, the scenarios are much lower-dimensional than the initial set of objects, and the scenario encoding for a scene tends to be sparse. Subsequently, compared to the explanations based on the initial object set, the scenario-based explanations are much simpler for humans to interpret. The scenario-based model also generally matches or outperforms the object-based model in terms of accuracy on the downstream scene classification task. This improved accuracy is likely because scenarios are explicitly designed to be discriminative, it is harder for the scenario-based model to overfit since it is so low-dimensional, and scenarios capture contextual information, which is useful for scene classification.

4.6.6 Comparisons to Other Methods for Scene Classification

In the next set of experiments, we frame the predictive accuracy of ScenarioNet on the scene classification task in the context of other models. In particular, we compare ScenarioNet to traditional convolutional neural network architectures, other object-based features, and visual attribute-based methods. For experiments not involving a CNN, we train a logistic regression model on top of the given features. We report our experimental results in Table 4.2.

Method	SUN-RGBD	ADE20K
Baseline CNNs		
AlexNet	0.469	0.786
GoogLeNet	0.541	0.796
VGG-16	0.531	0.809
ResNet-18	0.548	0.802
ResNet-18 + Dimensionality Reduction	0.490	0.776
Object-Based Representations		
Object Bank + PCA	0.296	0.511
Object Detection (YOLOv2)	0.399	0.639
ResNet-18-Objects	0.551	0.724
Visual Attribute-Based Representations		
SUN-Attribute	0.429	0.705
Classemes	0.309	0.581
Meta-Classes	0.360	0.635
Proposed Model		
ScenarioNet	0.555	0.760

Table 4.2: We evaluate scene classification accuracy on a number of different features for the SUN-RGBD and ADE20K datasets.

Comparison to Baseline CNNs

In this set of experiments, we compare ScenarioNet to the AlexNet [238], GoogLeNet [254], VGG-16 [255], and ResNet-18 [239] convolutional neural networks. AlexNet, GoogLeNet, and VGG-16 are pre-trained on the Places dataset [256] and fine-tuned on the SUN-RGBD and ADE20K datasets. ResNet-18 is pre-trained on the ImageNet dataset [218] and fine-tuned on the SUN-RGBD and ADE20K datasets. Recall that the experimental results in Table 4.2. ScenarioNet outperforms all of the other models on the SUN-RGBD dataset in terms of accuracy. ScenarioNet underperforms the other models on the ADE20K data by about 3-4%, but ScenarioNet has several advantages over these models. In particular, ScenariNet can generate human-understandable explanations for all predictions, and the scenario encodings are useful for tasks beyond scene classification, e.g., semantic comparison of two images.

The baseline convolutional neural networks operate over feature spaces with dimensionality in the hundreds and thousands, whereas ScenarioNet generates an encoding space that is much lower dimensionality (23 scenarios for the SUN-RGBD dataset and 67 scenarios for the ADE20K dataset). For more fair comparisons, we modify the

ResNet-18 network such that the output of the final feature layer is the same dimensionality as the scenario-based representation. In this case, we see sharp decreases (of about 3-6%) of the predictive accuracy of the ResNet-18 network on both the SUN-RGBD and ADE20K datasets. In this case, ScenarioNet outperforms the compressed ResNet-18 network by about 6-7% on the SUN-RGBD dataset and decreases the gap in accuracy to only 1-2% on the ADE20K dataset.

Comparison to Other Object-Based Representations

In Section 4.6.5, we compared ScenarioNet to a neural network that bottlenecks through an object recognition layer. In this set of experiments, we look at several alternative object-based representations. We consider a logistic regression model that uses Object Bank features [236] compressed to 8000 dimensions using PCA and a logistic regression model that uses the object probability scores output by a trained YOLOv2 object detector [257] as features. The YOLOv2 detector is trained to detect all objects in the SUN-RGBD and ADE20K datasets that appear in at least one percent of the data (55 objects for the SUN-RGBD dataset and 193 objects for the ADE20K dataset). For context regarding how well the object detector performs: on the multi-object recognition task, YOLOv2 results in a mean average precision of 0.442 on the SUN-RGBD dataset and a mean average precision of 0.379 on the ADE20K dataset.

ScenarioNet outperforms these two other object-based representations for the scene classification task in terms of predictive accuracy. We hypothesize that this increase in accuracy is a result of the scenario representation's ability to better capture global scene information compared to individual object-based representations. This increase in accuracy is also likely due in part to the fact that ScenarioNet is trained to recognize objects and scenes jointly, and thus, the scenario representation receives feedback from the scene classification task.

Comparison to Visual Attribute-Based Representations

Our final set of experiments for the scene classification task involves comparing ScenarioNet to visual attributes. As introduced in Chapter 2, visual attributes are high-level

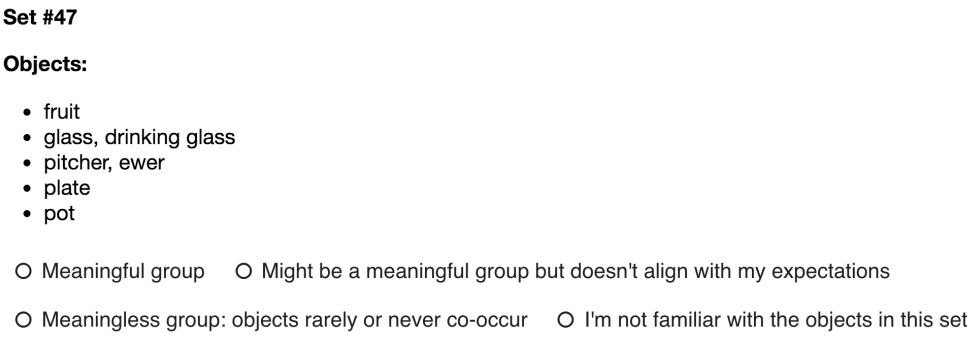


Figure 4.13: We show an example question for evaluating whether a human participant believes a scenario is meaningful.

semantic properties shared between multiple classes [18]. We feel these comparisons are important because attributes are a common alternative semantically-grounded representation. We consider three attribute-based representations: SUN Attributes [168] (attributes designed for scene classification), Classemes [258], and Meta-Classes [259]. ScenarioNet outperforms all of these methods in terms of accuracy on the scene classification task.

4.6.7 Evaluating ScenarioNet's Explanations via Human Subject Experiments

In our final set of experiments for this chapter, we want to verify that humans find the learned scenarios meaningfully group objects, and we also want to verify that the explanations output by ScenarioNet are plausible (i.e., we want to ensure that the explanations output by ScenarioNet are meaningful to humans). It should be noted that we use a small sample size in this set of experiments, but in Chapter 6, we will repeat similar experiments on a similar dataset with much larger sample sizes.

We start by asking the question: do the object groupings make sense? In the first experiment, we want to evaluate if the scenarios learned using PBMF (i.e., with no visual feedback and outside of the ScenarioNet) form logical groups of objects. Using Amazon's Mechanical Turk (AMT) service, we collect data from five English-speaking annotators with no knowledge of the dataset or method. We generate 100 scenarios on the ADE20K using only PBMF and the ground truth object data, and we discard

scenarios containing only one object. The human participants are shown one scenario at a time where each scenario is represented as a set of objects. For each scenario, the participants are asked to determine how likely the majority of the objects in the set would be seen together based on their prior knowledge. Specifically, they are asked if 1) the objects form a meaningful group, 2) the objects might form a meaningful group, but the group does not align with the participant's expectations, 3) the objects form a meaningless group, or 4) the group consists of objects that the participant is not familiar with. They were further asked to assign a label to the set of objects, if possible. This additional labeling task was to ensure that the participants were carefully studying the scenarios and to aid in post-hoc analysis of the experimental results. An example question is shown in Figure 4.13. After taking the modal response for each question, **respondents found 75.0% of the scenarios were meaningful, 15.3% might be meaningful, and 9.7% were meaningless**. Upon examining the meaningless scenarios, we found that these were often considered "meaningless" for several reasons. Sometimes, a scenario would cluster two sets of objects that shouldn't go together; sometimes, the scenarios captured context between objects that was very specific to the ADE20K dataset; and sometimes, the dataset uses obscure, verys specific, or foreign (i.e., British English vs. American English) terminology for specific objects that make the scenarios appear meaningless. For example, the felt part of a pool table is referred to as a "bed", and without knowing this information, people might be confused about why the "bed" object is clustered with "pool cue" and "pool ball".

We then repeated this experiment using 75 scenarios output by ScenarioNet (discarding all scenarios with just one object). Unlike in the previous experiment, the learning of these scenarios is influenced by visual feedback. Once again using Amazon's Mechanical Turk (AMT) service, we collect data from five English-speaking annotators with no knowledge of the dataset or method. These annotators are not the same as the ones used in the previous experiments. The human participants are shown one scenario at a time where each scenario is represented as a set of objects, and for each scenario, each participant was once again asked to rate the meaningfulness of the object grouping was and to assign a label to the grouping, if possible. After taking the modal response

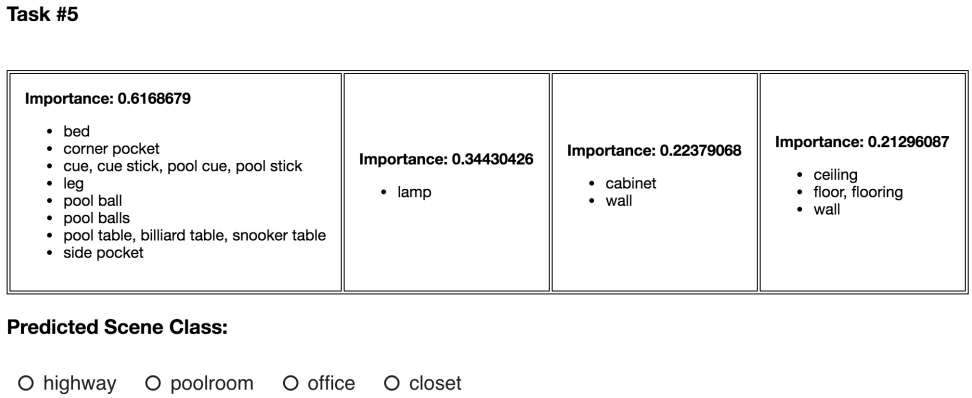


Figure 4.14: We show an example question for evaluating whether a human participant can predict the scene class of a given scene instance based only on detected scenarios that the network deems important for making the true classification decision.

for each question, we see similar results as before. **Respondents found 80.6% of the scenarios were meaningful, 8.3% might be meaningful, 9.7% were meaningless, and 1.4% contained objects the participants were not familiar with**.

In the final experiment for this section, we evaluated whether humans could accurately identify the category of a scene when presented with only the most influential scenarios and their influence score. Recall that the influence score consists of the predicted scenario probability output by the neural network multiplied by the corresponding weight in the logistic regression model. We identified all scenarios detected by the network for a given image. We then identified which of these scenarios were positive indicators of the true class as determined by ScenarioNet (i.e., which scenarios provided positive evidence of the true class). For thirty random scenes that were correctly classified by ScenarioNet, given only the "positive evidence" scenarios, humans were then asked to predict the target class from a choice of one true class and four false classes. The humans never saw any of the images. An example question is shown in Figure 4.14. After taking the modal response for each question, **86.7% of the scene classifications made by humans were correct**, suggesting that the model output plausible explanations. It should be noted that the network does make mistakes about which scenarios are present in a scene instance and might be able to generate plausible-sounding explanations that are not 100% percent accurate. Thus, we conclude that the

explanations output by ScenarioNet is often plausible, but not necessarily totally accurate. That being said, because scenarios are recognized with much higher accuracy than individual objects, the scenario-based explanations should be more trustworthy than individual object-based representations. Thus, ScenarioNet is a good first step towards creating neural networks that are capable of generating grounded explanations.

Chapter 5

Deriving New Visual Concepts from Auxiliary Data Sources: Jointly Learning Topic Models and Visual Classifiers

Note: This chapter is based on work originally published as "Exploiting Visual and Report-Based Information for Chest X-Ray Analysis by Jointly Learning Visual Classifiers and Topic Models" [4].

5.1 Introduction

In the previous two chapters, we introduced two approaches for grounding the decisions made by convolutional neural networks to human-understandable visual concepts. However, these methods made many assumptions that do not always hold in practice. First, these two approaches assumed that for every training image, a human hand-labeled the presence or absence of all of the relevant visual concepts from a human-curated prespecified set of visual concepts. Often, acquiring such fine-grained annotations is a costly and time-consuming procedure, and thus, it is often not feasible to collect images paired with visual concept information. Second, we tested our methods on the relatively simple task of scene classification with relatively clean data. Most images in the dataset are very prototypical views of the given scene, and most images are labeled with care, and thus, the annotations are relatively accurate. In this chapter, we look at a harder problem that is more representative of the types of problems encountered in real-world applications. Furthermore, we consider an application where clean annotations for visual concepts are not directly provided with the image data. Instead, we focus on extending the ScenarioNet model introduced in the previous chapter to extract meaningful and grounded representations from noisier auxiliary data sources.

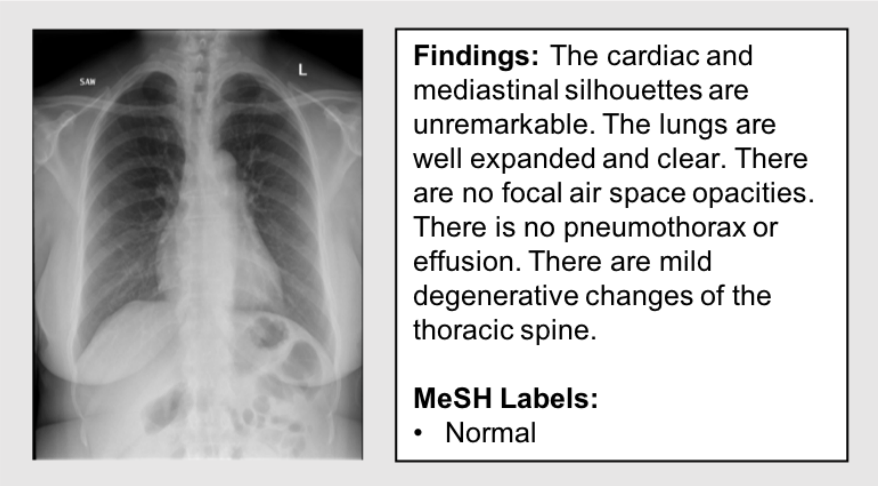


Figure 5.1: We show an example ground-truth medical report for the chest x-ray analysis problem. Image from [4].

Specifically, the data we consider in this chapter consists of images paired with natural language text that describes the content of the image. Such data is available from many sources including encyclopedia entries paired with images, images from news stories paired with captions, and medical images paired with medical reports. In this chapter, our motivating application consists of making computer-aided diagnoses of chest-related diseases based on chest x-ray data paired with medical reports written by expert doctors. In particular, these medical reports highlight the important characteristics of the chest x-rays that explain the specific diagnoses made by a doctor.

We now describe the basic problem addressed in this chapter and our proposed approach. The data consists of frontal-view x-rays, a set of corresponding natural language findings, and one or more Medical Subject Headings (MeSH) labels that describe the diagnosis using a controlled vocabulary. An example of a medical report provided by the data set is shown in Figure 5.1.

Our goal is to predict the illnesses present in each chest x-ray image. The diagnoses we consider consist of "normal/no findings", "atelectasis", "cardiomegaly", "effusion", and "emphysema", and one or more of these illnesses may be present in the chest x-ray. We propose an extension of the ScenarioNet architecture introduced in the previous chapter. Instead of bottlenecking through the scenario recognition layer, our proposed convolutional neural network bottlenecks through a topic modeling layer that learns to

cluster key terms from the findings into meaningful groups. For example, the network might learn to cluster "lungs", "clear", and "expanded" to form a topic. See [260] for an overview of topic modeling.

During training, our proposed convolutional neural network architecture simultaneously 1) constructs a topic model, 2) predicts the presence or absence of each topic for a given image based on learned visual features, and 3) uses an image's predicted topic encoding as features for predicting one or more diagnoses. At test time, using only images as input, the network predicts which topics are present in each image, and using these topic predictions, predicts one or more diagnoses. Since the neural network jointly learns both the topic model and the classifier for the downstream diagnosis task, it is useful for investigating which semantic concepts the CNN might be exploiting during the decision-making process for the diagnosis task. Likewise, since the neural network predicts topics based on expert-annotated reports and uses the learned topics to make diagnoses, the net is forced to "think" like an expert. It should be noted that the original goal of this work was to improve automated diagnosis and inject expert knowledge into the neural network by grounding it to information typically accessible only through medical reports. The explainability aspect of our approach was a secondary focus. As such, throughout this chapter, we make some sacrifices in terms of explainability in order to focus on improving the predictive power of the model on the diagnosis task.

5.2 Related Work

Most related work dealing with interpretable models for visual recognition tasks has been covered in previous chapters. In this section, we will specifically highlight some existing work demonstrating how deep neural networks can be used for chest x-ray analysis and other work combining natural language processing with biomedical image and text analysis.

Chest x-rays are often used for diagnosing certain life-threatening diseases such as pneumonia. However, manually inspecting chest x-rays is a time-consuming process and requires significant time and effort by expert radiologists. In recent years, computer

vision and machine learning have been shown to be necessary for developing powerful support tools for (semi-)automated chest x-ray analysis. Machine learning-based decision support systems allow radiologists to make faster diagnoses, and as a result, radiologists can spend more time focusing on challenging cases. Deep learning, in particular, has significantly pushed the field forward. Most recent work in this area has focused on learning how to map from images to multi-label diagnoses using deep neural networks (e.g. [261–263]). However, these models are generally treated as black boxes. In models where interpretability is important, typically, an attention mechanism is used to explain the model's decision-making process by showing the parts of the image that the network attends to when predicting a specific diagnosis. However, these attention maps are insufficient because they do not explain why the attended areas are important. Similarly, neural networks learn everything from raw visual data and are not designed to incorporate expert knowledge. Thus, they do not exploit the rich medical knowledge that humans have developed over hundreds of years and must learn to make diagnoses virtually entirely from scratch.

Some of this knowledge comes in the form of radiologist-dictated reports that justify a doctor's diagnostic impressions. It should be noted that natural language processing (NLP) has been used extensively to analyze biomedical text (e.g. [264–267]). However, typically the information extracted from this natural language data has not been combined with visual data. Recently, this has started to change. Several new methods have been proposed that integrate information extracted from medical reports with deep neural networks in order to improve automated biomedical image analysis (e.g. [268–270]). Furthermore, another currently popular research topic in the medical analysis community involves generating natural language reports directly from images (e.g. [271–279]).

5.3 Problem

We briefly review the problem that we are addressing. Our data consists of frontal-view x-rays, a set of corresponding natural language findings, and one or more Medical Subject Headings (MeSH) labels that describe the diagnosis using a controlled vocabulary

(see Figure 5.1 for an example data instance). During training, we have access to all three of the aforementioned pieces of information (images, reports, and labels). At test time, we only have access to the chest x-ray images, and we aim to predict which illnesses are present in a given chest x-ray image using only visual data as input. We consider five common chest-related illnesses: "normal/no findings", "atelectasis", "cardiomegaly", "effusion", and "emphysema". Multiple illnesses may appear present in the same chest x-ray; thus, we need to train a model to perform multi-label classification.

5.4 Data

We work with two datasets. For most of our experiments, we work with the OpenI dataset [280]. From the OpenI dataset, we use 3,821 frontal-view chest x-ray images along with their corresponding "FINDINGS" annotations. Since the OpenI dataset is relatively small, and deep neural networks often require large amounts of data to learn good features, we pre-train the feature extraction portion of neural network on 86,524 training images from the ChestX-Ray14 dataset [261] using all 14 diagnosis labels. We cannot use the ChestX-Ray14 dataset for evaluating our proposed method because it contains only images and not medical reports. Instead, we subsequently finetune the feature extraction layers, learn the top modeling layers, and learn the final classification layers on the OpenI dataset. We do not consider all of the labels in the OpenI dataset. We restrict our experiments to the five diagnosis labels shared between the OpenI and Chest X-Ray14 datasets that appear most often. These labels include "normal/no findings", "atelecstasis", "cardiomegaly", "effusion", and "emphysema". We split the data based on 5-fold cross validation using splits based on individual patients, not individual images.

5.5 Methodology

We now present an overview of our method including how to pre-train the feature extraction neural network, how to extract key terms from natural language text, how to learn topic models using matrix factorization, and how to integrate topic modeling

with convolutional neural networks.

5.5.1 Pre-Training the Feature Extraction Neural Network

In order to evaluate our method, we need both frontal-view chest x-ray images and medical reports describing the contents of these images. There are not many publicly available datasets that include both of these components. The OpenI dataset [280] is the most commonly used dataset that includes both image and text data for chest x-ray analysis. However, the OpenI dataset is not without its issues. Large amounts of data are needed when training deep neural networks, and the subset of OpenI that we use in our experiments contains very limited data: only includes 3,821 images. A popular approach to overcoming the limited data problem is by using transfer learning [281]. In the context of deep neural networks, transfer learning is often done by training a neural network on a huge source of data similar to the dataset of interest, and then, finetuning the feature extraction portion of the network and relearning the classification layers on the smaller dataset of interest. We employ this practice in this chapter by pre-training the feature extraction layers of a ResNet-22 model [239] on the larger ChestX-Ray14 dataset [261] and subsequently fine-tuning the entire network on the OpenI dataset. We use a weighted variant of the multi-label cross-entropy as our loss function for training the initial network:

$$\mathcal{L}_c = \sum_{l \in L} \left(\frac{1}{|P_l|} \sum_{y_{li} \in P_l} -y_{li} \log \hat{y}_{li} + \frac{1}{|N_l|} \sum_{y_{lj} \in N_l} -(1 - y_{lj}) \log \left(1 - \hat{y}_{lj}\right) \right) \tag{5.1}$$

L is the set of all labels. l is an individual label. P_l and N_l are the sets of positive examples and negative examples in a batch for label l, respectively. $|X|$ denotes the number of examples in a batch for set X. $y_{li} \in \{0, 1\}$ is the label of ith positive instance of label l, and $y_{lj} \in \{0, 1\}$ represents the jth negative instance. $\hat{y}_{li} \in [0, 1]$ and $\hat{y}_{lj} \in [0, 1]$ are the predicted scores of instances i and j.

