Inductive Biases for Higher-Order Visual Cognition

by

Shashank Shekhar

A Thesis
presented to
The University of Guelph

In partial fulfilment of requirements
for the degree of
Master of Applied Science
in
Engineering
(Collaborative Specialization in Artificial Intelligence)

Guelph, Ontario, Canada

ABSTRACT

INDUCTIVE BIASES FOR HIGHER ORDER VISUAL COGNITION

Shashank Shekhar
University of Guelph, 2022

Advisor:
Dr. Graham W. Taylor

A key factor driving the success of deep neural network (DNN) based machine learning models is the use of inductive biases. Inductive biases allow human designers to incorporate knowledge about the learning task into the models, the data used to train them, as well as their training regime. These biases enable DNNs to learn structured representations of data that capture problem semantics for well-specified tasks and generalize to test data drawn from the same distribution. However, there are several problems in higher-order cognitive thinking that deep learning has yet to solve. For example, explainability of DNN decision making, reasoning on the learned data abstractions, and out-of-distribution (OOD) generalization to unseen domains are still open research problems. In this thesis, we try and address these issues by devising novel inductive biases derived from human cognition. For this purpose, we utilize dynamic DNNs that perform a variable amount of computation based on their input. Our first project focuses on generating black-box explanations for DNNs. We adapt response time evaluation in human psychophysics for deep learning to calculate Neural Response Times (NRT) by profiling dynamic DNNs. We verify that NRT is able to corroborate known effects about the feature space composed by object recognition models when tested on OOD view points. We further demonstrate that NRT can be used to causally verify Scene Grammar effects in identifying semantic and syntactic inconsistencies across visual scenes. For our second project, we utilize Structure Mapping, a cognitive theory of how humans make analogies, to design deep models capable of out-of-distribution generalization. We explicitly map the structure of the reasoning problem to the DNN architecture using neural module networks. Our Neural Structure Mapping (NSM) approach successfully isolates the visual relationship from the underlying visual domain. Furthermore, mapping the relationship structure to the reasoning model demonstrates systematic OOD generalization in drawing analogies across novel visual domains. Our results demonstrate the utility of building inductive biases derived from human cognition for addressing problems in higher-order computer vision.

DEDICATION

To my grandmother Prabhavati Devi

Her care and support enabled three generations of my family to pursue higher education. Despite never having the opportunity to receive schooling herself, she always instilled into me the importance of a good education. I owe mine to her.

ACKNOWLEDGEMENTS

First and foremost, I would like to acknowledge my advisor, Prof. Graham Taylor, for being an amazing and sympathetic mentor throughout my master's program. Working closely with Graham has helped me develop a better sense of planning and implementing a research idea. He also encouraged and guided me in pursuing my interests in teaching and entrepreneurship outside of the regular coursework and research. On a more personal note, he has helped me become a better person through advice and feedback as a mentor, teacher, and advisor. I sincerely hope that our collaboration continues into the future.

Next, I would like to thank my senior academic collaborators and co-authors, Dr. Eric Taylor and Boris Knyazev, who mentored and guided me through research projects and helped me shape my own research system. I would also like to thank Dr. Magdalena Sobol, for her mentorship and feedback with my academic writing and communication. I have learned a lot from all of them, and their mentorship and help has been very fruitful.

I would also like to thank all the other members of the Machine Learning Research Group (MLRG). MLRG has been an incredible part of my master's journey. I have been privileged to get to know, and call upon as friends, all the smart and kind people I met here. Several ideas and breakthroughs in my thesis were inspired by discussions, talks, and brainstorms with different MLRG members at various points during my time at MLRG. Beyond research, I would like to personally thank Michal Lisicki for inviting me to several of his outdoor adventures and helping me out at multiple occasions; Terrance DeVries for his advice and for organizing game nights during the pandemic; Nolan Dey for organizing daily stand-ups and helping me with mock interviews; Eu Wern Teh for helping me move apartments during the lockdown; Vikram Voleti for helping me with job applications; and Adam Balint for letting me share an apartment in Seoul.

I would like to thank my friends and family for their consistent love and support. I want to thank my parents, Sudha Yadava and Krishna Mohan Singh, for always being there for me and encouraging me to pursue my dreams. I would like to thank my sister, Sukriti Yadav, for being an incredible role-model. I would like to thank my friends Prabodh Tripathi, Manoj Kumar, Janpreet Singh, Syed Danish Haider, Abdelrahman Allam, and Rupali Bhati for their help on multiple occassions during my master's. And lastly, I would like to thank my girlfriend Jasmine Muszik, for her constant encouragement and support during our time together.

Finally, I would like to thank the University of Guelph, Vector Institute, National Sciences and Engineering Research Council, Tata Trusts, and the Computer Vision Foundation, for directly contributing to my success in this program through research funding, scholarships, and travel grants.

BIBLIOGRAPHIC NOTE

Major portions of this thesis are based on prior peer-reviewed publications. NRT was originally published in Taylor et al. (2020), and an extended version (Taylor et al., 2021) has been accepted for journal publication. Initial work on NSM has been published in Shekhar and Taylor (2021) and an extended version is currently under peer review.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The field of machine learning aims to build learning agents which can improve their performance on intelligent tasks through additional experience. A long-standing goal of machine learning has been to improve these agents to rival and surpass human performance on non-trivial tasks. In certain narrow areas, this goal seems to have been realized as deep learning has demonstrated strong performance on several problems in computer vision (Krizhevsky et al., 2012; Ren et al., 2015) as well as other domains like natural language processing (Vaswani et al., 2017; Chollampatt and Ng, 2017), robotics (Andrychowicz et al., 2020), and network analysis (Bronstein et al., 2017). Deep learning models are now able to surpass human performance on computer vision tasks like canonical object recognition in images (He et al., 2015), activity recognition in videos (Jobanputra et al., 2019), facial recognition (Balaban, 2015), handwriting generation (Graves, 2013) and a myriad of other problems once considered intractable for machines. This rapid pace of advancement in computer vision, driven primarily through deep learning, begs the question: "Is computer vision 'solved' or will it be 'solved' in the near future?"

The spectrum of cognitive tasks that humans can perform ranges from simplistic tasks such as recognition to extremely complex generalization and reasoning. There are no rigid delineations between the various faculties involved in cognition such as perception, memory, reasoning etc as multiple brain regions are involved simultaneously in generating a perceived cognitive output. There have been attempts at developing a taxonomy of cognitive skills (Bloom et al., 1956) to guide the development of human learning programs, with the lower-order tasks presenting easier challenges and aimed at early learners, while higher-order tasks requiring complex thinking and multiple cognitive requirements and aimed at advanced learners. The tasks which deep-learning based computer vision models have excelled at, tend to involve discriminative learning (such as image or video classification) which tests for memory and recall. On the hierarchy of cognitive tasks, these tasks would constitute 'lower-order' tasks requiring fewer and easier cognitive skills. Recent works in deep learning research (Goyal and Bengio, 2020) have highlighted the need to design problem statements, models, as well as learning regimes that can address problems that involve higher-order cognition. Among a myriad of other requirements, higher-order vision tasks

require models to understand the semantics of a problem, reason about the task, and make decisions that are explainable to humans. Visual question answering (Antol et al., 2015), understanding object dynamics from videos (Luo et al., 2017), and inferring scene knowledge graphs from images (Shi et al., 2019) are a few examples of computer vision tasks that involve higher-order cognition. While deep learning models have shown promising performance in constrained environments on such tasks (Johnson et al., 2017a), their success is far less resounding than on lower-order tasks (Bahdanau et al., 2019).

One of the central factors driving the success of deep learning (Zhang et al., 2016; Xu et al., 2020) in vision tasks was the incorporation of **inductive biases** (Mitchell, 1980) into these models. Inductive biases, or priors, are design choices made while formalizing the learning task, modelling the underlying data-generating phenomenon, as well as training the model itself. These priors help constrain the hypothesis space of possible solutions and, thus, facilitate the learning of good models by searching over the reduced space. Deep learning itself is based on the fundamental inductive bias of learning hierarchical representations of data that enables it to learn functional compositionality. Convolutional neural networks (CNNs) rely on the convolution operator, which introduces a group equivariance prior on input data that translates to spatial equivariance in the vision domain. The ability to learn features that are spatially equivariant enables CNNs to treat the same object occurring at different spatial locations in a uniform manner, driving their success and widespread adoption in computer vision.

For computer vision to succeed at higher-order tasks, it is important to understand the requirements for solving these tasks and then deriving new inductive biases, or re-purposing existing biases, in a way that meets these requirements. There have been several innovations in interpretability and reasoning in computer vision in utilizing existing priors for higher-order tasks. For example, learning a common representational embedding between images and natural language is widespread in interpretable machine learning (Hendricks et al., 2016) and visual reasoning research (Antol et al., 2015). Since language enables an excellent abstraction to represent higher-order semantic concepts, it is only logical that these embeddings are common for higher-order vision tasks. This example clearly demonstrates the two steps required for building higher-order models: (1) understanding the underlying cognitive requirements (abstraction for semantic concepts), and (2) building inductive biases that address these requirements (learning a shared embedding).

Another requirement of higher-order cognition is the ability to generalize to out-of-distribution (OOD) data. The overarching goal of artificial intelligence research has always been to not only rival human performance at specific tasks but at a large variety of general tasks (sometimes dubbed artificial general intelligence.) The ability to generalize to OOD data is considered central to developing such models. Although, the possible sample space of OOD data is infinite, the space of low-level semantic concepts is much smaller. A key insight into human knowledge and reasoning from cognitive science is that human knowledge is composed of a reasonable number of low-level semantic concepts (Lake et al., 2015). These basic concepts are then combined in a multitude of ways to enable rapid learning and systematic generalization across several cognitive functions in humans (Nam

and McClelland, 2021). Systematic generalization benchmarks in natural language (Lake and Baroni, 2018) and computer vision (Johnson et al., 2017a) aim to quantify this ability in machine learning models.

In this thesis, we concentrate on the **dynamic inference** inductive bias in computer vision models. Unlike regular deep learning models that are fixed at inference time, this inductive bias enables models to perform variable computation for different input data at inference time. While dynamic inference was originally developed as a means to perform resource-efficient machine learning Graves (2017), it has since evolved to serve other purposes in deep learning such as improved training with increased depth (Szegedy et al., 2015) and adversarial robustness (Hu et al., 2019). We consider the dynamic inference prior in two different contexts. First, we consider the early-exit strategy which allows intermediate classifiers in a model to return an output as soon as a threshold is met. We use early-exits to derive a novel explainability measure for CNNs. Second, we utilize the modular prior that assembles neural network modules for reasoning at inference time in a dynamic manner. We then use the modular network to perform abstract visual reasoning . Thus, we successfully demonstrate the use of the dynamic inference prior in two higher-order vision tasks, namely explainability and reasoning.

1.1 Contributions

1.1.1 Neural Response Time Analysis

Explainable Artificial Intelligence (XAI) methods often rely on access to the internals of a machine learning model such as parameters, gradients, and activations. However, the internals of a model are often not available to users and practitioners of AI since the release of such details is up to the developers. Several popular methods rely on post-hoc explanations. For instance, saliency-based methods (Simonyan et al., 2013) were developed to aid humans' ability to anticipate network predictions by visualizing the input feature space for components important for classification. Yet, saliency-based methods lack sensitivity to features (Sundararajan et al., 2017) which makes them prone to making incorrect assessments on the importance of features. Furthermore, saliency-based methods lack invariance to input transformations that do not affect model predictions or weights (Kindermans et al., 2019) which means that they reflect changes in the input even when they should not. Finally, it is also clear that saliency methods do not impact humans' ability to anticipate network predictions (Alqaraawi et al., 2020), the key desiderata of an explainability method. While the drawbacks with saliency methods are perhaps the most studied owing to their popularity, other post-hoc approaches to XAI also suffer from similar shortcomings. As a result of the unreliability of existing interpretability methods, more robust XAI methods are needed to provide causal rather than correlation-based explanations. In order to do so, it is necessary that such XAI methods are able to satisfy the requirments of the classical scientific method of performing falsifiable hypothesis testing (Leavitt and Morcos, 2020).

In this study, we propose the metric of Neural Response Time (NRT) for dynamic inference models inspired from human response time studies. The premise of calculating NRT is very simple —we take a dynamic inference model and from it return both the model outputs as well as the time it took for the model to perform the computation. Controlling for system hardware and output accuracy, we test if NRT can distinguish between canonical and non-canonical object recognition for the same classes in ImageNet (Deng et al., 2009) and ObjectNet (Barbu et al., 2019). Furthermore, we test if NRT may be used to demonstrate Scene Grammar phenomena (Öhlschläger and Võ, 2017) through careful hypothesis testing on controlled input scenes. We test NRT on the background controlled Massive Memory dataset (Konkle et al., 2010) for object recognition. We also investigate whether NRT is simply a factor of input complexity and find that it bears specificity for class labels and features.

Individual Contribution

The idea of utilizing response time for explainability came from Dr. Eric Taylor from his prior work in human psychology. Prof. Graham Taylor suggested the exploration of early-exit models to enable response time calculations in CNNs. I implemented the calculation for NRT and conducted experiments. Statistical analyses of the resulting NRT were done by Dr. Eric Taylor. The manuscript was written by Dr. Eric Taylor and I with feedback from Prof. Graham Taylor and Dr. Magdalena Sobol.

1.1.2 Neural Structure Mapping

Building conceptual abstractions from sensory information and then reasoning about them is central to human intelligence. Abstract reasoning (Barrett et al., 2018) both relies on, and is facilitated by, our ability to make analogies about concepts from known domains to novel domains. Structure Mapping Theory of human analogical reasoning posits that analogical mappings (Gentner, 1983) rely on (higher-order) relations and not on the sensory content of the domain. This enables humans to reason systematically about novel domains, a problem with which machine learning (ML) models tend to struggle. Systematic generalization in visual reasoning requires the ability to reason across novel compositions of objects, attributes, and relationships. Bahdanau et al. (2018) have shown previously that modular networks (Andreas et al., 2016; Johnson et al., 2017b) that are aligned with the structure of a reasoning problem are significantly better at systematic generalization compared to monolithic models.

In this study, we introduce a two-stage neural framework, which we call Neural Structure Mapping (NSM), to learn visual analogies from Raven's Progressive Matrices (Raven, 2000), an abstract visual reasoning test of fluid intelligence. Our framework uses (1) a multi-task visual relationship encoder to extract constituent concepts from raw visual input in the source domain, and (2) a neural module net-based analogy inference engine to reason compositionally about the inferred relation in the target domain. We show that our NSM

approach (a) isolates the relational structure from the source domain with high accuracy, and (b) successfully utilizes this structure for analogical reasoning in the target domain. Furthermore, we show that the mapping process is central to drawing the correct analogy in our model, and that our approach to explicitly map the structure of the analogy to the neural network can compensate for the lack of a structure mapping prior in the data.

Individual Contribution

The idea of working on abstract reasoning came from my literature survey on concept learning, and the connection to systematic generalization was pointed to by Prof. Graham Taylor. The structure mapping approach, experimental plan, and development of the NSM codebase was done by me in consultation with Prof. Graham Taylor. The manuscript was written by me with the assistance of Prof. Graham Taylor and Dr. Magdalena Sobol.

1.2 Organization

The rest of this thesis is organized as follows:

- Chapter 2 provides a brief background on fundamental deep learning concepts. Next, a taxonomy of cognitive tasks is presented and used as a framework to classify computer vision tasks into lower and higher order. The chapter wraps up with a discussion on the challenge of systematic generalization in machine learning.
- Chapter 3 analyses the role of inductive biases in learning. It starts with a discussion of the convolution operator and then proceeds to examine the dynamic inference prior.
- Chapter 4 covers our Neural Response Time Analysis framework for explainability and discusses its results on object and scene classification tasks.
- Chapter 5 introduces the task of abstract visual reasoning and presents our Neural Structure Mapping framework for learning visual analogies.
- Chapter 6 provides a concluding discussion of this master's thesis.

Chapter 2

Background

This master's thesis considers the role of inductive biases in machine learning on higher-order computer vision problems. This chapter focuses on introducing the fundamental research questions addressed in this thesis. Since we primarily consider the class of algorithms that fall under the umbrella of **deep learning** for building vision models, the chapter begins with a brief discussion of deep learning and the neural network architectures underpinning our work. Next, we present a taxonomy of cognitive skills in humans and then use it to classify computer vision tasks into two hierarchical classes. We wrap up this chapter with a discussion on systematic generalization, an emerging challenge in building machine learning models for higher-order cognitive tasks.

2.1 Deep Learning

Deep learning (LeCun et al., 2015; Schmidhuber, 2015) encompasses the field of machine learning concerned with using **neural network** models for learning hierarchical feature representations of data and the learning algorithms for training these neural networks. Features refers to meaningful information about the data. Deep learning focuses on learning features directly from data as opposed to traditional machine learning models which relied on using features designed by humans. The features learned by a deep neural network are constituted by its parameters which represent the non-linear function learned by the network.

The learning procedure itself consists of an optimization problem, where an objective (or loss) function of the neural network's outputs is optimized using an algorithm. The most widely-used algorithms for neural network optimization involve stochastic gradient descent (Bottou, 2010) and its variations. Since gradient descent relies on computing gradients of the loss function wrt each network parameter, the optimization of deep neural networks benefited greatly from the development of the **backpropagation** (Rumelhart et al., 1988) algorithm. Backpropagation is a specific instantiation of automatic differentiation (Griewank and Walther, 2008), which evaluates derivatives of a function specified

as a computer program. The invention of backpropagation enabled efficient computation and storage of gradients of deep neural networks and laid the foundation for later work on developing both software and hardware implementations of deep learning routines as well as their optimization procedures.

The deep learning approach is different from traditional machine learning approaches where the feature extraction process is separate from the data modelling process. The idea of representation learning (Bengio et al., 2013a) of a hierarchy of features is central to deep learning's ability to perform well on complex learning tasks as features learned from vast amounts of data have been shown to outperform hand-crafted features on multiple learning tasks. The traditional approach dominated in the 20$^{\text{th}}$ and early 21$^{\text{st}}$ century because learning features, especially complex hierarchical features, was a computationally hard problem. There were several breakthroughs that enabled the learning of features from data: the advent of computational advancements particularly in hardware and software for performing fast tensor computations, large availability of rich data from the internet, as well as several novel developments in deep learning algorithms. These advancements have led to deep learning breakthroughs on challenging tasks in computer vision (OMahony et al., 2019), natural language processing (Nagarhalli et al., 2021), high-energy physics (Guest et al., 2018), and other domains with challenges that involve finding and exploiting patterns in highly complex data.

We will now introduce common deep learning models and briefly discuss their architectures. We do not go into the details of neural network training and optimization algorithms in this thesis since they are not central to our thesis' question. Readers are referred to Nielsen (2015) for an accessible introduction to this subject. Chapter 3 provides additional commentary on deep learning architectures from the perspective of inductive biases and their role in learning.

2.1.1 Feed-forward neural networks

One of the original neural networks was proposed by Rosenblatt (1958) and called the 'Perceptron'. It consisted of a single artificial neuron that took in as input a vector $\mathbf{x}$, multiplied it by a set of weights represented by the vector $\mathbf{w}$, and added a scalar bias term b to compute its activation value a . The output y of the perceptron is given by applying a binary thresholding function to the activation.

$$y = \begin{cases} 1, & \text{if } \mathbf{w}^\top \mathbf{x} + b \geqslant 1 \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

While the perceptron was initially quite popular, it fell out of use after Minsky and Papert (1969) showed its limitations. A key limitation was their inability to represent a wide class of functions, as Minsky and Papert showed that the Perceptron cannot represent a binary XOR function. Despite its limited success, the perceptron was a historically fundamental concept in the development of neural networks.

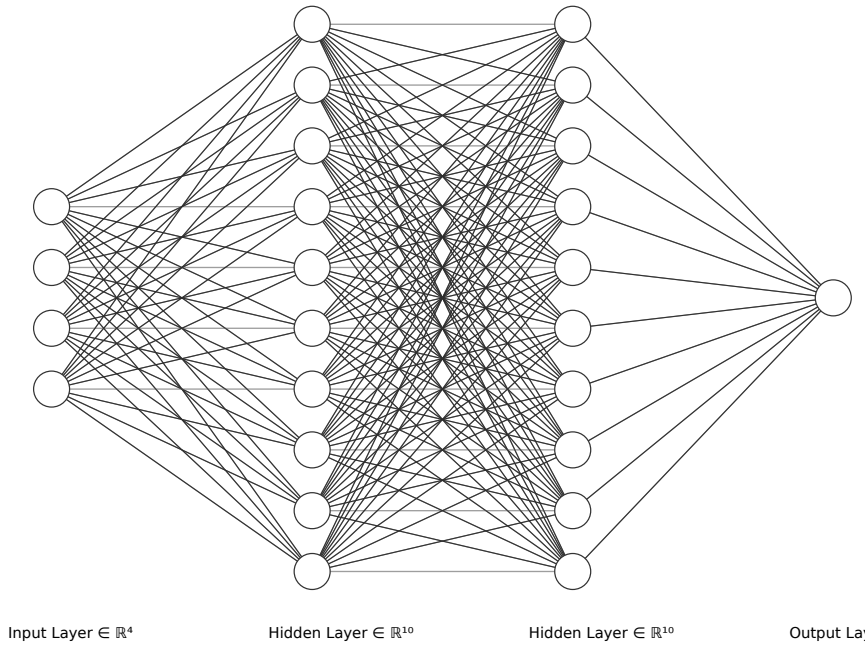


Figure 2.1: Schematic of a feed-forward neural network with two hidden layers

While singular perceptron units might be limited in their representational capacity to learn complex functions, a network consisting of multiple such units linked together through a series of non-linear transformations is quite a powerful approximator (Hornik, 1991). This model came to be known as the **Feedforward Neural Network** (FFNN), sometimes also called the Multi Layer Perceptron despite tenuous links to the perceptron. FFNNs are realized as a sequence of hidden layers of neurons, followed by an activation function. A FFNN layer can be expressed mathematically as:

$$\mathbf{h}_i = \mathbb{f}_i(\mathbf{W}_i{}^\top \mathbf{h}_{i-1} + \mathbf{b}_i), \tag{2.2}$$

where $\mathbf{h}_i$ is a hidden layers of neurons (or units), $\mathbf{W}_i$ is a matrix of weights for the layer i of the neural network, $\mathbf{b}_i$ is a bias vector, and $\mathbb{f}_i$ is a non-linear activation function. The input feature vector of the FFNN can be expressed as h_0, and we can express h_L as the output of an FFNN with L layers. Common activation functions include the Sigmoid function $\mathbb{f}(\mathbf{x}) = 1/(1 + e^{-\mathbf{x}})$ or ReLU $\mathbb{f}(\mathbf{x}) = \max(\mathbf{x}, 0)$.

The most significant improvement in FFNNs over Perceptrons comes from their vastly superior ability to learn useful representations of data. FFNNs are extremely strong function approximators (Hornik, 1991). Under mild assumptions on the activation function (continuous, bounded and non-constant), even a single-layer FFNN can approximate any continuous function arbitrarily well given enough hidden units. This result came to be

known in learning theory as the Universal Approximation Theorem. While in theory single layer FFNNs can approximate any function; in practice their representation capacity is often increased by increasing the depth of the network (Szegedy et al., 2015). This came to be reflected in the eponymous title of the field 'deep' learning. Figure 2.1 shows a FFNN with two hidden layers of units. The input vector is connected to the neurons in the first hidden layer through weight matrix $\mathbf{W}_1$, which are in turn connected to the neurons in the second hidden layer through weight matrix $\mathbf{W}_2$. Finally, the hidden neurons of the second layer feed into the output unit. The number of hidden layers is used to denote the depth of the neural network (`depth` = 2).

In modern deep learning architectures, FFNNs are rarely used by themselves. Instead FFNNs are used in conjunction with more complex neural network architectures. While in theory FFNNs can learn to approximate any function, doing so requires networks with a large amount of parameters. This makes training of FFNNs on large amounts of data hard and as a result FFNNs become computationally inefficient to train. As a result, other deep model artchitectures are utilized alongside FFNNs. Two popular architectures used in conjunction with FFNNs are convolutional neural networks and recurrent neural networks. These models are designed to exploit different characteristics in the training data as we discuss next.

2.1.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) (LeCun et al., 1989, 1998) have been the driving force behind the adoption of deep learning in computer vision. It was Krizhevsky et al. (2012)'s success on the ImageNet (Deng et al., 2009) dataset for object recognition which popularized their use in almost all computer vision problems. CNNs utilize the convolution operation to model data arranged in a grid-like topology. The convolution operation introduces spatial equivariance in the CNN features learned from this data. This enables to CNNs to utilize fewer parameters in modelling as they are able to utilize the parameters learned for a visual feature in one spatial position at other spatial positions too. Further detail about the equivariance and its effects on the CNN features are in Section 3.2.

The key difference between FFNNs and CNNs is in how the neurons between consecutive layers are connected to each other. Unlike FFNNs where each neuron in a layer is connected to all neurons from the previous layer, a CNN neuron is connected to only a few other neurons in the previous layer based on spatial proximity. For CNNs that operate on images, this connectivity is enabled through a 2D convolution operation or kernel. The input $\mathcal{I}$ to a kernel $\mathcal{K}$ comprises of a 3D tensor, with two spatial dimensions (e.g. `width` and `height` dimensions of an image) and usually one channel dimension (e.g. 3 channels for RGB images, 1 channel for grayscale images). The kernel $\mathcal{K}$ takes this input $\mathcal{I}$ and maps it to the output $\mathcal{O}$, another 3D tensor with different spatial and channel dimensions.

The kernel $\mathcal{K}$ itself is a 4D tensor of parameters: one dimension corresponding to the input tensor channels, one dimension that becomes the channels of the output tensor, and

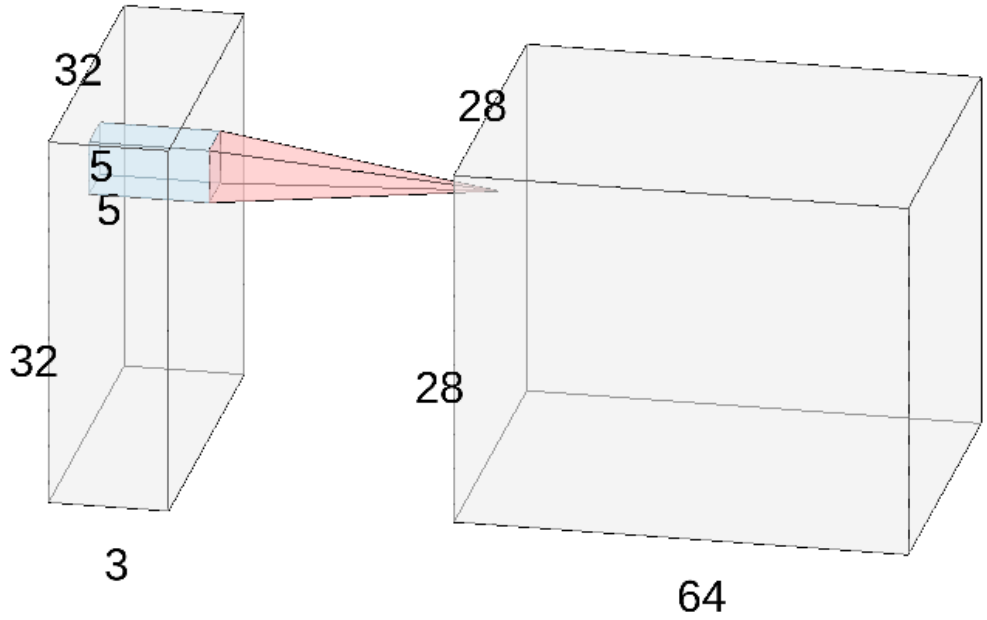


Figure 2.2: A Convolution Neural Network layer. An input volume (left) representing a 32x32x3 image, and an example volume of neurons (5x5x3, in blue) from one convolution filter are shown. The convolution layer shown here consists of 64 such filters. Connection (in red) shows output of convolution operation in next set of feature maps.

two spatial dimensions similar to the input and output. By design, these spatial dimensions are engineered to be usually much smaller than the input feature tensor $\mathcal{I}$'s spatial dimensions. For example, kernel dimensions are usually of the order of 3×3 or 5×5 pixels whereas images are usually have hundreds or sometimes thousands of pixels along their height and width. Thus, a convolutional kernel observes only a limited region of the input image determined by the kernel dimensions. This region is called the receptive field of the convolution. In a CNN, where several layers of convolutions are stacked on top of each other, the network's receptive field 'tunnels out' as feature maps in deeper layers have a wider receptive field of the input image. With sufficient number of convolutional layers, a CNN can have a receptive field covering the entire input image. Receptive fields in CNNs are discussed in the context of feature learning for computer vision in Section 3.2.

The convolution operation itself is defined as a series of vector dot product between the input tensor and the kernel as the kernel is moved element-wise across the input tensor. Each element in the output tensor $\mathcal{O}$ is operating on only a small spatial region of the input

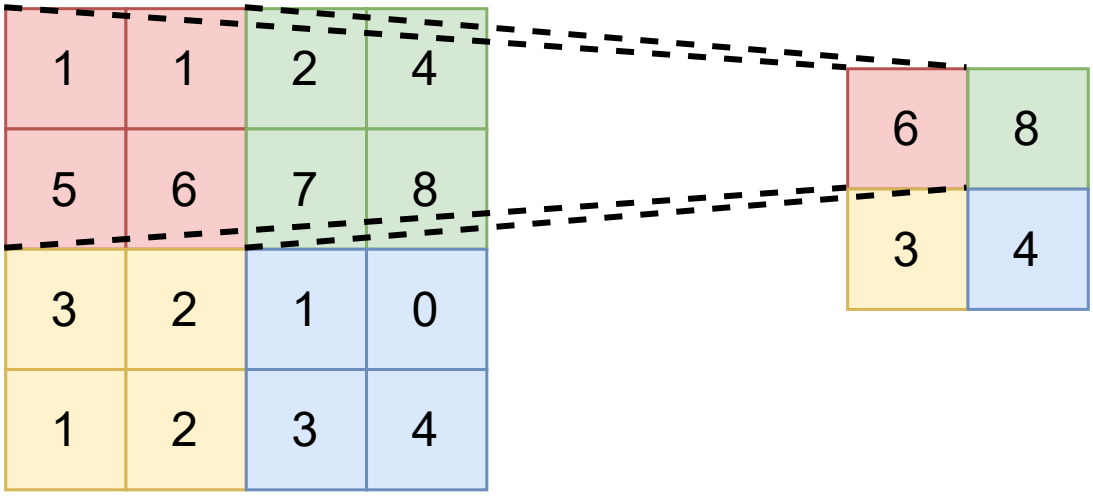


Figure 2.3: Illustration of the max pool operation with window of size 2×2 and stride 2.

tensor corresponding to the kernel size. Mathematically, each element $\mathcal{O}_{i,j,k}$ of the output tensor, where i is the index of the output channel and j and k are the spatial indices of the output element, is obtained via a sum over dot products on the vectorized kernel and a vectorized slice of a small input spatial region (Equation 2.3). The summation index l iterates over the input channels, while indices m and n are restricted by the spatial dimensions of the kernel, e.g. for a 3×3 kernel, $m, n \in \{-1, 0, 1\}$. Figure 2.2 depicts the interaction between two subsequent layers of a CNN: the kernel $\mathcal{K}$ (inner box on the right) is multiplied with a volume of the input $\mathcal{I}$ (outer box on the left) to produce a single unit consisting of a column of values at a single spatial location in the output $\mathcal{O}$ (outer box on the right)

$$\mathcal{O}_{i,j,k} = \sum_{l,m,n} \mathcal{I}_{l,j+m,k+n} \mathcal{K}_{i,l,m,n} \tag{2.3}$$

The exact arrangement of convolution kernels and their layers defines the architecture of the CNN. Each layer in a CNN is usually made up of a few convolutional kernels. In turn, the network itself is made of a series of convolutional kernels applied to the output of the previous kernels. Apart from convolutions, pooling is another common component used in CNNs. Pooling introduces a layer of subsampling between two convolutional kernels. Pooling takes the result of a non-linear activation applied to the convolution outputs, and replaces the activations in a small sub-region with their summary statistics. In Figure. 2.3 the 'max pool' operation is illustrated. Max pooling takes the outputs of convolution kernels, and replaces the outputs of a small region (pooling window) with the maximum output in the region.