For many illnesses, the population of people with the illness is often much lower than the population of people without the illness. As a result, when training machine learning models for automated medical diagnosis tasks, it is crucial to consider that the data is frequently imbalanced in terms of the class label; i.e., there will often be many

more negative training examples for a given illness than positive training examples. We make some adjustments to our training procedure to address the issue of imbalanced multi-label classification. In particular, in our loss function, we weigh the entropies for the positive instances by $\frac{1}{|P_l|}$, and we weigh the entropies for the negative instances by $\frac{1}{|N_l|}$. Furthermore, we use stratified sampling with replacement to construct each mini-batch, whereas the standard procedure for constructing mini-batches entails random sampling without replacement. Specifically, for each mini-batch, we select one label, and subsequently, we randomly select an even number of positive examples and negative examples for this label. This process is repeated by iterating through all of the labels. The rationale behind this sampling scheme is that it allows the net to see instances with rare labels more frequently than the traditional sampling scheme.

To pre-train our feature extracting neural network, we 2,703 total mini-batches per epoch where each batch contains 32 training instances. We train the network for 50 epochs. We also carefully apply data augmentation via random cropping and small perturbations to the brightness, contrast, and saturation of the cropped images. Images are of size 512-by-512 pixels.

5.5.2 Extracting Key Terms from Natural Language Text

Our goal is to use information that has been extracted from natural language text reports to improve the predictive performance and interpretability of deep neural networks. Specifically, we are most curious if grounding the decision-making process of a deep learning model to semantic information present in both images and text (i.e., forcing the network to "think" more like an expert when evaluating medical images) will improve the network's capability for discovering more discriminative features. To accomplish this objective, we propose extending the ScenarioNet architecture that was introduced in the previous chapter. Specifically, instead of learning how to group a base set of visual concepts into scenarios, we use a ScenarioNet-like architecture to learn how to group key terms present in the medical reports into topics, each of which should express some meaningful semantic concept. We propose a novel deep neural network architecture that learns to predict one or more diagnoses as it concurrently builds

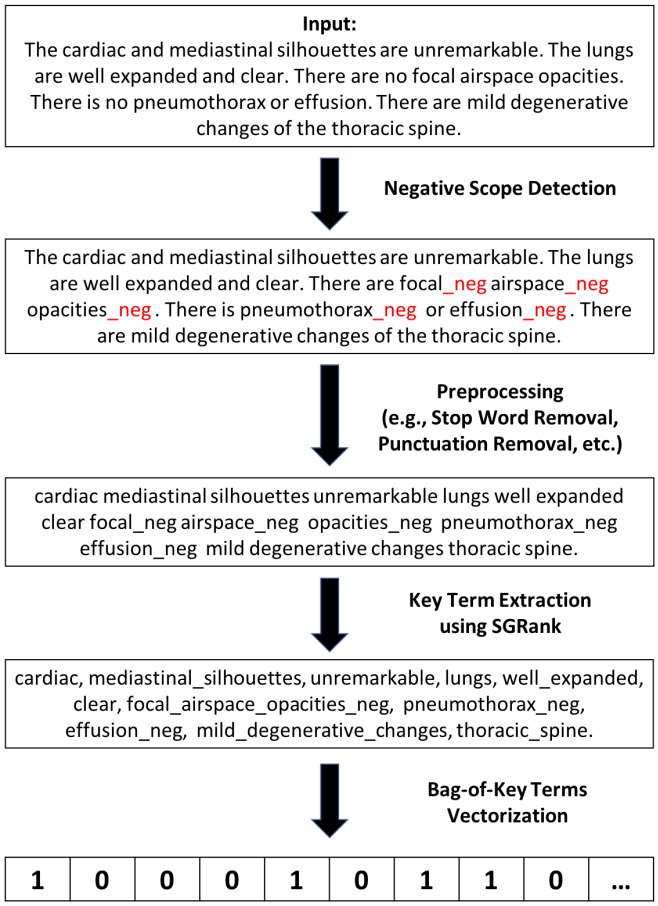


Figure 5.2: We show the pipeline for extracting a set of key terms from a natural language medical report. The pipeline consists of four stages: negative scope detection, basic pre-processing, key term extraction using SGRank, and vectorizing the key terms in a bag-of-key terms representation.

a topic model consisting of 1) a dictionary which clusters related key terms together and 2) an encoding matrix that decomposes each input instance into a set of topics. In this section, we explain how to extract the initial set of key terms from a collection of natural language medical reports.

We present the pipeline for extracting a set of key terms from natural language medical reports in Figure 5.2. The first stage of the pipeline performs simple rule-based negative scope detection for each report in the training set. Negative scope detection involves determining which phrases are affected by negation (i.e., we want to determine which words are modified by words like “no” and “not”). For example, the phrase “no pleural effusion” is parsed as “pleural_neg effusion_neg”, meaning that pleural effusion

cannot be seen in the corresponding medical image. The second stage of the key term extraction pipeline is applying basic natural language pre-processing to each report in the training set in order to remove stop words (i.e., common words like "the"), remove punctuation, remove capitalization, and a number of other common operations to remove unnecessary information and structure from the text. The third stage of the key term extraction pipeline involves using SGRank [282] to identify important phrases from all documents in the training set (e.g., "pleural effusion", "focal airspace disease", and "pneumothorax"). In the final stage of the key term extraction pipeline, a *bag-of-key terms (BOKT)* representation is extracted from all documents, and the dictionary for the bag-of-key terms is pruned so that terms must appear at least ten times in the training dataset. It should be noted that in our experiments, we use k-fold cross validation, so when we refer a "training set", we mean a training fold. In total, the key term extraction process results in 600-700 key terms per fold.

5.5.3 Learning Topic Models using Matrix Factorization

The naïve way to incorporate text information into a visual DNN is to treat each key term as its own visual concept and directly predict the bag-of-key terms for each image. However, this approach is flawed. First, the key term extraction process is far from perfect. For example, "pleural effusion", "pleural effusions", and "effusion" are all extracted as different key terms, but each of these terms is semantically equivalent. Thus, in the bag-of-key terms representation, they are incorrectly treated as independent visual concepts. As such, it is too tough to learn a visual classifier for each individual key term since there is too much noise in the training labels. Furthermore, synonyms and abbreviations pose the same problem, e.g. "copd" and "chronic obstructive pulmonary disease". The second problem with directly predicting the bag-of-key terms is that many times individual terms are not useful by themselves. Instead, many key terms become useful when paired with other key terms; i.e., context between key terms is necessary. For example, "cardiac silhouette" by itself offers no useful information for the downstream diagnosis task, but when it is paired with the key term"oversized" it becomes strongly indicative of cardiomegaly (a condition characterized by the abnormal

enlargement of the heart). Finally, the bag-of-key terms representation includes a great deal of redundancy as a result of co-occuring key terms. For example, "normal cardiac silhouette" and "normal mediastinum size" frequently appear together, and thus, by knowing one of these concepts is present in an image strongly suggests that the other is also present. As discussed in previous chapters, to maximize the interpretability of the model, we desire grounded representations that are as succinct as possible. The bag-of-key terms representation does not satisfy this property. The last issue with directly predicting the bag-of-key terms representation is that for many of the key terms, there aren't enough training examples to learn a key term extraction model that can be trusted with any degree of confidence. To overcome these problems, we can exploit context between key terms. By doing so, we can identify a lower-dimensional set of topics and use a convolutional neural network to predict the presence of each topic in a given x-ray image.

As in the previous chapter, we will use pseudo-Boolean matrix factorization (PBMF) to discover how to group the key terms into topics. Suppose we're given a binary input matrix A where each row represents a key term and each column represents a medical report. A_{ij} is 1 if key term i is present in document j and 0, otherwise. The A matrix can then be decomposed into two other matrices: the dictionary matrix W, which clusters key terms into topics, and the encoding matrix H, which decomposes reports into the union of topics. In this chapter, we use the following formulation of PBMF where m is the number of key terms, k is the number of documents, and n is the number of medical reports:

$$\begin{aligned}&\min_{W,H} ||\Omega \bullet (A - \min(WH, 1 + 0.01WH))||_F^2 + \alpha_1||W||_1 \\ &+ \alpha_2||H||_1 + \alpha_3||W^\intercal W - diag(WW)||_F^2 \; s.t. \; W \in [0,1]^{m \times k}, H \in [0,1]^{k \times n} \end{aligned} \tag{5.2}$$

The first term penalizes for bad reconstructions of the key term-report matrix. The second and third terms encourage sparsity in the encoding and dictionary matrices. The fourth term encourages orthogonality between topics, forcing similar topics to merge. The αs are hyperparameters designed to balance the different penalty functions. The model must pay attention to rare key terms because, in medicine, one often cares more

about rare occurrences, which tend to be more indicative of medical problems. To force the net to pay more attention to the rare key terms, we adjust the reconstruction error using the inverse document frequency (idf): $\Omega_{i,j} = max(idf_i \bullet A_{i,j}, 0.25), idf_{term} = (1 + log\left(\frac{N_{all}}{N_{term}}\right))$ where N_{all} is the total number of training instances and N_{term} is the number of training instances containing a specific term.

In Figure 5.6, we show examples of the topics learned by our approach.

5.5.4 Incorporating Topic Modeling into Convolutional Neural Networks

The PBMF optimization problem (Equation 5.2) can be reformulated as a loss function $\mathcal{L}_t$. The goal will be to use a convolutional neural network to estimate the encoding matrix $\hat{H}$ for one or more scenes from some set of input images X. Furthermore, the dictionary W is treated as a variable of the network. As in the last chapter, the elements of W are constrained to lie in the interval $[0, 1]$ using a sigmoid function. The topic model loss can be combined with the classification loss via weighted summation to get the total loss function that guides the learning of the neural network: $\mathcal{L} = \mathcal{L}_t + \beta\mathcal{L}_c$.

We make minor changes to the neural network architecture introduced in the previous chapter. As mentioned previously, the network is designed to predict the topic encoding for each training instance based on an input chest x-ray image, update the dictionary to adjust for the noise in the predicted topic encodings and based on feedback from the downstream diagnosis task, and use the topic encodings as the input features to a classifier for the diagnosis task. Ultimately, the goal is to force the network to make decisions in a manner similar to expert radiologists by encouraging the network to attend to properties derived from the radiologist-annotated natural language reports. The model proposed in the previous chapter used a linear logistic regression classifier because the primary goal of the previous chapter was maximizing the interpretability of the network. In contrast, for the network proposed in this chapter, we replace the linear classification block with a non-linear block because we prioritize predictive performance on the diagnosis task. As in the previous chapter, we solve for an initial dictionary outside of the network and finetune it inside the net on order to promote

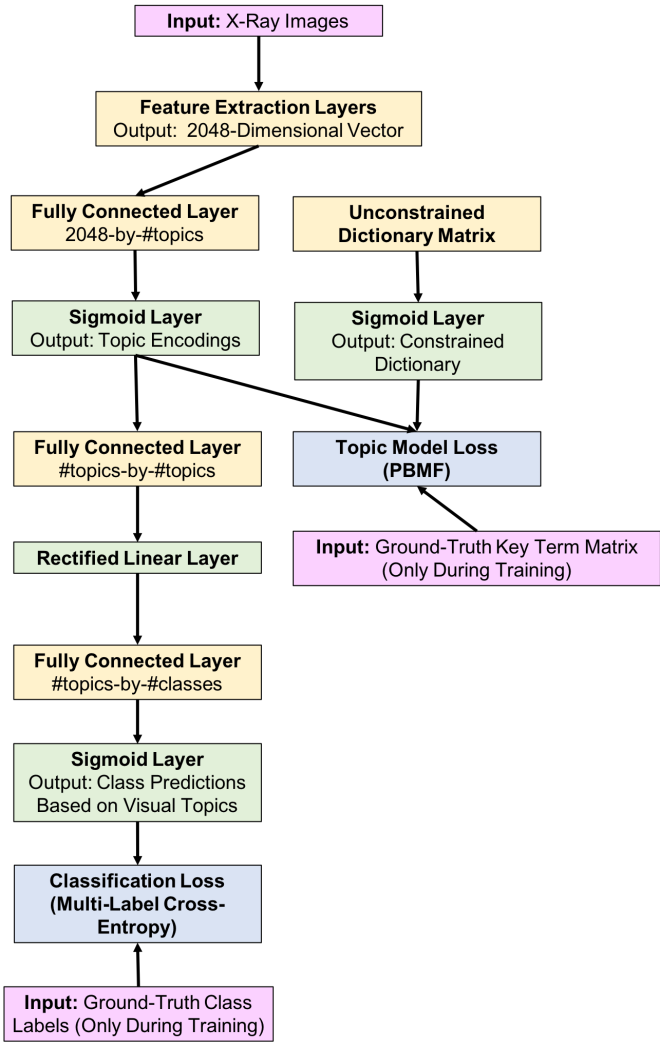


Figure 5.3: We present the structure of the proposed neural network model for combining topic modeling with convolutional networks. Image from [4].

faster convergence to a better topic dictionary. Our architecture appears in Fig. 5.3.

Our final network is trained for 100 epochs with 166 batches per epoch using a batch size of 16. We use the same sampling and data augmentation strategies that were introduced in Section 5.5.1. We set $\alpha_1 = 0.01, \alpha_2 = 0.001, \alpha_3 = 0.1$, and $\beta = 10$. These values were determined experimentally on the first fold of the data using a holdout validation set.

Features	m-AUROC	mAP
All Key Terms	0.969	0.814
Doc2Vec	0.927	0.613
Our Approach	0.928	0.674

Table 5.1: We evaluate the predictive power of various features extracted from natural language medical reports for diagnosing chest-related illnesses.

5.6 Experimental Results and Analysis

In this section, we evaluate the strengths and weaknesses of the various components of our model. We evaluate the predictive power of the topics learned using only the ground truth text in order to get a basic understanding of how much useful information is contained in the natural language reports. We also evaluate the effect of integrating the topic modeling component with a convolutional neural network.

5.6.1 Evaluation Metrics

Before we begin evaluating the strengths and weaknesses of the proposed model experimentally, we introduce the evaluation metrics used in our experiments. Typically, the macro-area under the receiver-operator curve (macro-AUROC) is used to evaluate automated chest x-ray analysis systems. However, this metric is often ill-suited for problems exhibiting large amounts of class imbalance, as is the case with this problem. In automated chest x-ray analysis, there is a higher priority associated with identifying rare positive cases (i.e., patient X has disease Y) than common negative cases (i.e., patient X does not have disease Y). Ideally, in such cases, it is better to use metrics based on the precision-recall curve. Thus, when evaluating our methods, we also use the mean average precision (mAP). The mean average precision provides a single value that summarizes the precision-recall curve over all labels.

5.6.2 Text-Based Experiments

First, we want to get a basic understanding of how much useful information is embedded in the medical reports without considering image data. Moreover, we want to see how much of this information can be preserved via our proposed PBMF-based topic model.

We train classifiers to predict the chest-related illnesses mentioned in earlier sections using various text-based features. Results are reported in Table 5.1.

We achieve the highest high macro-AUROC (about 0.97) and mAP(about 0.81) when we use the full bag-of-key term vectors as features. We also compare with the distributed bag-of-words Doc2Vec embedding [283] with 200 dimensions trained on the reports as a baseline method. Doc2Vec is a very common method in the natural language processing community for embedding text as a numerical vector. Finally, we evaluate our approach: a PBMF topic model with 200 topics. Our approach and the Doc2Vec representation lose a significant amount of predictive power compared to the raw bag-of-key terms representation, but results are still promising. It is worth mentioning that while our PBMF-based representation matches the Doc2Vec embedding in terms of macro-AUROC, our representation outperforms the Doc2Vec representation by about 6% in terms of mean average precision. Overall, our results show that if possible, we should train a model to recover the full bag-of-key terms representation from visual data if no text report is provided. Unfortunately, as we have discussed in previous sections, it is virtually impossible to recover this information with any degree of accuracy for a number of reasons.

5.6.3 Imaging-Based Experiments

Next, we consider how well a set of diagnostic labels can be predicted from visual inputs (chest x-ray images). We train and test a standard ResNet-22 neural network and a modified ResNet-22 neural network that bottlenecks through the topic modeling layer (using 200 topics). We report the experimental results in Table 5.2. On average, our proposed model outperforms the standard model by about 1% in terms of the macro-AUROC and 2% in terms of the mAP. We also outperform the standard neural network in terms of AUROC for three of the five diagnostic labels, and we outperform the standard neural network in terms of average precision for four of the five diagnostic labels, sometimes by several percentage points.

The improvements made by our model are relatively small compared to what Table 5.1 suggests should be possible. This is because the topic recognition component is

	Standard	Our Approach
macro-AUROC	0.857	0.867
mAP	0.459	0.477

		Standard		Our Approach	
Diagnosis	# Instances	AUROC	AP	AUROC	AP
Normal	1395	0.774	0.647	0.783	0.655
Atelectasis	309	0.785	0.303	0.812	0.332
Cardiomegaly	329	0.930	0.592	0.927	0.587
Effusion	148	0.926	0.514	0.921	0.535
Emphysema	110	0.868	0.241	0.892	0.276

Table 5.2: We evaluate the performance of learned visual classifiers. The top table looks at the overall predictive performance of the model while the bottom table looks at predictive performance on individual diagnostic labels (bottom). We compare a standard ResNet-22 architecture with our modified architecture that bottlenecks through the topic modeling layer.

far from perfect. If we compare our visually recognized topic encodings to *approximate* ground-truth encodings (i.e., using the dictionary learned on the training data, we encode the ground truth test data), we only achieve an mAP of about 0.24 when considering all 200 topics. This means most topics are not recognized with high accuracy, and this is likely because the net overfits the rare topics. To better understand the limitations of our model, we can focus on topics that appear in at least 75 (about 2%) of the training instances which accounts for about 60% of the learned topics. In this case, the mAP for topic recognition significantly rises to 0.39, suggesting that the network is encoding some useful information. However, these numbers are far from desirable (especially for medical applications) and also significantly limit the interpretability of the model.

5.6.4 Analysis of Quantitative Results

The proposed model's biggest weakness is its ability to overfit to rare topics, which are often "noisy" because of the imperfection extraction of the key terms This tendency to overfit ultimately leads to suboptimal performance when recovering topics from test data and prevents the model from reaching its full potential on the downstream diagnosis task. There are several ways to mitigate this flaw. First, better methods can be

applied during the key term extraction phase. For example, our negative scope detection method is overly simplistic and makes a non-trivial number of mistakes. Second, the method sometimes produces topics that make sense from a semantic perspective but are visually meaningless. For example, we saw that the network learned one topic that grouped words relating to the direction (e.g., "left", "right", "top", and "bottom"); while it makes sense why these words were grouped, there is no way to map visual data to this topic. Third, and similar to the previous solution, we can prune topics that are recognized with poor accuracy. This would significantly improve the interpretability of the model and might improve performance on the downstream classification task by forcing the model to rely on more stable and better-grounded features. Fourth, we could implement some quality control on the key term annotations to ensure that they accurately reflect the text that they are extracted from and to improve the information-completeness of the annotations. One issue we have not addressed is that doctors have different philosophies about what they find important. Some doctors will record all relevant information, whereas other doctors focus on only reporting the abnormal characteristics of an image while ignoring reporting the normal characteristics. For example, they might assume that saying the heart size is normal should be assumed and not explicitly stated. Thus, medical reports are often information-incomplete. Adding quality control would likely require the addition of a human-in-the-loop component of the system. Fifth and finally, the most important way to prevent the network from overfitting to rare topics is to collect more training data. We are working with a very small dataset (for deep learning), and a large portion of the topics appear in very few training samples. Thus, there is simply not enough data to capture the intricate patterns needed to properly recognize many of the topics.

Despite the flawed topic recovery from visual data, some improvement is still seen in terms of performance on the target task. We hypothesize that there are several potential reasons for this. First, our approach employs privileged information during training in the form of expert-annotated "findings". This privileged information might guide the neural network to explore different regions of the feature and parameter space beyond the regions that would be explored when only considering visual data. By injecting

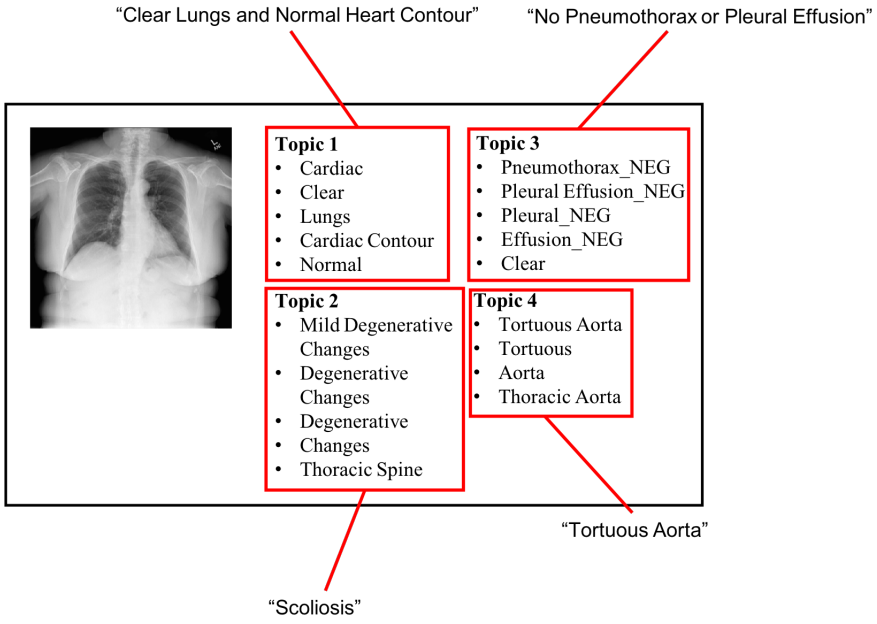


Figure 5.4: We show an example of an x-ray and some of its highly-ranked topics.

expert-annotated prior knowledge into the system, the model can learn patterns it otherwise would not. Second, forcing the neural network to jointly learn the topic model and classifier might act as a form of regularization. Third, there are minor differences in the base network architecture between the standard ResNet-22 architecture and our proposed architecture. For example, we use a non-linear classification layer because there is no guarantee that the topics will result in a feature space that ensures different diagnosis classes are linearly separable. In contrast, the standard ResNet-22 network explicitly enforces this assumption with respect to the learned feature space. Even these small changes might affect performance.