Subsampling through pooling provides two key benefits: (1) it enables the model to be

invariant to small translations (equal to the window size) in the input features, and (2) it reduces the number of parameters in the subsequent layers and thus enables computational efficiency. For models where precise feature locations are important, such as semantic segmentation models where pixel-wise labels of the image need to be generated, pooling is generally not used. Further discussion on CNNs is available in the corresponding chapter in Goodfellow et al. (2016).

2.1.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) (Rumelhart et al., 1986) are neural network models that are designed to work on sequential data of arbitrary length. As a result, RNNs have been widely adopted in sequence processing tasks such as time-series modelling and forecasting (Hewamalage et al., 2021), as well as natural language processing (Goodfellow et al., 2016), where sentences and paragraphs can be considered as sequences of character or word tokens. In computer vision, recurrent neural networks are commonly used in conjunction with CNNs for video processing tasks, such as activity recognition (Murad and Pyun, 2017) or object tracking (Ondruska and Posner, 2016), since videos are made of sequences of image frames.

The reason behind the widespread use of RNNs in modelling sequential data comes from the fact that their architecture is designed to represent sequential relationships in data. RNNs operate on variable-length sequences in a recursive manner. They compute a hidden state activation of the input at each time step in the input sequence by operating on the hidden state of the previous time step and the input at the current time step. Mathematically, the hidden state of an RNN is given by:

$$\mathbf{h}^{(t)} = \phi\big(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}\big) \tag{2.4}$$

where ϕ represents the recurrence function of the unit. Thus, the hidden representation $\mathbf{h}^{(t)}$ learned at each time step t is a recursive function of the hidden representations of all previous time steps as well as all previous hidden representations.

In a vanilla RNN, ϕ is obtained by combining $\mathbf{h}^{(t-1)}$ and $\mathbf{x}^{(t)}$ using an affine function, and then performing a non-linear activation such as the $\tanh$ or sigmoid. Thus, the hidden unit may be broken down into two steps: the pre-activation and the non-linear activation. The hidden state $\mathbf{h}^{(t)}$ is obtained by applying the non-linear activation to the pre-activation $\mathbf{a}^{(t)}$ of a given timestep t by the link function $\mathbb{f}$, i.e. $\mathbf{h}^{(t)} = \mathbb{f}(\mathbf{a}^{(t)})$. This operation is similar to applying the non-linear activation to the neuron outputs computed in a FFNN or a CNN. The key differentiator for RNNs is how the pre-activation step is computed. The pre-activation $\mathbf{a}^{(t)}$ at each timestep t is a sum of matrix vector products of the RNN parameter matrices with the hidden state from the previous timestep $\mathbf{h}^{(t-1)}$ and the input $\mathbf{x}^{(t)}$ of the current time step:

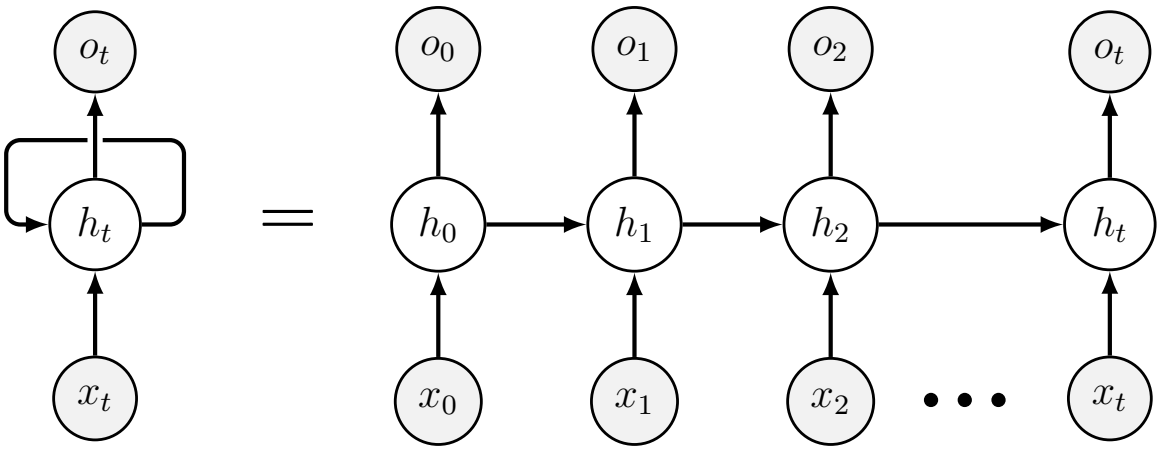


Figure 2.4: A recurrent neural network (unrolled)

$$\mathbf{a}^{(t)} = \mathbf{W_1}\mathbf{h}^{(t-1)} + \mathbf{W_2}\mathbf{x}^{(t)} + \mathbf{b}, \tag{2.5}$$

The weight matrices $\mathbf{W_1}$, $\mathbf{W_2}$ and bias vector $\mathbf{b}$ constitute the parameters of the RNN. RNNs share parameters over time as they remain consistent across each time step in the sequence, instead of sharing parameters spatially as in CNNs. This parameter reuse enables RNNs to learn similar features across the sequence, which is desirable since often the features of sequential data remain consistent across time. For example, the features learned for a video at time step = t would also be useful at time step = $t + n$. Sharing parameters across time also makes training easier compared to a FFNN on sequence data. Despite this, vanilla RNNs are hard to train for very long sequences and improvements on the base architecture such as the Long-Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Cho et al., 2014) are more commonly used in practice. Detailed discussion on these variants as well on training RNNs may be found in the corresponding chapter in Goodfellow et al. (2016).

2.2 Taxonomy of Human Learning

Curricula for human learning have tried to classify human educational objectives into levels of complexity based on their cognitive requirements. For example, a common learning hierarchy in the cognitive domain is Bloom's taxonomy of educational objectives (Bloom et al., 1956) that breaks down human learning objectives into six tiers of cognitive skills (Fig 2.5). While there have been several revisions of Bloom's taxonomy (Krathwohl, 2002; Adams, 2015), the central idea posits that the cognitive skills involved in human learning follow a tiered approach. In order to understand how this hierarchy might be relevant for

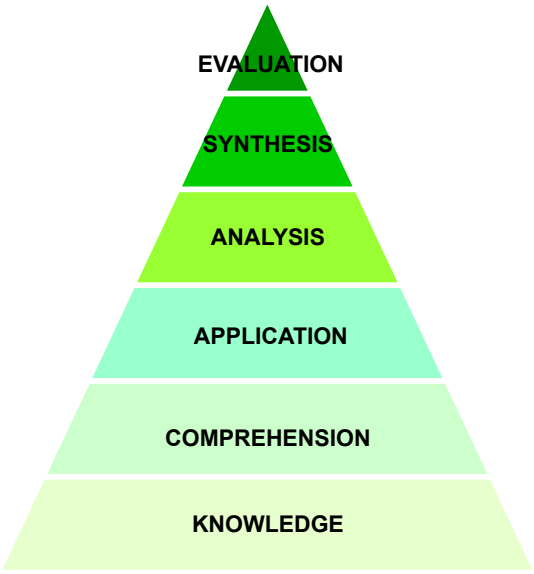


Figure 2.5: Bloom's taxonomy for cognitive domain

machine learning, it is important to understand what constitutes lower and higher-order cognitive skills in human learning.

The lower-order skills involve knowledge, comprehension and application (Fig 2.5). At the lowest tier, knowledge of semantic concepts simply involves recognizing and remembering, without necessarily understanding what they mean. Comprehension involves the next tier of understanding through organizing, summarizing, and generalizing the main ideas (Fig 2.5). Application involves the ability to utilize acquired knowledge in solving problems in new situations. The lower order of cognitive skills thus corresponds to a superficial understanding of concepts via direct recall, understanding and application.

At the higher-tiers of the taxonomy analysis involves the ability to break down knowledge into component parts, determining their compositional relations, and finding evidence to support generalizations. The penultimate tier, synthesis, involves building new ideas from diverse elements through modification and combination. At the highest level, evaluation involves the ability to make judgements about information, evaluating the validity of hypotheses, as well as presenting and justifying potential outcomes. The higher order of cognitive skills thus corresponds to a deep understanding of concepts beyond their surface level meaning and applications. Deep learning models currently struggle with the tasks involved in this higher-order. In order to develop new models or training methods to address these tasks, we need to break down these requirements and addressing each of them individually through proper inductive biases (see Section 3.1)

While there has not been a consolidated effort towards classifying machine learning tasks along an equivalent hierarchy, there is a growing body of work (Zellers et al., 2019; Zhang et al., 2019a; Goyal and Bengio, 2020) that aims to understand what is required

to incorporate higher-order cognitive skills into machine learning models. Next, we draw similarities between the requirements on a computer vision model for solving various tasks and Bloom's taxonomy of cognitive skills in order to better position the research on higher-order cognition.

2.2.1 Lower-order vs Higher-order Cognition in Computer Vision

Before the advent of deep learning, low-level computer vision consisted of feature extraction such as image matching (Barrow et al., 1977; Grauman and Darrell, 2005), inferring geometry (Torr and Zisserman, 2000), optical flow computation (Horn and Schunck, 1981). On the other hand, high-level computer vision involved inferring semantics such as object detection (Viola et al., 2001) and scene recognition (Wu, 2003). With the introduction of CNNs, this delineation has been blurred as both the feature extraction and semantic inference are performed by the same model. However, higher-order visual cognition extends beyond inferring object-level semantics for recognition (Zellers et al., 2019). If we draw a correspondence with human learning objectives, higher-order learning in computer vision involves reasoning about visual concepts, external knowledge, inferring object dynamics and relationships, as well being able to explain the decision-making process involved. Furthermore, humans tend to learn from few training examples and generalize very well to novel environments, something traditional deep learning models struggle with.

While CNNs excel at recognition-level understanding for computer vision problems, there has been limited progress in higher-order cognitive understanding in computer vision. Generally, the computer vision problems where deep learning models perform very well on IID test sets are analogous to the 'lower-order' domains of Bloom's taxonomy. Since the cognitive skills required in these tasks require learning class-specific visual features without deeply understanding the object class (Zhang et al., 2019a), the features learned end up relying on surface-level properties without developing an understanding of the higher-order semantics of the visual information (Geirhos et al., 2018). Higher-order visual cognition is dependent on the results of lower-order tasks such as object recognition, pose estimation, image and video segmentation which serve as building blocks for further understanding and reasoning about visual information. However, it also requires computer vision tasks and learning approaches to grow beyond these recognition-level problems where good performance on an IID test set is conflated with 'solving' the learning problem.

Since the advent of computer vision, there have been continuous attempts to formulate approaches that drive it from low-level to high-level visual cognition. This has been done both through novel tasks (Barrett et al., 2018), as well as imposing requirements on the models for solving these tasks that go beyond a simple class label or a binary yes/no answer (Hendricks et al., 2016). One attempt at higher-order visual cognition is the task of Visual Question Answering or VQA (Antol et al., 2015). VQA requires computer vision models to express their semantic understanding of an image or a video (Lei et al., 2018) by answering natural language questions about the image or video. VQA has slowly evolved to include answering questions based on external knowledge (Wu et al., 2016;

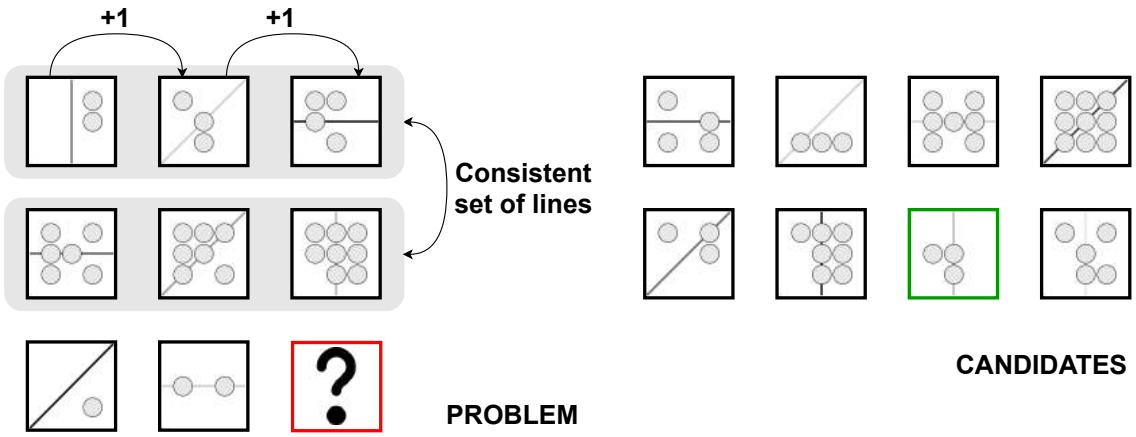


Figure 2.6: Abstract visual reasoning using Raven's progressive matrices. The example shows a progression in the quantity of shapes across the columns of the image matrix, and a consistent union in the types of lines across each row. Only the correct candidate (highlighted in green) satisfies both these relations with the missing panel (highlighted in red). Example from Barrett et al. (2018)'s PGM dataset.

Wang et al., 2017; Singh et al., 2019), reasoning about common sense (Zellers et al., 2019) including reasoning about physical kinematics (Mottaghi et al., 2016; Ye et al., 2018), social interactions (Alahi et al., 2016; Chuang et al., 2018), and understanding procedures for performing a task (Alayrac et al., 2016; Zhou et al., 2018). This rich set of reasoning problems has propelled VQA as an important task in computer vision research and several advancements in deep learning models for reasoning has come from VQA research (Hudson and Manning, 2018). However, language-grounded reasoning such as in VQA is only the beginning, as more complex reasoning problems such multi-modal reasoning, reasoning over multiple forms of knowledge (common-sense, factual, memory recall), reasoning in a multi-agent system, reasoning in an interactive environment are yet to be explored sufficiently in computer vision.

Besides reasoning, the ability to explain and recommend ideas is the highest tier of cognitive skills in Bloom's taxonomy. The best analogue for this skill in computer vision is the field of explainable/interpretable machine learning. While computer vision models are often correct in their predictions, the exact reasons for their predictions are vague, uncertain, and sometimes outright wrong (Taylor and Taylor, 2020). This has motivated research in explainable artificial intelligence (XAI) to provide explanations for models' behaviour. In computer vision models, these explanations most commonly take the form of a natural language phrase (Hendricks et al., 2016; Chen et al., 2017; Kim et al., 2018) or a saliency map (Park et al., 2018; Hendricks et al., 2018; Hu et al., 2018).

2.2.2 Abstract Visual Reasoning

Beyond VQA, a separate class of computer vision tasks formulated to address higher-order visual cognition involves abstract visual reasoning. These tasks test the ability of computer vision models to develop abstractions of visual data and the reason about these abstractions. A common task for abstract reasoning aims to mimic the human IQ test of Raven's Progressive Matrices (Carpenter et al., 1990). Each reasoning problem consists of eight context panels arranged in 3×3 matrix with the bottom right panel missing. A set of eight candidate panels is provided. The model is required to select the candidate that best fits the missing panel based on the visual relation present in the context panels. The challenges with reasoning in progressive matrices are discussed in further detail in Section 2.2.3 and Chapter 5.

Another important test-bed in abstract reasoning is that of human-level learning of novel visual concepts and reasoning about them (Lake et al., 2015; Nie et al., 2020). In this task, the computer vision model is expected to learn, distinguish, and generate visual concepts based on few examples (few-shot learning). An important difference from human learning is that the motor or computer program used to generate the symbols is available for the models to train on. Abstract reasoning tasks are not as well studied in computer vision as VQA but provide a great test bed for exploring the higher-order cognitive abilities of analysis, synthesis, and evaluation.

2.2.3 Systematic Generalization

Abstract visual reasoning problems and datasets such as the PGM dataset (Barrett et al., 2018) (Figure 2.6) and the RAVEN dataset (Zhang et al., 2019a), were created with test generalization splits of the images based on unseen objects, attributes and relationships. This is an important test scenario for deep learning to be successful at higher-order cognition tasks as it requires the ability to generalize to out-of-distribution (OOD) data. The ability of deep models to generalize to unseen test data is not yet understood very well theoretically (Zhang et al., 2016; Xu et al., 2020). However, what is clear is that this ability is limited to testing their performance on independent and identically distributed (IID) test sets. Deep learning models tend to overfit the distribution from which the training data is drawn (Geirhos et al., 2018, 2020). While this overfitting is diagnosable when the model overfits to the exact training set, it also leads to strong generalization performance when testing on IID data if the overfitting is on the data distribution. However, this generalization performance often breaks down in the face of OOD test data with minor changes from the training distribution. An example of this would be a self-driving car performing well in train and test scenarios in summer weather, and performing poorly in test scenarios in rainy or snowy weather.

Borrowing from its use in linguistics (Lake and Baroni, 2018), Bahdanau et al. (2018) popularized the evaluation setting of systematic generalization in machine learning. The

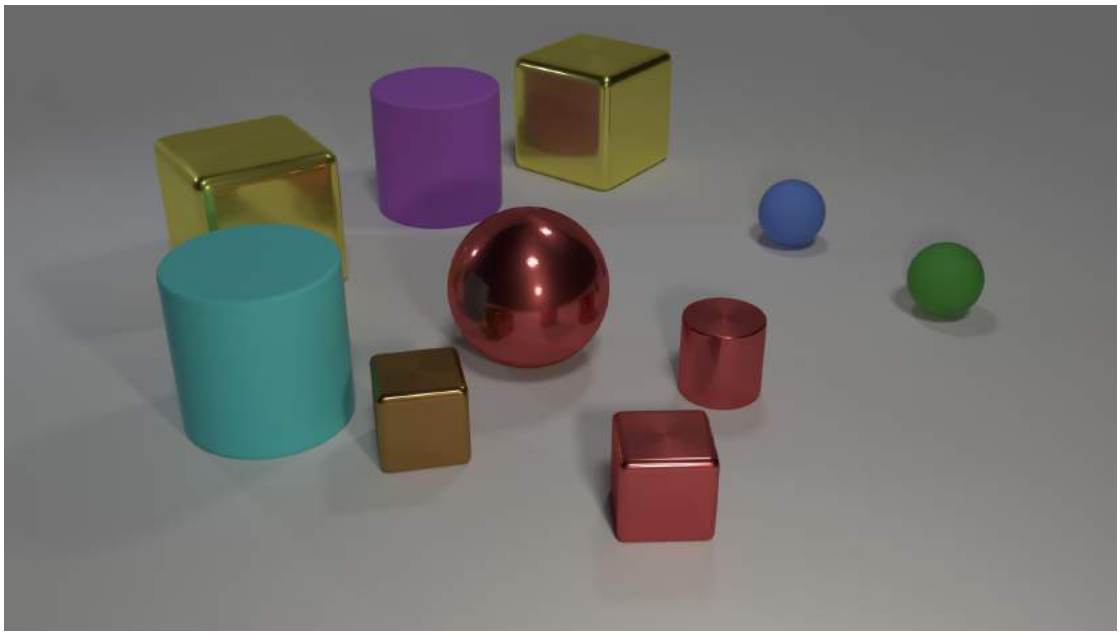


Figure 2.7: An image from the CLEVR dataset (Johnson et al., 2017a). As seen in the image, objects in the dataset have several attributes such as color, size, material; and are arranged in specific spatial layout in each image. Questions in CLEVR test various aspects of visual reasoning including attribute identification, counting, comparison, spatial relationships, and logical operations. Systematic generalization is required to understand these reasoning concepts across object and attribute values.

idea behind systematic generalization states that the meaning of a high-level semantic concept, e.g. 'red traffic light', can be derived systematically from the composition of the low-level semantic concepts it is made up of, 'red' and 'traffic light'. Furthermore, systematic generalization also involves 'productivity' and 'localism' in understanding concepts, e.g. generalizing from 'Boy's dog' to 'Boy's dog's ball' and vice versa. Systematic, or compositional, generalization is a fundamental property of human thought, and as a consequence, human language (Fodor, 1975). Systematic generalization enables humans to understand novel combinations of concepts by combining previously known concepts (Fodor and Pylyshyn, 1988). For deep learning models to learn from a finite set of training concepts and then generalize to a combinatorial set of test concepts they must be able to generalize systematically.

In computer vision, the idea of systematic generalization was first introduced in joint

vision and language based tasks, like visual question answering. The CLEVR dataset by Johnson et al. (2017a) was a procedurally generated dataset of images and questions designed to measure the ability of VQA models to reason compositionally. The CLEVR dataset can be split in multiple ways to test generalization on concepts like attributes and quantity and relationships like comparison. A VQA model that only memorizes visual correlations will not be able to generalize to the novel scenarios raised in the corresponding questions as shown in Figure 2.7 since they do not co-occur in the training data. Since CLEVR, several other VQA datasets like compositional-VQA (Agrawal et al., 2017) and GQA (Hudson and Manning, 2019) have been created that test the ability of models to generalize in a systematic manner.

In this thesis, we look particularly at the problem of systematic generalization as it applies to abstract visual reasoning tasks in computer vision. Detailed discussion on the differences between the training and test data distributions and how it measures for systematic generalization is provided in Section 5.2

Chapter 3

Inductive Biases in Deep Learning and Computer Vision

Now that we have discussed the fundamentals of deep learning, we will focus our attention on the role of inductive biases in deep learning and computer vision. This chapter begins with a introduction of inductive biases. We next discuss in detail the priors introduced by the convolution operator and how they enable the success of CNNs in computer vision. Next, we introduce the central theme of this thesis, the dynamic inference inductive bias and conclude with a discussion of two relevant use cases of this inductive bias: **early-exit** models and **neural module networks**.

3.1 Inductive Bias

The aim of machine learning is to learn from past experiences, training data, to deal with novel situations, test data, related to past experience. This ability to perform well on unseen data is called generalization. To enable generalization there are certain assumptions that the designer takes while building a machine learning model, or its training procedure including both the data and the training algorithm. These assumptions are called an inductive bias or a prior. In other words, inductive biases are decisions used in modelling a learning problem and in formalizing the learning algorithm that exploit the specifics of the problem at hand to enable the model to perform better at generalization.

The importance of using inductive biases while building and training machine learning models can be understood from the **No Free Lunch Theorem** in learning theory. The No Free Lunch theorem relates the potential space of problems to the potential solutions (algorithms) that can be learned for these problems. Initially, the theorem was derived for search (Wolpert and Macready, 1995) and optimization (Wolpert and Macready, 1997) problems. A learning problem can framed as a search problem over the space of possible solutions (sometimes called hypothesis space), or as an optimization problem where the learner has to optimize over an objective function. This naturally led to the adaptation of

the “No Free Lunch" for learning problems Wolpert (1996).

The theorem states that the improved performance of any algorithm on one class of problems is offset by poorer performance on another class of problems. In mathematical terms, if the possible space of solutions:

1. is a probability function, then the computational cost of finding a solution, averaged over all problems in the class, is the same for any solution method.

2. has underlying structure, then the outputs of all procedures solving a particular type of problem are statistically identical.

To put it simply, the theorem states that if no assumptions are made on the probability distribution of possible objective functions, then there is no reason to prefer one learning algorithm over another (Kochenderfer and Wheeler, 2019). This means that on average, all learning algorithms are equally effective when averaged across all possible learning problems. In principle, if deep neural networks are much more effective that random predictions on problem type A, then there exists a problem type B where random predictions are better than deep neural networks.

What are the implications of the ‘No Free Lunch Theorem’ on how we build machine learning models? According to Domingos (2015), *“...the practical consequence of the no free lunch theorem is that theres no such thing as learning without knowledge. Data alone is not enough."* In fact, the ability to generalize for an unbiased model would be “no better than if it simply stored all the training instances and performed a lookup” as posited by Mitchell (1980). Given that there is no single best machine learning algorithm across all possible prediction problems, it necessitates the need to develop new learning algorithms as well as trying to understand what makes existing algorithms succeed on different problems (Brownlee, 2020).

As a result, machine learning algorithms used to model a learning problem are built with several inductive biases to constrain them to succeed at the problem at hand. Mitchell (1980) described five kinds of inductive biases that can potentially be used by designers of learning algorithms:

- Domain-knowledge of the problem, e.g.using CNNs to model grid-like data such as images, and RNNs to model sequential data like time series as discussed in Section 2.1.3.

- Intended use of generalization, e.g.adversarial training of models when the learning task is sensitive and prone to adversarial attacks (such as cybersecurity) (Bai et al., 2021).

- Knowledge about the training data distribution, e.g.training self-supervised approaches with the data augmentation of random crops and random color distortion in order to generate enhance contrastive learning on ImageNet images (Chen et al., 2020a)

- Simplicity bias, e.g.simple optimization procedures like stochastic gradient descent tend to learn simple neural network architectures (Arpit et al., 2017; Kalimeris et al., 2019).
- Analogy with previously learned generalizations i.e.if a system is learning a collection of related concepts (generalizations), then generalization on one might imply generalization on others. This bias is at play in the practice of transfer learning where model features learned on one problem (such as image classification) are used as-is or as starting off points for another problem (such as object detection) (Torrey and Shavlik, 2010; Bengio, 2012).

The research on designing better architectures, training regimes and learning tasks in deep learning is in many ways a search for better inductive biases. The possible space of priors that human operators can introduce on a learning algorithm goes vastly beyond Mitchell (1980)'s recommendations. Several inductive biases have been discovered, and several others engineered for deep learning to succeed at a huge range of intelligent tasks in the past. Next, we will discuss the role of convolutions and CNNs in processing visual data, and how this inductive bias enables deep models to understand images and videos.

3.2 What priors do Convolution and CNNs introduce?

3.2.1 Translation Equivariance and Invariance

In mathematics, a function $\mathbb{f}$ is considered equivariant to another function $\mathbb{g}$, if the following relation between the two functions holds:

$$\mathbb{f}(\mathbb{g}(x)) = \mathbb{g}(\mathbb{f}(x)) \tag{3.1}$$

Equivariance implies that the output of applying the function $\mathbb{f}$ to the output of the operation $\mathbb{g}(x)$, will be exactly the same as if the operation $\mathbb{g}$ were applied to the output of $\mathbb{f}(x)$. To make this concrete, consider a linear function of vector **x**:

$$\mathbb{f}(\mathbf{x}) = 3x_1 + 2x_2 \tag{3.2}$$

and consider a scalar multiplication function $\mathbb{g}$:

$$\mathbb{g}(y) = 7 * y \tag{3.3}$$

We can see that the linear function is equivariant to scalar multiplication as follows:

$$\begin{aligned} \mathbb{f}(\mathbb{g}(\mathbf{x})) &= 3(7 * x_1) + 2(7 * x_2) = 21x_1 + 14x_2 \\ \mathbb{g}(\mathbb{f}(\mathbf{x})) &= 7 * (3x_1 + 2x_2) = 7 * 3x_1 + 7 * 2x_2 = 21x_1 + 14x_2 \end{aligned} \tag{3.4}$$

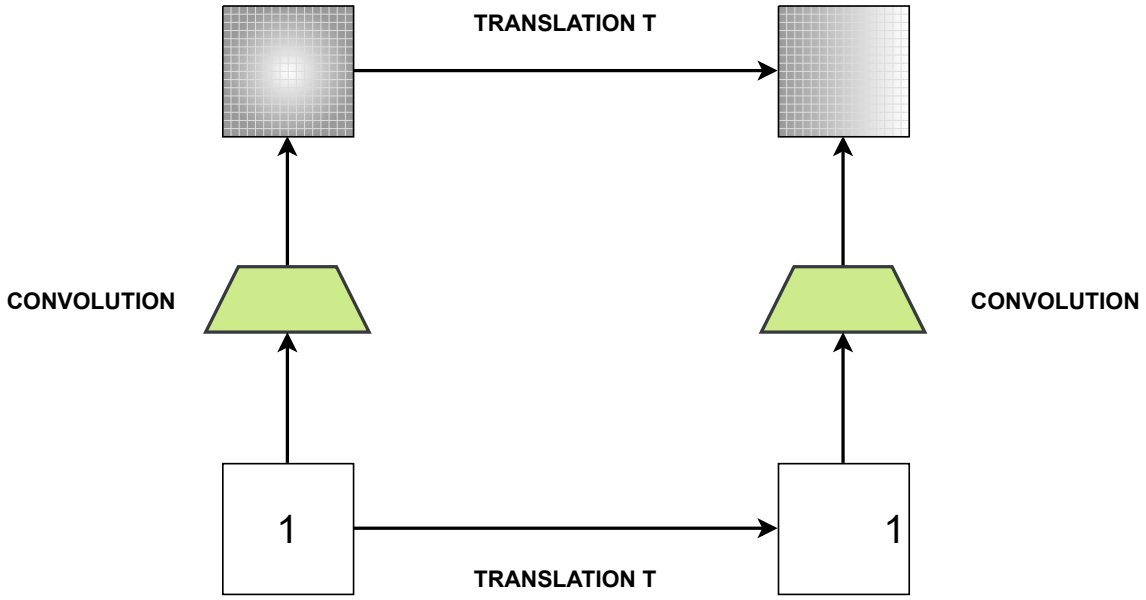


Figure 3.1: Translation equivariance in visual features learned by convolutional filters.

We know from the Universal Approximation Theorem that neural networks can learn to approximate any possible function to an arbitrary degree of accuracy. However, this also means that they technically could learn functions that are actually not helpful to solving an intelligent task. A common inductive bias in neural networks is to constrain the set of functions they can learn in order to reduce the function space explored by the learning procedure. The convolution operation does this by learning functions that are equivariant to translation.

We can see how this works for visual data in Figure 3.1. In the first instance, an object in the center of the image is processed by a convolution filter to generate visual features that are also centered in the middle of the feature map. Next, the object in the center is translated to the right in the image. Upon processing this translated image via the same convolution filter, we obtain a feature map that looks exactly the same as the original feature map translated to the right. This operation would have been thus equivalent to translating the original feature map to the right.

Since convolutions are equivariant to translation, they encode the translation symmetries present in data i.e.the features learned by convolutions for input data get translated with a translation in the input data. This is specially useful in visual data such as images, since we would want the model to learn similar features for an object irrespective of where it is present in the image. Translational equivariance enables convolutions to be robust to spatial translations of objects when generalizing to out-of-distribution data. Translation equivariance has been one of the central reasons for the success of convolutions on visual

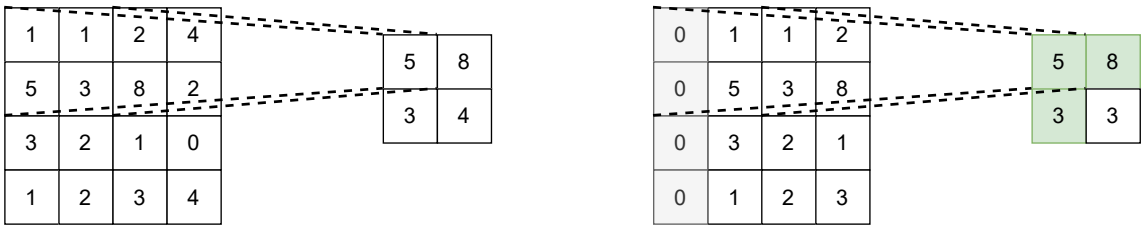


Figure 3.2: An example of localized translation invariance due to (max) pooling. The image (L) is translated by one pixel (R). For the given pixel distribution in the image, the outputs of 3/4 of max pooling operations remains the same i.e.invariant to the small translation.

data. We will discuss next how being translation equivariant lets CNNs get away with much fewer parameters than FFNNs.

However, before we do that, we consider another useful property of CNNs: translation invariance. In mathematics, a function $\mathbb{f}$ is considered invariant to another function g, if the following relation between the two functions holds:

$$\mathbb{f}(\mathrm{g}(x)) = \mathbb{f}(x) \tag{3.5}$$

Convolutions are not directly invariant to translations as we have seen: the features learned by a convolution filter translate with a translation in the input. This is a desirable property since we would want to preserve location specific information in the features learned by our network in order to enable spatial reasoning across object features. Even though convolutions are not directly implemented with translation invariance (Kayhan and Gemert, 2020), it is also desirable to have models that are invariant to small translations in data. Several vision tasks rely on determining whether a visual feature is present in the image and not on the exact location of the feature. Furthermore, being invariant to translations enables models with the ability to recognise an object irrespective of its location in the image.

In CNNs, convolution operations are often followed by pooling operations as discussed in Section 2.1.2. Pooling operations such as taking the maximum or average over a window of pixels are not a function of the location of the pixels. In fact, the output of the pooling operation are roughly translation invariant, as translating all pixels in an image by a small amount does not impact the maximum or average value of a pixel in the pooling window significantly (Goodfellow et al., 2016). This is visualized in Figure 3.2 where the output of three our of four max pool operation remains the same as before after a translation of one pixel. Hence, pooling operations impart a localized translation invariance to CNNs. This also enables the success of CNNs on computer vision tasks such as object recognition where translation invariance is desired.

3.2.2 Parameter Sharing

Translation equivariance in convolutions leads to an efficient sharing (or re-use) of the parameters in CNNs, as the same filter is applied across different spatial locations in the image, leading to learning the same features (with translations) across an image. Intuitively this is a sensible assumption for visual data as we want the model to learn similar features for similar objects irrespective of their physical location in the data. However, this efficiency in reusing parameters is useful beyond just learning similar features across spatial locations. It also leads to several practical consequences that enable the success of CNNs on visual data.

Before we discuss these implications, let us consider the case of the first convolution layer in the AlexNet (Krizhevsky et al., 2012) architecture to understand the magnitude of parameter efficiency in CNNs. This layer takes an input image of size $227 \times 227 \times 3$ and produces convolutional feature maps of size $55 \times 55 \times 96$ using $11 \times 11 \times 3$ filters. A fully-connected layer would require $227 \times 227 \times 3 \times 55 \times 55 \times 96 = 44,892,064,800$ parameters[1] to model this operation. In comparison, a convolution layer where the filters were not shared across the image i.e.each unit in the output feature map has its own filter will require a total of $55 \times 55 \times 96 \times 11 \times 11 \times 3 = 105,415,200$ parameters for this operation. Comparatively, the first layer in AlexNet uses $96 \times 11 \times 11 \times 3 = 34,848$ parameters to model this operation, which is less than fully-connected layer by an order of 10^6 and less than a convolution layer without parameter re-use by an order of 10^4. Across the depth of a several layer deep neural network this leads to an efficiency of the order of trillions of parameters when compared to an MLP and billions of parameters when compared to a CNN without parameter re-use.

As mentioned before, higher efficiency in utilizing the neural network parameters leads to several practical implications that directly relate to learning improved models. A non-exhaustive list of these benefits include:

- The number of parameters in a model determines the size of the learning problem as the loss function being optimized is a function of all the parameters, and is optimized by varying these parameters. A model having fewer parameters for a learning problem thus translates to a faster optimization step as fewer parameters need to be updated. This consequently leads to faster training as the computation performed in one optimization step requires fewer number of floating point operations.

- Models with fewer parameters require less physical memory to store and operate on. Since the physical memory is shared across the model and data in a processor, this means that each model can train on more data points per each mini-batch. Training with larger mini-batches has been shown to empirically improve the learning process. Smith et al. (2018) showed that increasing batch size in training leads to the

[1] We are ignoring bias units for simplification here. Please refer to CS231n (CS231n) where this example is borrowed from for the full calculations.

same effect as reducing the learning rate and leads to higher training and validation accuracy with fewer training epochs. Chen et al. (2020b) showed that training with very large batch sizes (4096) enables self-supervised learning of CNNs to reach a very high degree of accuracy in learning without labels.

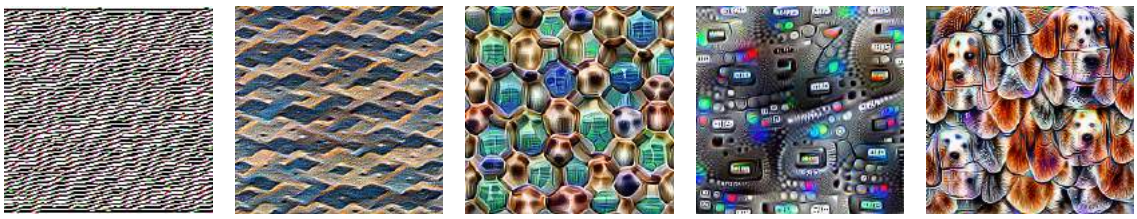
- Parameter efficiency of CNNs also enables us to design and train bigger neural networks for learning. In particular, deeper models with over a hundred or even over a thousand layers (He et al., 2016) were enabled by convolutions and shown to outperform shallower models on image recognition problems. Similarly, wider neural networks (Zagoruyko and Komodakis, 2016) were demonstrated to outperform narrow networks when it came to recognition tasks. Both types of increase in the scale of neural networks would not have been possible without the parameter efficiency of convolutions.

There are several other useful practical implications of parameter re-use in CNNs, and thus the inductive bias of sharing filters across spatial dimensions directly contributes to learning better models. Next, we will discuss how convolutions enable the learning of a hierarchy of visual features and how these features are useful in solving visual tasks.

3.2.3 Hierarchy of Visual Features

One of the most powerful inductive biases in deep neural networks is that their structure enables them to learn higher-level features in deeper layers that are composed of non-linear combinations of lower-level features. This ability, which has come to be known as **representation learning** (Bengio et al., 2013a), is the key differentiator between deep learning and other approaches to machine learning. The ability to learn a hierarchy of features in neural networks lends itself naturally to visual information as it can be deconstructed into lower-level features like edges and curves which can then be combined into complex features such as objects and shapes. In fact, studies in neuroscience and cognitive science have confirmed that the human brain does perform feed-forward computations of heirarchical features in order to perform visual object recognition (DiCarlo et al., 2012). This has driven research in understanding how CNNs learn visual features from images and if there are similarities to human vision in this process.

One approach to understanding the visual features composed by a CNN is visualizing the features learned across the network. Visualizations help humans understand the feature space of a CNN and compare it to visual features that we can understand. A common approach to visualizations is activation maximization (Nguyen et al., 2019), where an input image that maximally (or minimally) activates a particular unit (neuron, feature map, feature maps) of a neural network is generated by optimizing the input and keeping the parameters of the network fixed. This helps isolate the visual features that are responsible for activating the different units in a neural network while controlling for confounds that are present in natural images.



(a) Layer Conv2d0 Edges (b) Layer Mixed3a Textures (c) Layer Mixed4a Patterns (d) Layer Mixed4c Parts (e) Layer Mixed4c Objects

Figure 3.3: A sample of images that maximally activate neurons in Inception v3 from shallow to deeper (L-R) layers. We can see that the initial layers get activated by low-level visual features like edges (horizontal) and textures, while deeper layers are activated by higher-level visual features like object parts (car dashboards) and objects (dogs). Adapted from Olah et al. (2017).

Olah et al. (2017) conducted an extensive empirical investigation of the features learned by a CNN, Inception v3 (Szegedy et al., 2015), through activation maximization. In Figure 3.3 a sample of figures they generated via maximizing the activatation of one neuron in different layers of this CNN are shown. It can be seen that the initial layers of a CNN learn low-level visual features like edges (e.g.horizontal edge) and textures (e.g.grids), which slowly make way for higher-level features like patterns (e.g.hives), object parts (e.g.dashboards), and objects (e.g.dogs) in deeper layers. While these instances are cherry-picked, the general consensus about CNNs learning a hierarchy of visual features that corresponds (roughly) to how human vision recognizes visual features has been validated across multiple visualization methods and multiple CNN models (Qin et al., 2018).

The ability of CNNs to learn these visual features without explicit supervision from data can be explained based on two of the inductive biases used in the design of CNNs. The first bias is the use of the convolution operator itself. The convolution operation can lend itself naturally to the operation of edge detection (Marr and Hildreth, 1980). An edge in the visual domain can be understood as a sharp change in the pixel intensity of the image (Figure 3.4). Thus, the gradient of the intensity gives the direction perpendicular to an edge. The gradient can be approximated using a finite difference operation, which is a linear and spatially equivariant operation and thus can be obtained using a convolution. A concrete example of how this works is shown in Figure 3.4. Thus, convolution filters can operate as edge detectors and the features obtained after a convolution can represent an image's edge features.

The next inductive bias that leads to a hierarchy of visual features is the use of deep neural networks in CNNs. All shapes can approximated locally as straight lines, and thus coarse image features like textures and shapes can be interpreted as combinations of edges. Once the initial layers of a CNN have learned to perform edge detection, the later layers which compose linear functions of these lower layer features can learn such features. This

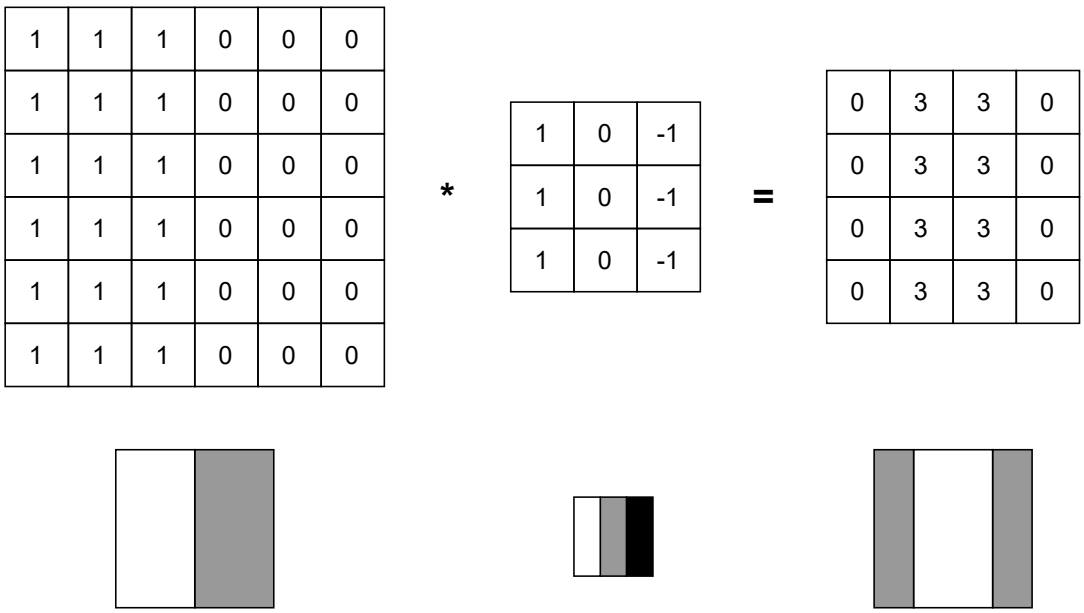


Figure 3.4: An example of vertical edge detection in an image using a convolution filter (L) Original image with an edge running down its middle, (M) a convolution filter, (R) Output of convolution with the center containing the features representing an edge

results in the learning of convolution filters which detect sequentially higher-level features in the layers of CNNs and lead to the hierarchy of features as seen in Figure 3.3. Thus, these two priors of performing a convolution and using deep neural networks combine to learn features that are similar to how human vision builds visual features and leads to superlative performance on computer vision tasks.

Now that we have seen several ways in which convolutions and CNNs help in learning features that are better suited to solve visual tasks, we shift our attention to a different inductive bias: dynamic inference. We introduce this bias, explain the different approaches to implementing it in deep neural networks, and discuss two different manifestations of it in the next section.

3.3 The Dynamic Inference Inductive Bias

Most machine learning models are built to perform a fixed amount of computation given an input. This is unlike humans whose cognitive effort spent on a problem varies depending on the complexity of the problem at hand (Donders, 1868). Dynamic inference, also called conditional computation (Bengio et al., 2013b), or input-adaptive inference (Hu et al., 2019), is an inductive bias that enables machine learning models to perform a different amount of computation at inference time. The amount of computation is determined either by an operational budget in terms of number of floating point operations (Huang

et al., 2018) or by the input to the model (Kaya et al., 2019). The machine learning models built with this inductive bias can be equipped to have several benefits which are not possible in standard models. We propose that these benefits could possibly borrow from features of human cognition and enable higher-order cognitive abilities in computer vision.

Perhaps the most well-known example of a model with the dynamic inference bias is the decision tree (Quinlan, 1986) model. Decision trees are made up of a learned rule-based nodes and the model performs a different traversal of the rule tree for each input. Thus, one input may be processed through a shorter root to leaf path than the other enabling decision trees to perform dynamic inference based on the input data. The structure of decision trees, as well as their dynamic inference nature, lends itself very well to explainability (Molnar, 2020), since humans have an easier time understanding branching decisions versus say high-dimensional features in a deep model. This has naturally led to their use in computer vision for explainability (Si and Zhu, 2013).

Despite their utility in explainability, decision trees are not widely used in computer vision since they present a huge computational cost in learning visual features. As discussed previously, deep neural networks have become the de facto approach to computer vision because of their exceptional ability to learn visual features from data. However, they also suffer from major drawbacks such as not being very interpretable (Taylor and Taylor, 2020) or learning dataset biases which prevent generalization (Geirhos et al., 2020). We propose that the ability to perform dynamic inference can be utilized to address some of these drawbacks.

However, deep neural networks are generally not engineered to perform dynamic inference by default. Each network operates as a series of matrix multiplications on a batch of inputs, and training the network involves a series of parallel gradient descent steps and matrix multiplications for each batch. As a result, performing dynamic inference presents a challenge in terms of sub-optimal performance on current software and hardware infrastructure. This is because a dynamic inference model performs different operations on each input in the batch that cannot be parallelized as easily with current infrastructure.

While they might be harder to get working due to engineering limitations, dynamic inference models are very flexible. They allow for a very fine trade-off between performance and resource efficiency at both training and inference time compared to other approaches like distillation (Hinton et al., 2015) or pruning (Molchanov et al., 2019). As a result, dynamic inference based deep models are quickly becoming a popular approach to resource-efficient computer vision and machine learning. During inference time these models allocate more computational resources and processing time to examples based on their difficulty such that harder examples receive more compute while easier example receive less compute.

There have been several attempts to engineer deep learning models that can perform dynamic inference (Wu et al., 2018; Bengio et al., 2013b; Graves, 2017; Wang et al., 2018; Zhang et al., 2019b; Kaya et al., 2019; Phuong and Lampert, 2019). We try to classify these approaches into a few general directions based on their approach:

- **Layer Dropping:** Deep learning models can introduce intermediate classifiers to allow the model to stop processing if a pre-specified criterion is met. Examples of this approach are the Branchy-Net (Teerapittayanon et al., 2016) and Shallow-Deep Networks (Kaya et al., 2019). Layer dropping models are discussed in further detail in Section 3.3.1.

- **Layer Skipping:** These models allow the skipping of certain layers either by introducing skip-connections or implementing controller modules for the flow of the input through the model's computation graph. The Block-Drop (Wu et al., 2018) and SkipNet (Wang et al., 2018) models are examples of layer skipping.

- **Channel Skipping:** These models are build to enable the skipping of different channels in each layer through. The most well-known example of a channel skipping approach to dynamic inference would be Slimmable Neural Networks (Yu et al., 2019).

- **Dynamic Computation Graph:** While the previously discussed strategies for dynamic inference exploit architectural nuances in networks, the most general approach to building dynamic inference models is to have a dynamic computation graph of the model arranged for each input. Several models that focus on reasoning and building compositional representations. For example, neural module networks (Andreas et al., 2016; Johnson et al., 2017b) and Tree-LSTM (Tai et al., 2015) take this approach to enable dynamic inference. While this approach is the most general, it comes with added limitations since efficient batching becomes an additional challenge for training these models (Neubig et al., 2017; Agarwal, 2019).

Although dynamic inference was conceived of as an inductive bias to enable performance-efficiency trade-off in deep learning models, researchers have started utilizing it for other purposes. For example, Hu et al. (2019) showed that besides improving efficiency, training with input-adaptive inference can be used to boost accuracy and adversarial robustness. We discuss a few other uses of dynamic inference outside of computational efficiency in Section 3.3.1. Building on this line of work, we aim to utilize this inductive bias as a mechanism to introduce priors for higher-order visual cognition in deep learning models. We achieve this by exploiting two different approaches to the dynamic inference inductive bias:

1. We use the early-exit prior (Section 3.3.1), which encompasses both layer dropping and skipping models, to serve as a probe into intermediate visual representations in a CNN. This helps us to introduce a new measure of explainability for these representations.

2. We use the modular prior (Section 3.3.2) to build models which can learn compositional visual representations for objects, attributes, and relationships. This helps us

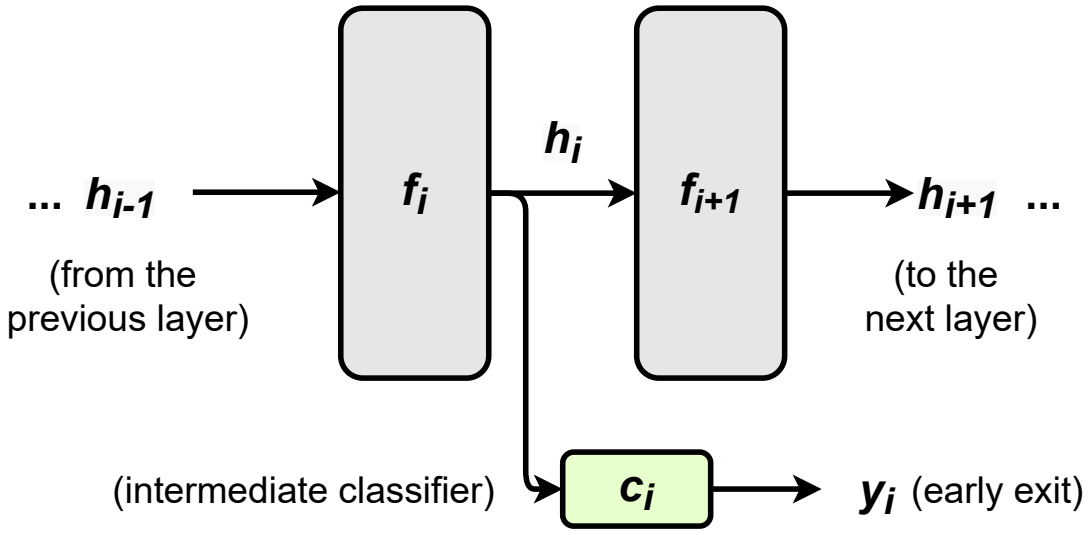


Figure 3.5: An early-exit in a deep neural network

to perform abstract visual reasoning in a manner that is able to generalize systematically.

We will next discuss these two approaches to the dynamic inference prior in some more detail.

3.3.1 Early-Exit Models

Since deep neural networks are made of sequential matrix computation operations (layers), a natural way to introduce dynamic inference in models is to halt the sequential computation at different intervals (depths) of the network. This approach to dynamic inference gives rise to early-exit models. Early-exit models (Huang et al., 2018; Phuong and Lampert, 2019; Zhang et al., 2019b; Hu et al., 2019) are obtained by adding auxiliary/intermediate classifiers between neural network layers. The 'early-exit' circuit gets triggered based on when a pre-specified criterion is fulfilled. In most models, this criterion is set to be the softmax value of classification. As soon as the highest softmax value crosses a threshold for an intermediate classifier, the output of the network is returned from that classifier and no further computation is done.

The reason behind using softmax outputs from early-exit networks as criterion for conditional computation is that they serve as proxies for probability of a class as well as confidence in the results. A threshold softmax value being met denotes that the network is predicting with high confidence that it knows the correct answer. However, Guo et al. (2017) showed that softmax values in deep neural networks are over-confident, i.e., the

softmax value predicting probability estimates is not a representative of the likelihood of the class label being correct. This drawback of softmax outputs has triggered more research into alternative 'early-exit' criteria. 'Patience' was proposed as one such criterion, where the change in values across intermediate classifiers is noted, and once the change is below a patience threshold the early-exit is triggered (Zhou et al., 2020). While the exact mechanism for triggering early-exits can vary, the general idea behind it remains the same: assigning more compute to harder data points and less to easier data points.

Early-exits provides a way to reduce computation in a deep neural network. As result, early-exit models are primarily aimed at efficient machine learning use cases. However, there have been several other use-cases of having early-exits in neural networks. One of the most well-known use of early-exit models is the Inception network (Szegedy et al., 2015). The Inception network used the outputs from early-exit intermediate classifiers to calculate training losses. These losses were then used as additional training signals for the Inception network allowing for better training of very deep models compared to a single-exit network. More recently, Kaya et al. (2019) demonstrated the phenomenon of 'overthinking' using early-exit networks. Overthinking is the phenomenon in deep CNNs where the neural network reaches the correct classification in the intermediate layers but returns a wrong output upon further processing. The authors used early-exit classifiers as probes to investigate the overthinking problem in neural networks. We build upon this line of work in our own research and use early-exit models for explainable artificial intelligence.

3.3.2 Neural Module Networks

The dynamic inference inductive bias often takes the form of a compute budget per input or across a dataset in order to maximize resource-efficiency. However, dynamic inference can also be utilized in designing neural network layouts to match the problem posed by the input. This is specially useful in reasoning based learning problems, such as VQA. It has been shown theoretically that a network whose structure aligns algorithmically with a reasoning problem is able to learn and generalize with fewer training samples (Xu et al., 2020). This utility in matching input problems to their corresponding neural network layout is the idea behind the modular approach to dynamic inference.

The neural module network framework was first introduced for visual question answering tasks (Andreas et al., 2016). The authors designed small neural networks which they term neural 'modules' to serve as building blocks of their framework. Each module takes as input a combination of the feature maps of the image and question words to achieve a particular lower-order visual task. For example, the 'attend' module takes an image feature map and object label as input, and returns a saliency map with the object highlighted. The authors hypothesized that individual neural network modules are good at performing lower-order recognition of features as well as transformation of features. Bringing together these neural modules into a dynamic neural network, akin to combining LEGO blocks, enables them perform more complex tasks such as reasoning for VQA since these tasks can be realized as a composition of several lower-order tasks.

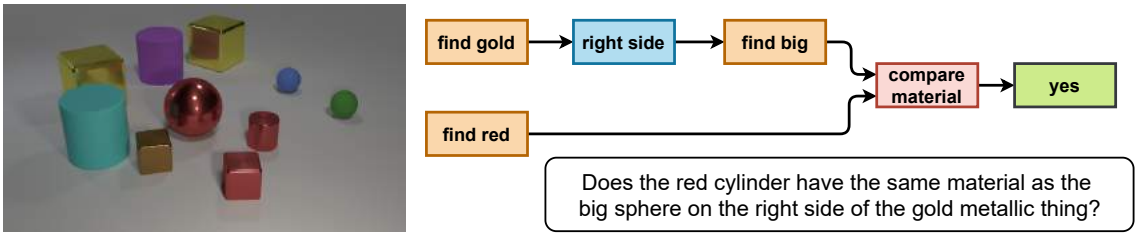


Figure 3.6: A neural module network for VQA, arranged to answer the question “Does the red cylinder have the same material as the big sphere on the right side of the gold metallic thing?". Each module (shown as different colored boxes) consists of a learnable function implemented via a neural network. Modules share parameters across function instantiations for different objects, attributes, and relations and the module network is arranged based on the question’s dependency parse. This flexibility allows them to generalize systematically to out-of-distribution compositions of objects, attributes, and relations.

For VQA, the individual modules represent different neural networks aimed at performing specific tasks in the reasoning process, such as object recognition and counting (Fig. 3.6). The layout of the full model is obtained by first generating a universal dependency parse (De Marneffe and Manning, 2008) of the question using the Stanford parser (Klein and Manning, 2003). Each node in the parse tree corresponds to a neural module in the network layout whose inputs are determined by the operand of the node. The resulting neural network is trained using only the final answers of the natural language question as the training signal. The neural module network learns the function utility of its modules during the training process as it is not fixed by the designer. This leads to great flexibility in learning these modules and consequently in the possible neural module networks that can be built out of them. As a result, neural module networks have been empirically shown to outperform fixed neural network architectures (Bahdanau et al., 2018, 2019) when it comes to systematically generalizing to out-of-domain reasoning problems.

The flexibility of the neural module networks is constrained by relying on a fixed non-differentiable parser. In order to address this limitation, Hu et al. (2017) did away with using a fixed parser in order to generate the semantic dependency parse for the questions. Instead, the authors used a differentiable Sequence-2-Sequence (Sutskever et al., 2014) model to learn the transformation from a natural language query to a semantic layout for the modules. The switch from using a fixed parser to a learnable neural parser enabled the entire pipeline of neural module networks to be end-to-end trainable. Since both the layout of the modules as well as the neural modules themselves could now be learned from data, the modular inductive bias became even more general. This led to neural module networks being adapted for tasks outside of VQA such as explainability (Hu et al., 2018) and visual grounding (Liu et al., 2019).

In order to further simplify the use of neural module networks, Johnson et al. (2017b)

proposed to remove the priors introduced on different modules by setting their input and output types. The original module network consisted of some modules which received images and text as inputs, some which only received text, and yet others which only received saliency maps. Instead of having specialized neural modules for different tasks, a general approach to defining modules based on their input arity was proposed (Johnson et al., 2017b). Thus, only three types of modules remain. Unary modules receive a singular intermediate feature vector and question word as inputs. Binary modules receive a question word and two intermediate features and operated on their combination. Scene modules take as input the entire visual context. Despite having simplistic modules, modular networks were able to learn complex function representations and adapt to the semantic purpose as required by the dependency parse of the question. In fact, neural module networks with simple modules that relied only on the input arity of the function they modelled were shown to be better at generalization in VQA (Bahdanau et al., 2018, 2019) when compared to specialized modules that relied on knowledge of the task performed by the function being modelled.

Chapter 4

Neural Response Time Analysis

The majority of techniques developed for XAI depend on privileged access to the architecture and parameters of the model in question (Rahwan et al., 2019). If XAI as a field is to provide satisfying explanations for decisions and behaviours to end users, researchers will need ways to generate explanations from "outside" the black-box — without having to inspect the model internals, rather, by interacting with it by curating inputs and observing outputs. Explanations from outside the black box are desirable because they empower any user to investigate the cause and consequence of otherwise inscrutable model processes.

The black-box problem in XAI is similar to the challenge faced in building explainable models of an analogous black box — the human mind. Some machine learning researchers have already begun to model AI decisions using methods from cognitive psychology (e.g. (Ritter et al., 2017; Bojarski et al., 2017; Leibo et al., 2018; Geirhos et al., 2018; Kim et al., 2019; Roig et al., 2018; Osband et al., 2019; Gulordava et al., 2018; Rajalingham et al., 2018; Richard Webster et al., 2018; Henderson and Serences, 2020).) Cognitive psychology is characterized by inferring hidden mental processes from observed behaviour. XAI without privileged model access must also infer hidden processes from output, so cognitive psychology's methods are attractive when explanations are desired from outside the black box.

In humans, an easily-measured behaviour that corresponds to the complexity of mental operations is response time (RT) (Donders, 1868; Sternberg, 1969). For example, difficult visual search tasks take more time (Treisman and Gelade, 1980). RT methods could in principle be adapted to machines with only a simple system clock and stimuli curated to test a given hypothesis. However, unlike human brains, most deep learning models for computer vision (e.g. (Simonyan and Zisserman, 2014; He et al., 2016)) perform a fixed number of operations over a static interval of time between receiving an input and yielding an output. This results in an uninformative RT distribution.

A new class of "dynamic inference" models offers a potential use case for RT methods. As deep learning models continue to grow and are deployed in resource and time-critical environments, there is a growing push towards models which can adapt their processing dynamically based on input complexity. This feature is desirable in embedded

systems and autonomous vehicles, among other things. Initially proposed to address the issue of resource-efficiency (Bengio et al., 2013b), the scope of dynamic inference models has quickly grown to enable several desirable properties, such as semantic layout enabled models (Andreas et al., 2016), adversarial robustness (Hu et al., 2019), or mitigating overthinking (Kaya et al., 2019). Not only does this class of models have a performance and resource advantage, they can provide sufficient variance in RT to draw meaningful conclusions for XAI. By modelling when to exit successfully in a hierarchical framework, dynamic inference models have "baked-in" interpretability into their decision making. Our approach, which we call Neural Response Time Analysis, exploits this underappreciated property of this class of models.

Given a dynamic inference model, our technique works by measuring only the softmax output and system clock time elapsed for a given input known to contain certain features. Thus, NRTA is agnostic to model architecture, its implementation, and the dynamic inference strategy used, making it adaptable to any use-case where variable RTs are available. We then analyse RTs recorded in response to carefully controlled input stimuli using both quantitative comparisons in values and trends of the NRT as well as by performing hypothesis testing on the nature of response across these stimuli.

Our first experiment compares RT profiles for input from ImageNet versus ObjectNet (Barbu et al., 2019), which was designed to contain non-stereotyped features. Results from this experiment provide a useful proof of concept for using RT to make inferences about the complexity of a model's composed feature space. In addition, we demonstrate how intermediate classifiers can be trained and appended to any standard model after training, broadening the applications of RT analyses. The first experiment shows that RT can reliably detect differences in feature complexity. The second experiment makes specific predictions about how dynamic inference models will process stimuli from the SCEGRAM dataset — a set of images specifically designed to display different values of the high-level, abstract notion of scene grammar studied in humans (e.g. (Vo and Henderson, 2009)). We perform *a priori* hypothesis testing for when RT is and is not sensitive to scene grammar. The RT effects in Experiments 1 & 2 are observed despite insensitivity in accuracy measures, demonstrating that RT profiles can be more informative than item difficulty. This second experiment shows that we can test hypotheses about the relative complexity of composed features simply by observing differences in RT. Finally, we show that inputs with shared features exhibit RT profiles that follow similar trajectories. As such, RT profiles allow inferences about the model's feature space and can describe more than the difficulty of a prediction task.

4.1 Related Work

This work bridges several sub-fields within psychology and artificial intelligence. We briefly outline them in this section.