It should also be noted that while the topics are recovered with too low accuracy, and thus, the explanations output by the network cannot be trusted with high confidence, bottlenecking through the topic modeling layers does help us better understand what the network might be attempting to do during its decision-making process. The topic model learned by the network tells us something about what the network thinks it is learning in order to make decisions, even if it fails to actually learn this information. This information could be beneficial for debugging the behavior of the model and developing improvements to correct flaws in the decision-making process.

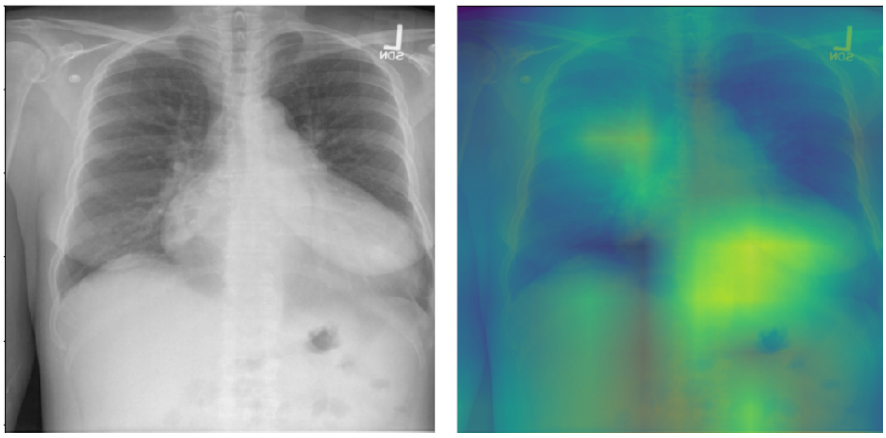


Figure 5.5: We show an example of the net attending to a topic relating to enlarged hearts.

There are other changes that can be made to our model that are more application-specific. Promising future work includes exploring how multi-view data can be utilized (i.e., using both frontal- and lateral-view x-ray images), including patient history and demographics into the learning process, and going beyond topic models and instead generating complete sentences from the topic encoding.

5.6.5 Qualitative Results

Finally, we want to show some qualitative results highlighting how when the topics can be accurately recognized from visual data, they provide useful information. Figure 5.4 shows an example of an x-ray and some of its highly-ranked topics. This highlights some of the types of concepts the network attempts to recognize while also calling attention to the utility and potential of using the predicted topics as an interpretable intermediate feature representation. It should also be noted that, as in the previous chapter, our proposed model is compatible with the class activation mapping (CAM) technique [2]. As such, for each topic, we can generate attention maps showing which parts of an image the network focuses on when predicting a specific topic. Figure 5.5 shows an example attention map for a category relating to the "enlarged heart" topic. Figure 5.6 shows some additional example topics (some diagnostically relevant and others diagnostically irrelevant) learned on the OpenI dataset using our proposed model.

Some Example Topics
Diagnostically Relevant:

- {changes, degenerative changes, degenerative, spine, minimal degenerative changes}
- {aorta, tortuous, ectatic, tortuous aorta}
- {normal heart size, normal heart size mediastinal, normal, normal heart, clear lungs}
- {focal lung consolidation: NEG, pulmonary consolidation: NEG, focal air space consolidation: NEG, consolidation: NEG, alveolar consolidation: NEG, upper mediastinum}
- {atelectasis, subsegmental atelectasis, xxxx atelectasis, basilar atelectasis}
- {lung, low, low lung volumes, lung volumes}
- {cardiomegaly, stable cardiomegaly, mild cardiomegaly}
- {focal infiltrate, focal airspace disease, focal}
- {airspace opacities, xxxx opacities, opacities}

Diagnostically Irrelevant:

- {thoracolumbar spine, cervical spine, lumbar spine, spine, midthoracic spine}
- {view, pa view, lateral, lateral view, frontal view}
- {examination, prior exam. similar, prior examination, prior granulomatous disease, prior}

Figure 5.6: We show several examples of some of the more interesting topics learned by our approach on the OpenI dataset. Many of the learned topics make sense from a visual diagnosis perspective. However, we also see that the model sometimes extracts topics that make sense from a natural language processing perspective but not from a diagnostic or visual perspective. For example, our model learns to group different terms related to the spine, but this group doesn't express any information about the visual appearance of the spine. Similarly, it groups terms related to view of an x-ray, but this topic is visually and diagnostically meaningless. Lastly, we see it group terms related to examinations (in particular "prior examinations"), which once again, doesn't offer any meaningful information for the diagnosis, and it also picks on the keyword "prior" and incorrectly groups the term "prior granulomatous disease" with other terms about "prior examinations".

Chapter 6

Adapting Visual Concepts: Utilizing Scenario-Based Models in Dynamic Settings

6.1 Introduction

Up to this chapter, we have only considered problems that involve learning or utilizing human-understandable representations in static settings. In such settings, once the representation is learned or specified, it is never updated after-the-fact. The static nature of such representation is often insufficient for real-world problems. In the real-world, things change over time. For example, if an agent is deployed in a real-world environment, we cannot assume that it will have perfect knowledge of all of the types of scenes it might encounter. Instead, we must assume that any model that underlies the agent must have the capability to adapt to unforeseen scenes and situations. In this chapter, we develop a model similar to the ScenarioNet architecture introduced in Chapter 4 that can operate in dynamic settings where new types of scenes may be encountered. The model must be capable of identifying if a scene is atypical and subsequently, have the ability to efficient update its structure and parameters.

We once again consider scene classification as our motivating application. Scene classification is a critical computer vision problem with many practical use cases in a wide range of domains, including remote sensing, robotics, autonomous driving, defense, and surveillance. Many approaches to scene classification make simplifying assumptions about the data, and many of the algorithms for scene classification are ill-suited for real-world use cases. We begin by discussing some of the reasons existing methods for scene classification are insufficient for real-world use cases.

The first issue with existing approaches for scene classification is that they assume

the input data is always highly representative of the scene's category. This issue becomes apparent when one examines many of the popular datasets used to train scene classification models (e.g., [21, 245, 256, 284, 285]). Most of these datasets only consider single views that are extremely representative of a limited set of known scene categories. In real-world applications, such perfect data is rarely, if ever, encountered. Consider a robot exploring an indoor environment. If the robot is randomly placed within the environment, it will likely observe both views that are uninformative (e.g., the robot is facing a blank wall) and views that are adversarial (e.g., the robot is looking through a doorway or window into a different type of scene). When constructing agents for real-world settings, one must assume such antagonistic views will be encountered and instead embed the agents with the ability to explore unexpected and adversarial environments in an intelligent and efficient manner.

The second issue with existing approaches for scene classification is that they make closed set assumptions about the types of scenes a model might encounter. However, closed set assumptions about scene categories are rarely satisfied in real-world applications. Sometimes, an agent will meet with a scene unlike any it has previously encountered. Instead of making a misleading and incorrect prediction about the scene category based on the false assumption of a closed world, the agent should have the capability to refuse to make a prediction. Subsequently, the agent should have the capability to query a human for help. The human would then collect more data about the new type of scene. Finally the agent can update its internal knowledge and machine learning models using this new data. In summary, the agent should be capable of performing open set recognition [286, 287] and continuous/active learning [288].

The third issue with existing approaches for scene classification is that most do not prioritize interpretability of the model's decision-making process. Most modern approaches for scene classification rely on complex, black-box models such as deep convolutional neural networks [63]. Deep neural networks are such complex models with such a large number of parameters that humans often find it difficult to understand their decision-making process [5, 42]. However, many very important applications of scene classification (especially safety-critical tasks) require models that can be debugged and

models that can generate explanations to help support decisions that must ultimately be made by humans. For example, in the autonomous driving setting, it is vitally important to understand the limitations of a model to prevent fatal mistakes by the autonomous vehicle. In military applications, where machine learning models are used for decision support systems, models that provide poor recommendations for action based on inaccurate and unreliable evidence can cost the loss of (often innocent civilian) lives.

The focus of this chapter is on building explainable models for scene classification in dynamic settings. Specifically, it addresses the case where an agent with an imaging sensor is placed in some environment, and based on its sensory input, the agent needs to assign a label to the perceived scene. The agent can adjust its sensor to capture more details about the scene, but there is a cost associated with manipulating the sensor. Ideally, the agent must understand the scene in an efficient manner. In order to understand the global state of a scene (i.e., the scene category), the agent must extract properties about the scene from multiple views and use these properties to generate human-understandable explanations about why it made specific predictions. If the agent encounters an unknown type of scene, it should reject assigning a label to the scene, request additional data about the new scene category from a human, and update its underlying knowledge base and machine learning models. Specifically, this chapter is motivated by a specific application in scene understanding: the active explanation-driven classification of indoor scenes.

To address the problem mentioned above, we present an approach based on the ScenarioNet architecture introduced in Chapter 4. For each view encountered by the agent, ScenarioNet is used to extract a human-understandable feature representation. As the agent encounters new views in the same scene, it fuses the predicted scenarios from each view using max-pooling. It then uses Weibull-calibrated support vector machines (W-SVMs) [289] to determine if a scene belongs to an unknown category and for assigning a class prediction. The exploration process (i.e., determining whether to assign a label to a scene or change views) is facilitated using reinforcement learning.

Specifically, our reinforcement learning algorithm consists of linear function approximation of the Q-value with experience replay. Finally, we present extensions of the pseudo-Boolean matrix factorization and ScenarioNet for use in a continuous learning paradigm.

6.2 Related Work

The work in this chapter touches upon many subjects. Some of these subjects include active vision, dynamic data-driven applications systems, high-level information fusion, and open set recognition. In this section, we frame our approach in the context of all of the aforementioned subjects.

6.2.1 Active Vision and Dynamic Data-Driven Applications Systems

Our work is closely related to the active vision/perception paradigm [290–294], which considers the task of visual perception as a dynamic and purpose-driven process whereby observers actively control some imaging sensor. In particular, our work is similar to Li and Guo's work on active learning for scene classification [295]. In Li and Guo's work, object-based features are used for explainable scene classification, and an active learning component is introduced in order to improve the model based on unexpected scenes. Unlike our model, [295] relies on classic methods for feature extraction (e.g., bags-of-patches and SIFT) and only operates on data consisting of clean, single views (i.e., there is no exploration of the scene). Our work is also similar to active scene recognition [296], which employs object-based high-level knowledge to actively guide attention in scene images and videos for improved scene classification. Once again, this method relies on classic visual features and assumes clean, single view images and videos. They do not address the case where unknown scenes are encountered. There are a number of other less-related works which attempt to merge active learning/vision with scene classification [297–301]. Our work also shares similarities to active scene exploration [302, 303], viewpoint selection [304–307], and active object localization and recognition (e.g., [308–312]). Finally, it is worth mentioning that our approach follows

the dynamic data-driven applications systems paradigm [313] where there is a feedback loop between sensor manipulation and data-driven modeling.

6.2.2 High-Level Information Fusion

Our work is also related to the problem of high-level information fusion (HLIF) (e.g., see [314–316]). HLIF involves fusing information captured by multiple sensors based on high-level symbolic information. Our approach follows this paradigm by making predictions about the global states of scenes based on symbolic semantic information captured from different views obtained by manipulating a sensor.

6.2.3 Open Set Recognition

One task that is a central focus of our approach is automatically understanding when new, previously unseen situations are encountered by an agent. This involves constructing models capable of quantifying uncertainty in their predictions and operating based on an open set assumption. Open set recognition (e.g., [286, 287, 289, 317–321]) is a task whereby a classification model can either 1) assign a label from a known closed set of labels, or if confronted with an instance of a new class, 2) reject making a decision about the class assignment and flag the instance as belonging to a new class. Some recent work introduces methods capable of incrementally updating as new classes are encountered [286, 320, 322–328]. Unlike other methods, our model utilizes open set recognition to update both the underlying machine learning models and the underlying knowledge representation.

6.3 Problem

We consider the problem of *active explanation-driven classification of indoor scenes* (see Fig. 6.1). In this setting, an agent is placed in the center of an indoor room and with few sensor adjustments, must assign a label to the scene (e.g., kitchen, bathroom, office) or identify the scene as atypical and update its internal models and knowledge base. After the agent is situated, it 1) captures an image, 2) based on the captured image,

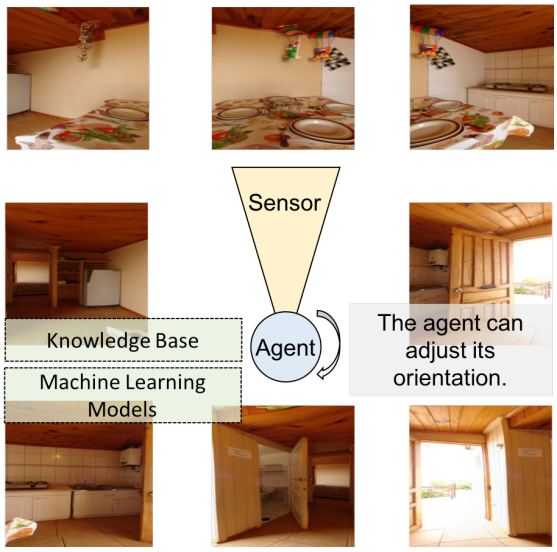


Figure 6.1: We show a visual overview of the problem of active explanation-driven classification of indoor scenes. In this problem, an agent is placed in the center of an indoor room and with few sensor adjustments, must assign a label to the scene or identify that the scene is atypical and and update its internal models and knowledge base.

extracts relevant human-understandable semantic information about the scene, and 3) using this information must make a *grounded* decision about which action to take next. The agent can 1) assign a label to the scene, 2) adjust the orientation of its camera to gather more information about the scene, or 3) determine that the scene is unlike any it has seen before and request additional feedback from humans. If the agent adjusts its sensor, it must be capable of fusing existing measurements/information with the newly obtained measurements/information. If the agent identifies a new type of scene, it must be able to 1) augment/update its existing knowledge base, and 2) augment/update its visual recognition models.

6.4 Data

We extend the SUN360 dataset [329] of *panoramic* scene images. This dataset enables us to easily simulate manipulating an agent with a camera in order to obtain different views. We select 14 common scene categories: atrium, bathroom, bedroom, child's room, church, classroom, conference room, dining room, kitchen, living room, office, restaurant, theater, and workshop. We annotate 35 instances for each scene category.

For each instance, we extract eight views at evenly spaced intervals. Thus, our dataset consists of a total of 3,920 images. Furthermore, for each image, we annotate the presence or absence of 316 unique object classes (e.g., bed, chair, lamp). In our experiments, we only consider the 201 objects that appear in at least five images. For each scene category, we use twenty scene instances for training (160 images), five scene instances for validation (40 images), and ten scene instances for testing (80 images), for a total of 280 training scenes (2,240 images), 70 validation scenes (560 images), and 140 test scenes (1,120 images).

6.5 Methodology

We now provide a summary of our proposed approach:

1. The agent captures an image.
2. The agent extracts human-understandable semantic information from the visual data. This conversion from visual data to human-understandable features is performed using ScenarioNet.
3. The view-level semantic information is fused with existing scene-level semantic information discovered from earlier exploration (i.e., in earlier time steps).
4. The agent computes the predicted probabilities for each scene category based on the scene-level semantic information. This is done using a Weibull-calibrated support vector machine (W-SVM) [289].
5. The agent must then make a decision about which action to take.
    (a) If the agent is highly confident about the category of the scene, it can output a final prediction for the scene category.
    (b) If the agent is highly confident that the scene is atypical, i.e., the agent has never encountered the scene category in the past, it can reject making any decision about the scene category, end exploration, and request human intervention.

(c) If the agent is not confident enough to assign a label to the scene or reject assigning a label to the scene, it can adjust its sensor to either the nearest unseen view or furthest unseen view.

6. If the agent adjusts its sensor, the process begins anew from step 1.
7. If the agent rejects making a decision about the scene category, it queries the human to collect additional data about the unknown scene category, and using this newly collected data, the agent updates the knowledge representation and visual recognition models. This involves extending pseudo-Boolean matrix factorization and ScenarioNet to be compatible with a continuous learning paradigm.

To decide which action to take, we formalize a Markov Decision Process (MDP), and use a linear function approximation of the Q-value with experience replay as our reinforcement learning algorithm for addressing this MDP.

We can break the our proposed framework into four major stages: sensing, processing, decision making/predicting, and updating. Throughout this portion of the chapter, we consider each stage of our proposed framework in more detail. Before we discuss each of these stages, we first present the common training procedure for all neural networks discussed in this chapter.

6.5.1 Neural Network Training Procedure

As a preliminary, we define the standard procedure we follow for training all neural networks in this paper. When a neural network is used, it consists of a ResNet-18 [239] which is pre-trained on the Places-365 scene classification dataset [256] and fine-tuned on individual images from our dataset. The CNN is trained for a maximum of 100 epochs with early stopping based on the validation data. Typically the model converges in fewer than ten epochs. A batch size of 16, learning rate of 1.0e-4, and a weight decay of 1.0e-5 are used. The AMSGRAD optimizer [330] is used. The learning rate is reduced when the training loss plateaus.

6.5.2 Sensing: Understanding the Input Data

The first stage of our framework is to collect data from some set of sensors about a single "view". We define a view as the set of sensor measurements at a single point in time focused on a specific region of interest; i.e., in this work, a view consists of a single image capturing a specific portion of a scene. This stage is relatively simple and does not require us to propose any technical innovations. Therefore, in this section, we will briefly describe the sensing capabilities of our agent. We assume that our agent can capture individual RGB images of single views every time some action is taken. We simulate an active agent with a single camera that is capable of pivoting on some central axis. The agent can capture views at eight positions spaced at even intervals as shown in Figure 6.1. We choose to simulate the agent in order to thoroughly and systematically evaluate our method in a controlled manner.

6.5.3 Processing: From Pixels to Human-Understandable Representations

Next, each image captured by the camera needs to be processed in order to extract representations that are grounded to semantic concepts and which can be understood by humans. As in our earlier chapters, these human-understandable, semantic representations will be used as input features for the machine learning models that decide which action to take next. The processing stage involves three parts. First, we must define the representation. Second, we must define and train a model to map from the raw sensor data to the representation. Third, as new views are encountered, we must define a way to fuse the view-level semantic features to form scene-level features.

Scenarios as Grounded and Interpretable Representations

As is a common theme throughout this thesis, we need to specify a representation grounded to human-understandable visual concepts that is discriminative for the scene classification task and capable of being extracted from visual data with acceptable levels of accuracy. We once again return to using scenarios as this representation. To extract

the scenarios, we once again use the formulation introduced in Chapter 4 with some minor adjustments.

We re-introduce our pseudo-Boolean matrix factorization (PBMF). The problem setup is the same as in Chapter 4, except, instead of decomposing an entire scene instance into its scenarios, we decompose a single *view* into its constituent scenarios.

From Chapter 4:

> Assume that there exists a training set of scene instances and a finite set of pre-specified objects. For each scene instance [view] in the training set, ground-truth annotations for the presence (or absence) of every object in the object set are provided via human labeling. For each of the training [views], we encode object presence information in the form of a binary vector. In this vector, an element is one if the corresponding object is present in the scene instance [view]; otherwise, if the corresponding object is absent in the image, the element is zero. The vectors for all training [views] are concatenated to form a matrix A. Each row of A corresponds to a specific object, and each column corresponds to a specific training [view].
>
> Next, we specify the number of desired scenarios k... Then, A is decomposed into two *approximately* binary matrices. The first matrix resulting from this decomposition is a dictionary matrix W which assigns objects to scenarios. The second matrix resulting from this decomposition is an encoding matrix H that tells us how each scene instance [view] can be represented as a combination of the learned scenarios. Each column of W represents a single scenario, and each row of W represents a specific object. If element W_{ij} is zero (or very small), object i does not belong in scenario j. The closer W_{ij} is to one, the stronger influence object i exerts on scenario j. Each column of H represents a specific scene instance [view], and each row of H represents a specific scenario. If element H_{ij} is zero (or very small), then scenario i is not a component of scene instance [view] j. The closer H_{ij} is to one, the more influence scenario i exerts on scene instance [view] j.

Our formulation of PBMF is similar to the formulation from Chapter 4:

$$
\begin{aligned}
&\min_{W,H} P0 + \alpha_1 * P1 + \alpha_2 * P2 + \alpha_3 * P3 + \alpha_4 * P4 \\
&s.t.\ W \in [0,1]^{m \times k}, H \in [0,1]^{k \times n}, \\
&\Omega_{ij} = \max\left(A_{ij} * \left(1 + \log\left(\frac{N_{instances\_all}}{N_{instances\_object}}\right)\right), 0.5\right), \\
&P0 = ||\Omega \bullet (A - \min(WH, 1 + 0.01WH))||_F, \\
&P1 = ||H - H^2||_F,\ P2 = ||W - W^2||_F, \\
&P3 = ||H^\intercal||_{2,1},\ P4 = ||W^\intercal W - diag(W^\intercal W)||_F
\end{aligned}
\tag{6.1}
$$

m is the number of objects. n is the number of training instances. k is the number of scenarios. Ω is a weight matrix that decreases the importance of common objects and increases the importance of rare objects during the factorization. $\bullet$ denotes element-wise matrix multiplication. The αs represent tradeoff parameters. The key differences between the factorization introduced in Chapter 4 and the one introduced in this chapter is that we impose an $\ell 2, 1$-norm on the scenario encodings to automatically prune unnecessary scenarios. Another difference is that instead of solving a single optimization problem where hyperparameters are determined via a Bayesian optimization-based hyperparameter tuning process, we partially solve a series of optimizations problems in order to automatically adapt the trade-off parameters αs in order to enforce an ordering between the importance of each component of the loss function:

$$
\begin{aligned}
\alpha_1^{(t)} &= 0.5 * \min(P0^{(t)}/P1^{(t)}, 1) \\
\alpha_2^{(t)} &= 0.5 * \min(P0^{(t)}/P2^{(t)}, 1) \\
\alpha_3^{(t)} &= \alpha_1^{(t)} * 0.5 * \min(P1^{(t)}/P3^{(t)}, 1) \\
\alpha_4^{(t)} &= \alpha_2^{(t)} * 0.5 * \min(P2^{(t)}/P4^{(t)}, 1)
\end{aligned}
\tag{6.2}
$$

t represents the current iteration of the optimization problem. After the optimization concludes, we prune the scenario dictionary based on the $\ell 2, 1$-norm of $H^\intercal$.

Mapping from Scene Views to Scenarios

As before, we also require a method for mapping visual data to the scenario encodings. We explore two ways of predicting the scenario encodings H for each image that differs significantly from Chapter 4. In the first method, we threshold H at a value of 0.5 (where a scenario i is considered present in image j if its encoding value H_{ij} is greater than the threshold), and then train a standard ResNet-18 model to perform multi-label recognition (using weighted binary multi-label cross-entropy as the loss function). In this case, the dictionary is always held static and does not receive any feedback from the visual data. In the second method, we refine the dictionary as we train ResNet-18 to predict $H \geq 0.5$:

1. Perform PBMF to obtain an initial dictionary $W^{(0)}$ and ground truth scenarios $H^{(0)}$.
2. Prune the scenarios based on the $\ell 2, 1$-norm of $H^{(0)\intercal}$.
3. Threshold $H^{(t)} \geq 0.5$.
4. Train a CNN to estimate scenario presence from images.
5. Extract the predicted scenario probabilities $\hat{H}^{(t)}$ from all training examples.
6. Refine the dictionary by holding $\hat{H}^{(t)}$ constant and solving for $W^{(t)}$
7. Get new ground truth scenarios by holding $W^{(t)}$ constant and solving for $H^{(t+1)}$
8. Repeat 2-7 until the stopping criteria on the validation data is met.

Our experiments (discussed in later sections) show that updating the scenario dictionary based on visual feedback does improve the performance of the network with respect to how well scenarios can be recognized from previously unseen test data.

Unlike in previous chapters, in this chapter, we do not learn a model for both scenario recognition and scene classification in an end-to-end manner. By separating these two components of the model, we can much more easily adapt the scenario learning process and scene classification process in a continuous learning setting and for open set classification.

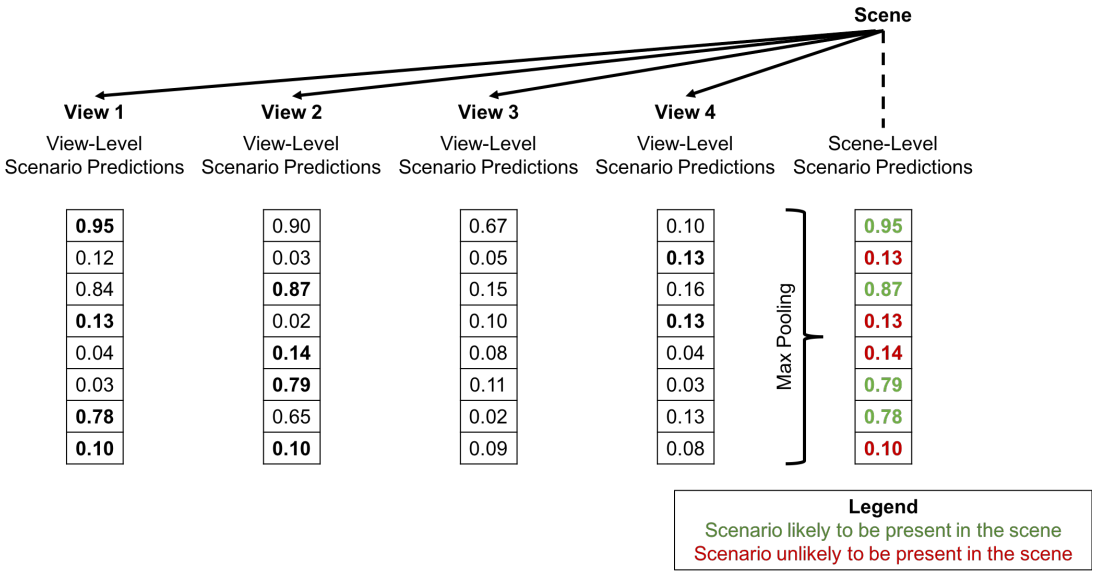


Figure 6.2: A visual representation of the max-pooling operator of view-level scenario predictions to determine scene-level scenario predictions.

Fusing View-Level Scenarios into Scene-Level Scenarios

We have defined a grounded and interpretable representation (the scenario), and we have defined a method for mapping from images to view-level scenarios. Now, we need to address the last component of the processing stage of our framework: how to fuse information as new views are encountered. When the representation consists of scenarios, this task becomes trivial. If a scenario is recognizing as being present in a view, it should have a value close to one, and if it is recognized as being absent, it should have a value close to zero. This property enables a simple method for performing high-level information fusion between views. For a given scenario, its scene-level encoding coefficient can be extracted by taking the maximum value for the view-level scenario encoding coefficient over all of the views encountered up to this current time step. Essentially, we assume that if a scenario is detected as being present in any view, then we assume it is present in the scene as a whole. Our max-pooling based fusion scheme is visually depicted in Figure 6.2. While there are likely better fusion schemes that are more robust to noisy scenario predictions, our experiments will show that this simple max-pooling scheme works well in practice.

6.5.4 Decision Making/Predicting: Classifying Scenes

Once the scenarios have been extracted for a given view or sequence of views, the model uses the extracted scenarios as features for deciding which action to take next. Three actions can be taken. First, the model can decide to assign a label to the scene with high confidence. Second, the model can determine that the scene is atypical of the scene classes encountered during training. Third, the model can conclude that there is not enough evidence to confidently make a prediction in favor of assigning a known class or identifying the scene as atypical. In such cases, the sensor must be manipulated to collect additional information about the contents of the scene from different viewpoints before making a decision. In this section, we explore utilizing the Weibull-calibrated support vector machine (W-SVM) [289] as a means of performing open set recognition.

The Weibull-Calibrated Support Vector Machine for Open Set Classification

Explainability is an important component of the proposed framework. As mentioned in other chapters, there are two crucial characteristics that define the interpretability of a classification model. The input features should be understandable to humans, and the classifier should be easy to interpret, e.g., by probing its components. In the case of our proposed framework, scenarios serve as the human-understandable features. In other chapters, we have used linear multinomial logistic regression models as the interpretable classifier. However, logistic regression is not ideal for the framework introduced in this chapter because logistic regression is not designed for open set classification, where the model must be capable of either outputting a class prediction or rejecting making a decision.

Instead, we employ a Weibull-Calibrated Support Vector Machine (W-SVM) [289]. The W-SVM is an extension of the popular support vector machine (SVM) [331] designed for open set multi-class classification. The W-SVM is grounded in extreme value theory. Unlike logistic regression, the W-SVM is capable of refusing to assign a label when it encounters a novel type of scene without ever seeing an example of the unknown scene category in the training data.

The W-SVM formulation makes use of both a one-class SVM (OC-SVM) [332] and a one-vs-rest SVM. The two SVMs are trained for each of the known classes. After the SVMs are trained, Weibull distributions are fit based on the distances between a training sample (in feature space) and each decision boundary. For the one-class SVM, this enables us to compute the probability of inclusion for a given class $P_O(y|f(x))$. For the one-vs-rest SVMs, we fit two Weibull distributions. The first of these two distributes allows us to compute the probability of inclusion in the target class:

$$P_{R+}(y|f(x)) = 1 - e^{-\left(\frac{f(x)-v_{R+}}{\gamma_{R+}}\right)^{\kappa_{R+}}} \tag{6.3}$$

The second distribution is a reverse Weibull distribution, and it enables the model to compute the probability that the data does not belong to any of the other known classes:

$$P_{R-}(y|f(x)) = e^{-\left(\frac{f(x)-v_{R-}}{\gamma_{R-}}\right)^{\kappa_{R-}}} \tag{6.4}$$

It should be noted that the parameters of the aforementioned Weibull distributions (the location ν, scale γ, and shape κ parameters) can be fit using maximum likelihood estimation.

In our implementation, we use a radial basis function kernel for the one class-SVM, and we use a linear kernel for the one-vs-rest SVM. Because we use a linear kernel for the one-vs-rest SVM, we have a classifier that is relatively simple to understand, and it is used in combination with human-understandable features, so the entire model can be considered to be relatively interpretable. At test time, to determine if a new data point belongs to one of our known classes, we run the following procedure:

1. We begin by testing $P_O(y|f(x)) > \delta_O$ for each class y where δ_O is a small threshold. If no class satisfies this condition, then we can reject making a class assignment and say the scene is atypical.

2. For those classes that pass the first test, we perform a second test by checking $P_{R+}(y|f(x)) * P_{R-}(y|f(x)) > \delta_R$ where δ_R is some threshold. This tests the probability that the input is 1) from the positive class and 2) not from any of the known negative classes. If no class satisfies this condition, then we can reject making a class assignment and say the scene is atypical.

3. Finally, we select our predicted class y^* from the remaining classes by selecting y^* as the argmax of $P_{R+}(y|f(x)) * P_{R-}(y|f(x))$.

In our experiments, the thresholds (δs) are determined empirically via cross-validation. Furthermore, we can compute class-specific probabilities by calculating $P_{R+}(y|f(x)) * P_{R-}(y|f(x)) * I(P_O(y|f(x)) > \delta_O)$ for each y (note: $I(.)$ is the indicator function). These probabilities will become useful in later sections when we start discussing how to incorporate exploration into the proposed pipeline.

It is briefly worth mentioning why we select the W-SVM as our algorithm of choice for the open set recognition problem. We do so for several reasons. First, it is a probabilistic approach, so not only do we gain the capability to reject a class, but we also get a quantitative and principled measure of how likely the class should be rejected. Second, W-SVM is designed for multi-class classification whereas other methods are designed to handle only binary decisions (e.g., methods like the one-class SVM designed for anomaly/outlier detection). Third, if we utilize linear SVMs for the one-vs-rest component of the one class SVM, then we can use it to get an understanding of which features are most important for predicting specific classes. Fourth, experimentally, we have found that the W-SVM is a very competitive method, even when compared to more modern approaches like the extreme value machine [320] and OpenMax classifier [317].

6.5.5 Updating: Adapting the Models and Adjusting the Sensors

In the previous section, we discussed how to detect out-of-distribution scenes. In this section, we talk about what to do once we know that a scene is atypical; namely, we discuss how to update our representations and machine learning models to account for new types of scenes. It should be noted that we propose methods for updating our representations and models on a per-class basis, not a per-instance basis (i.e., the traditional streaming machine learning setting). Furthermore, in addition to updating the models, we also need to consider the problem of how to "update" an agent's sensors by adjusting their position. Thus, this section is also concerned with specifying the exploration component of our proposed framework and introducing some methods from reinforcement learning to implement this specification in practice. We can categorize

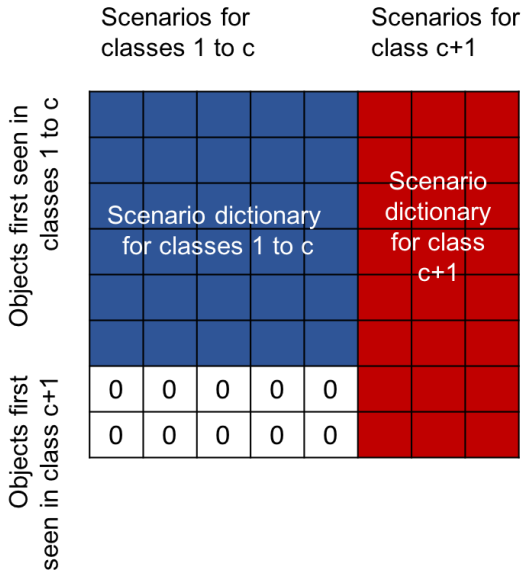


Figure 6.3: A visual representation of the continuous learning variant of PBMF where when a new class is encountered, the existing scenario dictionary is frozen, and we learn a new dictionary of scenarios specific to the new class that gets appended to the existing scenario dictionary.

the main topics of this section into three topics: 1) updating the scenario representation, 2) updating ScenarioNet and the scene classifier, and 3) exploring the scene.

Updating the Scenario Representation

In this section, we propose a simple extension to pseudo-Boolean matrix factorization. Specifically, we show how the scenario dictionary can be augmented using only instances from a new scene category. This augmentation involves solving for a small matrix $W^{(c)}$, which represents scenarios that are specific to the new class c. To discover $W^{(c)}$, we

only need to use ground truth object data from the new class instances $A^{(c)}$:

$$\begin{aligned}
&\min_{W^{(c)},H^{(c)}} P0 + \alpha_1 * P1 + \alpha_2 * P2 + \alpha_3 * P3 + \alpha_4 * P4 \\
&s.t.\ W^{(new)} = [W, W^{(c)}], W^{(new)} \in [0,1]^{m\times(k+k_c)}, W^{(c)} \in [0,1]^{m\times k_c}, H^{(c)} \in [0,1]^{k_c\times n_c} \\
&\Omega_{ij} = \max\left(A_{ij}^{(c)} * \left(1 + \log\left(\frac{N_{instances}}{N_{objects}}\right)\right), 0.5\right), \\
&P0 = ||\Omega \bullet (A^{(c)} - \min(W^{(new)}H^{(c)}, 1 + 0.01W^{(new)}H^{(c)}))||_F, \\
&P1 = ||H^{(c)} - H^{(c)2}||_F,\ P2 = ||W^{(new)} - W^{(new)2}||_F, \\
&P3 = ||H^{(c)\intercal}||_{2,1}, \\
&P4 = ||W^{(new)\intercal}W^{(new)} - diag(W^{(new)\intercal}W^{(new)})||_F
\end{aligned} \tag{6.5}$$

Every time a new class c (with n_c training instances) is added, we solve for k_c new class-specific scenarios $W^{(c)}$ and append them to the old dictionary $W^{(new)} = [W, W^{(c)}]$. See Figure 6.3 for a visual overview of this process. The intuition behind this formulation is that if we have enough "seed" classes when learning the initial dictionary, then the scenarios that are common throughout multiple types of scenes should already be captured in the initial dictionary and require little to no fine-tuning. The remaining scenarios in the initial dictionary should be essentially class-specific to scene classes that we have already seen, so these do not need to be updated either. Then, all that remains is learning how to augment the initial dictionary with scenarios that are specific to the new classes. Likewise, since we impose orthogonality constraints on the dictionary, little redundancy should be encoded within the new scenarios. The scenario selection penalty (using the $\ell 2, 1$-norm) is also useful in this situation because if the number of class-specific scenarios is estimated incorrectly, the unnecessary scenarios can be automatically pruned. Furthermore, we make the observation that we do not need to relearn using all existing data if we only care about class-specific scenarios, so we can save a significant amount of computational time by only optimizing using the new class-specific training instances.

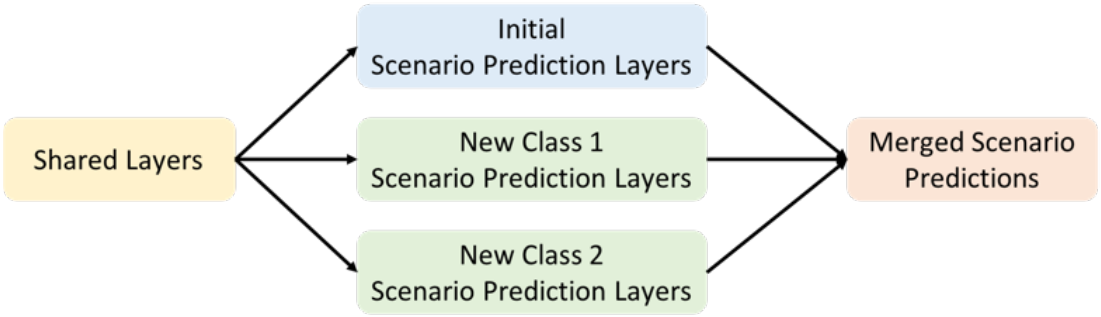


Figure 6.4: As new class-specific scenarios are discovered via the dynamic-variant of pseudo-Boolean matrix factorization, the scenario-predicting neural network is updated by learning new branches consisting of layers designed to recognize the new class-specific scenarios.

Updating ScenarioNet

We have proposed a method for updating the scenario dictionary in a dynamic manner; however, the agent also needs the capability to efficiently learn models that can recognize these new scenarios from visual data without having to retrain the convolutional neural network from scratch using all previously collected data. We propose a branching convolutional neural network model. Whenever a new class is encountered, a new set of class-specific scenarios is learned using the dynamic-variant of PBMF, and subsequently, a branch of the convolutional neural network is trained to perform multi-scenario recognition on just the new scenarios using only data for the new classes. This process results in separate scenario recognition models for each class-specific scenario dictionary. Then, the scenario predictions for each branch can be concatenated to form a unified scenario prediction. We show the model architecture in Figure 6.4.

In our proposed architecture, we have a set of shared layers that feed into class-specific scenario prediction layers. We assume these shared layers are learned on a diverse set of initial scene classes (i.e., the same classes used to learn the initial dictionary), and do not need to be fine-tuned as new classes are encountered. Our experiments show that this is a relatively safe assumption for the given dataset. However, if such an assumption cannot be made, then we would have to employ techniques for minimizing catastrophic forgetting [333, 334] in deep neural network such as elastic weight consolidation [335].

Updating the Scene Classification Model

Training the scene classification W-SVM is significantly less computationally expensive than training the scenario-recognizing neural network. Likewise, while there exist machine learning models capable of performing open *world* recognition [286] (where the model is updated as new classes are encountered), it is a much harder problem to perform open-world recognition when the dimensionality of the feature space is also changing (as is the case in this instance where the number of scenarios grows as new classes are encountered). As such, we do not make any major changes to the W-SVM formulation, and simply retrain the W-SVM from scratch every time a new class is encountered. However, we feel this is inefficient and as such, is important future work.

Formulating a Policy for Exploring the Scene

Finally, we need to model the exploration component of our approach. We define a Markov Decision Process for the problem of active explanation-driven classification of indoor scenes. We define a *state* to be the vector of class probabilities output by W-SVM (see Section 6.5.4) concatenated with the rejection score (one minus the maximum probability output by W-SVM), and the number of views seen. There are four *actions*: 1) make a class prediction and end exploration, 2) reject making any decision and end the exploration process, 3) adjust the camera to the nearest unseen view, and 4) adjust the camera to the furthest unseen view. We define the *rewards* as: -1 if the view is changed when the agent would have made the correct prediction, -#maximum_views if the model predicts an incorrect class, -#maximum_views if the model refuses to make a prediction and ends exploration when it would have predicted the correct class, and #maximum_views + (number of remaining unseen views)^v if a correct classification is made. v is a parameter which controls the trade-off between accuracy and exploration. The *terminal states* are when a prediction is made, when the agent rejects making a prediction and ends exploration, or when there are no more views left to consider. We use a linear function approximation of the Q-value with experience replay as our reinforcement learning algorithm.

Method	Single-View Accuracy	All-View Accuracy
Standard ResNet-18	0.504	0.586
Ground Truth Object Presence + Log. Regression	0.654	0.793

Table 6.1: Object-based representations for scene classification are very discriminative, especially when compared to purely visual features (using a fine-tuned ResNet-18). It is also apparent that using all views of a scene results in significant performance gains compared to using just a single view.

6.6 Experimental Results and Analysis

In this section, we empirically justify the various components of our system. We begin by reiterating the importance of object-based representations.

6.6.1 Understanding the Importance of Object-Based Representations

As in previous chapters, we begin by justifying the use of object-based representations. In Table 6.1, we investigate the discriminative power of object-based representations for scene classification. We train a multinomial logistic regression model using the *ground-truth* presence of objects as a binary feature vector. As a baseline, we consider purely visual features that are extracted using a ResNet-18 convolutional neural network fine-tuned on individual images from our dataset. We examine performance when we treat every single view as its own scene and also when we fuse information about all of the views for a scene. For the object-based classifier, it is trivial to fuse information for all views by making a binary feature vector signifying which objects exist in any of the views and then training a logistic regression model on the scene-level object vectors. For the ResNet-18 baseline, we naïvely fuse information from all views by outputting the scene category that has the maximum predicted probability when all individual view prediction probabilities are max-pooled. Results suggest object presence is a powerful representation for scene classification, outperforming the visual features.

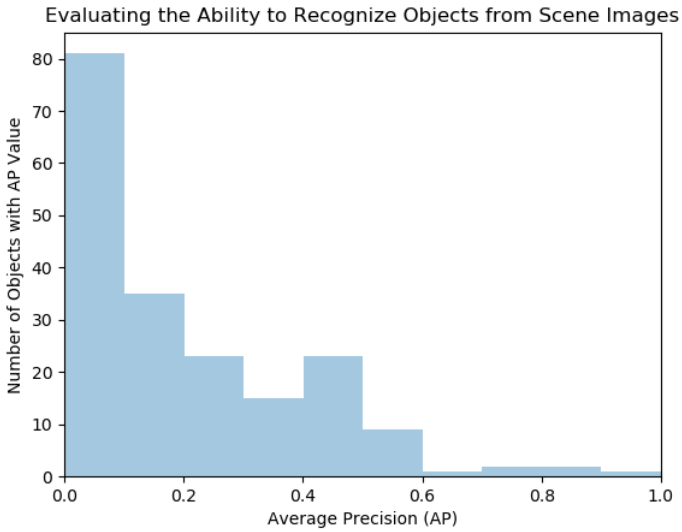


Figure 6.5: Results show that the majority of objects in our extended SUN360 dataset cannot accurately recognized from scene images.

Method	Single-View Accuracy	All-View Accuracy
Ground Truth Object Presence + Log. Regression	0.654	0.793
Predicted Object Probabilities + Log. Regression	0.473	0.564

Table 6.2: Classification performance diminishes when noisy predicted object presence features are used for scene classification.

6.6.2 Understanding the Limitations of Using Object Presence as Features

In practice, we do not have access to ground truth object information, and instead, the object presence information must be estimated from visual data. If a ResNet-18 convolutional neural network is trained to perform multi-object recognition (using weighted binary multi-label cross-entropy as the loss function) from scene images, we start to see some of the problems with using object presence as features for scene classification. Unsurprisingly, results are consistent with those obtained in earlier chapters on similar datasets. Figure 6.5 shows us that the majority of objects in our dataset cannot be accurately recognized. We hypothesize that this is because there is not enough data to learn all possible deformations and variations of appearances for most of the objects.