4.1.1 Artificial Cognition for XAI

Most XAI techniques depend on privileged access to the model being explained. Some of these popular techniques involve: training an adjacent model to explain another using captioning (Anne Hendricks et al., 2018), visualizing the gradient of the loss function with respect to the input at the penultimate layer (Simonyan et al., 2013), optimizing random input to visualize the "preferred" stimulus of a given neuron or layer (Simonyan et al., 2013; Olah et al., 2017), or systematically perturbing input to see which elements affect the output (Zeiler and Fergus, 2014; Ribeiro et al., 2016). Although these methods are indispensable and ingenious, they share two vulnerabilities: (1) they can be difficult to implement from "outside" the black box, and (2) because they explore the space of explanations iteratively and automatically, interpretation depends on many observer degrees of freedom, whether through hyperparameter selection or through confirmation bias of the interpreter.

A complementary approach to XAI, adapted from experimentation in human cognitive psychology, shows how explanations can also be generated by testing specific *a priori* hypotheses of how models ought to behave under certain assumptions. Observing model behaviour under fair attempts to falsify the hypothesis results in intuitive explanations for how the model behaves. This is exactly how cognitive psychologists approach explaining the human mind, and it can be re-purposed for explaining any black box. We form a falsifiable hypothesis for how the model ought to behave under a set of contrived and highly controlled experimental stimuli that vary only along the experimental dimension.

Studies inspired by the cognitive approach to understanding the mind are becoming more common in XAI. Ritter et al. devised an experiment to determine whether image classification models would rely on shape rather than colour — like humans — for one-shot object recognition (Ritter et al., 2017). Using two sets of matching stimuli: one matching in shape of the familiar object vs. other matching the color (and controlling all other low-level features), they found the model to more reliably identify the shape-matching stimulus as belonging to the same category. This was interpreted as demonstrating shape-bias in object learning. A similar line of hypothesis-testing driven research contrived different variations of same input either with/without features highlighted by a saliency algorithm for their PilotNet model, while controlling other variables (Bojarski et al., 2017). PilotNet is a NN-based computer vision model for autonomous driving that takes images of the road ahead as input and outputs steering angle. PilotNet responded to untampered input in a similar manner as the salient input alone, and it did not respond to input with salient information removed. The behavioural results clearly supported the explanation that the information discovered by their saliency algorithm did in fact control steering. Geirhos et al. used a similar hypothesis-testing approach when they created a stimulus set that directly pitted texture and shape cues against each other to observe whether CNNs overindex on either dimension preferentially (Geirhos et al., 2018). By creating mash-up stimuli that contained the global shape of one input and the texture of another, the CNNs were shown to reliably identify the stimuli according to their texture class.

Whereas most XAI techniques explore the range of possible explanations automatically

(e.g. through optimizing for preferred input or systematic perturbation), we pit alternative explanations against each other from "outside" the black box by measuring response times. Core to the approach is the process of contriving a test that may discern between competing hypotheses. We use stimulus sets that are known to contain different features, specifically non-stereotyped viewpoints (Experiment 1) and syntactically or semantically inconsistent object-object relationships (Experiment 2). We assume that there is a correlation between the hierarchical feature complexity learned by a neural network layer and its depth. Consequently, we can attribute differences in RTs between conditions of our experiment to these features since RT distribution is a function of the depth at which the discerning feature will be utilized by the network.

4.1.2 RT methods for explaining human behaviour

RT is an easily-measured external behaviour for humans that corresponds to the complexity of mental operations (Donders, 1868). In the original demonstration from 1868, Donders showed that repeating a prepared syllable was about 75 m/s faster than repeating an unknown syllable. The mental process of classifying an incoming syllable takes a human neural network an additional serial processing step requiring 75 m/s. Although Donders' Subtractive Method has seen some major revisions in the intervening 150 years (e.g. (Sternberg, 1969; Sternberg et al., 1973)), the core logic persists. Assuming there is a relationship between processing time and hierarchical feature space or task complexity, we can apply a similar approach to the inference of deep learning black box processes. However, unlike human brains, most DNNs for computer vision perform a fixed number of operations over a static time interval, resulting in a uniform and uninformative RT distribution. If we want to use RT methods to explain DNN behaviours, we require a *distribution* of RTs.

4.1.3 Input-Adaptive Dynamic Inference

Input-adaptive dynamic inference, also called conditional computation, or simply adaptive/dynamic inference, is an emerging paradigm for learning resource-efficient deep learning models with representative works proposed in (Wu et al., 2018; Bengio et al., 2013b; Graves, 2017; Wang et al., 2018; Wu et al., 2018; Zhang et al., 2019b; Kaya et al., 2019; Phuong and Lampert, 2019). These models allocate more computational resources and processing time to harder examples and less to "easy" examples during inference time. Thus the models do not need to sacrifice accuracy for efficiency at training time. Dynamic inference has become an increasingly popular approach to resource efficient ML due to its flexible speed-accuracy trade-off.

Among dynamic inference models, early-exit models (Huang et al., 2018; Phuong and Lampert, 2019; Zhang et al., 2019b; Hu et al., 2019) operate by adding auxiliary classifiers to intermediate layers. This allows for "early" and fast exits from neural network computations for simple instances, and later exits for harder instances. Huang et al. (2018)

utilized multi-scale features for early-exit models to reach near state-of-the-art performance on ImageNet (Deng et al., 2009) and other standard image recognition tasks. While most prior works have utilized softmax-based confidence scores for choosing their early exit criteria, more recent works explored other criteria like patience or change in intermediate predictions (Zhou et al., 2020), and surprise or negative log-likelihood from a secondary auto-regressive model (Lugosch et al., 2020). The early exits can be used as probes for interpretability to identify and mitigate "overthinking", the phenomena where deep neural nets reach correct decisions in intermediate classifiers but a wrong decision on further processing (Kaya et al., 2019).

Most of the prior work on dynamic inference is focused on either designing efficient architectures (Huang et al., 2018) or better learning algorithms (Li et al., 2019) that exploit input-adaptivity. Our approach is an orthogonal effort to explore the input-adaptive inductive bias as a tool for interpretability. For our purposes, dynamic inference produces the key requirement for using RT methods: a distribution of response times corresponding to model depth and abstraction of feature space. We limit the scope of our NRT analysis to confidence-based early-exit models. However, NRT analysis can directly be extended to other early-exit criteria which mitigate the issues with softmax-based confidence estimates (Guo et al., 2017) as well as other forms of dynamic inference models. Finally, while the scope of this study is black-box XAI for dynamic inference models, we also present a use-case for a universal extension of our technique. Based on (Kaya et al., 2019) we also add early-exit probes to a standard ResNet-56 architecture to extend our analyses to *off-the-shelf* neural networks.

4.2 Method

We studied the Multi-Scale DenseNet (MSDNet) (Huang et al., 2018) and Shallow Deep Network (SDN) (Kaya et al., 2019), specifically the ResNet-56 variant) as our reference early-exit models. In terms of number of layers, the auxiliary internal classifiers (ICs) were equally spaced across the depth of the model following the same procedure as described in (Huang et al., 2018) and (Kaya et al., 2019). In addition to a final classifier, MSDNet and SDN have a total of 4 and 6 of these ICs respectively.

All models were trained on ImageNet using a sum of cross-entropy losses across all ICs and final classifier unless otherwise mentioned. The top-5 test accuracies for each model is reported across all classifiers. MSDNet trained in this manner achieved accuracy of 80, 86.2, 88.6, 89.4, and 90.4%. For the SDN, we trained a standard ResNet-56 for 100 epochs, froze its parameters, added the ICs and trained only the IC parameters for another 25 epochs. This training strategy (IC-only) resulted in top-5 accuracies of 17.4, 20.6, 31.3, 34.9, 54.8, 67.8, and 80.2%. SDN was also trained end-to-end in a standard manner which led to an accuracies of 31.3, 35.2, 51.6, 59.6, 69.8, 76.9, and 78% respectively.

Algorithm 1: Time Series Step Function Generator

Require: $N \times K$ measurement matrix R reporting RT for each exit,
top-1 confidence $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_k, \ldots, \theta_K]$
for each classifier k,
confidence step size η

Result : $N \times S$ matrix R' representing RT to reach each confidence level

```
for all images, n
  for all steps, i
    for all auxiliary classifiers, k
      if θ_k > (i * η)
        R'_{n,i} = R_{n,k}, break
      else if max_k θ < (i * η)
        R'_{n,i} = R_{n,K}
```

4.2.1 Measuring Neural Response Times

The models were implemented with the PyTorch (Paszke et al., 2019) framework. To ensure that the RT statistics were model and implementation independent, we did not query the internals of the model architecture for calculating timing values. Instead, we relied entirely on the system clock time elapsed during the forward pass of a test input. The system used to perform the RT calculations had a 2 × E5-2620 v2 Hex-core processor (12 , physical CPU cores) and 128 GB RAM. Inference was performed on each input stimuli using two 11 GB NVIDIA 2080Ti graphics processing units. We avoided spurious correlations in RTs across stimuli by passing each input individually during test time, as is commonly done for evaluation of dynamic inference methods.

The models decide to make an “early exit" based on some confidence threshold or an anytime prediction (Huang et al., 2018) at an arbitrary time. The classifiers' top-5 softmax estimates were used as a proxy for confidence; if the estimate was greater than the confidence threshold, it was recorded as sufficiently confident for an early exit. Thus, the classifier's RT was recorded to be equal to the clock time at the moment of that classifier's output minus the initial time. One of the core assumptions in our approach to RT methods is that this time will correspond directly to the number of operations. As each layer and classifier within our model performs a fixed number of operations, the RT to reach a decision for each classifier should be constant, with a small amount of error allowed for external variations in system processing. To verify this, we measured the correlation between FLOPs and RT to be $r = 0.996$. Instead of relying on exact measures of model operations, unavailable without access to the black-box, the RT measure is a very strongly correlated measure of the operations involved while only relying on the system clock time. Since the exact number of ICs is discernible from the number of discrete RT plateaus in the output, our method maintains a black-box approach for any dynamic inference model.

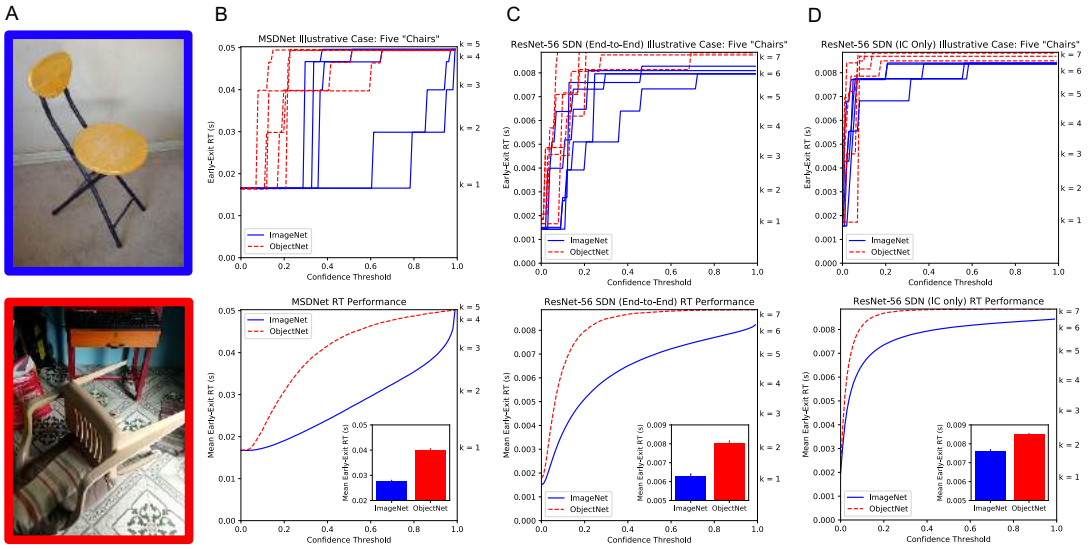


Figure 4.1: (A) Examples of chairs from ImageNet (blue) and ObjectNet (red). (B, above) Illustrative example of the step function describing the early-exit RT for five randomly-selected chairs from both ImageNet and ObjectNet. (B, below) Mean early-exit RT across all test images that had top-5 accuracy in the final auxiliary classifier. The values on the right vertical axis indicate the mean processing time for each of $K = 5$ auxiliary classifiers. Subplot displays the grand means for both datasets. (C) The same analyses for an alternate architecture, ResNet-56 SDN, trained end-to-end with $K = 7$ early exits. (D) The same analyses for ResNet-56 SDN, with $K = 7$ ICs appended after initial training. This shows that an off-the-shelf model can be converted and analyzed with NRT Analysis post hoc. Note differences in y-axis scale.

RT was modeled as a function of the range of confidence values (0 to 1 with a step size of .01) to fully profile how quickly the model would make decisions. For each image, the lowest auxiliary classifier with a decision at the given confidence was identified and its RT was saved. In some cases, higher classifiers made decisions with lower confidence than the preceding classifier, an artefact of the “overthinking" problem in deep neural networks (Kaya et al., 2019). In these cases, RT was recorded from the lower level as the early-exit architecture would prevent the higher classifier from ever making a decision. If the confidence threshold was higher than the confidence estimate across all exits, the RT was assigned to be the RT of the final classifier, since input processing will proceed up to the last exit. The full procedure is outlined in Algorithm 1.

4.3 Experiments and Results

4.3.1 ObjectNet

To quantify the over-representation of canonical viewpoints and backgrounds in ImageNet, Barbu & Mayo et al. created ObjectNet, a 50,000-image test set for object recognition tasks that features common objects (many overlapping with ImageNet classes) viewed from non-canonical viewpoints on non-stereotyped backgrounds (Barbu et al., 2019). To achieve a more diverse set of features, those authors recruited thousands of workers to photograph objects from specified angles communicated via smartphone. The result is a test set that is more inclusive of objects' non-canonical features. When popular object recognition models are tested on ObjectNet, they exhibit performance decrements of up to 45%.

Because ObjectNet is difficult for object recognition models, it is a candidate proof-of-concept test case that RT can be used to measure performance in DNNs. More importantly, because ObjectNet deliberately includes many non-canonical and presumably complex object features, which may be over-represented in later layers, it ought to display strong RT effects for models with hierarchical representation and dynamic inference, including MSDNet.

MSDNet

RT was recorded from the 45,182 top-5-correct ImageNet images (90.3% top-5 accuracy) and 4,753 ObjectNet images that shared ImageNet labels and were also top-5 correct (25.6% top-5 accuracy on images with shared labels; while this is low, it is in the range of reported top-5 accuracy with overlapping labels by the ObjectNet team (Barbu et al., 2019)). We analyzed inputs that were top-5 correct to ensure that we fairly compared object classes across datasets. Figure 1 illustrates how quickly MSDNet can make a decision given a range of confidence values between 0 and 1. Results from five randomly-selected images of chairs from both test sets are plotted to show how confidence propagates through the model, occasionally increasing RT in steps. The best performance would be a reverse-L shape, where the classifier $k = 1$ is sufficiently confident to identify the chair across the full range of thresholds.

RT was aggregated across all images in each set for every level of confidence. These values were submitted to an independent-samples *t*-test to affirm that RT can indeed be used as a reliable indicator of performance ($t = 9.29$, $p < .001$). Looking at the mean RT across all confidence thresholds, MSDNet processes ImageNet test stimuli 31.11% faster than ObjectNet stimuli with overlapping labels (27.40 ms vs. 39.77 ms). Because ObjectNet is characterized by a range of rotational viewpoints and non-canonical backgrounds, we infer that the higher-order features required to identify these items are better represented across MSDNet's deeper auxiliary classifiers. These inputs are all top-5 correct, so it is worth noting that RT is sensitive to the discrepancy in feature space even when accuracy is not. These data are strong evidence that RT can be used to measure the complexity of feature

space — a tidy proof of concept for the following experiments.

ResNet56 SDN

Repeating the analyses as above, we found that both the end-to-end SDN (t = 7.35, p < .001) and the version with auxiliary classifiers appended after initial training (t = 5.12, p < .001) showed the same effect (although smaller in magnitude for the latter). The steeper curves indicate that the features required for correct classification are composed in deeper layers. Comparing across models, we can see that the end-to-end SDN composes the features required to achieve ∼50% confidence on ObjectNet a full IC earlier compared to the IC-only variant. This was expected since the IC-only model is initially a standard DNN optimized for the final classifier only later converted to a dynamic inference model. However, we chose to demonstrate the replication of our conclusion on it since this dramatically expands the usefulness of RT as an XAI technique beyond models with a baked-in input-adaptive inductive bias to any DNN. Going forward, we restrict our analyses to MSDNet.

4.3.2 SCEGRAM

The visual world is populated with regularities, enabling the learning of implicit rules for the relationships between objects and scenes (Biederman et al., 1982). Humans depend on scene grammar, or object-scene congruities, to guide perception and attention (Võ et al., 2019). A semantic violation occurs when an objects identity is statistically uncorrelated with that of other scene elements (for instance, a roll of toilet paper placed in a microwave), whereas a syntactic violation occurs when the statistically reliable interposition of objects in a scene is upset (for instance, a roll of toilet paper placed in the bathroom sink). Evidence for the human phenomenon of scene grammar comes from studies showing that these violations produce performance decrements on our ability to identify the gist of scenes, guide visual attention (Pereira and Castelhano, 2014; Vo and Henderson, 2009), and to plan movements and object interactions (Draschkow and Võ, 2017). Scene grammar effects also occur in artificial neural networks, with evidence for decreased performance in object and scene classification (Bayat et al., 2018). If MSDNet composes higher-order object features representing these relationships, they ought to occur in deeper layers, manifesting as a slower RT. In other words, scene grammar manipulations should yield RT effects.

The SCEGRAM database (Öhlschläger and Võ, 2017) is a set of images of 62 indoor scenes with carefully curated manipulations of scene grammar. For each scene, there are four images (see Figure 2): consistent scene grammar (CON), inconsistent semantics (SEM), inconsistent syntax (SYN), and inconsistent semantics and syntax (SEMSYN). The semantic and syntactic manipulations are fully crossed. So for a given scene, say a kitchen counter, there are four versions of the image: a pot in a pile of dishes (CON); a clock in a pile of dishes (SEM); a pot affixed to the dishwasher door (SYN); and a clock affixed to the

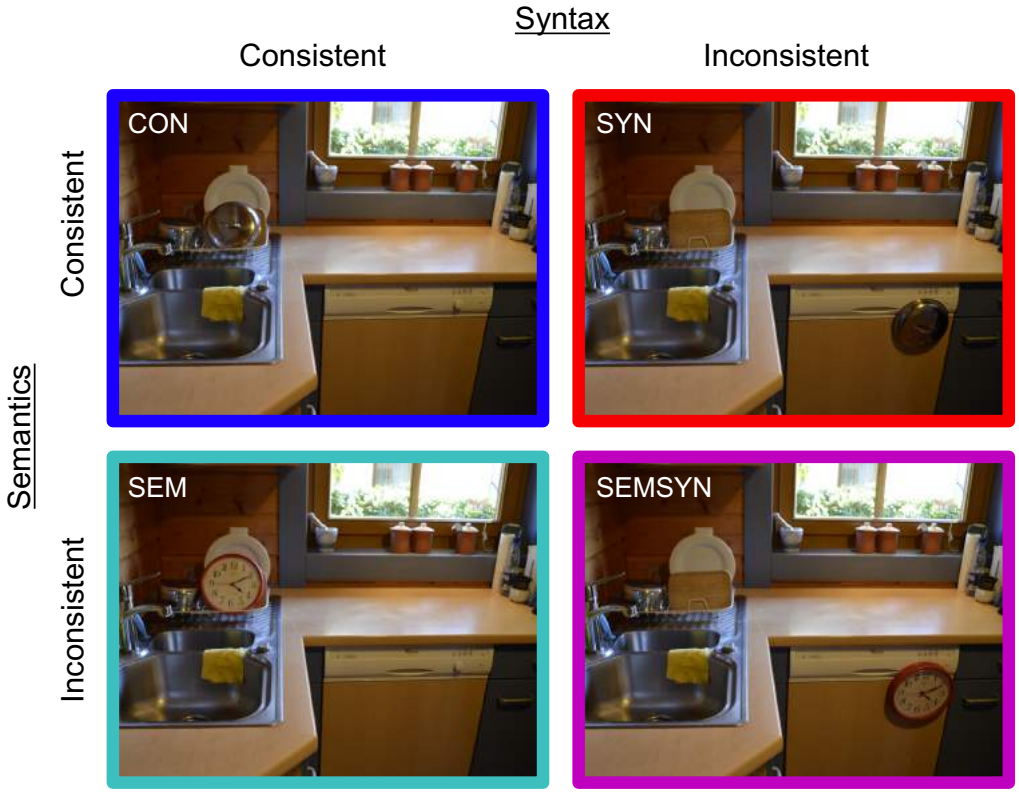


Figure 4.2: Illustrative example of SCEGRAM's test stimuli. Images feature two orthogonal manipulations: semantic consistency and syntactic consistency. A single scene is therefore presented to the model four times, with different combinations of scene grammar.

dishwasher door (SEMSYN). All other visual features in the scene are identical, allowing for experimental inferences about the scene grammar manipulations. We profiled MSDNet object recognition performance on 248 SCEGRAM images, taking the same measurements as in the ObjectNet experiment. SCEGRAM object labels do not match ImageNet perfectly so we manually checked that classification was correct.

The experimental considerations embedded in SCEGRAM allow us to make powerful, specific predictions about how MSDNet should process these inputs with respect to its RT. If MSDNet composes higher-order object features in later layers, then early-exit decisions should be made using coarse features. If high-confidence classification depends on processing higher-order object features such as inter-object/object-scene relationships, then images with inconsistent scene grammar ought to be processed slower on average than images with consistent scene grammar. SCEGRAM also carefully controls salience, modeled by state-of-the-art human attention models (Walther and Koch, 2006) and validated with human foveal fixation data (Öhlschläger and Võ, 2017), ensuring that the four test images within a scene do not differ in salience. Consequently, differences in model performance

cannot be attributed to the minimal low-level differences between scenes. SCEGRAM also contains normative data for consistency ratings, to assure that the scene grammar conditions violate expected inter-object relationships from a human perspective. Because of these controls, we can confidently attribute differences in the model's RT to the independent variable: scene grammar — or the features that present scene grammar to MSDNet. Finally, because MSDNet was pre-trained for object recognition, we can predict that RT effects should be specific to semantic, rather than syntactic inconsistencies.

Step functions of RT given the full range of confidence thresholds are displayed in Figure 3. The mean early-exit RT across all 62 scenes for the full range of confidence thresholds was determined to generalize a profile of the relationship between RT, confidence, and scene grammar (see Figure 3B). Visual inspection of these means reveals better performance for scenes with consistent grammar with local troughs around 0.2 and 0.4 confidence. In human subjects experiments with electro-physiological or other time series data, it is common to specify a window of interest within which to compare RTs from different conditions (Võ and Wolfe, 2013). We defined the boundaries for these windows as the mean confidence required by the model to reach the subsequent classifier's RT (e.g. if RT for $k = 2$ is 0.03 s, what is the mean confidence at which MSDNet reaches 0.03 s?). To characterize whether RT differences were reliable across scenes, we submitted the data to a three-way repeated-measures ANOVA with semantics (consistent, inconsistent), syntax (consistent, inconsistent), and classifier window (thresholds described above) as within-subjects factors. As expected, the classifier produced a strong effect on RT ($F(3,183) = 222.59$, $p < .001$), indicating that RTs were slower as deeper layers were employed for visual recognition. The critical result is that, as expected, there was a significant effect of semantics ($F(1,61) = 4.87$, $p = .031$), indicating that SCEGRAM images with inconsistent semantic information were classified reliably slower than images with consistent semantics. No other effects reached statistical significance.

Unlike Experiment 1, which compares entirely different images of the same classes, the scenes depicted in SCEGRAM are pixel-perfect comparisons of each other except the critical object identity and location. The backgrounds are the same. So the relatively small effects are expected, given that the model is dealing with almost identical information. That we still observe statistically significant effects despite this handicap is what makes the finding so striking. MSDNet is selectively slower due to pixel-level differences that correspond to scene grammar. The observed differences cannot be attributed to anything other than the objects and their position relative to other objects in the scene.

We can also predict when scene grammar effects should not emerge on RT. RTs were collected for the object-absent clone images corresponding to each of the same 248 SCEGRAM images used above. These images are identical except that the critical object has been removed, resulting in images with no inconsistencies. As predicted, there was no effect of semantic inconsistency ($F(1,61) = 0.36$, $p = 0.55$).

Analyzing the object-absent images provides some additional insight into the usefulness of RT measures. The object-present and object-absent images perform similarly in

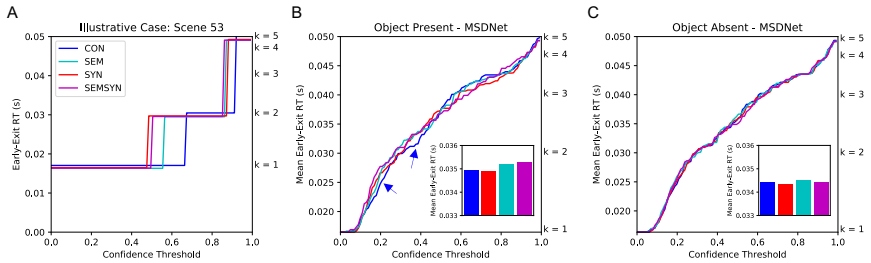


Figure 4.3: (A) Illustrative example of the step function describing the early-exit RT for a single scene in SCEGRAM. (B) Mean early-exit RT across all SCEGRAM scenes, grouped by scene grammar condition. Consistent scene grammar (CON) has visibly faster RT for responses around 0.2 and 0.4 confidence (blue arrows). The values on the right vertical axis indicate the mean processing time for each of $K = 5$ auxiliary classifiers. (C) Same analysis for the object-absent clone images in SCEGRAM. These are technically all semantically consistent. As predicted, they share the same RT profile as CON. Subplots display grand means for each condition. Best viewed in colour.

terms of recognition accuracy, despite the images having some strange objects in the inconsistent conditions. While top-5 accuracy failed to capture these oddities, the RT analysis revealed them. NRT analysis shows how model evaluation can explore other creative ways of documenting performance to capture subtleties in model processing.

4.3.3 Massive Memory

A shrewd reader might rightfully point out that RT is conflated with classification difficulty in both previous experiments. It was essential to demonstrate that RT methods could capture differences in feature space, but these differences also correspond to difficulty; ObjectNet possesses non-stereotypical features, and that's what makes it harder and that's also what makes responses slower. Ideally, RT profiles can tell us about more than just difficulty. One approach to confirm this would be to show that RT profiles for inputs that rely on the same features for classification follow similar trajectories. To demonstrate this, we used the Massive Memory dataset (Konkle et al., 2010), which contains images of 200 categories with 15 exemplars each. We asked whether RT profiles for inputs from the same class were more similar to each other than they were to RT profiles of inputs from other classes. This dataset is experimentally useful because the backgrounds have been omitted and background features that might otherwise complicate interpretation of the RT plots have been controlled for.

We profiled MSDNet performance on 3,000 Massive Memory dataset images (200 classes $\times$ 15 unique exemplars per class). RT data from each of the Massive Memory images was collected with the same process described in Section 4.3.1. We measured system clock time as RT, top 5 classes for each of $k = 5$ classifiers, and the confidence judgments of those classifications. All of these behaviours are observable from outside the

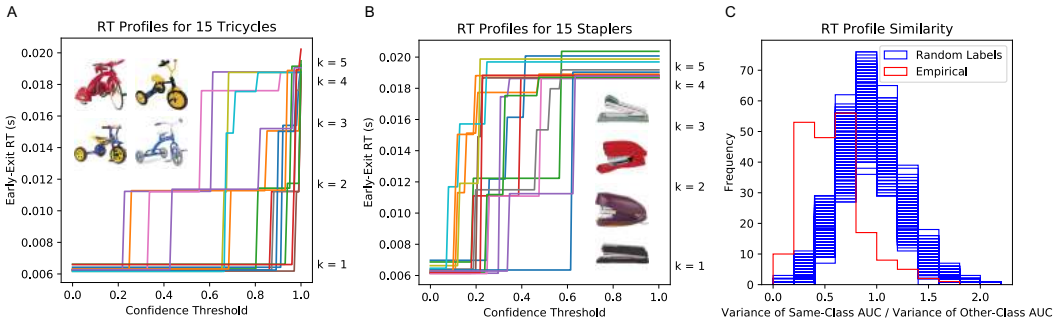


Figure 4.4: (A) Early-exit RTs as a function of confidence budget for all 15 tricycles in the Massive Memory dataset. Four examples displayed inset. (B) Early-exit RTs as a function of confidence budget for all 15 staplers in the Massive Memory dataset. Four examples displayed inset. The values on the left vertical axis indicate the mean processing time for each of $K = 5$ auxiliary classifiers. Visual inspection shows common trajectories for same-class exemplars. (C) This observation is quantified with a Monte Carlo analysis of a measure for assessing the same- versus other-class similarity of RT profiles. We calculated similarity as the variance of AUC for all inputs of a given class divided by the same variance for all inputs of all other classes. Empirical values for all 200 classes are depicted on a histogram in red. Lower values indicate more intra-class similarity of RT profiles. We repeated this analysis on 1,000 simulations of the same data with randomly permuted labels. Our empirical sample occurred well outside the distribution of all random samples.

black box. If inputs from the same class tend to share activations throughout feature space, then RT profiles for same-class exemplars should follow similar trajectories compared to inputs from other classes. Step functions of RT given the full range of confidence thresholds are displayed in Figure 4A and 4B, illustrating all RT profiles for the inputs associated with two exemplar classes.

To determine whether inputs with similar feature space follow similar RT profiles, we asked whether measures indicative of the RT profile's shape were more similar for same- versus different-class inputs. We calculated AUC for each individual RT profile and the ratio of the variance for AUC from inputs belonging to the same class versus variance for AUC for inputs from all other classes. This process was repeated for all 200 classes, and the distribution of these ratios are plotted in red in Figure 4C. Lower values on the x-axis indicate more similarity for AUC of same-class RT profiles compared with other-class.

To assess whether the observed distribution of our similarity metric could have occurred due to chance, we conducted a Monte Carlo simulation by randomly permuting labels in the Massive Memory dataset and repeating the above calculations with 1,000 such simulations. If there is no systematic trajectory for RT profiles of same-class inputs, randomly permuting the labels should have no effect on the observed distribution of our similarity metric.

Instead, we observed that all 1,000 simulated distributions resulting from randomly

permuting labels occurred to the right of our empirical distribution. This is very strong evidence that the empirical distribution of our similarity metric did not occur due to chance. From this analysis, we conclude that RT profiles are indicative of the feature space, and therefore the class, of their input, and can theoretically be used to infer more than just the difficulty of classifying a particular input. This experiment demonstrates that RT plots provide a more nuanced description of the features involved in classifying an image than simply examining accuracy. In addition, given the finding that RT trajectories are diagnostic of the features required to classify an input, it may be possible to train a new model to predict the input given to any model using dynamic inference based on its RTs.

4.4 Conclusions and Future Work

We introduce NRT analysis as an XAI method and experimentally demonstrate its value by probing the inner workings of otherwise opaque models. We were able to test *a priori*, falsifiable hypotheses about the relationship between input space and response time using three different test sets. We showed that classification that depends on access to higher-layer features takes longer for dynamic inference models using conditional computation. These analyses could be used to form expectations for when and how models should perform in situations where explanations are desirable, but privileged access to a model is denied. We also showed with our ImageNet vs. ObjectNet experiments that NRT may be useful for identifying aberrant CNN behaviours that do not show up in classification accuracy or other standard measures of performance. We predict that analyzing NRT profiles for datasets where CNNs end up learning 'shortcuts'(Geirhos et al., 2020) may help easily identify errors in generalization due to dataset bias. We hope to explore this direction of research in future work. The rate limiting factor on NRT analyses is the existence of test sets with meticulously controlled input features. Going forward, we would like to see the creation of more suitable test sets to expand the range of testable hypotheses.