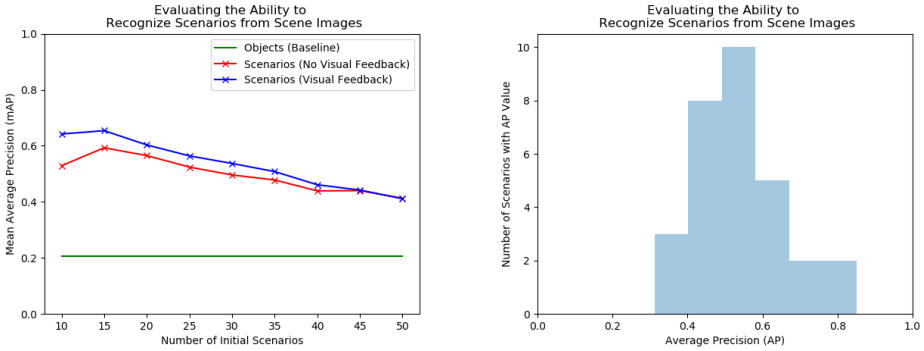


Figure 6.6: Left: We attempt to measure how scenario recognition performance is affected by the number of scenarios. We also see that refining the dictionary based on visual feedback is useful for improving scenario recognition. Right: We train a model to learn to recognize 30 scenarios. Results show that scenarios can be relatively accurately recognized from scene images, especially when compared to predicting individual object classes (Figure 6.5).

Furthermore, if the predicted object probabilities are used as features for scene classification, the model can no longer be considered explainable because the features can no longer be trusted. Similarly, in Table 6.2, we see that scene classification performance diminishes when noisy predicted objects are used as features for scene classification. This provides justification for why converting from an object presence-based representation to a scenario-based representation is a reasonable action.

6.6.3 Justifying Scenarios as Discriminative, Human-Understandable Features

In Figure 6.6, we evaluate how scenario recognition is affected by the number of scenarios as well as how the recognition of scenarios is affected by the use of visual feedback. The average scenario is much easier to recognize than the average object, suggesting that a scenario-based model is more trustworthy than a model that relies only on object presence. This is further validated by comparing Figures 6.5 and 6.6.

We also evaluate the use of scenarios for scene classification by training a logistic regression model on the predicted scenario probabilities. Information about scenarios over multiple views can be easily fused by simple max-pooling. Fig. 6.7 shows

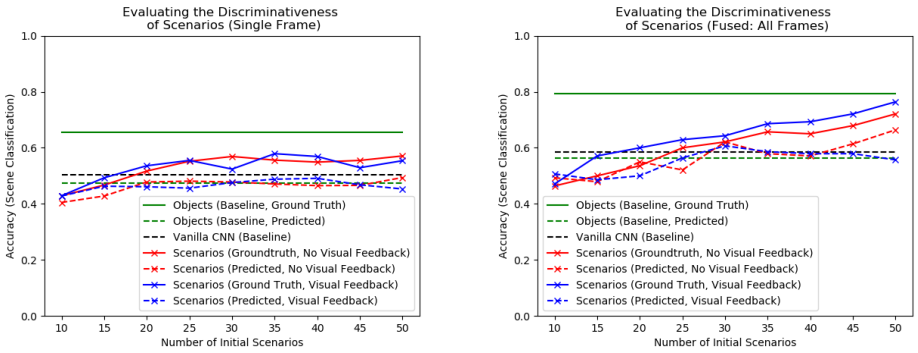


Figure 6.7: We attempt to understand how well a scenario-based model works for scene classification when using only single-views and when using all available views of a scene.

that visual scenario-based models can achieve scene classification accuracy on par with standard CNNs and object-based representations despite being low-dimensional. This property is likely due to the scenarios' ability to compress semantic information by efficiently exploiting context between objects effectively. Scenario-based models have the additional benefit that the features are human-understandable, and since logistic regression is a simple linear model, it is easy to evaluate the importance of each scenario when making a specific classification decision.

We also conducted human evaluation experiments to justify that humans are capable of understanding scenarios and understanding scenario-based explanations for scene classification. This set of experiments is very similar to those conducted in Chapter 4, but we use a different dataset and a much larger sample size. In the first experiment, we presented 30 scenarios (learned by our model with visual feedback) to 20 English-speaking participants using Amazon's Mechanical Turk (AMT) service. We asked participants to say whether each scenario "is a meaningful group of objects", "might be a meaningful group of objects but doesn't align with my expectations", "is a meaningless group of objects", or "consists of objects I'm not familiar with". After taking the modal response for each question, respondents found **74.1% of the scenarios were meaningful, 11.1% might be meaningful, and 14.8% were meaningless.** Upon examining the meaningless scenarios, we found that these were often considered "meaningless" because sometimes the PBMF would add one or two seemingly random

objects to a meaningful scenario or accidentally merge two meaningful scenarios. These problems could likely be solved by using a slightly larger number of scenarios.

We then evaluated if humans could accurately identify the category of a scene when presented with only the most influential scenarios and their influence score (the predicted scenario probability output by the CNN multiplied by the corresponding weight in the logistic regression model). For this experiment, scenarios were pooled over all views. We gathered 15 English-speaking participants using AMT and for 50 random test scenes that were correctly classified by our model, the participants were given a list of all scenarios with an influence score greater than one and asked to predict the scene class from four choices (one true, three randomly chosen). After taking the modal response for each question, **98% of the scene classifications were correct**, suggesting that the model output plausible explanations.

6.6.4 Evaluating the W-SVM for Identifying Unknown Scene Categories

Next, we consider how predictive performance is affected when our problem shifts from the traditional classification paradigm to the open set classification paradigm. We conduct an experiment to discern whether performance is gained, lost, or remains unchanged when using a Weibull-calibrated support vector machine instead of a logistic regression model *when all classes are known (i.e., the traditional classification setting).* The scenario recognition model (with 30 scenarios) is trained on all classes, and then the predicted scenario probabilities are used as features for both a logistic regression model and a W-SVM for scene classification. From Table 6.3, we see that the W-SVM slightly underperforms the logistic regression model by 1-2%.

We also want to see how well using scenarios with W-SVM works for scene classification *when there are unknown scene categories (i.e., the open set classification setting.* Ten trials are run. In each trial, seven known and seven unknown classes are randomly selected. The scenario recognition model is trained using only the known classes. Information from all views is used. The thresholds for rejection are determined based on the held-out validation set. We measure the accuracy with respect to the known classes

Method	Single-View Accuracy	All-View Accuracy
Predicted Scenario Probabilities + Log. Regression	0.476	0.607
Predicted Scenario Probabilities + W-SVM	0.459	0.593

Table 6.3: We measure the performance of logistic regression and W-SVM (using 30 predicted scenario probabilities as features) when all classes are known.

Method	Known Class Acc.	Unknown Class Rec.	Unknown Class Prec.	Unknown Class AUPRC
Predicted Scenarios + NN-CAP	0.424	0.489	0.581	0.563
Predicted Scenarios + PI-SVM	0.511	0.594	0.613	0.594
Predicted Scenarios + EVM	0.513	0.542	0.632	0.615
Predicted Scenarios + OpenMax	0.460	0.554	0.609	0.574
Predicted Scenarios + W-SVM	0.525	0.577	0.617	0.603

Table 6.4: We measure the performance of combining scenarios with various popular classifiers for open set recognition task using 30 scenarios and all views with 7 known classes and 7 unknown classes. Results are averaged over 10 random trials.

as well as the precision, recall, and area under the precision-recall-curve (AUPRC) for the unknown classes. Table 6.4 shows that using W-SVM as the classifier, results in a decrease in performance in terms of known class accuracy, but we gain the ability to perform open set recognition, achieving promising results in terms of the unknown class precision, recall, and AUPRC. We hypothesize that the likely reason for the decrease in accuracy on the known classes is that only half of the data is used for training compared to the previous experiment.

We also compare the W-SVM to several other popular open set recognition models, specifically the nearest neighbor + compact abating probability (NN-CAP) model [289], "probability of inclusion" support vector machine (PI-SVM) [318], extreme value machine (EVM) [320], and OpenMax classifier [317] (albeit using a single-layer neural network, which it was not designed for). Results appear in Table 6.4. In general, the W-SVM is competitive with all of the aforementioned methods, and compared to the other models, does an especially good job balancing performance on the known class multi-class classification task and unknown class rejection task.

Method	Single-View Accuracy	All-View Accuracy	Reconstruction Error
Ground Truth Scenarios (Static) + Logistic Regression	0.620	0.684	205.2
Ground Truth Scenarios (Dynamic) + Logistic Regression	0.573	0.678	247.6

Table 6.5: We compare the quality of a dictionary learned by Dynamic PBMF to one learned with regular PBMF.

Method	Scenario Recognition mAP	Single-View Accuracy	All-View Accuracy
Pred. Scenario Scores (Dynamic PBMF + Trad. Model) + Log. Reg.	0.414	0.503	0.649
Pred. Scenario Scores (Dynamic PBMF + Branch. Model) + Log. Reg.	0.410	0.543	0.648
Pred. Scenario Scores (Dynamic PBMF + Trad. Model) + W-SVM	0.414	0.488	0.587
Pred. Scenario Scores (Dynamic PBMF + Branch Model) + W-SVM	0.410	0.525	0.634

Table 6.6: Understanding the the performance of the branching convolutional neural network model for scenario recognitionin combination with different classifiers.

6.6.5 Evaluating the Dynamic-Variant of Pseudo-Boolean Matrix Factorization and the Branching Scenario-Recognition Neural Network

In the next set of experiments, we evaluate our proposed approach for updating the scenario dictionary using a dynamic-variant of pseudo-Boolean matrix factorization and our proposed approach for updating the scenario-recognizing convolutional neural network (our branching neural network). To validate our method for dynamically updating the scenario dictionary (Dynamic PBMF), we run experiments over ten trials where an initial dictionary is learned using seven classes and twenty scenarios. For each remaining class, ten class-specific scenarios are learned, pruned based on the $\ell 2, 1$-norm of $H^{(c)\intercal}$, and appended to the existing dictionary. For each trial, as a baseline, we learn a set of scenarios using all data for an equal number of scenarios as Dynamic PBMF outputs. We compare Dynamic PBMF to regular PBMF in terms of 1) reconstruction error on the entire dataset and 2) discriminability on both the single-view and all-view scene classification tasks. Results appear in Table 6.5. On average, Dynamic PBMF learns 43.5 scenarios. As expected, regular PBMF results in a lower reconstruction error and higher scene classification accuracies, but Dynamic PBMF is very competitive. The reconstruction error of Dynamic PBMF is only 1.2 times larger than regular PBMF, and scene classification accuracy is within 1% on the all-view classification task, and

within 5% on the single-view classification task.

Next, we evaluate using the dynamic-variant of pseudo-Boolean matrix factorization in combination with the branching scenario-recognizing neural network. Ten trials are run where an initial model was trained on data from half of the scene categories and then a cascade of models were learned on the remaining seven scene categories. Results appear in Table 6.6. On average, 39.3 scenarios are learned per trial. The branching CNN achieves comparable scenario recognition performance compared to a single model trained on all data at once. The branching model achieves superior single-view scene classification performance and equivalent all-view scene classification performance when compared to the traditional model. These results provide promising evidence that since class-specific scenarios are ideally independent between different scene categories, they can be learned with significantly limited data. It should be noted that the ability of the model to generalize quickly might be due in part because the ResNet feature extraction portion of the network is pre-trained on Places-365, a large-scale scene dataset, before being fine-tuned on the SUN360 dataset.

6.6.6 Understanding the Necessity of Exploration

As has been previously mentioned, most datasets for scene classification (e.g., [21, 245, 256, 284, 285]) consider "clean" single views that are very representative of a limited set of known scene categories. In real-world applications, if an agent (e.g., a robot) is placed in a new scene, it might encounter 1) views that are uninformative (e.g., the agent is facing a wall) or 2) views that are adversarial (e.g., the agent is looking through a doorway into a different type of scene). Thus, it is necessary for the agent to examine multiple views to understand its surroundings better. This is validated in the results in many of the previous experiments. For example, in Table 6.1, we see that for simple baselines, using information from all views of a scene results in significantly improved classification performance. Similar results are shown in Figure 6.6 and Tables 6.2, 6.3, and 6.6, which highlight how performance improves when all views are considered (over just single views) when the features are scenarios and not objects, regardless of what classification model is used (logistic regression or W-SVM), in both static and dynamic

v	Mean Number of Actions	Known Class Accuracy	Unknown Class Recall	Unknown Class Precision
0	6.66	0.49	0.78	0.61
1.5	4.19	0.46	0.56	0.59

Table 6.7: We explore how well the learned policy performs on the task of active explanation-driven classification of indoor scenes. We show the trade-offs between maximizing predictive performance and minimizing unnecessary exploration.

problem settings.

6.6.7 Evaluating the Exploration Component of our Proposed Framework

Next, we need to see if the reinforcement learning formulation for addressing the problem of the active explanation-driven classification of indoor scenes introduced earlier (see Section 6.5.5) is beneficial or a waste of time and computational resources. Ten trials are run. In each trial, we randomly select seven known classes and seven unknown classes. A single W-SVM model is trained using scene-level feature vectors, each constructed from 1-8 randomly sampled views per randomly sampled training instance. We then learn a policy as described in Section 6.5.5 and apply the policy to the unseen test data. We measure several important metrics including the average number of actions taken, the accuracy of the predictions on the known classes, and the precision and recall of rejecting the unknown classes. Results are reported in Table 6.7. Compared to the results in Table 6.4, which involved no active exploration and used all scene views, if we set $v = 0$ (i.e., exploration is not penalized), similar known class accuracy and unknown class precision are achieved while unknown class recall significantly improves. If we set $v = 1.5$ (we relatively heavily penalize unnecessary exploration), similar known class accuracy, unknown class precision, and unknown class recall are achieved as compared to the results in Table 6.4 while we only need to consider about four views on average. These results suggest that exploration is useful, and our proposed reinforcement learning formulation is a promising approach towards addressing the problem of the active explanation-driven classification of indoor scenes.

6.6.8 Qualitative Results

Finally, we want to show some qualitative results in order to highlight both the explanatory power of the proposed approach and the utility of using information from multiple views for scene understanding and classification. In Figures 6.8, 6.9, and 6.10 we show two scenes with eight views each. For each scene, we show the predicted scenarios. It should be noted that the scenarios learned on this dataset tend to be much denser (i.e., contain more objects) than those of Chapter 4. Specifically, there tend to be more class-correlated scenarios where one scenario covers most of the common objects for a specific class (e.g., there is a scenario that covers general bathroom appliances, another one that covers kitchen appliances, another one for tools that would appear in a workshop, etc.). As such, the scenarios produced on this dataset tend to result in less fine-grained, but still useful explanations.

View 1:

Scenario #1

- closet
- displaycase
- door

View 2:

Scenario #1

- closet
- displaycase
- door

View 3:

Scenario #1

- bathmat
- bathtub
- bidet
- counter
- exhaustfan
- hook
- lantern
- mirror
- scale
- shower
- sink
- soapbar
- soapdispenser
- supportbar
- tissuebox
- toilet
- toiletpaper
- toiletpaperholder
- towel
- towelrack
- towelring
- trashcan
- washingmachine

View 4:

Scenario #1

- bathmat
- bathtub
- bidet
- counter
- exhaustfan
- hook
- lantern
- mirror
- scale
- shower
- sink
- soapbar
- soapdispenser
- supportbar
- tissuebox
- toilet
- toiletpaper
- toiletpaperholder
- towel
- towelrack
- towelring
- trashcan
- washingmachine

View 5:

Scenario #1

- bathmat
- bathtub
- bidet
- counter
- exhaustfan
- hook
- lantern
- mirror
- scale
- shower
- sink
- soapbar
- soapdispenser
- supportbar
- tissuebox
- toilet
- toiletpaper
- toiletpaperholder
- towel
- towelrack
- towelring
- trashcan
- washingmachine

View 6:

Scenario #1

- bathmat
- bathtub
- bidet
- counter
- exhaustfan
- hook
- lantern
- mirror
- scale
- shower
- sink
- soapbar
- soapdispenser
- supportbar
- tissuebox
- toilet
- toiletpaper
- toiletpaperholder
- towel
- towelrack
- towelring
- trashcan
- washingmachine

View 7:

View 8:

Scenario #1

- bathmat
- bathtub
- bidet
- counter
- exhaustfan
- hook
- lantern
- mirror
- scale
- shower
- sink
- soapbar
- soapdispenser
- supportbar
- tissuebox
- toilet
- toiletpaper
- toiletpaperholder
- towel
- towelrack
- towelring
- trashcan
- washingmachine

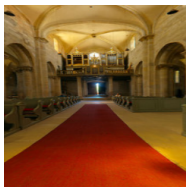
Figure 6.8: We show an example of a bathroom scene decomposed into eight views in order to highlight both the explanatory power of our proposed approach and the utility of using information from multiple views for scene understanding and classification. For each view, we show the predicted scenarios. This example presents an interesting case. If an agent were to start with the first view, it wouldn't be able to predict a bathroom scene because it only detects the "closet/door" scenario. However, if the agent looks at views 3, 4, 5, 6, and 8, then there is strong evidence that the scene is a bathroom. Another interesting property of this example is that it includes an uninformative view, specifically view 7 which is essentially a blank wall. In this case, our neural network doesn't detect any useful semantic information and as a result, doesn't output any scenario prediction with high confidence.

View 1:

Scenario #1

- altar
- bench
- booth
- ceilingfan
- cross
- fireplace
- paintedceiling
- pew
- pulpit
- rope
- stainedglass
- statue_model

View 2:



Scenario #1

- hangingart_painting

Scenario #2

- barrier_gate_fence
- decal_mural
- rope
- stairs
- supportbar

Scenario #3

- closet
- displaycase
- door

Scenario #4

- altar
- bench
- booth
- ceilingfan
- cross
- fireplace
- paintedceiling
- pew
- pulpit
- rope
- stainedglass
- statue_model

View 3:

Scenario #1

- altar
- bench
- booth
- ceilingfan
- cross
- fireplace
- paintedceiling
- pew
- pulpit
- rope
- stainedglass
- statue_model

View 4:

Scenario #1

- altar
- bench
- booth
- ceilingfan
- cross
- fireplace
- paintedceiling
- pew
- pulpit
- rope
- stainedglass
- statue_model

View 5:

Scenario #1

- hangingart_painting

Scenario #2

- altar
- bench
- booth
- ceilingfan
- cross
- fireplace
- paintedceiling
- pew
- pulpit
- rope
- stainedglass
- statue_model

View 6:

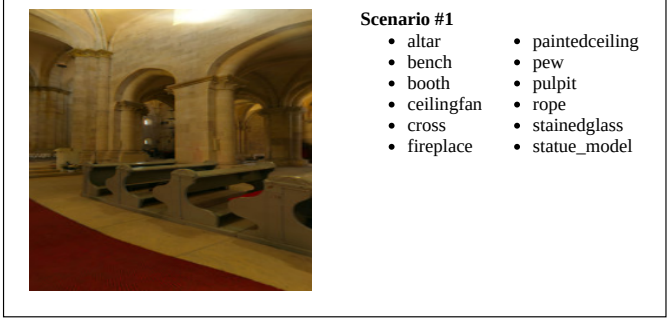
Scenario #1

- barrier_gate_fence
- decal_mural
- rope
- stairs
- supportbar

Scenario #2

- altar
- bench
- booth
- ceilingfan
- cross
- fireplace
- paintedceiling
- pew
- pulpit
- rope
- stainedglass
- statue_model

View 7:



View 8:

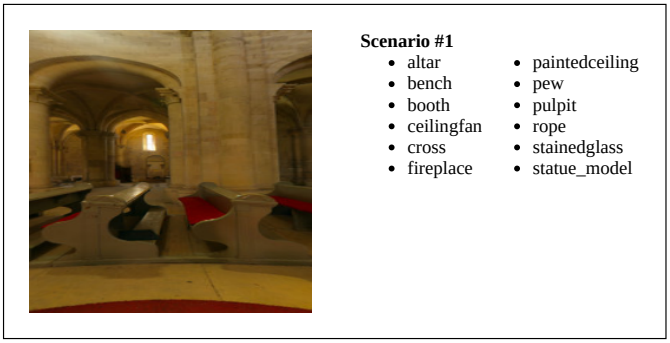
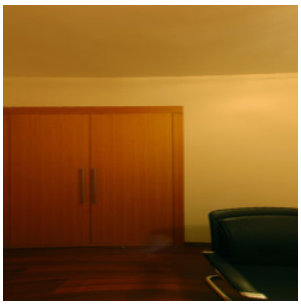


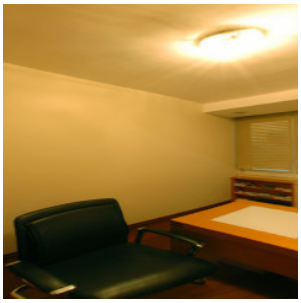
Figure 6.9: We show an example of a church scene decomposed into eight views in order to highlight both the explanatory power of our proposed approach and the utility of using information from multiple views for scene understanding and classification. For each view, we show the predicted scenarios.