4.5 Supplementary Experiments and Results

We provide additional results in this supplementary section to qualitatively motivate NRT as well as to illustrate an example of a null hypothesis where scene grammar effects are not expected to be observed. We also provide RT profiles for all scenes in SCEGRAM and exemplars in Massive Memory for completeness.

4.5.1 Additional ObjectNet vs. ImageNet Results and Qualitative Analysis

In Experiment 1, we showed that RT profiles could be used to account for differences in input difficulty that come with more complex feature space. We demonstrated this by

showing individual RT profiles for five exemplars from one class in each dataset (chairs) and the mean RT across all input. In this supplementary Figure A1, we show a wider range of RT profiles from different classes so readers can witness some variability at the class level and gain confidence that the chair example was not cherry-picked. For the most part, inputs from ImageNet appear to be classified faster than the more difficult ObjectNet. In these plots, response time is plotted as a function of the confidence threshold. The function displays the minimum time required to make a classification at the given confidence. So a RT profile that plateaus at $k = 1$ until a confidence threshold of 0.5 indicates that the first intermediate classifier tops out at a softmax confidence of 0.5 and must proceed to the next intermediate classifier, taking more time. Inputs that can be confidently classified at earlier classifiers will have faster RTs.

In Figures A2 & A3, we display an illustrative case that shows how complex features may propagate through intermediate classifiers in time. These examples were selected to illustrate this relationship. ObjectNet is difficult for object classification because it contains images of objects seen from non-stereotypical viewpoints. When we examine the RT profiles, we access a new level of nuance. We can see how the "easy" examples cascade through the model over time, whereas the "hard" examples — viewed from different angles — are quickly abandoned by earlier classifiers. The authors of ObjectNet have stated that they intend to release rotational, viewpoint and scene annotations. When these are released, we can conduct a quantitative analysis examining these attributes' influence on RT.

4.5.2 Full RT profiles on SCEGRAM and Additional Null results

For completeness, Figure B4 displays RT profiles for every input in our analysis. The mean RT profiles for the four scene grammar conditions appear close to each other. This is expected, given that the dataset controls for salience and background features, and in some cases contains the same features but in different locations. Here, we wanted to display all individual RT profiles so readers could get a sense for the variability within the dataset.

In Experiment 2, we discussed how we could also predict when RT effects should *not* emerge when comparing inputs that varied in scene grammar. We used the object-absent control images from SCEGRAM as an example, but there are other cases where RT effects should not occur. We ran an identical analysis as reported in Experiment 2 using the SCEGRAM dataset but on a variant of MSDNet trained with an additional self-distillation loss based on (Phuong and Lampert, 2019). The intermediate classifiers are trained to imitate the predictions of the final classifier, resulting in earlier classifiers that employ features like those composed in the final layers. This change in training would erase the correspondence between feature space and depth that RT methods depend on. Predictably, this analysis produced no significant differences in RT between scene grammar conditions

(see Figure B5).

4.5.3 Individual RT Profiles for all Massive Memory exemplars

In Experiment 3, we showed that inputs that share features, such as exemplars from the same class, will exhibit RTs that proceed through the range of confidence thresholds in a very similar manner. We supported this claim with a Monte Carlo simulation of label-permuted data, taking the variance of same-class AUC divided by the variance of all other-class AUC. For completeness, we show the individual RT profiles for the entire MM dataset so that readers can form intuition for how exemplars from the same class share RT profiles (see Figures C6-C10).

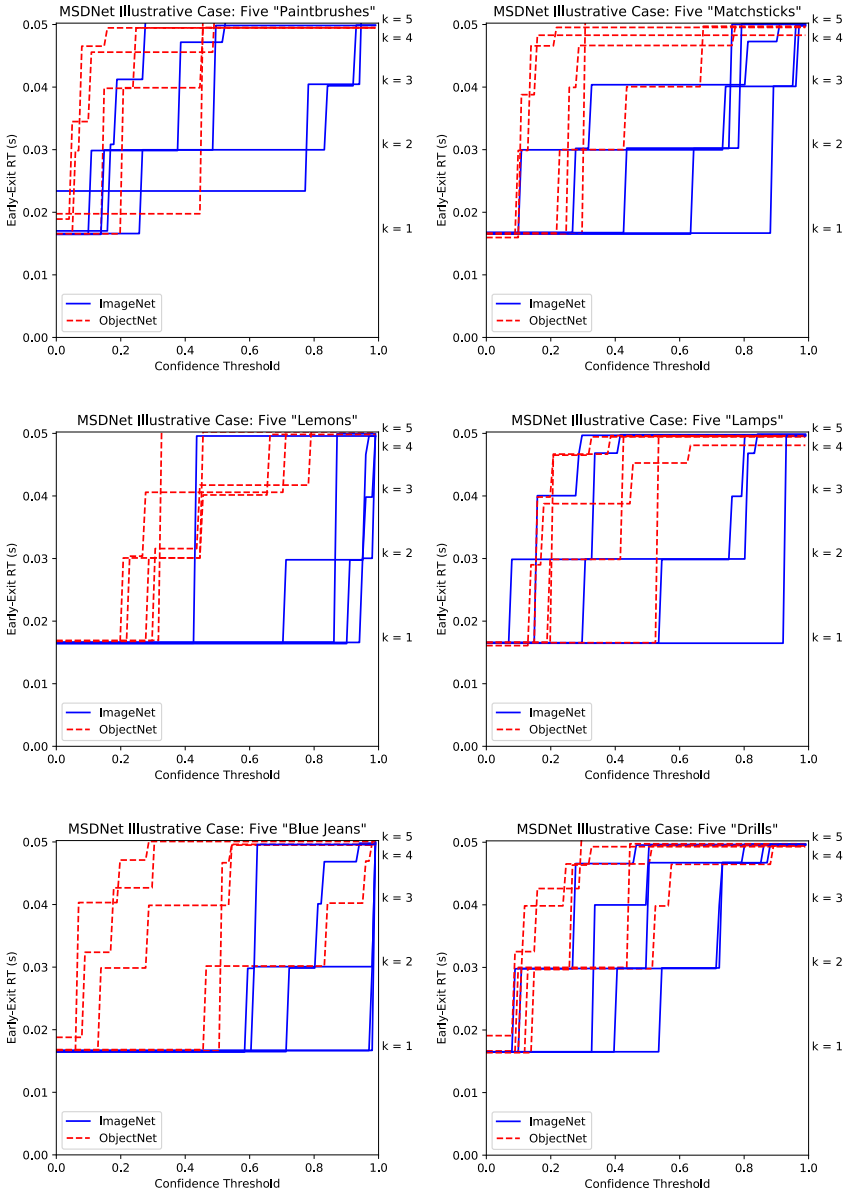
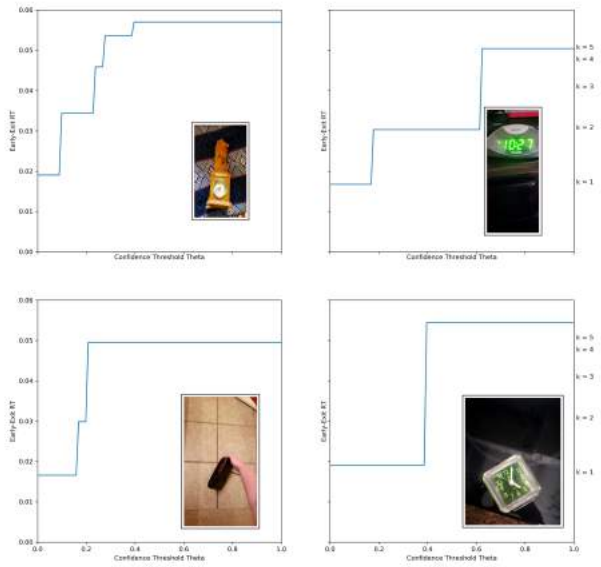
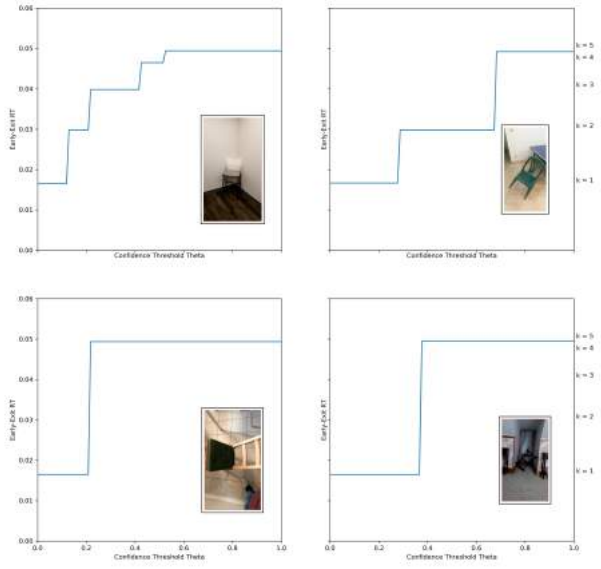


Figure 4.5: Additional illustrative examples of five exemplars for six different classes common to ImageNet and ObjectNet. These RT profiles show how quickly a classification can be made given a certain confidence threshold. The values on the right vertical axis indicate the mean processing time for each of $K = 5$ auxiliary classifiers. Throughout, we see that input from ImageNet is closer to the ideal reverse-L pattern that defines perfect performance (high confidence in the earliest classifier), whereas inputs from ObjectNet proceed to the intermediate classifiers faster given lower confidence thresholds.



(a) Alarm Clocks



(b) Chairs

Figure 4.6: Exploration of relationship between RT and a complex feature — in this case, rotation. The top row shows two stereotypical viewpoints and the corresponding RT profiles. The bottom row shows RT profiles for non-stereotyped viewpoints of the same class.

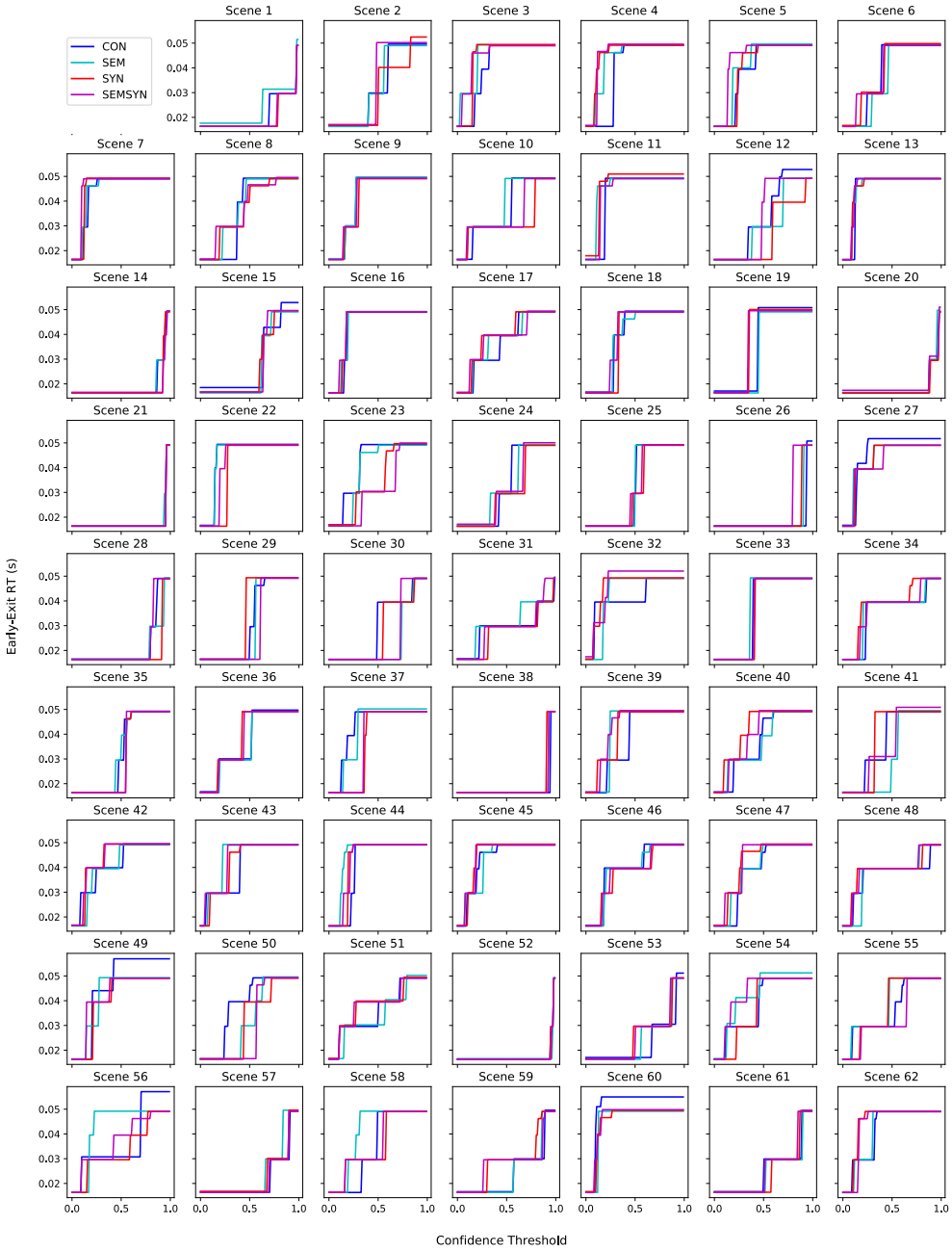


Figure 4.7: RT Profiles for all inputs from the SCEGRAM dataset used in Experiment 2, including all scene grammar conditions from all 62 scenes. These RT profiles show how quickly a classification can be made given a certain confidence threshold. The values on the right vertical axis indicate the mean processing time for each of $K = 5$ auxiliary classifiers.

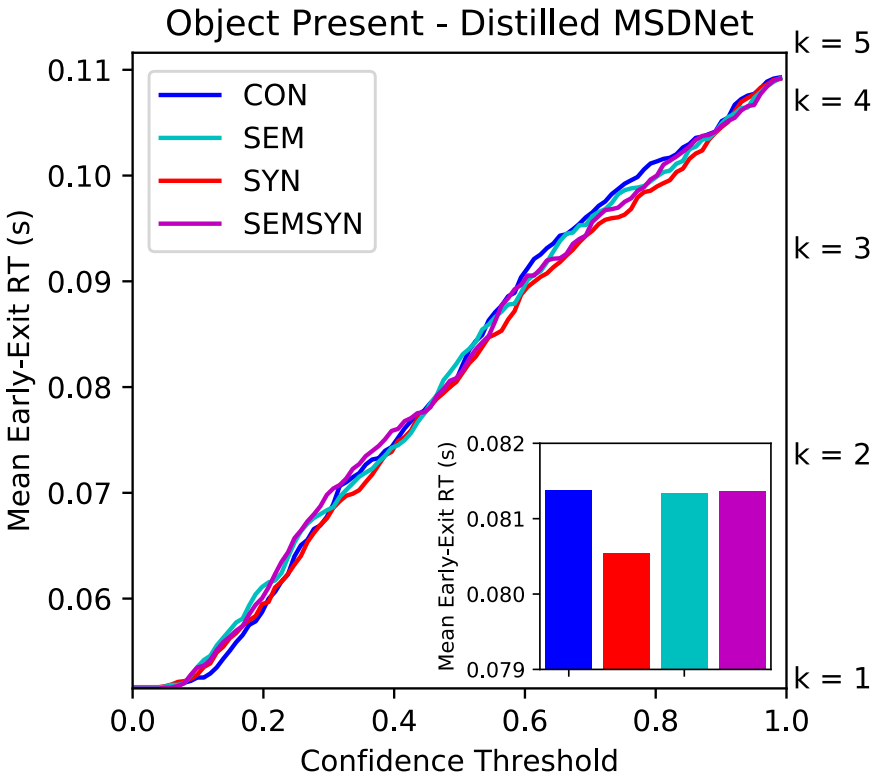


Figure 4.8: Mean early-exit RT across all SCEGRAM scenes, grouped by scene grammar condition for a distilled version of MSDNet. There are no statistically reliable differences between scene grammar conditions in this analysis. This was expected, as the distilled version of MSDNet gives earlier classifiers access to features composed in deeper layers, which ought to flatten any RT differences. The values on the right vertical axis indicate the mean processing time for each of $K = 5$ auxiliary classifiers. Subplots display grand means for each condition. Best viewed in colour.

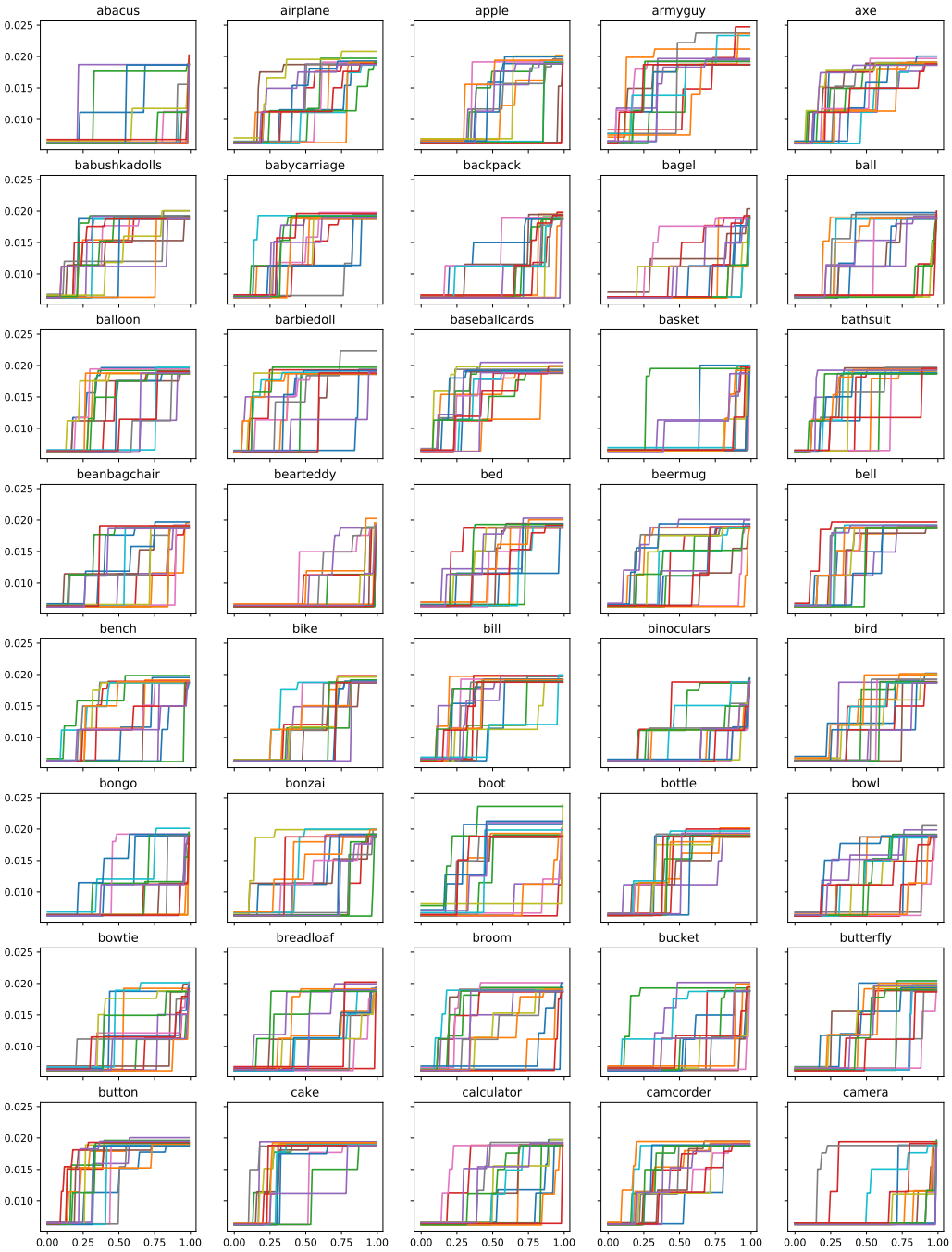


Figure 4.9: Classes 1-40 of the Massive Memory dataset. Note there are some classes that do not overlap with ImageNet. For our purposes, we are mostly interested in seeing how RT profiles for exemplars with similar features (regardless of dataset) proceed through time. MM is handy because objects are pictured on a featureless background, so any commonalities between RT profiles can be attributed to features unique to that input.

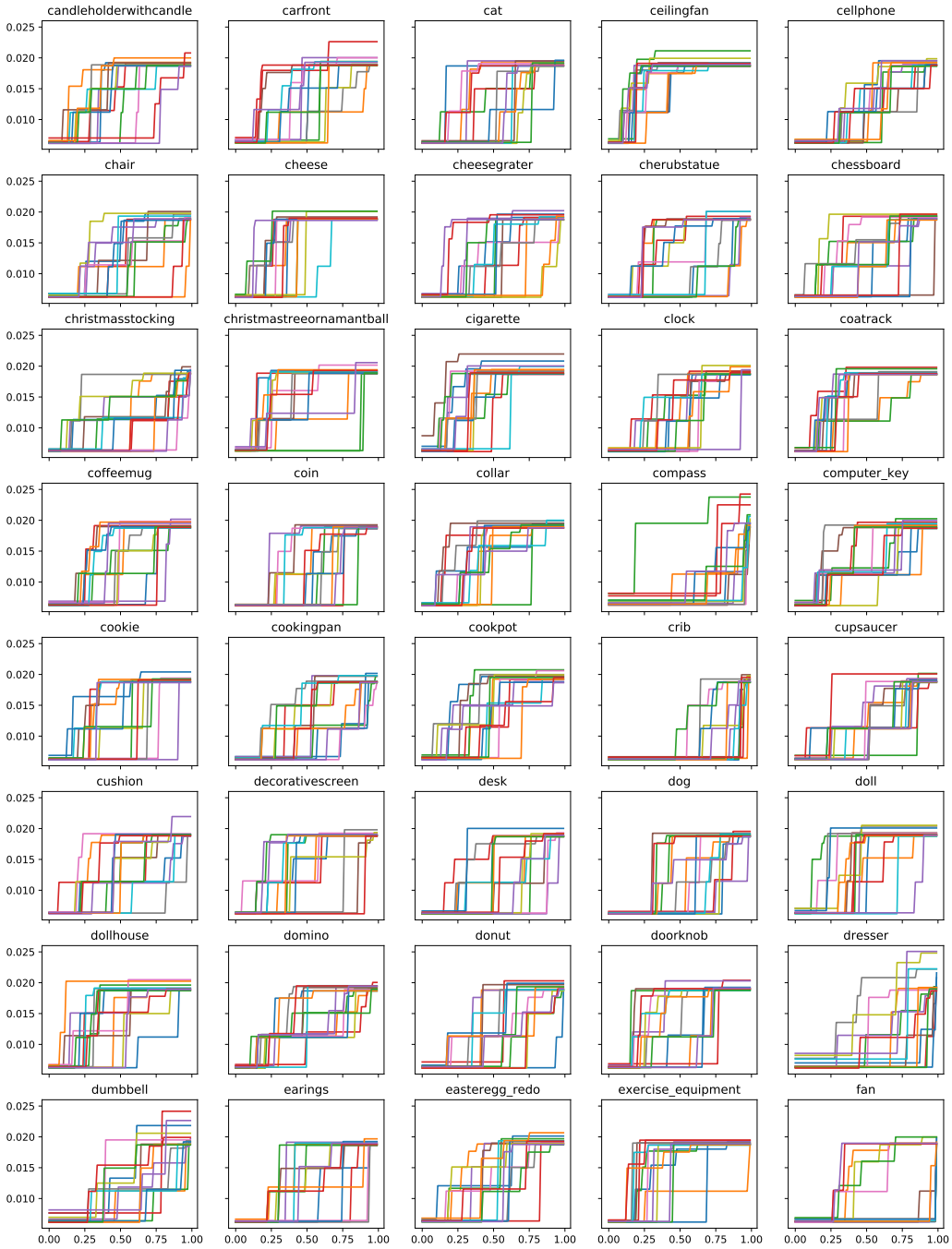


Figure 4.10: Classes 41-80 of the Massive Memory dataset. Note there are some classes that do not overlap with ImageNet. For our purposes, we are mostly interested in seeing how RT profiles for exemplars with similar features (regardless of dataset) proceed through time. MM is handy because objects are pictured on a featureless background, so any commonalities between RT profiles can be attributed to features unique to that input.

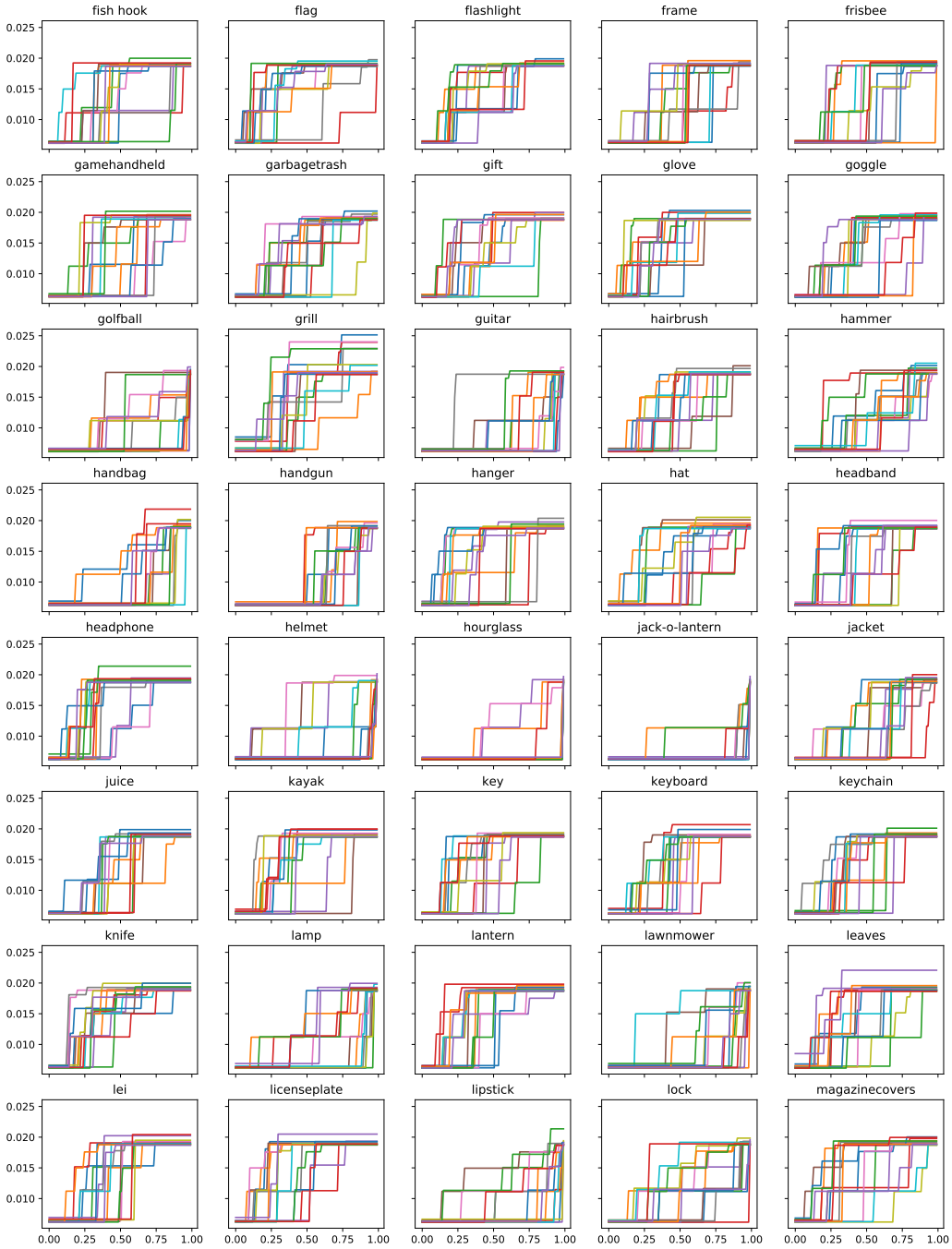


Figure 4.11: Classes 81-120 of the Massive Memory dataset. Note there are some classes that do not overlap with ImageNet. For our purposes, we are mostly interested in seeing how RT profiles for exemplars with similar features (regardless of dataset) proceed through time. MM is handy because objects are pictured on a featureless background, so any commonalities between RT profiles can be attributed to features unique to that input.

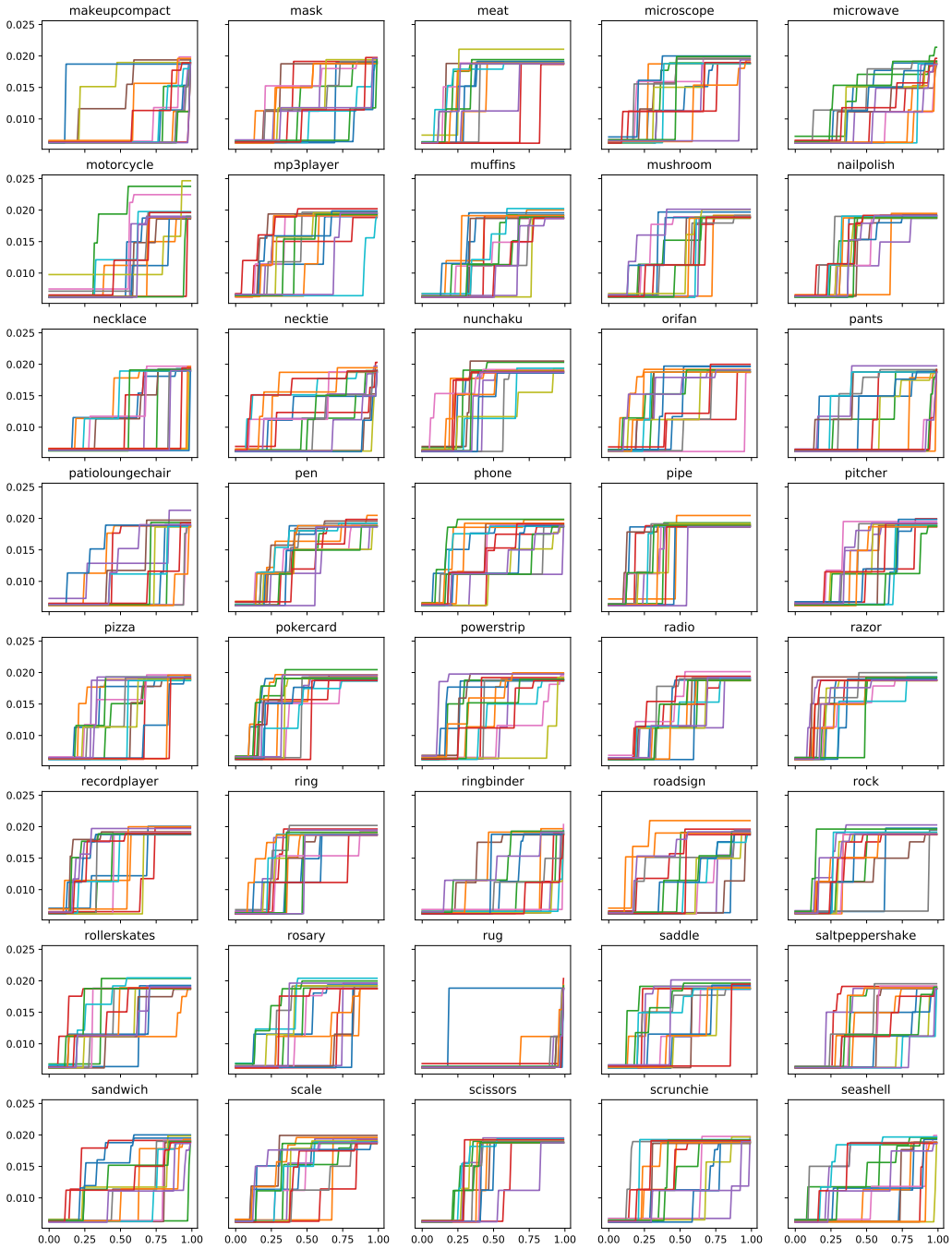


Figure 4.12: Classes 121-160 of the Massive Memory dataset. Note there are some classes that do not overlap with ImageNet. For our purposes, we are mostly interested in seeing how RT profiles for exemplars with similar features (regardless of dataset) proceed through time. MM is handy because objects are pictured on a featureless background, so any commonalities between RT profiles can be attributed to features unique to that input.

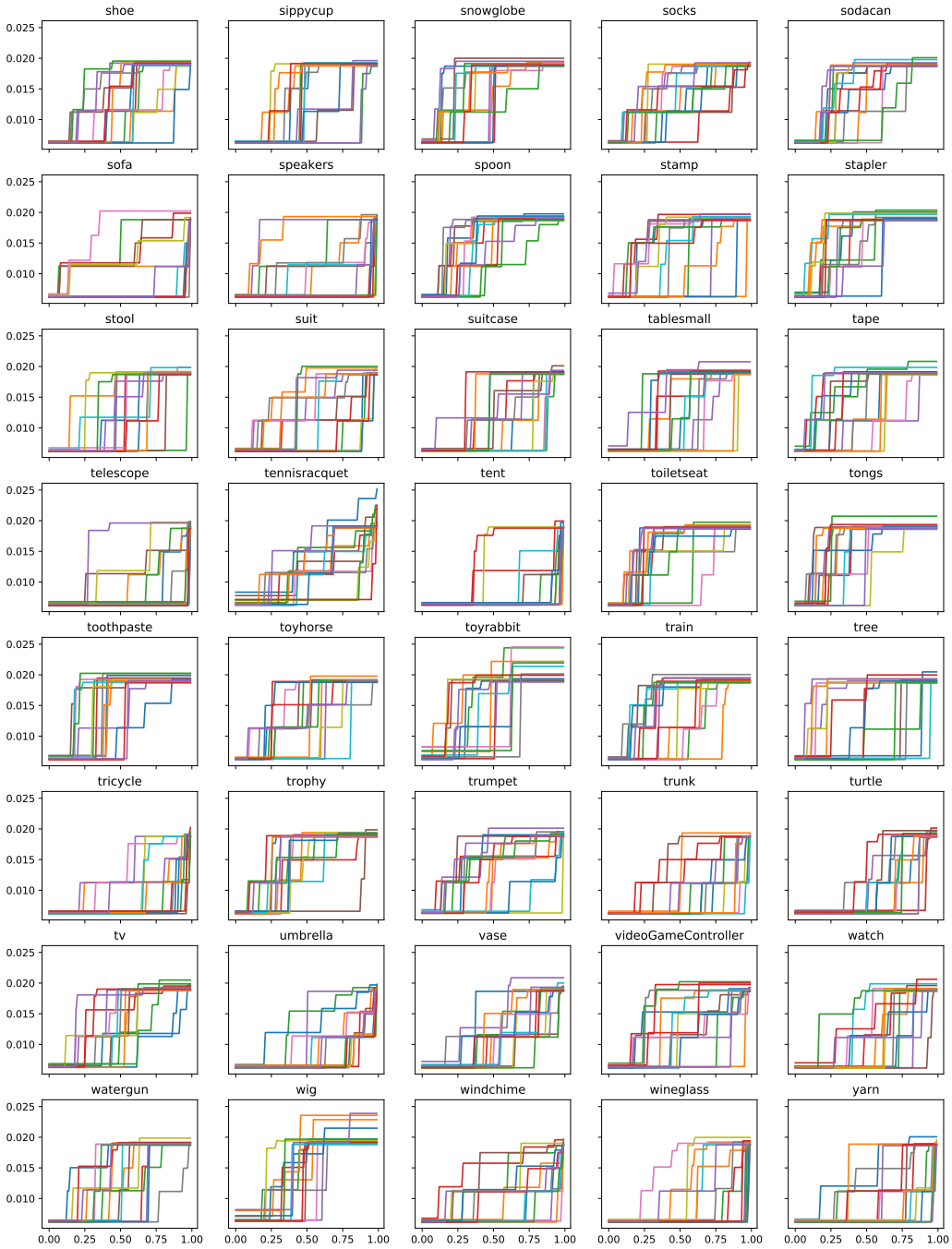


Figure 4.13: Classes 161-200 of the Massive Memory dataset. Note there are some classes that do not overlap with ImageNet. For our purposes, we are mostly interested in seeing how RT profiles for exemplars with similar features (regardless of dataset) proceed through time. MM is handy because objects are pictured on a featureless background, so any commonalities between RT profiles can be attributed to features unique to that input.

Chapter 5

Neural Structure Mapping

The ability to form abstractions of 'concepts' from information and then reason about them is central to human intelligence (Lake et al., 2015). Abstractions enable humans to quickly learn concepts from few examples, and then reason systematically about new concepts by composing previously understood concepts (Nam and McClelland, 2021). Over the last decade, artificial neural networks have demonstrated human-level performance in building useful abstractions from data when tested on well-defined and constrained tasks. In computer vision, deep learning (DL) models that learn visual abstractions from raw images show strong validation performance on curated test datasets for image recognition (Krizhevsky et al., 2012), object detection (Ren et al., 2015), and scene classification (Zhou et al., 2017). However, unlike humans, DL models struggle with isolating these abstractions and systematically applying them to out of distribution test scenarios (Greff et al., 2020).

One important way in which humans both build and reason about abstractions is through analogies (Mitchell, 2021). There are several theories in human cognitive science about how humans perform analogical reasoning. Structure Mapping Theory (SMT) posits that perceptual information can be broken down into a domain consisting of objects and attributes, and structural relations between the attributes in the domain (Gentner, 1983). Consequently, SMT defines an analogy as a mapping between the structural relations across two domains, with no mapping of the attributes. For example, to draw an analogy between the solar system and an atom, we can map the relational structure (`Planet` *revolves around* `Sun` $\Longrightarrow$ `Electron` *revolves around* `Nucleus`, `Sun` *more massive than* `Planets` $\Longrightarrow$ `Nucleus` *more massive than* `Electrons`). However, the domain attributes, such as the `Sun` being *yellow* or *hot* are not mapped to the `Nucleus` in drawing an analogy.

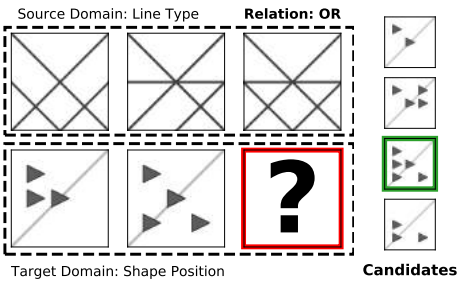


Figure 5.1: Abstract visual analogy problem

This idea behind structure mapping was utilized by Hill et al. (2019) as a prior in train-

ing ML models for learning analogies. They first constructed a dataset on learning abstract visual analogies from Raven's progressive matrices (RPMs) (Raven, 2000). The abstract visual analogy task is made up of two rows of five context panels, and four possible candidate panels to complete the visual analogy. The learner must understand the higher-order relation constituted by the first row of three context panels, and then choose the candidate that satisfies the same relation with the two context panels in the bottom row as shown in Figure 5.1. Hill et al. (2019) showed that if the candidates presented during training were curated to maximize differences in structural relations, while being perceptually similar, the models performed more accurately in learning visual analogies. Thus, the authors verifed that incorporating structural differences to maximize the importance of structure mapping in the learning procedure also works for ML models.

However, it is not feasible to build or curate datasets to always exploit the structure mapping prior. This would involve knowing a combinatorially large number of possible relational structures beforehand and manually selecting data to maximize learning these structures, a requirement that scales poorly with an increase in possible relational structures. To address this issue, we propose a two-stage Neural Structure Mapping (NSM) framework to learn abstract visual analogies (Figure 5.2). Our first-stage involves a visual relationship encoder that takes as input the visual context from the source domain and predicts the component visual concepts: `objects`, `attributes` and `relationship` in a multi-task manner. We then use the derived relation to dynamically build an analogy inference engine. Similar to structure mapping in humans, our engine utilizes only the relation from the source domain to pick the candidate panel that best fits the analogy. We incorporate compositional reasoning into our approach by using a neural module network (Andreas et al., 2016; Johnson et al., 2017b; Csordás et al., 2021b) as our engine. Our modular neural engine dynamically adapts its structure to align (Xu et al., 2020) with the structural `relationship`, thus explicitly incorporating the structure mapping process in our NSM model.

We train and evaluate our approach on five different generalization splits of the visual analogy dataset. These splits were proposed by Hill et al. (2019) to test systematic generalization (Bahdanau et al., 2018) across object and attribute values in learning visual analogies. We show that our relationship encoder achieves a high degree of accuracy in isolating the relationship from the source domain, and, as expected, there is no performance drop across generalization splits or training regimes. For our analogy inference engine, we empirically show that it explicitly captures the structure mapping process, while generalizing systematically in learning the structural relation across visual domains.

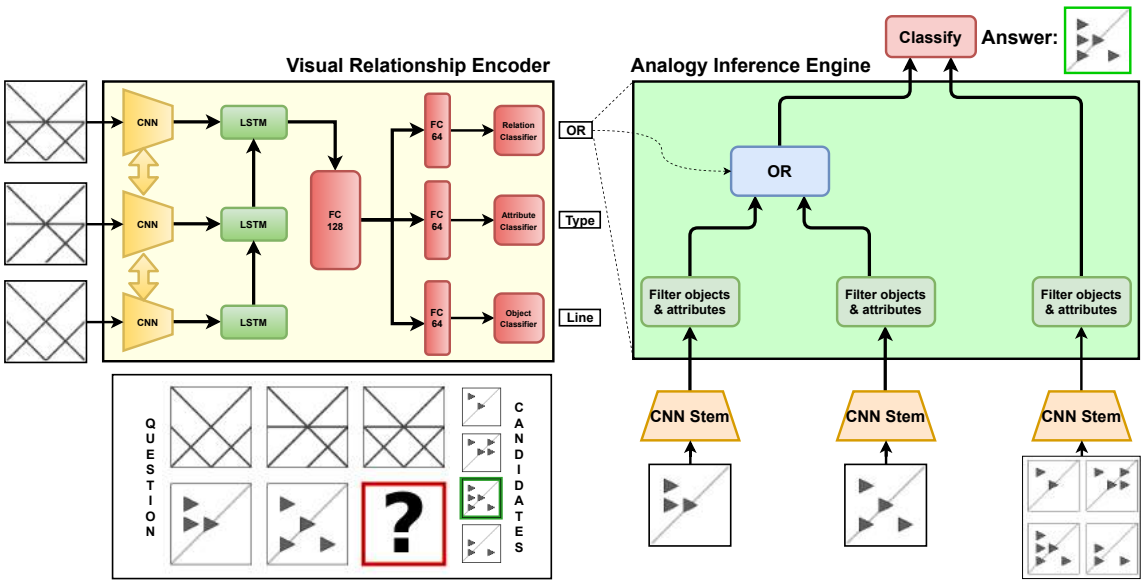


Figure 5.2: Neural Structure Mapping overview. Inset: RPM Analogy Problem from Fig 5.1. Left (yellow box): Visual Relationship Encoder to extract the object, attribute and relationship from the first row of panels. Right (green box): Analogy Inference Engine that uses the relationship label to configure a neural module net for matching the correct candidate to the second row of panels.

5.1 Related Work

5.1.1 Learning Analogies

Gentner (1983)'s SMT defines an analogy as a *comparison in which relational predicates, but few or no object attributes, can be mapped from base to target* domains. Symbolic models based on SMT (Falkenhainer et al., 1986), including those designed for RPMs (Lovett and Forbus, 2017), rely on first extracting a rule based representation of the domains from the perceptual input. Other cognitive analogy models based on Active Symbol theory (Hofstadter and Mitchell, 1994; Mitchell, 1993) or High-Level Perception theory (Chalmers et al., 1992) do not separate the structure extraction and mapping process, nor rely on mapping the structure syntax across domains. Our approach is not directly related to these models.

Hill et al. (2019) introduced the dataset for learning analogies from RPMs and utilized several neural network architectures to directly learn from visual data. Their key contribution was to introduce the SMT prior into the dataset during candidate selection. Webb et al. (2020) introduced Temporal Context Normalization to explicitly learn visual features that can support extrapolation, and tested it on the Visual Analogy Extrapolation Challenge (VAEC) dataset. Both of these approaches are complementary to ours. Other DL

approaches most similar to ours are the Part-Composition Model (Ichien et al., 2021) and Chen et al. (2019)'s approach. However, unlike our model, these models rely on availability of very structured intermediate representations like semantic segmentation maps, and do not use the extracted structural representations to explicitly configure the reasoning model.

5.1.2 Abstract Reasoning

Hill et al. (2019)'s dataset is directly derived from the Procedurally Generated Matrices (PGM) dataset introduced by Barrett et al. (2018) to test the ability of neural networks to perform abstract reasoning on RPM problems. RPM tests are an important measure of fluid intelligence in humans (Raven, 2000), and the PGM dataset provided a sufficiently large sample size for training neural networks on this problem. Following this work, two new RPM datasets were also released: RAVEN (Zhang et al., 2019a), which utilized additional rules and structured rule annotations, and V-PROM (Teney et al., 2020), which utilized real images.

DL approaches to RPM tasks can be roughly categorized into three types. The first is relation learning approaches, such as WReN (Barrett et al., 2018), which models all pairwise relationships between the matrix panels using relation networks (Santoro et al., 2017), and MXGNet (Wang et al., 2020), which learns row-based node embeddings and then performs a graph classification on the resulting candidate graphs. Second is rule learning approaches, such as DRT (Zhang et al., 2019a), that learns a structured Stochastic Image Grammar of the abstract rules, and LEN (Zheng et al., 2019) that utilizes a logic embedding network along with a curriculum to learn with increasingly distracting features. Third is object-centric learning, such as Rel-AIR (Spratley et al., 2020), that obtains object embeddings generated via Attend-Infer-Repeat for each panel, and Pekar et al. (2020)'s method that utilizes both Variational AutoEncoders and an adversarial loss for candidate generation. Our model can be broadly fit into rule learning by way of the encoder, followed by relation learning via the analogy inference engine. However, our general approach towards explicitly extracting structure and mapping it is complimentary to these ideas and can be used in conjunction with several of them.

5.1.3 Systematic Generalization

Systematic generalization refers to the ability to generalize to novel concepts (out of distribution data) by understanding them as compositions of known concepts (Bahdanau et al., 2018). It is underpinned by three important characteristics: (1) systematicity, the ability to generalize to semantically related concepts; (2) productivity, the ability to iteratively apply compounding to generalize from constituent concepts to their recurrence; and (3) localism, the ability to iteratively apply reductionism to generalize from repeated constituent concepts to the singular. The ability to generalize systematically has been previously explored in human cognition (Fodor and Pylyshyn, 1988), natural language processing models (Lake and Baroni, 2018), and language-grounded computer vision models

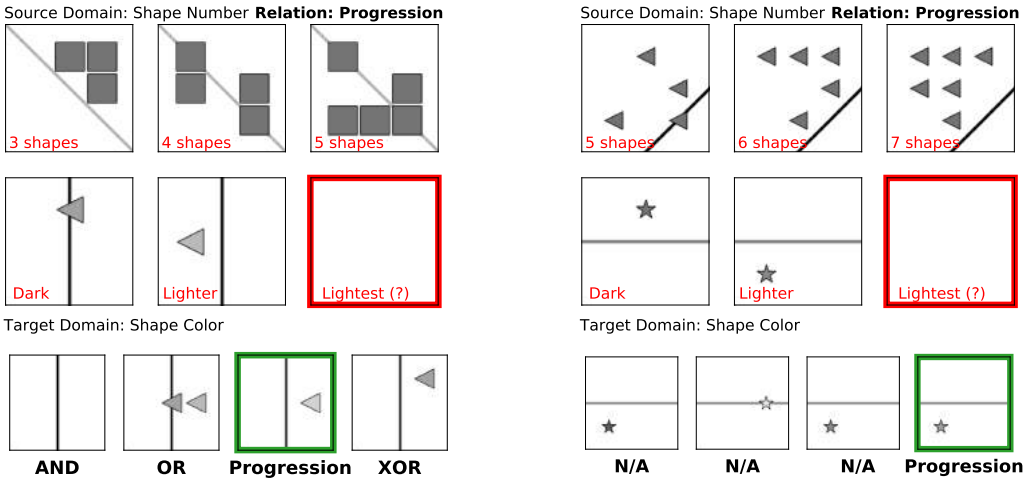
(Agrawal et al., 2017). It has its origins in human cognition and was formalized by Fodor (1975) under the 'Language of Thought' hypothesis.

Cognitive scientists have expressed reservations about the ability of connectionist models to generalize systematicitally (Fodor and Pylyshyn, 1988; Marcus, 2019). Investigation of neural language models showed that systematic generalization still poses a challenge for DL (Lake and Baroni, 2018; Loula et al., 2018). A key reason deep neural networks lack systematicity is due to their propensity towards 'shortcut learning' (Geirhos et al., 2018, 2020). This suggests that DL models rely on exploiting a few number of predictive features instead of considering all possible information about the data while drawing conclusions. Consequently, this leads to poor generalization when the key predictive features are changed, even though the central evidence remains the same. For example, DL approaches to solving RPM problems were discovered to rely on evaluating the mode across all candidates in the RAVEN dataset and a significant drop in generalization performance was observed when the models were tested without the shortcut on the RAVEN-Fair dataset (Spratley et al., 2020). On the other hand, neural module networks (Andreas et al., 2016; Johnson et al., 2017b) generalized well systematically when their layout aligned well with specific language-grounded reasoning problems (Bahdanau et al., 2018). Hence, we utilize a modular approach in building our analogy inference engine, which in turn utilizes the `relationship` structure inferred by our encoder to automatically align itself to the problem. This is a form of conditional computation.

5.2 Problem Setup

The abstract visual analogy task (Hill et al., 2019) falls under the umbrella of Raven's Progressive Matrices (Raven, 2000) designed to test abstract reasoning and fluid intelligence in human and artificial agents. Each question consists of five context panels P_{con}^{1-5}, and four candidate panels P_{can}^{1-4}. The context panels are arranged in a matrix with two rows of three columns each, with the final panel in the second row missing. The first row of panels P_{con}^{1-3} express a semantically related triplet $\mathcal{R} = \{o, a, r\}$, that is composed of two lower-order perceptual visual concepts: `object` o and `attribute` a, and one higher-order semantic visual concept: `relationship` r. The objective is to choose a candidate $c \in \bigcup_{i=1}^{4}\{P_{\text{can}}^{i}\}$ that, upon substitution in place of the missing panel, represents the same `relationship` as the first row of panels.

The `object` ($o \in$ {`line, shape`}) and `attribute` ($a \in$ {`quantity,colour, type,size,position`}) together constitute the visual domain d of the triplet $\mathcal{R}$. Each possible `attribute` can take ten values $v(a) \in \{1, 2, ..., 9, 10\}$ (normalized). The domain of the first row of panels P_{con}^{1-3} defines the source domain d_{source} of the analogy, and the domain of the second row of panels $\{P_{\text{con}}^{4-5}, c\}$ defines the target domain d_{target}. The dataset consists of seven unique possible domains such that $d \in$ {`shape quantity, shape colour, shape type, shape size, shape position, line type, line colour`}. To test the systematic generalization in learning visual analogies, the



(a) Contrasting (b) Normal

Figure 5.3: Two different types of candidates for the same target domain. (a) Contrasting candidates, each of which satisfies a relationship structure and highlights the Structure Mapping prior during candidate selection. (b) Normal candidates, which are merely perceptually similar to the context.

dataset has five different generalization splits that require the ability to recognize the `relationship` across both novel domain d and `attribute` a values:

- **Novel Domain Transfer:** The training set consists of 42 ordered pairs of d_{source} and d_{target} while the remaining 7 (7*7 - 42) pairs are present only in the test set i.e. $d^{\text{train}}_{\text{source}} \times d^{\text{train}}_{\text{target}} \cap d^{\text{test}}_{\text{source}} \times d^{\text{test}}_{\text{target}} = \varnothing$.
- **Novel Domain Type** (`line type` **and** `shape color`) **:** The training set does not contain any problems with the held-out domain i.e. $d^{\text{train}} \notin$ `{line type}` or $d^{\text{train}} \notin$ `{shape colour}`, while each problem in the test set involves the held-out domain.
- **Novel Domain Values (Interpolation and Extrapolation):** The training and test sets are made of two mutually exclusive sets of attribute values. For the extrapolation split, $v(a)^{\text{train}} \in \{1, 2, 3, 4, 5\}$ while $v(a)^{\text{test}} \in \{6, 7, 8, 9, 10\}$. For interpolation, $v(a)^{\text{train}} \in \{2, 4, 6, 8, 10\}$ while $v(a)^{\text{test}} \in \{1, 3, 5, 7, 9\}$.

Each of the splits presents a unique challenge in terms of generalization, yet shares the idea that understanding an analogy involves identifying the higher-order visual relation from the source domain and applying it to a target domain. The `relationship` ($r \in$ `{progression, AND, OR, XOR}`) in the visual analogy dataset can be of classified into two types: Unary and Binary. The Unary relation, `progression`, is composed of a function applied to a single panel to produce the next panel in a row. On the other hand, the Binary relations, (`AND, OR, XOR`), are composed of a function applied to the first two panels in a row to generate the final panel. Thus, the `relationship` defines the

semantics across a row of panels and is the core abstraction of analogical reasoning.

Hill et al. (2019) hypothesized that introducing a prior on the learning process that requires the learner to correctly identify the `relationship` would align with how humans learn structure for analogical mapping. They designed this prior by carefully controlling the candidate panels P_{can}^{1-4} presented during model training (Figure 5.3). This regime, called Learning Analogies By Contrasting (LABC), consists of candidates that are all semantically possible to complete the visual analogy. Each candidate satisfies one of the four possible `relationships` with the target domain panels. However, only the correct panel satisfies the `relationship` in the source domain panels. The authors found that LABC training significantly improved the ability to learn analogies in DL models compared to training with merely perceptually possible candidates (Normal training). In fact, even a mixed training regime consisting of both LABC and Normal problems showed significant improvement on the Normal training regime. To simplify presentation, in what follows we will use the term "Contrasting" for the use of such examples during training (LABC) and in evaluation.

5.3 Method

In our work, we take a complimentary approach to Hill et al. (2019) for learning abstract visual analogies in RPMs. Instead of depending on explicitly labeled candidates for mapping relational structure, we build a model that dynamically configures its structure based on the relational structure extracted from the source domain. Internally, our approach is made up of two different neural networks that correspond to the two separate tasks in our pipeline. The first step is the Visual Relationship Encoder, $\mathcal{R}_{\text{pred}} = \phi(P_{\text{con}}^{1-3})$, that predicts the visual relationship triplet $\mathcal{R} = \{o, a, r\}$ encoded in the source domain panels P_{con}^{1-3}. Next is the Analogy Inference Engine, $c = \pi(P_{\text{con}}^{4-5}, P_{\text{can}}^{1-4}, r)$, that takes the predicted source relation r, and selects the candidate $c \in \bigcup_{i=1}^{4}\{P_{\text{can}}^{i}\}$ that maximizes the probability of this relation in the target domain P_{con}^{4-5}. We next provide detailed discussion on each of these networks in our pipeline. For implementation details of the networks please refer to Appendix 5.6.5.

5.3.1 Visual Relationship Encoder

The Visual Relationship Encoder, $\mathcal{R}_{\text{pred}} = \phi(P_{\text{con}}^{1-3})$, segregates the abstract relationship triplet in the source domain panels into its constituent `object, attribute`, and `relationship`. This step of our NSM model corresponds to the structure extraction phase in SMT in which humans segregate the higher-order relation concepts in perceptual information from the underlying domain concepts.

The encoder ϕ is made up of a multi-task neural network. We use a convolutional neural network (CNN) C_ϕ to first extract a visual feature vector $vis_\phi^i = C_\phi(P_{\text{con}}^i)$ for each context panel. The extracted visual features are then sequentially fed into a long short term memory

network (LSTM) R_ϕ to generate visual sequence features seq_ϕ of the source domain. The hidden layer vector after the third sequence panel is taken as the encoded features of the source domain $seq_\phi = R_\phi(vis^{1-3})$.

The source domain feature vector seq_ϕ is then passed to a fully connected layer $l_\phi^{\text{shared}} = FC_\phi^{\text{shared}}(seq_\phi)$ that is shared between all three visual components. The shared linear feature vector l_ϕ^{shared} is then passed through three different fully connected layers $l_\phi^{\text{task}} = FC_\phi^{\text{task}}(l_\phi^{\text{shared}})$ for each task $t \in$ `{object, attribute, relation}` classification. Finally, each l_ϕ^{task} is passed through a fully connected layer $out_\phi^{\text{task}} = FC_\phi^{\text{out}^{\text{task}}}(l_\phi^{\text{task}})$ of sizes 2, 5, and 4 for the object, attribute, and relation prediction tasks respectively. A Softmax over the outputs out_ϕ^{task} yields a probability distribution over the possible $\{o, a, r\}$ values that constitute the triplet $\mathcal{R}$.

5.3.2 Analogy Inference Engine

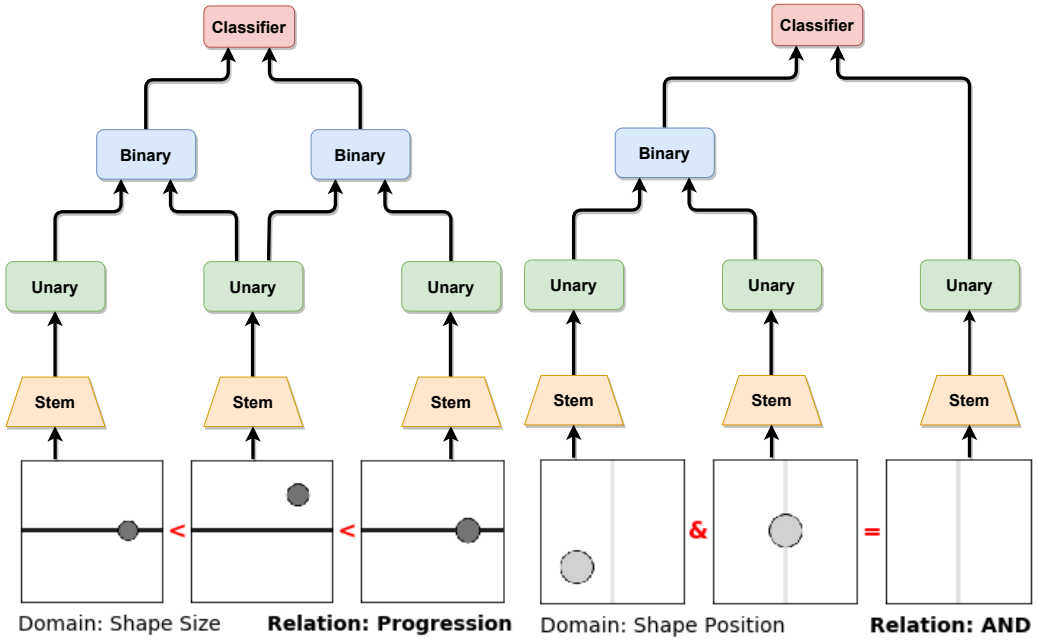
The Analogy Inference Engine, $c = \pi(P_{\text{con}}^{4-5}, P_{\text{can}}^{1-4}, r)$, maps the source relationship r extracted by the Visual Relationship Encoder to the target domain P_{con}^{4-5} to identify the correct candidate c that completes the visual analogy. This step of our NSM model corresponds to the structure mapping phase in SMT, in which humans apply relations extracted from the source domain to a new domain to reason about concepts in the target domain. To generalize systematically to new domains and attribute values, we utilize a modular approach (Andreas et al., 2016) to build our analogy inference engine. Similar to Johnson et al. (2017b) and Bahdanau et al. (2018), we use modules with a generic architecture. Our engine is made up of four different types of neural modules:

- The `Stem` module performs a series of convolution operations with stride=2. It takes the original grayscale images of size $1 \times 160 \times 160$ as input and returns a visual feature map of size $8 \times 9 \times 9$ $(C \times H \times W)$. Each of P_{con}^{4-5} and P_{can}^{1-4} panels is first passed through the `Stem` module to extract visual features before being processed by the rest of the engine.

- The `Unary` module is a residual block with two 3×3 convolution layers. It receives one feature map $(C \times H \times W)$ as input, and returns one feature map of $(C \times H \times W)$ as output.

- The `Binary` module receives two feature maps, concatenates them along the channel dimension, and uses a 1×1 convolution to project them to C dimensions. It then passes the combined feature maps through a residual block, and returns one feature map of $(C \times H \times W)$ as output.

- The `Classifier` module also receives two feature maps, and projects them to C dimensions. It then flattens the feature maps and passes them through two fully connected layers to generate a probability distribution over the relations.



(a) Unary Relation (b) Binary Relation

Figure 5.4: Types of `relationship` and corresponding Analogy Inference Engine layout. Only correct candidate panel shown. See Section 5.3.2 for details

Each candidate panel is fed parallely to the engine to generate its predictions for all four relations. In order to generate the candidate selection, we subset the probability of `relationship` r (r_{pred} during inference) for all candidates, and select the candidate with the highest corresponding probability.

Module Net Architectural Layout

The layout of the module network is chosen adaptively from two possible layouts based on the source domain relation at inference time. Each layout first processes the individual panels through the `Stem` module to extract visual features, followed by the `Unary` module. In the first layout, the `Unary` module outputs for the target domain panels P_{con}^{4-5} are combined using a `Binary` module, and the outputs for the second panel P_{con}^{5} and each candidate panel P_{can}^{1-4} are combined using another `Binary` module (Figure 5.4a). The outputs of both the `Binary` modules are then fed into the `Classifier` module. In the second layout, the `Unary` module outputs for the target domain panels P_{con}^{4-5} are similarly combined using a `Binary` module. However, the output of this `Binary` module and the `Unary` candidate output is directly fed into the `Classifier` module (Figure 5.4b). We provide further discussion on these layouts in Appendix 5.6.1.

5.4 Experiments and Results

5.4.1 Visual Relationship Encoder

Training

The Visual Relationship Encoder ϕ is trained using the source domain context panels P_{con}^{1-3} and the visual relationship labels $\mathcal{R} = \{o, a, r\}$. The cross-entropy loss for each task is obtained using the multi-task network's predictions and triplet labels. The total encoder loss is calculated as a weighted sum of the individual losses for object, attribute, and relation prediction:

$$\mathcal{L}_{\text{encoder}} = \alpha * \mathcal{L}_{\text{object}} + \beta * \mathcal{L}_{\text{attribute}} + \gamma * \mathcal{L}_{\text{relationship}} \tag{5.1}$$

Empirically, we found that the values of $\alpha = 0.5, \beta = 0.5, \gamma = 2.0$ led to higher validation accuracy. The model was trained for 100 epochs using the Adam optimizer (Kingma and Ba, 2015) with a learning rate of $1e-4$. For each generalization split, the model was trained with contrasting, normal, as well as mixed candidates. The model with the best validation accuracy was chosen for testing (the validation set consisted of a similar set of analogy candidates as the training set).