View 1:



Scenario #1
- closet
- displaycase
- door

View 2:



Scenario #1
- blackboard_whiteboard
- desk
- remote
- television

Scenario #2
- computer
- desktop
- keyboard
- laptop
- monitor
- mouse
- mousepad
- printer_copier_scanner
- projector
- telephone

View 3:

Scenario #1
- chair
- window
- windowshade

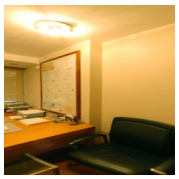
Scenario #2
- table

Scenario #3
- bulletinboard
- paper

Scenario #4
- blackboard_whiteboard
- desk
- remote
- television

Scenario #5
- computer
- desktop
- keyboard
- laptop
- monitor
- mouse
- mousepad
- printer_copier_scanner
- projector
- telephone

View 4:



Scenario #1
- chair
- window
- windowshade

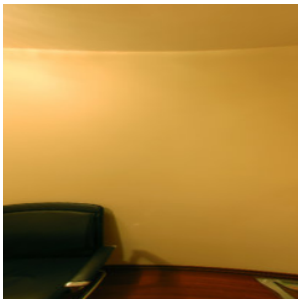
Scenario #2
- table___

Scenario #3
- bulletinboard
- paper

Scenario #4
- blackboard_whiteboard
- desk
- remote
- television

Scenario #5
- computer
- desktop
- keyboard
- laptop
- monitor
- mouse
- mousepad
- printer_copier_scanner
- projector
- telephone

View 5:



View 6:

Scenario #1

- hangingart_painting

Scenario #2

- closet
- displaycase
- door

Scenario #3

- cabinet
- coffeemaker
- dishrack
- drawer
- dresser
- exhausthood
- fridge
- kettle
- kitchenisland
- mediaplayer
- microwave
- oven
- sink
- stovetop
- toaster

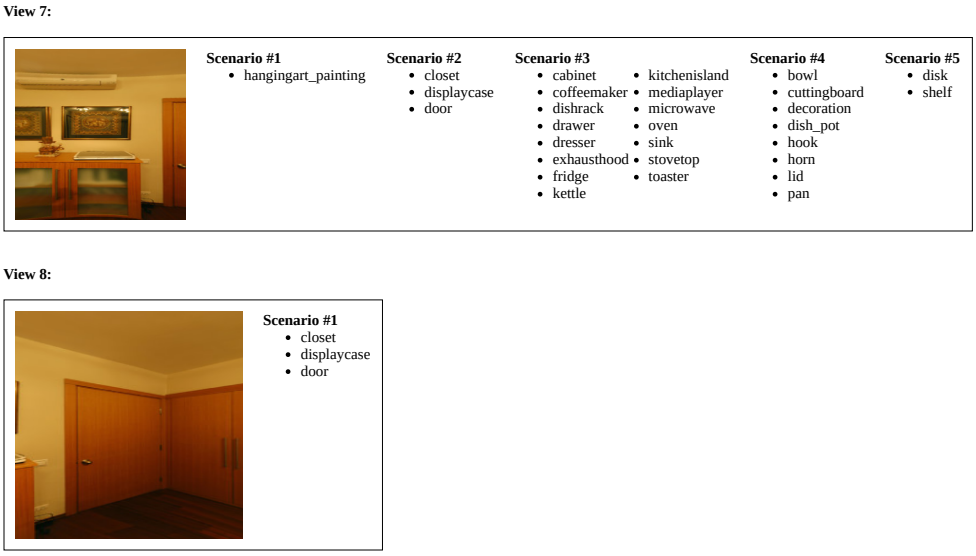


Figure 6.10: We show an example of an office scene decomposed into eight views in order to highlight both the explanatory power of our proposed approach and the utility of using information from multiple views for scene understanding and classification. For each view, we show the predicted scenarios. In this case, we see that our method is good but not perfect. The proposed model detects some scenarios that aren't present in the actual image, e.g., the computer scenario in view 2. Similarly, in views 6 and 7, we see that the model predicts a scenario related to kitchen appliances because it sees a cabinet, but most of the other concepts in the scenario are missing.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this thesis, we focused on general problems related to visual recognition, the task of automatically assigning some label to an image or video. Visual recognition problems often appear in safety-critical applications such as autonomous driving, human-machine teaming, and medical image analysis. The deep neural network [63], and specifically the convolutional neural network [62, 238] is a recent machine learning method that has revolutionized the field of computer vision and is now considered the defacto standard for solving visual recognition tasks. However, deep neural networks are typically treated as black-box models: models that take in data and output decisions without the user understanding the inner mechanisms of how the decisions were reached and without providing evidence to support predictions, i.e., why certain decisions were made. Black-box models are often incompatible with safety-critical applications such as those mentioned earlier. Safety-critical applications often require interpretable models and require the use of computational agents that are capable of understanding and reasoning about the high-level content of real-world scene images in order to make rational and grounded decisions that can be trusted by humans. As such, improving the interpretability of deep neural networks has emerged as a popular topic in the computer vision research community recently. Many of the more popular approaches for improving the interpretability of deep neural networks for visual recognition tasks take a given model that is already trained and try to explain the decision-making process of the model in a post-hoc (after-the-fact) manner. However, post-hoc explanations suffer from several problems. Namely, the post-hoc explanations are not always faithful to what the original model computes; sometimes the post-hoc explanations do not

make sense or provide enough detail to understand what the black box model is truly doing; and the post-hoc explanations are often overly complicated, leading to human error in subsequent decision making. Furthermore, many of the post-hoc explanation models for deep learning-based visual recognition models operate by highlighting which parts of an image are important when the model makes a specific decision and thus, highlight *where* the network is looking but not *why* the attended areas are important. To overcome the previously discussed weaknesses of existing explanation methods, we propose several novel alternative approaches for making convolutional neural networks more interpretable by utilizing explainability as a guiding principle when designing the model architecture. Specifically, our methods make explainability a core principle of the model by forcing the model to explicitly learn grounded and interpretable features.

The methods introduced in this thesis all rely on deep neural network that learn two components: 1) a mapping from visual input to a set of visual concepts and 2) a classification model mapping the visual concepts to class assignments. However, the visual concepts used in the introduced methods must satisfy two properties to maximize interpretability and trust; the set of visual attributes must be 1) able to be easily recognized from visual data given some dataset, and 2) discriminative for the given target task. As such, simply training a model to predict some known set of visual concepts and then using these visual concepts as features for the downstream classification task is insufficient. Instead, we must make non-trivial modifications to this pipeline.

In the first technical approach that we discussed, we demonstrated that when training a model to recognize a large set of visual concepts, there are often large subsets of visual concepts that can't be recognized from visual data with acceptable accuracy. The inability to recognize these visual concepts is due to several reasons. First, the object labels included in many datasets are often noisy and incomplete. Because these datasets are often created via crowdsourcing, labels can sometimes be redundant and ambiguous, e.g., if we use objects as our visual concepts, then many objects that are semantically equivalent might be labeled differently because humans have different names for the same objects. Sometimes labels are overly specific, e.g., if we use objects as our visual

concepts, an object might be labeled as a "wooden statue" while not being labeled using higher-level categories such as "statue" and "artwork". Humans also occasionally make mistakes, so they occasionally incorrectly label visual concepts that do not exist in a image and vice versa. Some mistakes are not due to human error but are instead due to limited training data. For example, the majority of objects (which can serve as our visual concepts) in the ADE20K dataset appear fewer than ten times, and thus, there is often not enough training data for the model to learn how to recognize these sparsely-appearing objects because deep neural networks are incredibly data-hungry. Similarly, some objects appear so small in the images (i.e., only hundreds of pixels in area), that deep neural networks cannot learn the fine-grained visual patterns necessary to accurately them. To overcome these issues, we propose expanding the set of visual concepts using an external knowledge graph relating visual concepts in a hierarchy, and then filtering out very rare concepts. By using a knowledge graph to expand and prune the visual concept set, some of the aforementioned issues can be addressed:

- Ambiguous concepts are merged.
- Concept information is captured at multiple levels of granularity.
- Ancestral categories appear more frequently than some of their children, so while there may not be enough training data to learn to recognize some visual concept, there might be enough training data to recognize one of its ancestors.

Thus, the proposed method for augmenting visual concepts is useful in many ways for reducing noise in the visual concept recognition stage of a semantically-grounded model. Furthermore, by exploiting the known structure of the visual concept hierarchy, we can improve the recognition of the individual visual concepts.

However, the method proposed in the first technical approach is not without its flaws. Predicting all of the visual concepts in the expanded set of visual concepts results in a very high-dimensional representation. The more information that a human has to consider for an explanation, the more the interpretability of the model is reduced. Likewise, many visual concepts express similar information, so the aforementioned representation often contains large amounts of redundant information. Finally, many of the

visual concepts are not discriminative for the downstream classification task, and thus, the model exerts unnecessary effort and computational resources trying to recognize concepts that are not necessary for our true task of interest.

To overcome these issues, we introduce the scenario representation, a novel low-dimensional, interpretable data-driven representation. Scenarios identify sets of frequently co-occurring visual concepts. We introduce the pseudo-Boolean matrix factorization to find these scenarios and propose a convolutional neural network architecture (the ScenarioNet) that bottlenecks through a scenario recognition layer in order to generate low-dimensional, human-understandable explanations for visual recognition tasks. As has been mentioned, many of the base visual concepts are often hard to recognize from visual data. In contrast, scenarios can be recognized from visual data with much higher accuracy; thus, decisions based on scenarios can be trusted with much higher confidence than those based on the base visual concepts. The scenario representation is also much lower-dimensional than a representation derived from predicting the entire base set of visual concepts, and the scenario representation has very little redundancy in the information it encodes compared to the representation consisting of predictions over the initial set of visual concepts. Thus, it is much easier for a human to evaluate decisions based on tens of scenarios compared to hundreds or thousands of the initial visual concepts. Similarly, by the formulation of ScenarioNet, in contrast to purely human-specified visual concepts, scenarios are naturally discriminative for the downstream target task.

A significant flaw exists with the scenario representation and ScenarioNet. In order to learn to identify and recognize scenarios, a human must hand-label the presence or absence of all of the relevant visual concepts from a human-curated pre-specified set of visual concepts for every training image. Often, acquiring such fine-grained annotations is a costly and time-consuming procedure, and thus, it is often not feasible to collect images paired with visual concept information. Second, we tested our methods on the relatively simple task of scene classification with relatively clean data. The third method we introduce attempts to circumvent this problem by extending the ScenarioNet model to extract meaningful and grounded representations from noisier auxiliary data

sources. Specifically, the auxiliary data we consider in our third approach is restricted to images paired with natural language text that describes the content of the image. This approach involves learning a convolutional neural network architecture simultaneously capable of 1) constructing a topic model, 2) predicting the presence or absence of each topic for a given image based on learned visual features, and 3) using an image's predicted topic encoding as features for a downstream classification task. Of course, this approach also has its flaws. Namely, we showed that grounding the decision-making process of the neural network to concepts hidden in the natural text resulted in models that have higher predictive power for the downstream classification task, but the topic extraction and recognition process was much noisier than the scenario extraction and recognition process, so the interpretability of the proposed model was much more limited than the original ScenarioNet.

In all of the previously discussed research directions, we assumed relatively simple classification settings. We assumed that the data consists of individual images where each image is very representative of its class. We also assumed all classes are known to the model a priori, and the model never needs to be updated to account for unexpected data. In our fourth research direction, we looked at a much harder problem domain: the active explanation-driven classification of indoor scenes. In this problem, an agent with an imaging sensor is placed in some environment, and based on its sensory input, the agent needs to assign a label to the perceived scene. The agent can adjust its sensor to capture more details about the scene, but there is a cost associated with manipulating the sensor. Ideally, the agent must understand the scene in an efficient manner. In order to understand the global state of a scene (i.e., the scene category), the agent must extract properties about the scene from multiple views and use these properties to generate human-understandable explanations about why it made specific predictions. If the agent encounters an unknown type of scene, it should reject assigning a label to the scene, request additional data about the new scene category from a human, and update its underlying knowledge base and machine learning models. To address this much more challenging problem, we extend the ScenarioNet architecture for use in setting requiring multi-sensor fusion and continuous learning. For each view encountered by the agent,

ScenarioNet is used to extract a human-understandable feature representation. As the agent encounters new views in the same scene, the predicted scenarios from each view are fused to obtain scene-level scenarios. A Weibull-calibrated support vector machine (W-SVM) [289] is then used to determine if a scene belongs to an unknown category and for assigning a class prediction. To facilitate the exploration process (i.e., determining whether to assign a label to a scene or change views), we formulate a Markov Decision Process and apply reinforcement learning.

In this thesis, we thoroughly discussed the importance of interpretable machine learning models for visual recognition tasks. We presented four technical approaches for grounding neural networks to human-understandable visual concept-based representations in order to improve the interpretability of the networks. We highlighted the strengths and weaknesses of each model, and while the models are far from perfect, we demonstrated their potential. In the following and final section of this thesis, we highlight some promising future directions of research for further improving the interpretability of deep neural networks.

7.2 Future Work

We now discuss several potential avenues of future research related to improving the interpretability of deep neural networks.

7.2.1 Expanding Scenarios Beyond Co-Occurrence Relations

Most of the methods introduced in this thesis are based on the scenario representation, which finds groups of visual concepts based on co-occurrence relations. However, many other relationships might lead to discovering more expressive yet semantically-meaningful representations. For example, even if we restrict our set of visual concepts to objects once again, there are many different types of relationships that might encode essential information. *Spatial* relationships are important because objects that are physically close to each other often are connected by some functionality, e.g., a computer mouse and keyboard are necessary for using a desktop computer, and so, they

will often be found together. Relative position between objects also provide subtle clues about how objects are used, e.g., a mouse is often to the right of a keyboard because most people are right-handed. We can also specifically consider *functional/interactional* relationships between objects. For example, the entire purpose of a baseball bat is to hit a baseball, so this "hitting" relationship is very important when attempting to deeply understand baseball bats, baseballs, the game of baseball, and the arrangement of baseball fields. We can also consider *temporal* relationships, or how objects move in relation to one another. As with spatial relationships, temporal relationships are useful for understanding how objects interact with one another and also for understanding the actions present in a scene. For example, the fact that darts move towards the dartboard and away from people tells us how darts are used, tells us that darts may be dangerous (because people avoid them), and tells us something about the functionality of the parent scene (e.g., sports bars are places people go to play darts). Finally, we can consider *causal* relationships, i.e., if a specific action is applied to an object, how will the object be affected? Once again, this helps inform us about how two objects might interact with, and causal relationships might also tell us something about how an object is supposed to be used. For other types of visual concepts, we might need to consider other types of relationships. As such, figuring out which relationships are important for various visual recognition problems and figuring out how to constrain neural networks (e.g., ScenarioNet) to make use of these relationships in a principled manner (especially for improving the interpretability of the neural network) are still very much important open research problems.

7.2.2 Outputting Richer, Easier-to-Interpret Explanations

The explanations output by our proposed methods are often very simple and sometimes too abstract for a human to understand. One interesting and important direction for future work is developing novel neural network architectures that can output explanations in forms that are very intuitive to humans. For example, can we train a neural network to output natural language explanations that we still ensure are grounded to

some known set of visual concepts? Similarly, we might be interested in developing neural networks that output graph-based explanations. For example, maybe future neural networks will be capable of outputting scene graphs and highlighting which nodes in the scene graph were most useful for making some prediction about a scene class.

7.2.3 Exploring Prototype-Based Approaches

As we have mentioned multiple times throughout this thesis, our methods often rely on predicting some set of visual concepts. The set of visual concepts must be defined by humans, and subsequently, large amounts of training must be collected via a time- and resource-intensive process in order to learn recognition models that can accurately recover these visual concepts from visual data. Even in the models introduced in this thesis, there is still a large amount of noise in the concept predictions, leading to models that cannot fully be trusted. These issues lead us to ask whether there are other ways to ground neural networks to some human-understandable representation that are more robust to noise and require less annotation effort by humans. One promising direction is incorporating prototype learning with deep neural networks [14, 15]. In this case, instead of grounding the decision-making process of deep neural networks to visual concepts, the network is grounded to other images (or parts of images) in the training set.

7.2.4 Utilizing Interpretable Non-Linear Classifiers

Lastly, in all of our work we've paired some semantically-meaningful and human-understandable representation with a linear classifer. We chose to use a linear classifier because it is among the simplest machine learning models to interpret, and it generally is trivially easy to implement in neural networks. However, there are many other machine learning models that are similarly interpretable while potentially having more predictive power because they can express more complex (i.e., non-linear) decision boundaries. Examples of such methods include decision trees, decision sets, and nearest neighbor classifiers, among others. However, it is difficult to integrate most of these methods with deep neural networks. One very interesting non-linear, but potentially

interpretable classifier that is compatible with deep neural networks are deep neural decision trees and forests [336, 337]. In these models, the linear classifier is replaced with a tree-structured classifier, and the routing probabilities for how information flows through the tree-model to arrive at a decision are learned parameterized functions of the model. This interesting structure results in models that learn fascinating decision boundaries while maintaining some possibility of interpretability.

References

[1] Z. A. Daniels, L. D. Frank, C. J. Menart, M. Raymer, and P. Hitzler, "A framework for explainable deep neural models using external knowledge graphs," *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, vol. 11413, p. 1141338, 2020.

[2] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*. IEEE, 2016, pp. 2921–2929.

[3] Z. A. Daniels and D. Metaxas, "Scenarionet: An interpretable data-driven model for scene understanding," *XAI 2018*, p. 33, 2018.

[4] Z. Daniels and D. Metaxas, "Exploiting visual and report-based information for chest x-ray analysis by jointly learning visual classifiers and topic models," in *IEEE International Symposium on Biomedical Imaging (ISBI)*, 2019.

[5] Z. C. Lipton, "The mythos of model interpretability," *ICML Workshop on Human Interpretability in Machine Learning*, 2016.

[6] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you?: Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 1135–1144.

[7] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Is object localization for free?-weakly-supervised learning with convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 685–694.

[8] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization." in *ICCV*, 2017, pp. 618–626.

[9] L. A. Hendricks, Z. Akata, M. Rohrbach, J. Donahue, B. Schiele, and T. Darrell, "Generating visual explanations," in *European Conference on Computer Vision*. Springer, 2016, pp. 3–19.

[10] D. H. Park, L. A. Hendricks, Z. Akata, B. Schiele, T. Darrell, and M. Rohrbach, "Attentive explanations: Justifying decisions and pointing to the evidence," *arXiv preprint arXiv:1612.04757*, 2016.

[11] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019.

[12] M. K. Sarker, N. Xie, D. , M. Raymer, and P. Hitzler, "Explaining trained neural networks with semantic web technologies: First steps," *International Workshop on Neural-Symbolic Learning and Reasoning*, 2017.

[13] L. Anne Hendricks, R. Hu, T. Darrell, and Z. Akata, "Grounding visual explanations," *European Conference on Computer Vision*, pp. 264–279, 2018.

[14] O. Li, H. Liu, C. Chen, and C. Rudin, "Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions," *AAAI Conference on Artificial Intelligence*, 2018.

[15] C. Chen, O. Li, D. Tao, A. Barnett, C. Rudin, and J. K. Su, "This looks like that: deep learning for interpretable image recognition," in *NeurIPS*, 2019, pp. 8928–8939.

[16] Z. A. Daniels and D. N. Metaxas, "Exploiting visual and report-based information for chest x-ray analysis by jointly learning visual classifiers and topic models," *IEEE ISBI*, pp. 1270–1274, 2019.

[17] R. S. Feris, C. Lampert, and D. Parikh, *Visual Attributes*. Springer, 2017.

[18] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth, "Describing objects by their attributes," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 1778–1785.

[19] V. Ferrari and A. Zisserman, "Learning visual attributes," in *Advances in Neural Information Processing Systems*, 2008, pp. 433–440.

[20] C. H. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 951–958.

[21] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ade20k dataset," *CVPR*, pp. 633–641, 2017.

[22] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[23] G. Montavon, W. Samek, and K.-R. Müller, "Methods for interpreting and understanding deep neural networks," *Digital Signal Processing*, vol. 73, pp. 1–15, 2018.

[24] C. Seifert, A. Aamir, A. Balagopalan, D. Jain, A. Sharma, S. Grottel, and S. Gumhold, "Visualizations of deep neural networks in computer vision: A survey," in *Transparent Data Mining for Big and Small Data*. Springer, 2017, pp. 123–144.

[25] Z. Qin, F. Yu, C. Liu, and X. Chen, "How convolutional neural network see the world-a survey of convolutional neural network visualization methods," *arXiv preprint arXiv:1804.11191*, 2018.

[26] I. Chalkiadakis, "A brief survey of visualization methods for deep learning models from the perspective of explainable ai," 2018.

[27] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM computing surveys (CSUR)*, vol. 51, no. 5, pp. 1–42, 2018.

[28] A. Nguyen, J. Yosinski, and J. Clune, "Understanding neural networks via feature visualization: A survey," in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 2019, pp. 55–76.

[29] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," in *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*. IEEE, 2018, pp. 80–89.

[30] C. Molnar, "A guide for making black box models explainable," *URL: https://christophm. github. io/interpretable-ml-book*, 2018.

[31] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artificial Intelligence*, vol. 267, pp. 1–38, 2019.

[32] T. Miller, P. Howe, and L. Sonenberg, "Explainable ai: Beware of inmates running the asylum or: How i learnt to stop worrying and love the social and behavioural sciences," *arXiv preprint arXiv:1712.00547*, 2017.

[33] Z. C. Lipton, "The mythos of model interpretability," *Queue*, vol. 16, no. 3, pp. 31–57, 2018.

[34] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *International Conference on Learning Representations (ICLR)*, 2015.

[35] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 582–597.

[36] D. Meng and H. Chen, "Magnet: a two-pronged defense against adversarial examples," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 135–147.

[37] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-gan: Protecting classifiers against adversarial attacks using generative models," *arXiv preprint arXiv:1805.06605*, 2018.

[38] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *International Conference on Learning Representations (ICLR)*, 2018.

[39] K. Kärkkäinen and J. Joo, "Fairface: Face attribute dataset for balanced race, gender, and age," *arXiv preprint arXiv:1908.04913*, 2019.

[40] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017.

[41] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu, "Interpretable machine learning: definitions, methods, and applications," *arXiv preprint arXiv:1901.04592*, 2019.

[42] D. Doran, S. Schulz, and T. R. Besold, "What does explainable ai really mean? a new conceptualization of perspectives," *arXiv preprint arXiv:1710.00794*, 2017.

[43] G. Ras, M. van Gerven, and P. Haselager, "Explanation methods in deep learning: Users, values, concerns and challenges," in *Explainable and Interpretable Models in Computer Vision and Machine Learning*. Springer, 2018, pp. 19–36.

[44] G. Shmueli *et al.*, "To explain or to predict?" *Statistical science*, vol. 25, no. 3, pp. 289–310, 2010.

[45] B. Herman, "The promise and peril of human evaluation for model interpretability," *arXiv preprint arXiv:1711.07414*, 2017.

[46] M. Ashoori and J. D. Weisz, "In ai we trust? factors that influence trustworthiness of ai-infused decision-making processes," *arXiv preprint arXiv:1912.02675*, 2019.

[47] P. Hall, *On Explainable Machine Learning Misconceptions: A More Human-Centered Machine Learning*, Jul 2019. [Online]. Available: https://github.com/jphall663/xai_misconceptions/blob/master/xai_misconceptions.pdf

[48] P. Hall and N. Gill, *Introduction to Machine Learning Interpretability*. O'Reilly Media, Incorporated, 2018.

[49] P. J. Lisboa, "Interpretability in machine learning–principles and practice," in *International Workshop on Fuzzy Logic and Applications*. Springer, 2013, pp. 15–21.

[50] A. Bibal and B. Frénay, "Interpretability of machine learning models and representations: an introduction." in *ESANN*, 2016.

[51] W. Samek, T. Wiegand, and K.-R. Müller, "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models," *arXiv preprint arXiv:1708.08296*, 2017.

[52] A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (xai)," *IEEE Access*, vol. 6, pp. 52 138–52 160, 2018.

[53] F. K. Došilović, M. Brčić, and N. Hlupić, "Explainable artificial intelligence: A survey," in *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*. IEEE, 2018, pp. 0210–0215.