Results

We test the relationship prediction accuracy of our encoder across all five generalization splits, with each possible set of analogy candidates (Contrasting, Normal, Mixed). Our relationship encoder was able to isolate the component relationship from the source domain with a high degree of accuracy (min: 82.4%, max: 86.04%) across all generalization splits and all three types of candidates (Figure 5.5). Furthermore, since the encoder utilizes only the source domain information in determining the relation, we observe no drop between training and test accuracy across all generalization splits.

By identifying the visual relationship in the source domain, the output of the visual relationship encoder is able to determine the layout for the subsequent analogy inference engine. Since the inference engine chooses between only two possible layouts, a baseline that always chooses the majority class layout would lead to a correct choice in 75% of problems. Hence, our encoder must achieve a high degree of accuracy in identifying the correct layout to beat this baseline. Our encoder is indeed able to identify the layout matching the reasoning problem with near perfect accuracy (min: 98.42%, max: 99.9%). The full results for layout prediction across all generalization splits and training regimes are in Table 5.15 in Appendix 5.6.6.

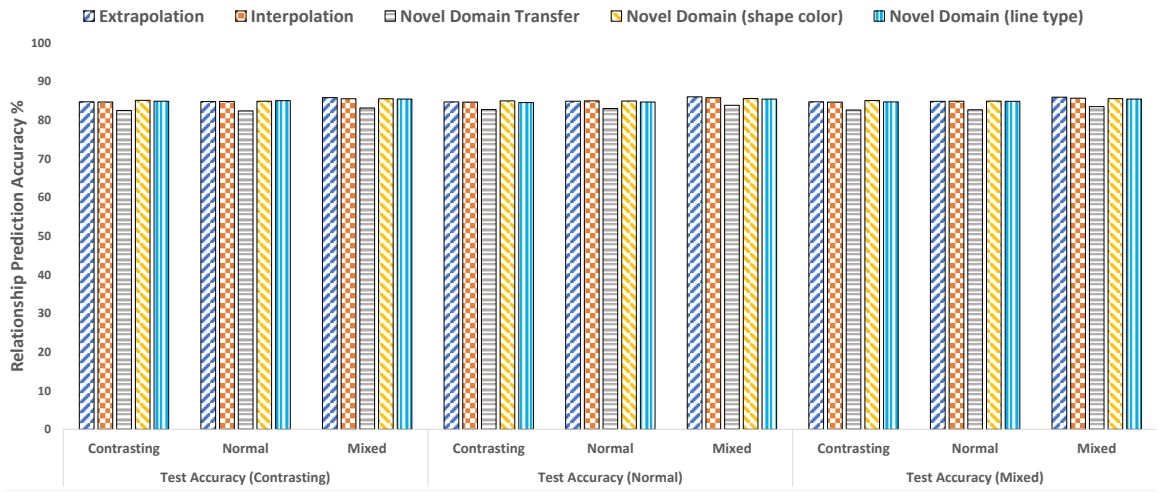


Figure 5.5: Relationship prediction test accuracy of our Visual Relationship Encoder. Results are first grouped (L-R) by test candidate regimes (contrasting, normal, and mixed candidates) and then sub-grouped (L-R) by training candidate regimes.

5.4.2 Analogy Inference Engine

Training

The Analogy Inference Engine π is trained using the target domain context panels P_{con}^{4-5}, the candidate panels P_{can}^{1-4}, the ground truth candidate label, and the ground truth visual relationship label r. The engine was trained for 100 epochs using the Adam optimizer with a learning rate of $1e^{-4}$. Similar to the first stage, the engine was also trained with contrasting, normal, as well as mixed candidates. The model with the highest validation accuracy was chosen for testing (validation set consisted of a similar set of analogy candidates as the training set).

Testing

For testing, we rely on the relationship label r_{pred} predicted by our encoder instead of ground truth r. While all baselines (see below) are trained with the full set of context panels P_{con}^{1-5}, our Analogy Inference Engine only utilizes the target domain panels P_{con}^{4-5} for context. Hence, we also provide the results of a full context model which utilizes all context panels P_{con}^{1-5}. For this purpose, we leverage our encoder which is trained with the remaining context panels P_{con}^{1} – P_{con}^{3}. We use it to first extract the `relationship` r from the source domain as discussed in Section 5.3.1. In addition, we also generate candidate probabilities from the encoder during inference, by considering each candidate in parallel followed by a Softmax over the probability of r across all candidates as described in Section 5.3.2. The candidate probabilities of the full context NSM model are calculated as a 1:1 ensemble

Model	Test Accuracy % (Contrasting/Normal)					
	Novel Domain (Line Type)			Novel Attribute Value (Extrapolation)		
	Contrasting	Normal	Mixed	Contrasting	Normal	Mixed
CNN-LSTM (Hill et al., 2019)	76/50	45/57	75/54	62/45	43/44	56/39
ResNet	25.01/24.95	41.8/46.96	25.01/24.95	**78.97**/55.2	48.5/49.55	62.59/52.73
ResNet-Parallel	79.35/**66.51**	52.43/**76.7**	79.67/**75.9**	61.97/57.23	54.86/56.69	65.71/57.17
WReN	73.71/57.49	53.42/62.64	61.1/49.93	74.25/**61.36**	61.23/**61.4**	**70.57/63.31**
NSM (ours)	78.14/59.55	70.64/64.31	78.53/64.1	65.24/57.14	62.33/58.08	58.4/50.98
NSM (full context)	**79.75**/59.55	**76.18**/62.43	**80.57**/65.74	73.36/59.94	**66.75**/59.84	63.47/52.92

Table 5.1: Candidate prediction test accuracy comparison for Novel Domain and Novel Attribute values. Our approach achieves the best generalization accuracy for the most number of possible train/test candidate scenarios across all models (4/12), and is within $< 1\%$ of the best generalization performance for half of all scenarios (6/12). Higher accuracy == Better systematic generalization.

between the probabilities from both the inference engine and the encoder.

Baselines

We compare the candidate selection accuracy of our analogy inference engine with each model utilized by Hill et al. (2019). These baselines include a CNN-LSTM model, a standard ResNet50 (with 9 input channels for 9 panels), a parallel ResNet50 (with 6 channels and each candidate fed in parallel), and a Wide Relation Network. Each baseline is trained for 100 epochs using the Adam optimizer with a learning rate of $1e^{-4}$, with contrasting, normal, and mixed candidates.

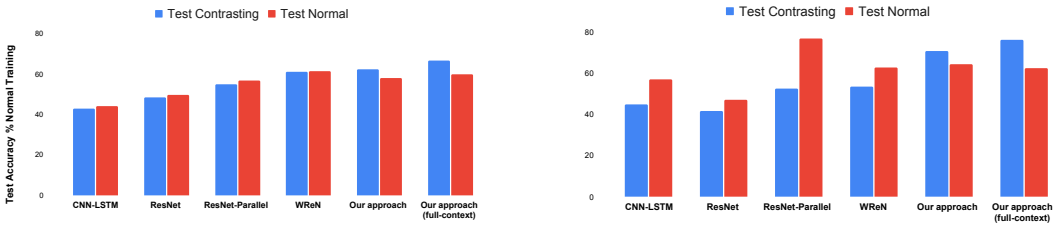
Results

Does the modular analogy inference engine generalize systematically?

We hypothesize that choosing a modular engine for analogy inference enables compositional reasoning about novel domain values (objects and attributes). In Table 5.2 we compare the generalization performance of our engine on the Novel Attribute Value: Extrapolation and Novel Domain: Line Type test splits of the analogy dataset. Our approach achieves the highest test accuracy across all three training candidate regimes in the Novel Domain setting when tested with contrasting candidates. Furthermore, our approach yields high test accuracy (highest or within $< 1\%$) for the Contrasting and Normal training candidate regimes in the Novel Attribute Value setting. Thus, our NSM approach is better at systematic generalization to novel visual domains than monolithic neural networks. For completeness, we provide the results for both our evaluation setups across all training and test regimes in Tables 5.16 and 5.17 in Appendix 5.6.6

Does our model's prior enable it to learn structure mapping?

We specifically evaluate the Normal training scenario where the candidates presented to the learner are drawn randomly and lack the structure mapping prior explicitly present when training with Contrasting. Since our NSM model explicitly captures and maps the structure of the visual analogy, thus making up for the lack of the structural prior in the candidates, we anticipate that the test performance of our model trained with normal candidates would outperform other baselines. We visualize the generalization performance of this training scenario for the Novel Domain Value: Extrapolation and Novel Domain: Line Type generalization splits in Figure 5.6. We observe that our model outperforms other networks when averaged across both types of candidates during testing, demonstrating that it learns the structure mapping prior better than other models.



(a) Novel Attribute Value: Extrapolation

(b) Novel Domain: Line Type

Figure 5.6: Systematic generalization test performance with Normal training candidates which lack the structure mapping prior at training time.

How important is structure mapping in drawing the correct analogy?

We verify the importance of explicitly mapping structure on our NSM approach by comparing our results when the `relationship` label inferred in the first step matches the ground truth `relationship` label. We present the confusion matrix in Table 5.2 where it can be seen that the test candidate selection accuracy for the correct structure mapping (diagonal entries) is significantly higher than incorrect structure mapping.

	Prog.	XOR	OR	AND
Prog.	**90.36**	16.72	22.53	10.88
XOR	20.88	**86.49**	57.67	12.32
OR	43.86	52.47	**94.80**	6.01
AND	5.86	6.18	2.76	**94.72**

(a) Test Accuracy (Mixed)

	Prog.	XOR	OR	AND
Prog.	**90.81**	2.60	9.12	0.01
XOR	6.61	**89.36**	50.04	1.94
OR	35.69	52.39	**96.19**	0.00
AND	0.30	3.27	0.00	**96.36**

(b) Test Accuracy (Contrasting)

	Prog.	XOR	OR	AND
Prog.	**89.93**	30.44	35.57	21.44
XOR	35.33	**83.58**	65.39	22.83
OR	51.98	52.55	**93.42**	11.97
AND	11.51	9.15	5.57	**93.06**

(c) Test Accuracy (Normal)

Table 5.2: Test accuracy with correct vs incorrect structure mapping. Correct Mapping == ground truth relation (rows) matches the relationship used to inform the analogy inference engine (columns).

5.5 Conclusions and Future Work

In this work, we introduce a two-stage neural framework for learning abstract visual analogies based on the Structure Mapping Theory of human analogy making. Our first stage is a multi-task Visual Relationship Encoder that corresponds to structure extraction from perceptual information in humans. Our second stage is a modular Analogy Inference Engine that corresponds to mapping higher-order relations for structure mapping in humans. Our approach is able to generalize systematically to novel target domains, compensate for the lack of a structure mapping prior in candidate selection, and successfully exploit the process of extracting and mapping the relationship structure. In future work, we plan to investigate search-based (e.g. Neural Architecture Search) as well as differentiable (e.g. attention-based) methods for generating the layouts of our Analogy Inference Engine. Furthermore, we would like explore approaches that combine our structure mapping framework into existing methods for reasoning about Raven's Progressive Matrices beyond analogy making.

5.6 Supplementary Experiments and Results

We provide additional ablations in this supplementary section to motivate our architectural choices, and also provide implementation details as well as the complete results of our model for readers looking to read about the specific aspects of our study.

5.6.1 How do we chose the module network layouts?

The generic architecture of our modules allows for a combinatorially large number of possible architectural layouts of our inference engine. Previous work in language-grounded reasoning tasks like VQA (Johnson et al., 2017b) or referring expression comprehension (Yu et al., 2018) uses a semantic parse tree generated from natural language to generate layouts for neural module networks. Since our inference engine $\pi(r)$ relies only on the relationship label from the source domain, it lacks the rich compositional structure available in language-grounded reasoning to arrange the modular network layout. However, generating the proper layout for neural module networks is fundamental to their ability to generalize systematically (Bahdanau et al., 2018).

Algorithm 1 To Identify Unary Relation

Require: Panels P_1, P_2, P_3
1: $D_1 \leftarrow$ *extractObjectsAndAttributes*(P_1)
2: $D_2 \leftarrow$ *extractObjectsAndAttributes*(P_2)
3: $D_3 \leftarrow$ *extractObjectsAndAttributes*(P_3)
4: $R_{12} \leftarrow$ *artihmeticOperation*(D_1, D_2)
5: $R_{23} \leftarrow$ *artihmeticOperation*(D_2, D_3)
6: Result $\leftarrow$ *isConsistent*(R_{12}, R_{23})
7: **return** Result

Algorithm 2 To Identify Binary Relation

Require: Panels P_1, P_2, P_3
1: $D_1 \leftarrow$ *extractObjectsAndAttributes*(P_1)
2: $D_2 \leftarrow$ *extractObjectsAndAttributes*(P_2)
3: $D_3 \leftarrow$ *extractObjectsAndAttributes*(P_3)
4: $R_{12} \leftarrow$ *binaryOperation*(D_1, D_2)
5: Result $\leftarrow$ *isConsistent*(R_{12}, D_3)
6: **return** Result

As discussed in Section 5.3.2, we dynamically chose between two layouts of our neural module network based on the `relationship` type. Our rationale behind choosing these layouts comes from the knowledge of the possible `relationships` our engine has to reason about. Each layout aligns algorithmically (Xu et al., 2020) with a reasoning algorithm for identifying the relationship across three panels. The first layout models Algorithm 1 to identify a Unary relationship like `progression` (not to be confused with a `Unary` module) between panels 1, 2 and 3. The second layout models Algorithm 2 to identify a Binary relationship (`OR, AND, XOR`) across panels 1 and 2 in the last panel.

We do not explicitly train our modules to correspond to any individual functional form. Instead, we expect the module instantiations to learn the corresponding function in the reasoning step simply by learning from the classification loss. For example, in the reasoning algorithms above, a `Unary` module can learn *extractObjectsAndAttributes()*, `Binary` module can learn *artihmeticOperation()* and *binaryOperation()*, and `Classifier` module can learn *isConsistent()* functions. This enables the modules to share parameters across relation functions (such as two different instantiations of *binaryOperation()* for `AND` and `OR`), as well as across objects and attributes. This also enables us to utilize general purpose modules which could potentially be arranged in several possible combinations depending on the reasoning problem, and can be easily extended beyond the abstract visual analogy setting of only four `relationships` to large number of possible `relationships` in datasets like Scene Graph (Johnson et al., 2015) and CLEVR (Johnson et al., 2017a).

5.6.2 Do algorithmically aligned engine layouts learn corresponding relations better?

We chose two possible layouts for our modular engine based on their alignment with algorithms for identifying a Unary or Binary relation in RPM problems. We expect Layout-A (Figure 5.4a) to better align with Unary (`progression`) problems and Layout-B (Figure 5.4b) to better align with Binary (`AND, OR, XOR`) problems. In order to verify this empirically we train each layout individually with similar training hyperparameters as the full engine. We control the overall validation accuracy to select trained models which perform similarly across the full dataset. Next, we compare the test accuracy for Unary and Binary relations across inference engines with Layout-A and Layout-B in Table 5.3.

Modular Layout	**Training Accuracy**			**Test Accuracy % (Contrasting)**			**Test Accuracy % (Normal)**			**Test Accuracy % (Mixed)**		
	Contrasting	Normal	Mixed	Contrasting	Normal	Mixed	Contrasting	Normal	Mixed	Contrasting	Normal	Mixed
Layout A	**88.47**/94.66	**91.81**/91.3	91.72/**93.91**	**87.63**/94.21	**89.74/90.89**	91.51/**95.19**	**79.54/76.32**	**90.76**/90.6	91.47/**92.03**	**83.53/85.3**	**90.26**/90.74	91.49/**93.61**
Layout B	86.44/**95**	90.79/**92.4**	**92.85**/93.54	85.35/**94.28**	77.6/90.28	**91.96**/94.73	74.83/75.05	89.5/**91.43**	**92.83**/91.86	80.01/84.7	83.64/**90.85**	**92.4**/93.3

Table 5.3: Comparison of the analogy inference engine layouts on corresponding (unary/binary) `relationship` types

In practice, we found significant stochasticity in the performance between Unary and Binary relation types across both layouts. From the observed results, we cannot conclu-

sively say that our layouts necessarily align better with one or the other relationship problem. One possible explanation for this could be that there can be alternate algorithms for evaluating the same relation, which align algorithmically with these layouts. This is similar to Xu et al. (2020)'s observation that a neural network which would have not aligned with the Bellman-Ford algorithm was found to align well with a different algorithm for the same dynamic programming problem.

5.6.3 Does having an adaptive engine enable better analogy inference?

We use an adaptive modular engine for analogy inference which chooses between two possible module net layouts at inference time. This enables our engine to compositionally reason about the underlying relation. We expect our adaptive setup to outperform a static layout since the engine can then chose to use a better fit layout according to the mapped `relationship` structure. To verify our hypothesis, we compare our input-adaptive engine to a fixed engine. We chose each candidate layout described in Section 5.3.2 as a possible inference engine. We call these fixed models Engine-A and Engine-B, and train them similar to our full model. We then compare the test accuracy performance of our full model, which we label Engine-Full, against the test accuracy fixed models.

Inference Engine	Training Accuracy %			Test Accuracy % (Contrasting)			Test Accuracy % (Normal)			Test Accuracy % (Mixed)		
	Contrasting	Normal	Mixed	Contrasting	Normal	Mixed	Contrasting	Normal	Mixed	Contrasting	Normal	Mixed
Engine-A	93.39	91.41	93.46	92.88	90.66	94.44	77.00	90.63	91.91	84.94	90.65	93.18
Engine-B	93.24	92.07	93.40	92.47	87.70	94.16	75.01	91.02	92.06	83.74	89.36	93.11
Engine-Full	93.12	91.04	93.32	91.07	90.40	93.30	74.57	88.92	90.02	82.81	89.66	91.65

Table 5.4: Comparison of fixed analogy inference engines versus the full adaptive engine

Empirically, we found that the adaptive engine was not able to outperform the fixed engines, contrary to our expectation. We believe there are two main reasons for this: (1) Our individual engines are not sufficiently differentiated in their performance on their respective reasoning tasks (as shown in Table 5.3), so there is not a significant performance gain in choosing one over the other, and (2) the number of possible reasoning scenarios is quite limited in the abstract visual analogy problem, and hence the adaptive engine is not able to benefit from learning a large number of mixture of experts.

5.6.4 Visual Relationship Encoder Ablations

Multi-task versus Multi-label Visual Relationship Encoder

In our two-step structure mapping approach, we treat the structure extraction problem as a multi-task learning problem where the `relationship`, `object`, `attribute` prediction are treated as separate learning tasks. We could have alternatively treated the structure extraction problem as a multi-label classification problem and instead predicted

the full triplet $\mathcal{R} = \{o, a, r\}$. A multi-label model could in principle learn from additional interaction between the classification layer weights for the domain components.

In order to draw a comparison between both these learning approaches, we trained a multi-label visual relationship encoder which predicts the triplet $\mathcal{R} = \{o, a, r\}$ as its output. We replaced the $l_\phi^{\text{task}} = FC_\phi^{\text{task}}(l_\phi^{\text{shared}})$ layers in out multi-task encoder with a $l_\phi^{\text{shared}_2} = FC_\phi^{\text{shared}_2}(l_\phi^{\text{shared}})$ layer, and the final classifiers $out_\phi^{\text{task}} = FC_\phi^{\text{out}^{\text{task}}}(l_\phi^{\text{task}})$ of sizes 2, 5, and 4, with one classifier $out_\phi = FC_\phi^{\text{out}}(l_\phi^{\text{shared}_2})$ of size 11 (2+5+4). For training, we used exactly the same procedure as the multi-task encoder (described in Section 5.4.1).

We compared the performance of these models by evaluating there test accuracy across three different generalization splits: Novel Domain Transfer, Novel Domain: Line Type, and Novel Attribute Value: Extrapolation. Since our encoder is not trained with the candidate panels, we restrict ourselves to one training regime (Contrasting) for this comparison. Our observations are reported below in Table 5.6.

Encoder	Extrapolation % Accuracy				Novel Domain Transfer % Accuracy				Novel Domain (line type) % Accuracy			
	Training	Test (Contrasting)	Test (Normal)	Test (Mixed)	Training	Test (Contrasting)	Test (Normal)	Test (Mixed)	Training	Test (Contrasting)	Test (Normal)	Test (Mixed)
Multi-task	86.23	84.78	84.75	84.76	86.96	82.52	82.75	82.64	86.16	84.93	84.58	84.75
Multi-label	86.11	84.84	84.98	84.91	85.31	84.13	84.38	84.25	85.80	85.16	85.15	85.16

Table 5.5: Comparison of `relationship` prediction accuracy of a multi-task versus a multi-label visual relationship encoder

We found the performance of the two types of relationship encoders to be quite similar in terms of the test accuracy for predicting the `relationship` from the source domain panels. In fact, the multi-label encoder was slightly better than the multi-task encoder across all generalization splits. Despite this, we still went ahead with using the multi-task encoder in our two-step approach since the multi-task learning network scales much better with the number of objects, attributes, and relationships in the perceptual domain.

Multi-task versus Relationship-Only Visual Relationship Encoder

We also performed an ablation study on predicting only the visual `relationship` from the encoder and comparing it with the full-task of predicting `relationship`, `object`, `attribute`. We found that predicting only the relationship structure is competitive with a multi-task predictor in the Novel Domain Transfer regime, and since our encoder does not perform generalization across domains or candidate types this should hold for other regimes as well. This does not take away from our central contribution of incorporating structure mapping into our model, rather it only highlights the importance of structure extraction as the first-step of our model.

Encoder	Novel Domain Transfer % Accuracy (Contrastive Training)				Novel Domain Transfer % Accuracy (Normal Training)			
	Training	Test (Contrasting)	Test (Normal)	Test (Mixed)	Training	Test (Contrasting)	Test (Normal)	Test (Mixed)
Multi-task	86.96	82.52	82.75	82.64	86.66	82.40	83.02	82.71
Relationship-only	84.80	83.61	83.6	83.61	85.74	83.68	83.80	83.74

Table 5.6: Comparison of `relationship` prediction accuracy of a multi-task versus relationship-only visual relationship encoder

5.6.5 Implementation Details

Visual Relationship Encoder

Index	Layer	Output Size
1	Panel Image Input	1 x 160 x 160
2	Conv(3 x 3, 1→ 8, stride 2)	8 x 79 x 79
3	BatchNorm	8 x 79 x 79
4	ReLU	8 x 79 x 79
5	Conv(3 x 3, 8→ 8, stride 2)	8 x 39 x 39
6	BatchNorm	8 x 39 x 39
7	ReLU	8 x 39 x 39
8	Conv(3 x 3, 8→ 8, stride 2)	8 x 19 x 19
10	BatchNorm	8 x 19 x 19
12	ReLU	8 x 19 x 19
13	Conv(3 x 3, 8→ 8, stride 2)	8 x 9 x 9
14	BatchNorm	8 x 9 x 9
15	ReLU	8 x 9 x 9
16	FC(8*9*9 → 128)	128
17	BatchNorm	128
15	ReLU	128

Table 5.7: Encoder CNN architecture

Index	Layer	Output Size
1	Encoder-CNN Output (3 panels)	3 x 128
2	LSTM (hidden dim = 128, sequence length = 3)	output = 3 x 128 final hidden state = 128 final cell state = 128

Table 5.8: Encoder LSTM architecture

Index	Layer	Output Size
1	Encoder-LSTM final hidden state	128
3	FC(128 → 128)	128
7	ReLU	128
4	FC_{task}(128 → 64)	64
7	ReLU	64
5	$\text{FC}_{classifier_{task}}$(64 → $\|Task\|$)	$\|Task\|$

Table 5.9: Encoder Multi-Task classifier architecture. $|Task|$ = 2, 5, 4 for `object`, `attribute`, and `relationship` classifiers respectively

Analogy Inference Engine

Index	Layer	Output Size
1	Panel Image Input	1 x 160 x 160
2	Conv(3 x 3, 1→ 8, stride 2)	8 x 79 x 79
3	BatchNorm	8 x 79 x 79
4	ReLU	8 x 79 x 79
5	Conv(3 x 3, 8→ 8, stride 2)	8 x 39 x 39
6	BatchNorm	8 x 39 x 39
7	ReLU	8 x 39 x 39
8	Conv(3 x 3, 8→ 8, stride 2)	8 x 19 x 19
10	BatchNorm	8 x 19 x 19
12	ReLU	8 x 19 x 19
13	Conv(3 x 3, 8→ 8, stride 2)	8 x 9 x 9
14	BatchNorm	8 x 9 x 9
15	ReLU	8 x 9 x 9

Table 5.10: `Stem` module architecture

Index	Layer	Output Size
1	Previous Module Output	8 x 9 x 9
2	Conv(3 x 3, 8→ 8)	8 x 9 x 9
3	ReLU	8 x 9 x 9
4	Conv(3 x 3, 8→ 8)	8 x 9 x 9
5	Residual: Add (1) and (4)	8 x 9 x 9
6	ReLU	8 x 9 x 9

Table 5.11: `Unary` module architecture

Index	Layer	Output Size
1	Previous Module Output	8 x 9 x 9
2	Previous Module Output	8 x 9 x 9
3	Concatenate (1) and (2)	16 x 9 x 9
4	Conv(1 x 1, 16→ 8)	8 x 9 x 9
5	ReLU	8 x 9 x 9
6	Conv(3 x 3, 8→ 8)	8 x 9 x 9
7	ReLU	8 x 9 x 9
8	Conv(3 x 3, 8→ 8)	8 x 9 x 9
9	Residual: Add (5) and (8)	8 x 9 x 9
10	ReLU	8 x 9 x 9

Table 5.12: `Binary` module architecture

Index	Layer	Output Size
1	Previous Module Output	8 x 9 x 9
2	Previous Module Output	8 x 9 x 9
3	Concatenate (1) and (2)	16 x 9 x 9
4	Conv(1 x 1, 16 → 8)	8 x 9 x 9
5	ReLU	8 x 9 x 9
6	FC(8*9*9 → 256)	256
7	ReLU	256
8	FC(256 → $\|r\|$)	$\|r\|$
9	Softmax($\|c\|$ x $\|r\|$ → $\|c\|$) (over subset of (8) along index r_{pred} for all $\|c\|$ candidates)	$\|c\|$

Table 5.13: `Classifier` module architecture. $|r|$ = 4. $|c|$ = 4.

5.6.6 Supplementary Results

Visual Relationship Encoder

GENERALIZATION SPLIT	Training Accuracy %			Test Accuracy % (Contrasting)			Test Accuracy % (Normal)			Test Accuracy % (Mixed)		
	Contrasting	Normal	Mixed	Contrasting	Normal	Mixed	Contrasting	Normal	Mixed	Contrasting	Normal	Mixed
Extrapolation	86.23	86.44	86.75	84.78	84.84	85.86	84.75	84.92	86.06	84.76	84.88	85.96
Interpolation	86.34	86.54	86.56	84.71	84.86	85.57	84.67	85.01	85.81	84.69	84.93	85.69
Novel Domain Transfer	86.96	86.66	86.30	82.52	82.40	83.18	82.75	83.02	83.88	82.64	82.71	83.53
Novel Domain (shape color)	86.68	86.48	86.27	85.16	84.91	85.55	85.00	84.98	85.62	85.08	84.95	85.58
Novel Domain (line type)	86.16	86.23	86.30	84.93	85.07	85.46	84.58	84.70	85.46	84.75	84.88	85.46

Table 5.14: Relationship prediction accuracy of our visual relationship encoder

GENERALIZATION SPLIT	Training Accuracy %			Test Accuracy % (Contrasting)			Test Accuracy % (Normal)			Test Accuracy % (Mixed)		
	Contrasting	Normal	Mixed	Contrasting	Normal	Mixed	Contrasting	Normal	Mixed	Contrasting	Normal	Mixed
Extrapolation	98.27	99.87	99.32	66.89	61.77	59.61	58.23	57.97	52.12	62.56	59.87	55.87
Interpolation	98.44	99.94	99.33	95.17	82.69	95.45	70.82	95.67	94.62	82.69	89.35	95.02
Novel Domain Transfer	98.72	97.11	97.86	90.93	90.34	93.35	74.49	88.89	89.95	82.70	89.63	91.66
Novel Domain (shape color)	96.63	98.23	98.39	78.80	76.41	83.51	60.71	66.74	70.81	69.76	71.58	77.15
Novel Domain (line type)	96.26	96.43	96.78	79.84	72.29	79.98	61.58	65.27	65.25	70.71	68.79	72.61

Table 5.15: Engine layout prediction accuracy of our visual relationship encoder

Analogy Inference Engine

GENERALIZATION SPLIT	Training Accuracy %			Test Accuracy % (Contrasting)			Test Accuracy % (Normal)			Test Accuracy % (Mixed)		
	Contrasting	Normal	Mixed	Contrasting	Normal	Mixed	Contrasting	Normal	Mixed	Contrasting	Normal	Mixed
Extrapolation	97.31	99.58	99.08	65.24	62.33	58.40	57.14	58.08	50.98	61.19	60.20	54.69
Interpolation	97.47	99.78	98.90	93.46	82.15	94.55	69.81	95.11	94.03	81.33	88.80	94.29
Novel Domain Transfer	97.31	95.70	96.44	87.96	87.48	90.81	73.07	86.78	87.94	80.50	87.13	89.38
Novel Domain (shape color)	95.82	97.61	97.75	77.79	75.64	82.69	58.72	66.27	70.26	68.25	70.96	76.47
Novel Domain (line type)	94.90	95.40	95.38	78.14	70.64	78.53	60.23	64.31	64.10	69.18	67.48	71.32

Table 5.16: Candidate prediction accuracy of our two-step model

GENERALIZATION SPLIT	Training Accuracy %			Test Accuracy % (Contrasting)			Test Accuracy % (Normal)			Test Accuracy % (Mixed)		
	Contrasting	Normal	Mixed	Contrasting	Normal	Mixed	Contrasting	Normal	Mixed	Contrasting	Normal	Mixed
Extrapolation	96.96	99.41	98.52	73.56	66.75	63.47	59.94	59.84	52.92	66.65	63.29	58.19
Interpolation	97.18	99.68	98.73	93.11	85.40	94.95	70.88	94.86	93.89	81.71	90.25	94.41
Novel Domain Transfer	90.81	88.75	91.25	88.57	88.61	91.40	73.02	86.80	88.15	80.80	87.71	89.78
Novel Domain (shape color)	95.26	96.91	97.23	78.43	79.50	83.15	58.08	66.18	69.66	68.25	72.83	76.40
Novel Domain (line type)	94.15	94.58	94.84	79.75	76.18	80.57	59.55	62.43	65.74	69.65	69.30	73.15

Table 5.17: Candidate prediction accuracy of our full-context ensemble

Chapter 6

Discussion

In this thesis we explored inductive biases to augment deep learning-based computer vision models with higher-order cognitive functions. We believe that this is a central research problem to solve in order for deep learning to grow beyond its current limitations, such as poor out-of-distribution (OOD) generalization, lack of black-box interpretability, and limited reasoning ability. We focused on the dynamic inference inductive bias, that enables models to perform a variable amount of compute for different inputs and designed the Neural Response Time (NRT) and Neural Structure Mapping (NSM) inductive biases to address some of the aforementioned issues.