[54] D. Gunning and D. W. Aha, "Darpa's explainable artificial intelligence program," *AI Magazine*, vol. 40, no. 2, pp. 44–58, 2019.

[55] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information Fusion*, vol. 58, pp. 82–115, 2020.

[56] W. Samek and K.-R. Müller, "Towards explainable artificial intelligence," in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 2019, pp. 5–22.

[57] E. Tjoa and C. Guan, "A survey on explainable artificial intelligence (xai): Towards medical xai," *arXiv preprint arXiv:1907.07374*, 2019.

[58] T. Miller, "" but why?" understanding explainable artificial intelligence," *XRDS: Crossroads, The ACM Magazine for Students*, vol. 25, no. 3, pp. 20–25, 2019.

[59] F. Emmert-Streib, O. Yli-Harja, and M. Dehmer, "Explainable artificial intelligence and machine learning: A reality rooted perspective," *arXiv preprint arXiv:2001.09464*, 2020.

[60] A. Kirsch, "Explain to whom? putting the user in the center of explainable ai," 2017.

[61] N. Xie, G. Ras, M. van Gerven, and D. Doran, "Explainable deep learning: A field guide for the uninitiated," *arXiv preprint arXiv:2004.14545*, 2020.

[62] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[63] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[64] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[65] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," *arXiv preprint arXiv:1506.06579*, 2015.

[66] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[67] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 2018–2025.

[68] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *European Conference on Computer Vision*, pp. 818–833, 2014.

[69] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," *University of Montreal*, vol. 1341, no. 3, p. 1, 2009.

[70] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[71] A. Mahendran and A. Vedaldi, "Visualizing deep convolutional neural networks using natural pre-images," *International Journal of Computer Vision*, vol. 120, no. 3, pp. 233–255, 2016.

[72] C. Olah, A. Mordvintsev, and L. Schubert, "Feature visualization," *Distill*, vol. 2, no. 11, p. e7, 2017.

[73] A. Nguyen, J. Yosinski, and J. Clune, "Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks," *arXiv preprint arXiv:1602.03616*, 2016.

[74] S. A. Cadena, M. A. Weis, L. A. Gatys, M. Bethge, and A. S. Ecker, "Diverse feature visualizations reveal invariances in early layers of deep neural networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 217–232.

[75] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[76] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," in *Advances in neural information processing systems*, 2016, pp. 3387–3395.

[77] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski, "Plug & play generative networks: Conditional iterative generation of images in latent space," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4467–4477.

[78] M. Godi, M. Carletti, M. Aghaei, F. Giuliari, and M. Cristani, "Understanding deep architectures by visual summaries," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2018.

[79] A. Mordvintsev, C. Olah, and M. Tyka, "Inceptionism: Going deeper into neural networks," 2015.

[80] C. Szegedy, W. Zaremba, I. Sutskever, J. B. Estrach, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *2nd International Conference on Learning Representations, ICLR 2014*, 2014.

[81] L. Engstrom, D. Tsipras, L. Schmidt, and A. Madry, "A rotation and a translation suffice: Fooling cnns with simple transformations," *arXiv preprint arXiv:1712.02779*, vol. 1, no. 2, p. 3, 2017.

[82] R. Fong, M. Patrick, and A. Vedaldi, "Understanding deep networks via extremal perturbations and smooth masks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 2950–2958.

[83] A. Azulay and Y. Weiss, "Why do deep convolutional networks generalize so poorly to small image transformations?" *Journal of Machine Learning Research*, vol. 20, pp. 1–25, 2019.

[84] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5188–5196.

[85] A. Dosovitskiy and T. Brox, "Inverting visual representations with convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4829–4837.

[86] E. Wong and J. Z. Kolter, "Neural network inversion beyond gradient descent."

[87] B.-L. Lu, H. Kita, and Y. Nishikawa, "Inverting feedforward neural networks using linear and nonlinear programming," *IEEE Transactions on Neural networks*, vol. 10, no. 6, pp. 1271–1290, 1999.

[88] C. A. Jensen, R. D. Reed, R. J. Marks, M. A. El-Sharkawi, J.-B. Jung, R. T. Miyamoto, G. M. Anderson, and C. J. Eggen, "Inversion of feedforward neural networks: algorithms and applications," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1536–1549, 1999.

[89] D. Wei, B. Zhou, A. Torrabla, and W. Freeman, "Understanding intra-class knowledge inside cnn," *arXiv preprint arXiv:1507.02379*, 2015.

[90] A. Zhmoginov and M. Sandler, "Inverting face embeddings with convolutional neural networks," *arXiv preprint arXiv:1606.04189*, 2016.

[91] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Object detectors emerge in deep scene cnns," *arXiv preprint arXiv:1412.6856*, 2014.

[92] A. Gonzalez-Garcia, D. Modolo, and V. Ferrari, "Do semantic parts emerge in convolutional neural networks?" *International Journal of Computer Vision*, vol. 126, no. 5, pp. 476–494, 2018.

[93] G. Alain and Y. Bengio, "Understanding intermediate layers using linear classifier probes," *arXiv preprint arXiv:1610.01644*, 2016.

[94] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba, "Network dissection: Quantifying interpretability of deep visual representations," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6541–6549.

[95] B. Zhou, D. Bau, A. Oliva, and A. Torralba, "Interpreting deep visual representations via network dissection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 9, pp. 2131–2145, 2018.

[96] B. Zhou, D. Bau, A. Oliva, and A. Torralba, "Interpreting visual representations of neural networks via network dissection," *Journal of Vision*, vol. 18, no. 10, p. 1244, 2018.

[97] B. Zhou, D. Bau, A. Oliva, and A. Torralba, "Comparing the interpretability of deep networks via network dissection," in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 2019, pp. 243–252.

[98] D. Bau, J.-Y. Zhu, H. Strobelt, B. Zhou, J. B. Tenenbaum, W. T. Freeman, and A. Torralba, "Visualizing and understanding generative adversarial networks," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=Hyg_X2C5FX

[99] R. Fong and A. Vedaldi, "Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8730–8738.

[100] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas *et al.*, "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav)," *International Conference on Machine Learning*, pp. 2673–2682, 2018.

[101] A. Ghorbani, J. Wexler, J. Y. Zou, and B. Kim, "Towards automatic concept-based explanations," in *Advances in Neural Information Processing Systems*, 2019, pp. 9273–9282.

[102] M. Raghu, J. Gilmer, J. Yosinski, and J. Sohl-Dickstein, "Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability," in *Advances in Neural Information Processing Systems*, 2017, pp. 6076–6085.

[103] U. S. Saini and E. E. Papalexakis, "A peek into the hidden layers of a convolutional neural network through a factorization lens," *arXiv preprint arXiv:1806.02012*, 2018.

[104] K. Dhamdhere, M. Sundararajan, and Q. Yan, "How important is a neuron," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=SylKoo0cKm

[105] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3319–3328.

[106] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 535–541.

[107] J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *Advances in neural information processing systems*, 2014, pp. 2654–2662.

[108] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[109] N. Frosst and G. Hinton, "Distilling a neural network into a soft decision tree," *arXiv preprint arXiv:1711.09784*, 2017.

[110] Q. Zhang, Y. Yang, H. Ma, and Y. N. Wu, "Interpreting cnns via decision trees," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6261–6270.

[111] X. Liu, X. Wang, and S. Matwin, "Improving the interpretability of deep neural networks with knowledge distillation," in *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2018, pp. 905–912.

[112] S. Tan, R. Caruana, G. Hooker, and Y. Lou, "Distill-and-compare: Auditing black-box models using transparent model distillation," in *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, 2018, pp. 303–310.

[113] S. Tan, R. Caruana, G. Hooker, and Y. Lou, "Detecting bias in black-box models using transparent model distillation," *arXiv preprint arXiv:1710.06169*, 2017.

[114] F. Grün, C. Rupprecht, N. Navab, and F. Tombari, "A taxonomy and library for visualizing learned features in convolutional neural networks," *arXiv preprint arXiv:1606.07757*, 2016.

[115] M. Oquab, L. Bottou, I. Laptev, J. Sivic *et al.*, "Weakly supervised object recognition with convolutional neural networks," in *Proc. of NIPS*, 2014, pp. 1545–5963.

[116] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," *Computer Vision and Pattern Recognition*, pp. 1717–1724, 2014.

[117] L. M. Zintgraf, T. S. Cohen, T. Adel, and M. Welling, "Visualizing deep neural network decisions: Prediction difference analysis," 2017.

[118] V. Petsiuk, A. Das, and K. Saenko, "Rise: Randomized input sampling for explanation of black-box models," *British Machine Vision Conference*, 2018.

[119] B. Vasu and C. Long, "Iterative and adaptive sampling with spatial attention for black-box model explanations," in *The IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 2960–2969.

[120] R. C. Fong and A. Vedaldi, "Interpretable explanations of black boxes by meaningful perturbation," *arXiv preprint arXiv:1704.03296*, 2017.

[121] C. Agarwal, D. Schonfeld, and A. Nguyen, "Removing input features via a generative model to explain their attributions to classifier's decisions," *arXiv preprint arXiv:1910.04256*, 2019.

[122] R. Fong, M. Patrick, and A. Vedaldi, "Understanding deep networks via extremal perturbations and smooth masks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 2950–2958.

[123] C.-H. Chang, E. Creager, A. Goldenberg, and D. Duvenaud, "Explaining image classifiers by counterfactual generation," *arXiv preprint arXiv:1807.08024*, 2018.

[124] J. Wagner, J. M. Kohler, T. Gindele, L. Hetzel, J. T. Wiedemer, and S. Behnke, "Interpretable and fine-grained visual explanations for convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9097–9107.

[125] Z. Qi, S. Khorram, and F. Li, "Visualizing deep networks by optimizing with integrated gradients," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 1–4.

[126] K. Schulz, L. Sixt, F. Tombari, and T. Landgraf, "Restricting the flow: Information bottlenecks for attribution," in *International Conference on Learning Representations*, 2019.

[127] A. Khakzar, S. Baselizadeh, S. Khanduja, S. T. Kim, and N. Navab, "Improving feature attribution through input-specific network pruning," *arXiv preprint arXiv:1911.11081*, 2019.

[128] A. Chattopadhay, A. Sarkar, P. Howlader, and V. N. Balasubramanian, "Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks," *IEEE Winter Conference on Applications of Computer Vision*, pp. 839–847, 2018.

[129] D. Omeiza, S. Speakman, C. Cintas, and K. Weldermariam, "Smooth grad-cam++: An enhanced inference level visualization technique for deep convolutional neural network models," *arXiv preprint arXiv:1908.01224*, 2019.

[130] L. Chen, J. Chen, H. Hajimirsadeghi, and G. Mori, "Adapting grad-cam for embedding networks," in *The IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 2794–2803.

[131] K. Fu, W. Dai, Y. Zhang, Z. Wang, M. Yan, and X. Sun, "Multicam: Multiple class activation mapping for aircraft recognition in remote sensing images," *Remote Sensing*, vol. 11, no. 5, p. 544, 2019.

[132] K. Gao, H. Shen, Y. Liu, L. Zeng, and D. Hu, "Dense-cam: Visualize the gender of brains with mri images," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–7.

[133] B. J. Kim, G. Koo, H. Choi, and S. W. Kim, "Extending class activation mapping using gaussian receptive field," *arXiv preprint arXiv:2001.05153*, 2020.

[134] H. Wang, M. Du, F. Yang, and Z. Zhang, "Score-cam: Improved visual explanations via score-weighted class activation mapping," *arXiv preprint arXiv:1910.01279*, 2019.

[135] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, "Residual attention network for image classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3156–3164.

[136] S. Jetley, N. A. Lord, N. Lee, and P. Torr, "Learn to pay attention," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=HyzbhfWRW

[137] L. Wang, Z. Wu, S. Karanam, K.-C. Peng, R. V. Singh, B. Liu, and D. N. Metaxas, "Sharpen focus: Learning with attention separability and consistency," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 512–521.

[138] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PloS one*, vol. 10, no. 7, 2015.

[139] G. Montavon, A. Binder, S. Lapuschkin, W. Samek, and K.-R. Müller, "Layer-wise relevance propagation: an overview," in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 2019, pp. 193–209.

[140] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller, "Explaining nonlinear classification decisions with deep taylor decomposition," *Pattern Recognition*, vol. 65, pp. 211–222, 2017.

[141] P.-J. Kindermans, K. T. Schütt, M. Alber, K.-R. Müller, D. Erhan, B. Kim, and S. Dähne, "Learning how to explain neural networks: Patternnet and patternattribution," *arXiv preprint arXiv:1705.05598*, 2017.

[142] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *International Conference on Machine Learning*, 2017, pp. 3145–3153.

[143] J. Zhang, S. A. Bargal, Z. Lin, J. Brandt, X. Shen, and S. Sclaroff, "Top-down neural attention by excitation backprop," *International Journal of Computer Vision*, vol. 126, no. 10, pp. 1084–1102, 2018.

[144] M. T. Ribeiro, S. Singh, and C. Guestrin, "Anchors: High-precision model-agnostic explanations," in *AAAI Conference on Artificial Intelligence*, 2018. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16982

[145] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, 2017, pp. 4765–4774.

[146] C. Burns, J. Thomason, and W. Tansey, "Interpreting black box models via hypothesis testing," *arXiv preprint arXiv:1904.00045*, 2019.

[147] J. Oramas, K. Wang, and T. Tuytelaars, "Visual explanation by interpretation: Improving visual feedback capabilities of deep neural networks," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=H1ziPjC5Fm

[148] D. Seo, K. Oh, and I.-S. Oh, "Regional multi-scale approach for visually pleasing explanations of deep neural networks (december 2019)," *IEEE Access*, 2019.

[149] P.-J. Kindermans, S. Hooker, J. Adebayo, M. Alber, K. T. Schütt, S. Dähne, D. Erhan, and B. Kim, "The (un) reliability of saliency methods," in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 2019, pp. 267–280.

[150] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim, "Sanity checks for saliency maps," in *Advances in Neural Information Processing Systems*, 2018, pp. 9505–9515.

[151] W. Nie, Y. Zhang, and A. Patel, "A theoretical explanation for perplexing behaviors of backpropagation-based visualizations," in *International Conference on Machine Learning*, 2018, pp. 3809–3818.

[152] Y. Goyal, Z. Wu, J. Ernst, D. Batra, D. Parikh, and S. Lee, "Counterfactual visual explanations," in *International Conference on Machine Learning*, 2019, pp. 2376–2384.

[153] S. Liu, B. Kailkhura, D. Loveland, and Y. Han, "Generative counterfactual introspection for explainable deep learning," *arXiv preprint arXiv:1907.03077*, 2019.

[154] L. A. Hendricks, R. Hu, T. Darrell, and Z. Akata, "Generating counterfactual explanations with natural language," in *ICML Workshop on Human Interpretability in Machine Learning*, 2018, pp. 95–98.

[155] X. Zhang, A. Solar-Lezama, and R. Singh, "Interpreting neural network judgments via minimal, stable, and symbolic corrections," in *Advances in Neural Information Processing Systems*, 2018, pp. 4874–4885.

[156] R. K. Mothilal, A. Sharma, and C. Tan, "Explaining machine learning classifiers through diverse counterfactual explanations," in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 2020, pp. 607–617.

[157] M. Pazzani, A. Feghahati, C. Shelton, and A. Seitz, "Explaining contrasting categories." in *IUI Workshops*, 2018.

[158] A. Feghahati, C. R. Shelton, M. J. Pazzani, and K. Tang, "CDeepex: Contrastive deep explanations," 2019. [Online]. Available: https://openreview.net/forum?id=HyNmRiCqtm

[159] S. Rathi, "Generating counterfactual and contrastive explanations using shap," *arXiv preprint arXiv:1906.09293*, 2019.

[160] A. Dhurandhar, P.-Y. Chen, R. Luss, C.-C. Tu, P. Ting, K. Shanmugam, and P. Das, "Explanations based on the missing: Towards contrastive explanations with pertinent negatives," in *Advances in neural information processing systems*, 2018, pp. 592–603.

[161] R. Luss, P.-Y. Chen, A. Dhurandhar, P. Sattigeri, K. Shanmugam, and C.-C. Tu, "Generating contrastive explanations with monotonic attribute functions," *arXiv preprint arXiv:1905.12698*, 2019.

[162] B. Zhou, Y. Sun, D. Bau, and A. Torralba, "Interpretable basis decomposition for visual explanation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 119–134.

[163] N. Xie, M. K. Sarker, D. Doran, P. Hitzler, and M. Raymer, "Relating input concepts to convolutional neural network decisions," *NIPS Workshop on Interpreting, Explaining, and Visualizing Deep Learning*, 2017.

[164] Z. Qi and F. Li, "Learning explainable embeddings for deep networks," in *NIPS Workshop on Interpreting, Explaining and Visualizing Deep Learning. Long Beach, CA, December*, vol. 9, 2017.

[165] C. H. Lampert, H. Nickisch, and S. Harmeling, "Attribute-based classification for zero-shot visual object categorization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 3, pp. 453–465, 2014.

[166] D. Parikh and K. Grauman, "Relative attributes," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 503–510.

[167] G. Patterson and J. Hays, "Coco attributes: Attributes for people, animals, and objects," in *European Conference on Computer Vision*. Springer, 2016, pp. 85–100.

[168] G. Patterson and J. Hays, "Sun attribute database: Discovering, annotating, and recognizing scene attributes," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2751–2758.

[169] S. Wang, J. Joo, Y. Wang, and S.-C. Zhu, "Weakly supervised learning for attribute localization in outdoor scenes," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE, 2013, pp. 3111–3118.

[170] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," California Institute of Technology, Tech. Rep. CNS-TR-2011-001, 2011.

[171] H. Chen, A. Gallagher, and B. Girod, "Describing clothing by semantic attributes," in *European conference on computer vision*. Springer, 2012, pp. 609–623.

[172] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3730–3738.

[173] S. Branson, C. Wah, F. Schroff, B. Babenko, P. Welinder, P. Perona, and S. Belongie, "Visual recognition with humans in the loop," in *European Conference on Computer Vision*. Springer, 2010, pp. 438–451.

[174] M. Palatucci, D. Pomerleau, G. E. Hinton, and T. M. Mitchell, "Zero-shot learning with semantic output codes," in *Advances in neural information processing systems*, 2009, pp. 1410–1418.

[175] A. Kovashka, D. Parikh, and K. Grauman, "Whittlesearch: Interactive image search with relative attribute feedback," *International Journal of Computer Vision*, vol. 115, no. 2, pp. 185–210, 2015.

[176] A. Kovashka, S. Vijayanarasimhan, and K. Grauman, "Actively selecting annotations among objects and attributes," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1403–1410.

[177] P.-Y. Laffont, Z. Ren, X. Tao, C. Qian, and J. Hays, "Transient attributes for high-level understanding and editing of outdoor scenes," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, p. 149, 2014.

[178] X. Yan, J. Yang, K. Sohn, and H. Lee, "Attribute2image: Conditional image generation from visual attributes," in *European Conference on Computer Vision*. Springer, 2016, pp. 776–791.

[179] Z. Al-Halah and R. Stiefelhagen, "Automatic discovery, association estimation and learning of semantic attributes for a thousand categories," in *CVPR*, 2017.

[180] T. L. Berg, A. C. Berg, and J. Shih, "Automatic attribute discovery and characterization from noisy web data," in *European Conference on Computer Vision*. Springer, 2010, pp. 663–676.

[181] M. Rastegari, A. Farhadi, and D. Forsyth, "Attribute discovery via predictable discriminative binary codes," in *European Conference on Computer Vision*. Springer, 2012, pp. 876–889.

[182] Q. Zhang, Y. Yang, Y. Liu, Y. N. Wu, and S.-C. Zhu, "Unsupervised learning of neural networks to explain neural networks," *arXiv preprint arXiv:1805.07468*, 2018.

[183] D. A. Melis and T. Jaakkola, "Towards robust interpretability with self-explaining neural networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 7775–7784.

[184] F. F. Li, J. Johnson, and S. Yeung, "Cs 231n: Visualizing and understanding," University Lecture, 2019.

[185] S. Lazebnik, "University of illinois' cs 498: Visualizing and explaining neural networks," University Lecture, 2019.

[186] M. Łopuszyński, "Awesome interpretable machine learning." [Online]. Available: https://github.com/lopusz/awesome-interpretable-machine-learning

[187] Y. Dong, "Robust-and-explainable-machine-learning." [Online]. Available: https://github.com/dongyp13/Robust-and-Explainable-Machine-Learning

[188] A. Nguyen, "Papers on explainable artificial intelligence." [Online]. Available: https://github.com/anguyen8/XAI-papers

[189] "Sig xai: Special interest group on explainable ai a special interest group within the association neural-symbolic learning and reasoning (nesy)." [Online]. Available: http://people.cs.ksu.edu/~hitzler/nesy/sig-xai/

[190] A. Karpathy, "Convnetjs: Deep learning in your browser," 2020.

[191] U. Ozbulak, "Pytorch cnn visualizations," https://github.com/utkuozbulak/pytorch-cnn-visualizations, 2019.

[192] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *International Conference on Learning Representations*, 2014. [Online]. Available: http://arxiv.org/abs/1312.6199

[193] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "Imagenet-trained cnns are biased towards texture: Increasing shape bias improves accuracy and robustness," *International Conference on Learning Representations*, 2018.

[194] K. Marino, R. Salakhutdinov, and A. Gupta, "The more you know: Using knowledge graphs for image classification," *Computer Vision and Pattern Recognition*, pp. 2673–2681, 2017.

[195] W. Goo, J. Kim, G. Kim, and S. J. Hwang, "Taxonomy-regularized semantic deep convolutional neural networks," *European Conference on Computer Vision*, pp. 86–101, 2016.

[196] Y. Guo, Y. Liu, E. M. Bakker, Y. Guo, and M. S. Lew, "Cnn-rnn: a large-scale hierarchical image classification framework," *Multimedia Tools and Applications*, vol. 77, no. 8, pp. 10 251–10 271, 2018.

[197] Z. Yan, H. Zhang, R. Piramuthu, V. Jagadeesh, D. DeCoste, W. Di, and Y. Yu, "Hd-cnn: hierarchical deep convolutional neural networks for large scale visual recognition," *International Conference on Computer Vision*, pp. 2740–2748, 2015.