NRT is one of the first approaches to explain neural network decisions in a black-box manner. It brings together two different directions of research: (a) Methods like LIME (Ribeiro et al., 2016) which aim to provide black-box explanations for any classifier, and (b) Methods like Shallow-Deep networks (Kaya et al., 2019) which utilized dynamic inference probes to understand the feature space of neural nets. Our central contribution in NRT is the proposal of a model agnostic explainable AI solution which can be used to test falsifiable hypotheses about neural network feature spaces without access to the model parameters. We have seen positive response from the research community regarding NRT. Our initial proposal was invited for an oral presentation at a workshop on the intersection between computer vision and cognitive science, and NRT has since been evaluated in a survey on AI assurance and given a score of 5 by the surveying authors. However, one potential barrier to its widespread adoption could be the general sparsity of falsifiable interpretability research in AI, as well as its reliance on controlled datasets for evaluating hypotheses. We look forward to seeing how other researchers utilize NRT.

Our NSM framework was able to demonstrate good performance on isolating relationship structure from abstract visual scenes and then perform analogical reasoning using this structure. However, it is currently limited by the availability of relationship labels in the training data. In order to make NSM more generally applicable, it would be prudent to develop an approach that is able to infer latent structure from data. A promising approach in this direction could be to learn self-supervised visual representations of the source domain, and then clustering said representations along 'perceptual' and 'semantic' axes. It

is unclear how semanticity can be quantified in an unsupervised manner and is an interesting open research problem. Another limitation of NSM was that using adaptive inference networks did not provide notable improvement in performance unlike language-grounded reasoning tasks like visual question answering. One potential reason behind this could be the lack of diversity in the analogy task in terms of types of semantic structures. Scaling up NSM to a larger set of relationship, and potentially higher-order relationships would provide a better understanding of the utility of our adaptive inference engine. We aim to pursue further experiments and refine our NSM framework to be more generally applicable for visual analogy making beyond abstract reasoning, as well as to serve as a component in multi-faceted visual reasoning problems.

The overarching theme in both our projects was the idea of utilizing the dynamic inference inductive bias for higher-order cognition: for generating explanations, or for performing reasoning that generalizes on OOD domains. We were successful in answering these research problems because the dynamic inference prior enabled us to probe into neural networks at various levels of processing and also to lay out neural networks based on semantic information. Since then, additional work exploring dynamic inference for similar problems has emerged in machine learning research. Banino et al. (2021) introduced PonderNet, which is a learning algorithm for neural networks that incentivizes exploration over spending additional computation. PonderNet was shown to outperform networks trained without the dynamic inference objective on reasoning tasks and also generalize well on OOD data. Csordás et al. (2021a) introduced Transformer Control Flows, which route the computation in transformer neural network (Vaswani et al., 2017) to dynamically apply compositional unitary functions. Again, dynamic inference was shown to enhance the ability of transformers to generalize systematically. Rahaman* et al. (2021) combined the ideas of control flow in transformers and our approach towards building modularity for systematic reasoning into their Neural Interpreter mechanism. A Neural Interpreter learn functions comprised of self-attention blocks (unlike our residual functions), and also learn a type-matching and interpreter mechanism to execute scripts (comprised of functions) in a dynamic manner. The authors showed that Neural Interpreters generalize to OOD domains in image recognition in a sample efficient manner. Furthermore, they also showed that their models outperform vision transformers (Dosovitskiy et al., 2020) and other specialized models in terms of systematic generalization on the PGM abstract reasoning task. These results are very promising and validate our central theme that dynamic inference could serve as a useful inductive bias to address the problems that we set out to solve.

This thesis is a step in developing inductive biases for addressing complex cognitive problems in computer vision. As computer vision moves beyond pattern recognition problems to higher-order cognition, the emphasis on building priors to solve new challenges is only going to increase. We demonstrated that our inductive bias of focus, dynamic inference, can be utilized to address some of these challenges. In the future, we look forward to computer vision and computational cognitive science research to continue exploring dynamic inference and other inductive biases for achieving higher-order cognition.

References

Nancy E Adams. Blooms taxonomy of cognitive learning objectives. *Journal of the Medical Library Association: JMLA*, 103(3):152, 2015.

Ashish Agarwal. Static automatic batching in TensorFlow. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 92–101. PMLR, 09–15 Jun 2019. URL `http://proceedings.mlr.press/v97/agarwal19a.html`.

Aishwarya Agrawal, Aniruddha Kembhavi, Dhruv Batra, and Devi Parikh. C-vqa: A compositional split of the visual question answering (vqa) v1. 0 dataset. *arXiv preprint arXiv:1704.08243*, 2017.

Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016.

Jean-Baptiste Alayrac, Piotr Bojanowski, Nishant Agrawal, Josef Sivic, Ivan Laptev, and Simon Lacoste-Julien. Unsupervised learning from narrated instruction videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4575–4583, 2016.

Ahmed Alqaraawi, Martin Schuessler, Philipp Weiß, Enrico Costanza, and Nadia Berthouze. Evaluating saliency map explanations for convolutional neural networks: a user study. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, pages 275–285, 2020.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 39–48. IEEE, 2016.

OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray,

et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Lisa Anne Hendricks, Ronghang Hu, Trevor Darrell, and Zeynep Akata. Grounding visual explanations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 264–279. ECVA, 2018.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015.

Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *International Conference on Machine Learning*, pages 233–242. PMLR, 2017.

Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron Courville. Systematic generalization: What is required and can it be learned? In *International Conference on Learning Representations*, 2018.

Dzmitry Bahdanau, Harm de Vries, Timothy J O'Donnell, Shikhar Murty, Philippe Beaudoin, Yoshua Bengio, and Aaron Courville. Closure: Assessing systematic generalization of clevr models. *arXiv preprint arXiv:1912.05783*, 2019.

Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. Recent advances in adversarial training for adversarial robustness. *arXiv preprint arXiv:2102.01356*, 2021.

Stephen Balaban. Deep learning and face recognition: the state of the art. In *Biometric and Surveillance Technology for Human and Activity Identification XII*, volume 9457, page 94570B. International Society for Optics and Photonics, 2015.

Andrea Banino, Jan Balaguer, and Charles Blundell. Pondernet: Learning to ponder. In *8th ICML Workshop on Automated Machine Learning (AutoML)*, 2021.

Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. *Advances in neural information processing systems*, 32:9453–9463, 2019.

David Barrett, Felix Hill, Adam Santoro, Ari Morcos, and Timothy Lillicrap. Measuring abstract reasoning in neural networks. In *International Conference on Machine Learning*, pages 511–520. PMLR, 2018.

Harry G Barrow, Jay M Tenenbaum, Robert C Bolles, and Helen C Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. Technical

report, SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER, 1977.

Akram Bayat, Anubhaw Kumar Nand, Do Hyong Koh, Marta Pereira, and Marc Pomplun. Scene Grammar in Human and Machine Recognition of Objects and Scenes. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2073–20737, Salt Lake City, UT, USA, June 2018. IEEE, IEEE. ISBN 978-1-5386-6100-0. doi: 10.1109/CVPRW.2018.00268. URL https://ieeexplore.ieee.org/document/8575436/.

Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 17–36. JMLR Workshop and Conference Proceedings, 2012.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013a.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *arXiv:1308.3432 [cs]*, August 2013b. URL http://arxiv.org/abs/1308.3432. arXiv: 1308.3432.

Irving Biederman, Robert J. Mezzanotte, and Jan C. Rabinowitz. Scene perception: Detecting and judging objects undergoing relational violations. *Cognitive Psychology*, 14(2):143–177, April 1982. ISSN 00100285. doi: 10.1016/0010-0285(82)90007-X. URL https://linkinghub.elsevier.com/retrieve/pii/001002858290007X.

Benjamin S Bloom et al. Taxonomy of educational objectives. vol. 1: Cognitive domain. *New York: McKay*, 20(24):1, 1956.

Mariusz Bojarski, Philip Yeres, Anna Choromanska, Krzysztof Choromanski, Bernhard Firner, Lawrence Jackel, and Urs Muller. Explaining how a deep neural network trained with end-to-end learning steers a car. *arXiv preprint arXiv:1704.07911*, 2017.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

Jason Brownlee. No free lunch theorem for machine learning, Oct 2020. URL https://machinelearningmastery.com/no-free-lunch-theorem-for-machine-learning/.

Patricia A Carpenter, Marcel A Just, and Peter Shell. What one intelligence test measures: a theoretical account of the processing in the raven progressive matrices test. *Psychological review*, 97(3):404, 1990.

David J Chalmers, Robert M French, and Douglas R Hofstadter. High-level perception, representation, and analogy: A critique of artificial intelligence methodology. *Journal of Experimental & Theoretical Artificial Intelligence*, 4(3):185–211, 1992.

Kezhen Chen, Irina Rabkina, Matthew D McLure, and Kenneth D Forbus. Human-like sketch object recognition via analogical learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1336–1343, 2019.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668, 2017.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020a.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in Neural Information Processing Systems*, 33:22243–22255, 2020b.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Shamil Chollampatt and Hwee Tou Ng. Connecting the dots: Towards human-level grammatical error correction. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 327–333, 2017.

Ching-Yao Chuang, Jiaman Li, Antonio Torralba, and Sanja Fidler. Learning to act properly: Predicting and explaining affordances from images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 975–983, 2018.

CS231n. Cs231n convolutional neural networks for visual recognition. URL `https://cs231n.github.io/`.

Róbert Csordás, Kazuki Irie, and Jürgen Schmidhuber. Learning adaptive control flow in transformers for improved systematic generalization. In *Advances in Programming Languages and Neurosymbolic Systems Workshop*, 2021a.

Róbert Csordás, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Are neural nets modular? inspecting functional modularity through differentiable weight masks. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=7uVcpu-gMD.

Marie-Catherine De Marneffe and Christopher D Manning. The stanford typed dependencies representation. In *Coling 2008: proceedings of the workshop on cross-framework and cross-domain parser evaluation*, pages 1–8, 2008.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

James J DiCarlo, Davide Zoccolan, and Nicole C Rust. How does the brain solve visual object recognition? *Neuron*, 73(3):415–434, 2012.

Pedro Domingos. *The master algorithm: How the quest for the ultimate learning machine will remake our world*. Basic Books, 2015.

Franciscus Cornelis Donders. On the speed of mental processes. *Acta psychologica*, 30: 412–431, 1868.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

Dejan Draschkow and Melissa L.-H. Võ. Scene grammar shapes the way we interact with objects, strengthens memories, and speeds search. *Scientific Reports*, 7(1): 16471, December 2017. ISSN 2045-2322. doi: 10.1038/s41598-017-16739-x. URL http://www.nature.com/articles/s41598-017-16739-x.

Brian Falkenhainer, Kenneth D Forbus, and Dedre Gentner. *The structure-mapping engine*, volume 1275. Department of Computer Science, University of Illinois at Urbana-Champaign, 1986.

Jerry A Fodor. *The language of thought*, volume 5. Harvard university press, 1975.

Jerry A Fodor and Zenon W Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71, 1988.

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

Dedre Gentner. Structure-mapping: A theoretical framework for analogy. *Cognitive science*, 7(2):155–170, 1983.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Anirudh Goyal and Yoshua Bengio. Inductive biases for deep learning of higher-level cognition. *arXiv preprint arXiv:2011.15091*, 2020.

Kristen Grauman and Trevor Darrell. Efficient image matching with distributions of local invariant features. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 627–634. IEEE, 2005.

Alex Graves. Generating sequences with recurrent neural networks. In *arXiv:1308.0850*, 2013.

Alex Graves. Adaptive Computation Time for Recurrent Neural Networks. *arXiv:1603.08983 [cs]*, February 2017. URL `http://arxiv.org/abs/1603.08983`. arXiv: 1603.08983.

Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. On the binding problem in artificial neural networks. *arXiv preprint arXiv:2012.05208*, 2020.

Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Society for Industrial and Applied Mathematics, USA, second edition, 2008. ISBN 0898716594.

Dan Guest, Kyle Cranmer, and Daniel Whiteson. Deep learning and its application to lhc physics. *Annual Review of Nuclear and Particle Science*, 68:161–181, 2018.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. Colorless green recurrent networks dream hierarchically. *arXiv preprint arXiv:1803.11138*, 2018.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. IEEE, 2016.

Margaret M Henderson and John Serences. Biased orientation representations can be explained by experience with non-uniform training set statistics. *bioRxiv*, 2020.

Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. Generating visual explanations. In *European conference on computer vision*, pages 3–19. Springer, 2016.

Lisa Anne Hendricks, Ronghang Hu, Trevor Darrell, and Zeynep Akata. Grounding visual explanations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 264–279, 2018.

Hansika Hewamalage, Christoph Bergmeir, and Kasun Bandara. Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37(1):388–427, 2021. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2020.06.008. URL https://www.sciencedirect.com/science/article/pii/S0169207020300996.

Felix Hill, Adam Santoro, David Barrett, Ari Morcos, and Timothy Lillicrap. Learning to make analogies by contrasting abstract relational structure. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SylLYsCcFm.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Douglas R Hofstadter and Melanie Mitchell. The copycat project: A model of mental fluidity and analogy-making. 1994.

Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1):185–203, 1981. ISSN 0004-3702. doi: https://doi.org/10.1016/0004-3702(81)90024-2. URL https://www.sciencedirect.com/science/article/pii/0004370281900242.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(91)90009-T. URL https://www.sciencedirect.com/science/article/pii/089360809190009T.

Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 804–813, 2017.

Ronghang Hu, Jacob Andreas, Trevor Darrell, and Kate Saenko. Explainable neural computation via stack neural module networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 53–69, 2018.

Ting-Kuei Hu, Tianlong Chen, Haotao Wang, and Zhangyang Wang. Triple wins: Boosting accuracy, robustness and efficiency together by enabling input-adaptive inference. In *International Conference on Learning Representations*. ICLR, 2019.

Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q. Weinberger. Multi-Scale Dense Networks for Resource Efficient Image Classification. *arXiv:1703.09844 [cs]*, June 2018. URL `http://arxiv.org/abs/1703.09844`. arXiv: 1703.09844.

Drew A Hudson and Christopher D Manning. Compositional attention networks for machine reasoning. In *International Conference on Learning Representations*, 2018.

Drew A. Hudson and Christopher D. Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

N Ichien, Q Liu, S Fu, KJ Holyoak, AL Yuille, and H Lu. Visual analogy: Deep learning versus compositional models. In *Proceedings of the 43rd Annual Meeting of the Cognitive Science Society*, 2021.

Charmi Jobanputra, Jatna Bavishi, and Nishant Doshi. Human activity recognition: A survey. *Procedia Computer Science*, 155:698–703, 2019.

Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David Shamma, Michael Bernstein, and Li Fei-Fei. Image retrieval using scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3668–3678, 2015.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910, 2017a.

Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2989–2998, 2017b.

Dimitris Kalimeris, Gal Kaplun, Preetum Nakkiran, Benjamin Edelman, Tristan Yang, Boaz Barak, and Haofeng Zhang. Sgd on neural networks learns functions of increasing complexity. *Advances in Neural Information Processing Systems*, 32:3496–3506, 2019.

Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. Shallow-deep networks: Understanding and mitigating network overthinking. In *International Conference on Machine Learning*, pages 3301–3310. PMLR, 2019.

Osman Semih Kayhan and Jan C. van Gemert. On translation invariance in cnns: Convolutional layers can exploit absolute spatial location. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Been Kim, Emily Reif, Martin Wattenberg, and Samy Bengio. Do Neural Networks Show Gestalt Phenomena? An Exploration of the Law of Closure. *arXiv:1903.01069 [cs, stat]*, March 2019. URL `http://arxiv.org/abs/1903.01069`. arXiv: 1903.01069.

Jinkyu Kim, Anna Rohrbach, Trevor Darrell, John Canny, and Zeynep Akata. Textual explanations for self-driving vehicles. In *Proceedings of the European conference on computer vision (ECCV)*, pages 563–578, 2018.

Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un) reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 267–280. Springer, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd Int. Conf. on Learning Representations (ICLR)*, 2015.

Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo, Japan, July 2003. Association for Computational Linguistics. doi: 10.3115/1075096.1075150. URL `https://aclanthology.org/P03-1054`.

Mykel J Kochenderfer and Tim A Wheeler. *Algorithms for optimization*. Mit Press, 2019.

Talia Konkle, Timothy F Brady, George A Alvarez, and Aude Oliva. Conceptual distinctiveness supports detailed visual long-term memory for real-world objects. *Journal of Experimental Psychology: General*, 139(3):558, 2010.

David R Krathwohl. A revision of bloom's taxonomy: An overview. *Theory into practice*, 41(4):212–218, 2002.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25: 1097–1105, 2012.

Brenden Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pages 2873–2882. PMLR, 2018.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Matthew L Leavitt and Ari Morcos. Towards falsifiable interpretability research. *arXiv preprint arXiv:2010.12016*, 2020.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature Cell Biology*, 521(7553):436–444, May 2015. ISSN 1465-7392. doi: 10.1038/nature14539.

Jie Lei, Licheng Yu, Mohit Bansal, and Tamara L Berg. Tvqa: Localized, compositional video question answering. In *EMNLP*, 2018.

Joel Z. Leibo, Cyprien de Masson d'Autume, Daniel Zoran, David Amos, Charles Beattie, Keith Anderson, Antonio García Castañeda, Manuel Sanchez, Simon Green, Audrunas Gruslys, Shane Legg, Demis Hassabis, and Matthew M. Botvinick. Psychlab: A Psychology Laboratory for Deep Reinforcement Learning Agents. *arXiv:1801.08116 [cs, q-bio]*, January 2018. URL `http://arxiv.org/abs/1801.08116`. arXiv: 1801.08116.

Hao Li, Hong Zhang, Xiaojuan Qi, Ruigang Yang, and Gao Huang. Improved techniques for training adaptive deep networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1891–1900. IEEE, 2019.

Daqing Liu, Hanwang Zhang, Feng Wu, and Zheng-Jun Zha. Learning to assemble neural module tree networks for visual grounding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

João Loula, Marco Baroni, and Brenden Lake. Rearranging the familiar: Testing compositional generalization in recurrent networks. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 108–114, 2018.

Andrew Lovett and Kenneth Forbus. Modeling visual problem solving as analogical reasoning. *Psychological review*, 124(1):60, 2017.

Loren Lugosch, Derek Nowrouzezahrai, and Brett H Meyer. Surprisal-triggered conditional computation with neural networks. *arXiv preprint arXiv:2006.01659*, 2020.

Zelun Luo, Boya Peng, De-An Huang, Alexandre Alahi, and Li Fei-Fei. Unsupervised learning of long-term motion dynamics for videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2203–2212, 2017.

Gary F Marcus. *The algebraic mind: Integrating connectionism and cognitive science*. MIT press, 2019.

David Marr and Ellen Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207(1167):187–217, 1980.

Marvin Minsky and Seymour Papert. Perceptrons. 1969.

Melanie Mitchell. *Analogy-making as perception: A computer model*. 1993.

Melanie Mitchell. Abstraction and analogy-making in artificial intelligence. *arXiv preprint arXiv:2102.10717*, 2021.

Tom M Mitchell. *The need for biases in learning generalizations*. Department of Computer Science, Laboratory for Computer Science Research , 1980.

P Molchanov, S Tyree, T Karras, T Aila, and J Kautz. Pruning convolutional neural networks for resource efficient inference. In *5th International Conference on Learning Representations, ICLR 2017-Conference Track Proceedings*, 2019.

Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020.

Roozbeh Mottaghi, Mohammad Rastegari, Abhinav Gupta, and Ali Farhadi. what happens if... learning to predict the effect of forces in images. In *European conference on computer vision*, pages 269–285. Springer, 2016.

Abdulmajid Murad and Jae-Young Pyun. Deep recurrent neural networks for human activity recognition. *Sensors*, 17(11):2556, 2017.

Tatwadarshi P. Nagarhalli, Vinod Vaze, and N. K. Rana. Impact of machine learning in natural language processing: A review. In *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, pages 1529–1534, 2021. doi: 10.1109/ICICV50876.2021.9388380.

Andrew Joohun Nam and James L McClelland. What underlies rapid learning and systematic generalization in humans. *arXiv preprint arXiv:2107.06994*, 2021.

Graham Neubig, Yoav Goldberg, and Chris Dyer. On-the-fly operation batching in dynamic computation graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 3974–3984, 2017.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Understanding neural networks via feature visualization: A survey. In *Explainable AI: interpreting, explaining and visualizing deep learning*, pages 55–76. Springer, 2019.

Weili Nie, Zhiding Yu, Lei Mao, Ankit B Patel, Yuke Zhu, and Anima Anandkumar. Bongard-logo: A new benchmark for human-level concept learning and reasoning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 16468–16480. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/bf15e9bbff22c7719020f9df4badc20a-Paper.pdf`.

Michael A Nielsen. *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, 2015.

Sabine Öhlschläger and Melissa Le-Hoa Võ. Scegram: An image database for semantic and syntactic inconsistencies in scenes. *Behavior research methods*, 49(5):1780–1791, 2017.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11):e7, 2017.

Peter Ondruska and Ingmar Posner. Deep tracking: Seeing beyond seeing using recurrent neural networks. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

Ian Osband, Yotam Doron, Matteo Hessel, John Aslanides, Eren Sezener, Andre Saraiva, Katrina McKinney, Tor Lattimore, Csaba Szepezvari, Satinder Singh, et al. Behaviour suite for reinforcement learning. *arXiv preprint arXiv:1908.03568*, 2019.

Niall OMahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. Deep learning vs. traditional computer vision. In *Science and Information Conference*, pages 128–144. Springer, 2019.

Dong Huk Park, Lisa Anne Hendricks, Zeynep Akata, Anna Rohrbach, Bernt Schiele, Trevor Darrell, and Marcus Rohrbach. Multimodal explanations: Justifying decisions and pointing to the evidence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8779–8788, 2018.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep

learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Niv Pekar, Yaniv Benny, and Lior Wolf. Generating correct answers for progressive matrices intelligence tests. *Advances in Neural Information Processing Systems*, 33, 2020.

Effie J. Pereira and Monica S. Castelhano. Peripheral guidance in scenes: The interaction of scene context and object content. *Journal of Experimental Psychology: Human Perception and Performance*, 40(5):2056, 2014. ISSN 1939-1277. doi: 10.1037/a0037524. URL https://psycnet.apa.org/fulltext/2014-31982-001.pdf.

Mary Phuong and Christoph H Lampert. Distillation-based training for multi-exit architectures. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1355–1364. IEEE, 2019.

Zhuwei Qin, Fuxun Yu, Chenchen Liu, and Xiang Chen. How convolutional neural networks see the world—a survey of convolutional neural network visualization methods. *Mathematical Foundations of Computing*, 1(2):149, 2018.

J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

N. Rahaman*, M. W. Gondal*, S. Joshi, P. Gehler, Y. Bengio, F. Locatello, and B. Schölkopf. Dynamic inference with neural interpreters. In *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*, December 2021. *equal contribution.

Iyad Rahwan, Manuel Cebrian, Nick Obradovich, Josh Bongard, Jean-François Bonnefon, Cynthia Breazeal, Jacob W. Crandall, Nicholas A. Christakis, Iain D. Couzin, Matthew O. Jackson, Nicholas R. Jennings, Ece Kamar, Isabel M. Kloumann, Hugo Larochelle, David Lazer, Richard McElreath, Alan Mislove, David C. Parkes, Alex 'Sandy' Pentland, Margaret E. Roberts, Azim Shariff, Joshua B. Tenenbaum, and Michael Wellman. Machine behaviour. *Nature*, 568(7753):477–486, April 2019. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-019-1138-y. URL http://www.nature.com/articles/s41586-019-1138-y.

Rishi Rajalingham, Elias B Issa, Pouya Bashivan, Kohitij Kar, Kailyn Schmidt, and James J DiCarlo. Large-scale, high-resolution comparison of the core visual object recognition behavior of humans, monkeys, and state-of-the-art deep artificial neural networks. *Journal of Neuroscience*, 38(33):7255–7269, 2018.

John Raven. The raven's progressive matrices: change and stability over culture and time. *Cognitive psychology*, 41(1):1–48, 2000.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99, 2015.

Marco Tulio Ribeiro, Sameer Singh, and Carlos 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*, pages 1135–1144. ACM, 2016.

Brandon Richard Webster, So Yon Kwon, Christopher Clarizio, Samuel E Anthony, and Walter J Scheirer. Visual psychophysics for making face recognition algorithms more explainable. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 252–270. ECVA, 2018.

Samuel Ritter, David GT Barrett, Adam Santoro, and Matt M Botvinick. Cognitive psychology for deep neural networks: A shape bias case study. In *International conference on machine learning*, pages 2940–2949. PMLR, 2017.

Gemma Roig, Anna Volokitin, and Tomaso Poggio. Do deep neural networks suffer from crowding? *Journal of Vision*, 18(10):902–902, 2018.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323, 1986.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. *Learning Representations by Back-Propagating Errors*, page 696699. MIT Press, Cambridge, MA, USA, 1988. ISBN 0262010976.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. *Advances in Neural Information Processing Systems*, 30, 2017.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

Shashank Shekhar and Graham W. Taylor. Neural structure mapping for learning abstract visual analogies. In *SVRHM 2021 Workshop @ NeurIPS*, 2021. URL `https://openreview.net/forum?id=l-TLGjxwajn`.

Jiaxin Shi, Hanwang Zhang, and Juanzi Li. Explainable and explicit visual reasoning over scene graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8376–8384, 2019.

Zhangzhang Si and Song-Chun Zhu. Learning and-or templates for object recognition and detection. *IEEE transactions on pattern analysis and machine intelligence*, 35(9): 2189–2205, 2013.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Ajeet Kumar Singh, Anand Mishra, Shashank Shekhar, and Anirban Chakraborty. From strings to things: Knowledge-enabled vqa model that can read and reason. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4602–4612, 2019.

Samuel L Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V Le. Don't decay the learning rate, increase the batch size. In *International Conference on Learning Representations*, 2018.

Steven Spratley, Krista Ehinger, and Tim Miller. A closer look at generalisation in raven. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII 16*, pages 601–616. Springer, 2020.

Saul Sternberg. The discovery of processing stages: Extensions of donders method. *Acta psychologica*, 30(0):276–315, 1969.

Saul Sternberg, Ronald L Knoll, et al. The perception of temporal order: Fundamental issues and a general model. *Attention and performance IV*, pages 629–685, 1973.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR, 2017.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, 2015.

Eric Taylor, Shashank Shekhar, and Graham W Taylor. Response time analysis for explainability of visual processing in cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 382–383, 2020.

Eric Taylor, Shashank Shekhar, and Graham Taylor. Neural response time analysis: XAI using only a stopwatch, 2021. URL `https://doi.org/10.22541/au.162316905.55031484/v1`. (under review).

J Eric T Taylor and Graham W Taylor. Artificial cognition: How experimental psychology can help generate explainable artificial intelligence. *Psychonomic Bulletin & Review*, pages 1–22, 2020.

Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469. IEEE, 2016.

Damien Teney, Peng Wang, Jiewei Cao, Lingqiao Liu, Chunhua Shen, and Anton van den Hengel. V-prom: A benchmark for visual reasoning using visual progressive matrices. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12071–12078, 2020.

Philip HS Torr and Andrew Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer vision and image understanding*, 78(1):138–156, 2000.

Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global, 2010.

Anne M Treisman and Garry Gelade. A feature-integration theory of attention. *Cognitive psychology*, 12(1):97–136, 1980.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30*. 2017.

Paul Viola, Michael Jones, et al. Robust real-time object detection. *International journal of computer vision*, 4(34-47):4, 2001.

M. L. H. Vo and J. M. Henderson. Does gravity matter? Effects of semantic and syntactic inconsistencies on the allocation of attention during scene perception. *Journal of Vision*, 9(3):24–24, March 2009. ISSN 1534-7362. doi: 10.1167/9.3.24. URL `http://jov.arvojournals.org/Article.aspx?doi=10.1167/9.3.24`.

Melissa L.-H. Võ and Jeremy M. Wolfe. Differential Electrophysiological Signatures of Semantic and Syntactic Scene Processing. *Psychological Science*, 24(9):1816–1823, September 2013. ISSN 0956-7976, 1467-9280. doi: 10.1177/0956797613476955. URL http://journals.sagepub.com/doi/10.1177/0956797613476955.

Melissa Le-Hoa Võ, Sage EP Boettcher, and Dejan Draschkow. Reading scenes: how scene grammar guides attention and aids perception in real-world environments. *Current Opinion in Psychology*, 29:205–210, October 2019. ISSN 2352250X. doi: 10.1016/j.copsyc.2019.03.009. URL https://linkinghub.elsevier.com/retrieve/pii/S2352250X18302574.

Dirk Walther and Christof Koch. Modeling attention to salient proto-objects. *Neural networks*, 19(9):1395–1407, 2006.

Duo Wang, Mateja Jamnik, and Pietro Lio. Abstract diagrammatic reasoning with multiplex graph networks. *arXiv preprint arXiv:2006.11197*, 2020.

Peng Wang, Qi Wu, Chunhua Shen, Anthony Dick, and Anton Van Den Hengel. Fvqa: Fact-based visual question answering. *IEEE transactions on pattern analysis and machine intelligence*, 40(10):2413–2427, 2017.

Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 409–424. ECVA, 2018.

Taylor Webb, Zachary Dulberg, Steven Frankland, Alexander Petrov, Randall OReilly, and Jonathan Cohen. Learning representations that support extrapolation. In *International Conference on Machine Learning*, pages 10136–10146. PMLR, 2020.

D. Wolpert and W. Macready. No free lunch theorems for search. 1995.

David H Wolpert. The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7):1341–1390, 1996.

D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997. doi: 10.1109/4235.585893.

Qi Wu, Peng Wang, Chunhua Shen, Anthony Dick, and Anton Van Den Hengel. Ask me anything: Free-form visual question answering based on knowledge from external sources. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4622–4630, 2016.

Ying Wu. An introduction to computer vision. 2003.

Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8817–8826. ICCV, 2018.

Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S. Du, Ken ichi Kawarabayashi, and Stefanie Jegelka. What can neural networks reason about? In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=rJxbJeHFPS`.

Tian Ye, Xiaolong Wang, James Davidson, and Abhinav Gupta. Interpretable intuitive physics model. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 87–102, 2018.

Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In *International Conference on Learning Representations*, 2019.

Licheng Yu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, Mohit Bansal, and Tamara L. Berg. Mattnet: Modular attention network for referring expression comprehension. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference 2016*. British Machine Vision Association, 2016.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. From recognition to cognition: Visual commonsense reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Chi Zhang, Feng Gao, Baoxiong Jia, Yixin Zhu, and Song-Chun Zhu. Raven: A dataset for relational and analogical visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5317–5327, 2019a.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. 2016.

Linfeng Zhang, Zhanhong Tan, Jiebo Song, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Scan: A scalable neural networks framework towards compact and efficient models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 4027–4036. Curran Associates, Inc., 2019b. URL `http://papers.nips.cc/paper/8657-scan-a-scalable-neural-networks-framework-towards-compact-and-efficient-models.pdf`.

Kecheng Zheng, Zheng-Jun Zha, and Wei Wei. Abstract reasoning with distracting features. *Advances in Neural Information Processing Systems*, 32:5842–5853, 2019.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

Luowei Zhou, Chenliang Xu, and Jason J Corso. Towards automatic learning of procedures from web instructional videos. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. Bert loses patience: Fast and robust inference with early exit. *arXiv preprint arXiv:2006.04152*, 2020.