[198] N. Srivastava and R. R. Salakhutdinov, "Discriminative transfer learning with tree-based priors," *Advances in Neural Information Processing Systems*, pp. 2094–2102, 2013.

[199] J. Fan, T. Zhao, Z. Kuang, Y. Zheng, J. Zhang, J. Yu, and J. Peng, "Hd-mtl: Hierarchical deep multi-task learning for large-scale visual recognition," *IEEE Transactions on Image Processing*, vol. 26, no. 4, pp. 1923–1938, 2017.

[200] Z. Kuang, J. Yu, Z. Li, B. Zhang, and J. Fan, "Integrating multi-level deep learning and concept ontology for large-scale visual recognition," *Pattern Recognition*, vol. 78, pp. 198–214, 2018.

[201] J. Zhang, K. Mei, Y. Zheng, and J. Fan, "Learning multi-layer coarse-to-fine representations for large-scale image classification," *Pattern Recognition*, vol. 91, pp. 175–189, 2019.

[202] D. Roy, P. Panda, and K. Roy, "Tree-cnn: A hierarchical deep convolutional neural network for incremental learning," *arXiv preprint arXiv:1802.05800*, 2018.

[203] J. Deng, N. Ding, Y. Jia, A. Frome, K. Murphy, S. Bengio, Y. Li, H. Neven, and H. Adam, "Large-scale object classification using label relation graphs," *European Conference on Computer Vision*, pp. 48–64, 2014.

[204] W. Ge, "Deep metric learning with hierarchical triplet loss," *European Conference on Computer Vision*, pp. 269–285, 2018.

[205] Z. Zhang and V. Saligrama, "Zero-shot learning via semantic similarity embedding," *International Conference on Computer Vision*, pp. 4166–4174, 2015.

[206] I. Donadello, L. Serafini, and A. D. Garcez, "Logic tensor networks for semantic image interpretation," *International Joint Conference on Artificial Intelligence*, pp. 1596–1602, 2017.

[207] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei, "Scene graph generation by iterative message passing," *Computer Vision and Pattern Recognition*, pp. 5410–5419, 2017.

[208] Y. Li, W. Ouyang, B. Zhou, K. Wang, and X. Wang, "Scene graph generation from objects, phrases and region captions," *International Conference on Computer Vision*, pp. 1261–1270, 2017.

[209] C. Lu, R. Krishna, M. Bernstein, and L. Fei-Fei, "Visual relationship detection with language priors," *European Conference on Computer Vision*, pp. 852–869, 2016.

[210] Y. Li, W. Ouyang, X. Wang, and X. Tang, "Vip-cnn: Visual phrase guided convolutional neural network," *Computer Vision and Pattern Recognition*, pp. 1347–1356, 2017.

[211] H. Zhang, Z. Kyaw, J. Yu, and S.-F. Chang, "Ppr-fcn: Weakly supervised visual relation detection via parallel pairwise r-fcn," *International Conference on Computer Vision*, pp. 4233–4241, 2017.

[212] S. Woo, D. Kim, D. Cho, and I. S. Kweon, "Linknet: Relational embedding for scene graph," *Advances in Neural Information Processing Systems*, pp. 560–570, 2018.

[213] E. Belilovsky, M. Blaschko, J. Kiros, R. Urtasun, and R. Zemel, "Joint embeddings of scene graphs and images," *ICLR Workshops*, 2017.

[214] J. Johnson, B. Hariharan, L. van der Maaten, J. Hoffman, L. Fei-Fei, C. Lawrence Zitnick, and R. Girshick, "Inferring and executing programs for visual reasoning," *International Conference on Computer Vision*, pp. 2989–2998, 2017.

[215] K. Yi, J. Wu, C. Gan, A. Torralba, P. Kohli, and J. Tenenbaum, "Neural-symbolic vqa: Disentangling reasoning from vision and language understanding," in *Advances in Neural Information Processing Systems*, 2018, pp. 1039–1050.

[216] S. Aditya, Y. Yang, and C. Baral, "Explicit reasoning over end-to-end neural architectures for visual question answering," *AAAI Conference on Artificial Intelligence*, 2018.

[217] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick, "Clevr: A diagnostic dataset for compositional language and elementary visual reasoning," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 2017, pp. 1988–1997.

[218] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," *Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[219] I. Kadar and O. Ben-Shahar, "Scenenet: A perceptual ontology for scene understanding," *Workshop on Computer vision + ONTology Applied Cross-disciplinary Technologies (CONTACT)*, pp. 385–400, 2014.

[220] C. Galleguillos and S. Belongie, "Context based object categorization: A critical survey," *Computer Vision and Image Understanding*, vol. 114, no. 6, pp. 712–722, 2010.

[221] C. Galleguillos, A. Rabinovich, and S. Belongie, "Object categorization using co-occurrence, location and appearance," *Computer Vision and Pattern Recognition*, pp. 1–8, 2008.

[222] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. J. Belongie, "Objects in context," *International Conference on Computer Vision*, vol. 1, no. 2, p. 5, 2007.

[223] K. Murphy, A. Torralba, W. Freeman *et al.*, "Using the forest to see the trees: A graphical model relating features, objects and scenes," *Advances in Neural Information Processing Systems*, vol. 16, pp. 1499–1506, 2003.

[224] P. Carbonetto, N. de Freitas, and K. Barnard, "A statistical model for general contextual object recognition." Springer, 2004, pp. 350–362.

[225] M. J. Choi, A. Torralba, and A. S. Willsky, "Context models and out-of-context objects," *Pattern Recognition Letters*, vol. 33, no. 7, pp. 853–862, 2012.

[226] A. Singhal, J. Luo, and W. Zhu, "Probabilistic spatial context models for scene content understanding," *Computer Vision and Pattern Recognition*, vol. 1, pp. I–235, 2003.

[227] N. Ahuja and S. Todorovic, "Learning the taxonomy and models of categories present in arbitrary images," *International Conference on Computer Vision*, pp. 1–8, 2007.

[228] E. B. Sudderth, A. Torralba, W. T. Freeman, and A. S. Willsky, "Learning hierarchical models of scenes, objects, and parts," *International Conference on Computer Vision*, vol. 2, pp. 1331–1338, 2005.

[229] T. Lan, M. Raptis, L. Sigal, and G. Mori, "From subcategories to visual composites: A multi-level framework for object detection," *International Conference on Computer Vision*, pp. 369–376, 2013.

[230] M. J. Choi, J. J. Lim, A. Torralba, and A. S. Willsky, "Exploiting hierarchical context on a large database of object categories," *Computer Vision and Pattern Recognition*, pp. 129–136, 2010.

[231] M. J. Choi, A. Torralba, and A. S. Willsky, "A tree-based context model for object recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 2, pp. 240–252, 2012.

[232] R. G. Cinbis and S. Sclaroff, "Contextual object detection using set-based classification." Springer, 2012, pp. 43–57.

[233] J. Fan, Y. Gao, and H. Luo, "Hierarchical classification for automatic image annotation," *ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 111–118, 2007.

[234] J. Fan, Y. Gao, and H. Luo, "Integrating concept ontology and multitask learning to achieve more effective classifier training for multilevel image annotation," *IEEE Transactions on Image Processing*, vol. 17, no. 3, pp. 407–426, 2008.

[235] J. Fan, Y. Gao, H. Luo, and R. Jain, "Mining multilevel image semantics via hierarchical classification," *IEEE Transactions on Multimedia*, vol. 10, no. 2, pp. 167–187, 2008.

[236] L.-J. Li, H. Su, L. Fei-Fei, and E. P. Xing, "Object bank: A high-level image representation for scene classification & semantic feature sparsification," *Advances in Neural Information Processing Systems*, pp. 1378–1386, 2010.

[237] L.-J. Li, H. Su, Y. Lim, and L. Fei-Fei, "Object bank: An object-level image representation for high-level visual recognition," *International Journal of Computer Vision*, vol. 107, no. 1, pp. 20–39, 2014.

[238] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[239] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[240] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," *International Conference on Machine Learning*, pp. 1321–1330, 2017.

[241] J. Platt *et al.*, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in Large Margin Classifiers*, vol. 10, no. 3, pp. 61–74, 1999.

[242] N. Ye, K. Chai, W. Lee, and H. Chieu, "Optimizing f-measures: A tale of two approaches," *International Conference on Machine Learning*, vol. 1, pp. 289–296, 2012.

[243] L. Feng and B. Bhanu, "Semantic concept co-occurrence patterns for image annotation and retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 4, pp. 785–799, 2015.

[244] C. Li, D. Parikh, and T. Chen, "Automatic discovery of groups of objects for scene understanding," in *CVPR*, 2012.

[245] S. Song, S. P. Lichtenberg, and J. Xiao, "Sun rgb-d: A rgb-d scene understanding benchmark suite," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 567–576.

[246] P. Miettinen, T. Mielikäinen, A. Gionis, G. Das, and H. Mannila, "The discrete basis problem," *IEEE TKDE*, 2008.

[247] J. Łukasiewicz, "O logice trójwartościowej," 1988.

[248] L. A. Zadeh, "Fuzzy sets," *Information and control*, vol. 8, no. 3, pp. 338–353, 1965.

[249] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *NIPS*, 2011.

[250] C. Ding, T. Li, W. Peng, and H. Park, "Orthogonal nonnegative matrix t-factorizations for clustering," in *SIGKDD*, 2006.

[251] Z. Zhang, T. Li, C. Ding, and X. Zhang, "Binary matrix factorization with applications," in *ICDM*, 2007.

[252] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[253] P. Paatero and U. Tapper, "Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values," *Environmetrics*, vol. 5, no. 2, pp. 111–126, 1994.

[254] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015.

[255] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv*, 2014.

[256] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE TPAMI*, 2017.

[257] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," *CVPR*, 2017.

[258] L. Torresani, M. Szummer, and A. Fitzgibbon, "Efficient object category recognition using classemes," *ECCV*, 2010.

[259] A. Bergamo and L. Torresani, "Meta-class features for large-scale object categorization on a budget," in *CVPR*, 2012.

[260] D. M. Blei and J. D. Lafferty, "Topic models," in *Text Mining*. Chapman and Hall/CRC, 2009, pp. 101–124.

[261] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers, "Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2097–2106.

[262] P. Rajpurkar, J. Irvin, K. Zhu, B. Yang, H. Mehta, T. Duan, D. Ding, A. Bagul, C. Langlotz, K. Shpanskaya *et al.*, "Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning," *arXiv preprint arXiv:1711.05225*, 2017.

[263] L. Yao, E. Poblenz, D. Dagunts, B. Covington, D. Bernard, and K. Lyman, "Learning to diagnose from scratch by exploiting dependencies among labels," *arXiv preprint arXiv:1710.10501*, 2017.

[264] P. Spyns, "Natural language processing in medicine: an overview," *Methods of information in medicine*, vol. 35, no. 04/05, pp. 285–301, 1996.

[265] M. Krallinger and A. Valencia, "Text-mining and information-retrieval services for molecular biology," *Genome biology*, vol. 6, no. 7, p. 224, 2005.

[266] E. Pons, L. M. Braun, M. M. Hunink, and J. A. Kors, "Natural language processing in radiology: a systematic review," *Radiology*, vol. 279, no. 2, pp. 329–343, 2016.

[267] Y. Wang, L. Wang, M. Rastegar-Mojarad, S. Moon, F. Shen, N. Afzal, S. Liu, Y. Zeng, S. Mehrabi, S. Sohn *et al.*, "Clinical information extraction applications: a literature review," *Journal of biomedical informatics*, 2017.

[268] H.-C. Shin, L. Lu, L. Kim, A. Seff, J. Yao, and R. M. Summers, "Interleaved text/image deep mining on a very large-scale radiology database," in *CVPR*, 2015, pp. 1090–1099.

[269] H.-C. Shin, L. Lu, L. Kim, A. Seff, J. Yao, and R. M. Summers, "Interleaved text/image deep mining on a large-scale radiology database for automated image interpretation," *Journal of Machine Learning Research*, vol. 17, no. 1, pp. 3729–3759, 2016.

[270] X. Wang, Y. Peng, L. Lu, Z. Lu, and R. M. Summers, "Tienet: Text-image embedding network for common thorax disease classification and reporting in chest x-rays," in *CVPR*. IEEE, 2018, pp. 9049–9058.

[271] H.-C. Shin, K. Roberts, L. Lu, D. Demner-Fushman, J. Yao, and R. M. Summers, "Learning to read chest x-rays: Recurrent neural cascade model for automated image annotation," in *CVPR*, 2016, pp. 2497–2506.

[272] B. Jing, P. Xie, and E. Xing, "On the automatic generation of medical imaging reports," *arXiv preprint arXiv:1711.08195*, 2017.

[273] Z. Zhang, Y. Xie, F. Xing, M. McGough, and L. Yang, "Mdnet: A semantically and visually interpretable medical image diagnosis network," in *CVPR*, 2017, pp. 6428–6436.

[274] M. Moradi, A. Madani, Y. Gur, Y. Guo, and T. Syeda-Mahmood, "Bimodal network architectures for automatic generation of image annotation from text," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2018, pp. 449–456.

[275] Y. Xue, T. Xu, L. R. Long, Z. Xue, S. Antani, G. R. Thoma, and X. Huang, "Multimodal recurrent model with attention for automated radiology report generation," in *MICCAI*. Springer, 2018, pp. 457–466.

[276] G. O. Gajbhiye, A. V. Nandedkar, and I. Faye, "Automatic report generation for chest x-ray images: A multilevel multi-attention approach," in *International Conference on Computer Vision and Image Processing*. Springer, 2019, pp. 174–182.

[277] S. Biswal, C. Xiao, L. Glass, B. Westover, and J. Sun, "Clinical report autocompletion," in *Proceedings of The Web Conference 2020*, 2020, pp. 541–550.

[278] M. M. A. Monshi, J. Poon, and V. Chung, "Deep learning in generating radiology reports: A survey," *Artificial Intelligence in Medicine*, p. 101878, 2020.

[279] Y. Zhang, X. Wang, Z. Xu, Q. Yu, A. Yuille, and D. Xu, "When radiology report generation meets knowledge graph," *arXiv preprint arXiv:2002.08277*, 2020.

[280] D. Demner-Fushman, M. D. Kohli, M. B. Rosenman, S. E. Shooshan, L. Rodriguez, S. Antani, G. R. Thoma, and C. J. McDonald, "Preparing a collection of radiology examinations for distribution and retrieval," *Journal of the American Medical Informatics Association*, vol. 23, no. 2, pp. 304–310, 2015.

[281] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *International conference on artificial neural networks*. Springer, 2018, pp. 270–279.

[282] S. Danesh, T. Sumner, and J. H. Martin, "Sgrank: Combining statistical and graphical methods to improve the state of the art in unsupervised keyphrase extraction," in **SEM*, 2015, pp. 117–126.

[283] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International Conference on Machine Learning*, 2014, pp. 1188–1196.

[284] A. Quattoni and A. Torralba, "Recognizing indoor scenes," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 413–420.

[285] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *NIPS*, 2014.

[286] A. Bendale and T. Boult, "Towards open world recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1893–1902.

[287] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, and T. E. Boult, "Toward open set recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 7, pp. 1757–1772, 2012.

[288] B. Settles, "Active learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2009.

[289] W. J. Scheirer, L. P. Jain, and T. E. Boult, "Probability models for open set recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 11, pp. 2317–2324, 2014.

[290] J. Aloimonos, I. Weiss, and A. Bandyopadhyay, "Active vision," *International journal of computer vision*, vol. 1, no. 4, pp. 333–356, 1988.

[291] R. Bajcsy, "Active perception," *Proceedings of the IEEE*, vol. 76, no. 8, pp. 966–1005, 1988.

[292] R. Bajcsy, Y. Aloimonos, and J. K. Tsotsos, "Revisiting active perception," *Autonomous Robots*, vol. 42, no. 2, pp. 177–196, 2018.

[293] D. H. Ballard, "Reference frames for animate vision." in *IJCAI*, vol. 89, 1989, pp. 1635–1641.

[294] S. Chen, Y. Li, and N. M. Kwok, "Active vision in robotic systems: A survey of recent developments," *International Journal of Robotics Research*, vol. 30, no. 11, pp. 1343–1377, 2011.

[295] X. Li and Y. Guo, "Multi-level adaptive active learning for scene classification," in *European Conference on Computer Vision*. Springer, 2014, pp. 234–249.

[296] X. Yu, C. Fermüller, C. L. Teo, Y. Yang, and Y. Aloimonos, "Active scene recognition with vision and language," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 810–817.

[297] J. H. Bappy, S. Paul, and A. K. Roy-Chowdhury, "Online adaptation for joint scene and object classification," in *European Conference on Computer Vision*. Springer, 2016, pp. 227–243.

[298] X. Li, R. Guo, and J. Cheng, "Incorporating incremental and active learning for scene classification," in *2012 11th International Conference on Machine Learning and Applications*, vol. 1. IEEE, 2012, pp. 256–261.

[299] X. Li and Y. Guo, "Adaptive active learning for image classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 859–866.

[300] S. Paul, J. H. Bappy, and A. K. Roy-Chowdhury, "Efficient selection of informative and diverse training samples with applications in scene classification," in *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2016, pp. 494–498.

[301] C. Zheng, Y. Yi, M. Qi, F. Liu, C. Bi, J. Wang, and J. Kong, "Multicriteria-based active discriminative dictionary learning for scene recognition," *IEEE Access*, vol. 6, pp. 4416–4426, 2017.

[302] D. Jayaraman and K. Grauman, "Learning to look around: Intelligently exploring unseen environments for unknown tasks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1238–1247.

[303] E. Sommerlade and I. Reid, "Information-theoretic active scene exploration," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2008, pp. 1–7.

[304] C. Brown, "Prediction and cooperation in gaze control," *Biological cybernetics*, vol. 63, no. 1, pp. 61–70, 1990.

[305] D. J. Coombs and C. M. Brown, "Intelligent gaze control in binocular vision," in *Proceedings. 5th IEEE International Symposium on Intelligent Control 1990*. IEEE, 1990, pp. 239–245.

[306] D. Wilkes and J. K. Tsotsos, "Active object recognition," in *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 1992, pp. 136–141.

[307] L. Wixson, "Viewpoint selection for visual search," in *CVPR*, vol. 94, 1994, pp. 800–805.

[308] J. C. Caicedo and S. Lazebnik, "Active object localization with deep reinforcement learning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2488–2496.

[309] X. S. Chen, H. He, and L. S. Davis, "Object detection in 20 questions," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2016, pp. 1–9.

[310] A. Garcia, A. Vezhnevets, and V. Ferrari, "An active search strategy for efficient object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3022–3031.

[311] E. Johns, S. Leutenegger, and A. J. Davison, "Pairwise decomposition of image sequences for active multi-view recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3813–3822.

[312] S. Mathe, A. Pirinen, and C. Sminchisescu, "Reinforcement learning for visual object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2894–2902.

[313] F. Darema, "Dynamic data driven applications systems: A new paradigm for application simulations and measurements," in *International Conference on Computational Science*. Springer, 2004, pp. 662–669.

[314] E. Blasch and D. A. Lambert, *High-level information fusion management and systems design*. Artech House, 2012.

[315] E. P. Blasch, D. A. Lambert, P. Valin, M. M. Kokar, J. Llinas, S. Das, C. Chong, and E. Shahbazian, "High level information fusion (hlif): Survey of models, issues, and grand challenges," *IEEE Aerospace and Electronic Systems Magazine*, vol. 27, no. 9, pp. 4–20, 2012.

[316] P. H. Foo and G. W. Ng, "High-level information fusion: An overview." *J. Adv. Inf. Fusion*, vol. 8, no. 1, pp. 33–72, 2013.

[317] A. Bendale and T. E. Boult, "Towards open set deep networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1563–1572.

[318] L. P. Jain, W. J. Scheirer, and T. E. Boult, "Multi-class open set recognition using probability of inclusion," in *European Conference on Computer Vision*. Springer, 2014, pp. 393–409.

[319] F. Li and H. Wechsler, "Open set face recognition using transduction," *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 11, pp. 1686–1697, 2005.

[320] E. M. Rudd, L. P. Jain, W. J. Scheirer, and T. E. Boult, "The extreme value machine," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 762–768, 2017.

[321] R. Zhang and D. N. Metaxas, "Ro-svm: Support vector machine with reject option for image categorization." in *BMVC*. Citeseer, 2006, pp. 1209–1218.

[322] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *Journal of Machine Learning Research*, vol. 7, no. Mar, pp. 551–585, 2006.

[323] R. De Rosa, T. Mensink, and B. Caputo, "Online open world recognition," *arXiv preprint arXiv:1604.02275*, 2016.

[324] A. Kapoor, S. Baker, S. Basu, and E. Horvitz, "Memory constrained face recognition," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 2539–2546.

[325] P. Laskov, C. Gehl, S. Krüger, and K.-R. Müller, "Incremental support vector learning: Analysis, implementation and applications," *Journal of machine learning research*, vol. 7, no. Sep, pp. 1909–1936, 2006.

[326] L.-J. Li and L. Fei-Fei, "Optimol: automatic online picture collection via incremental model learning," *International journal of computer vision*, vol. 88, no. 2, pp. 147–168, 2010.

[327] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka, "Metric learning for large scale image classification: Generalizing to new classes at near-zero cost," in *European Conference on Computer Vision*. Springer, 2012, pp. 488–501.

[328] M. Ristin, M. Guillaumin, J. Gall, and L. Van Gool, "Incremental learning of ncm forests for large-scale image classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 3654–3661.

[329] J. Xiao, K. A. Ehinger, A. Oliva, and A. Torralba, "Recognizing scene viewpoint using panoramic place representation," in *CVPR*, 2012.

[330] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," *arXiv preprint arXiv:1904.09237*, 2019.

[331] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[332] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[333] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*. Elsevier, 1989, vol. 24, pp. 109–165.

[334] R. Ratcliff, "Connectionist models of recognition memory: constraints imposed by learning and forgetting functions." *Psychological review*, vol. 97, no. 2, p. 285, 1990.

[335] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017. [Online]. Available: http://www.pnas.org/content/114/13/3521.abstract

[336] P. Kontschieder, M. Fiterau, A. Criminisi, and S. Rota Bulo, "Deep neural decision forests," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1467–1475.

[337] Y. Yang, I. G. Morillo, and T. M. Hospedales, "Deep neural decision trees," *arXiv preprint arXiv:1806.06988*, 2018.