THESIS

DEMONSTRATING THAT DATASET DOMAINS ARE LARGELY LINEARLY SEPARABLE IN THE FEATURE SPACE OF COMMON CNNS

Submitted by

Matthew R. Dragan

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Fall 2020

Master's Committee:

Advisor: J. Ross Beveridge

Francisco Ortega
Chris Peterson

ABSTRACT

DEMONSTRATING THAT DATASET DOMAINS ARE LARGELY LINEARLY SEPARABLE IN THE FEATURE SPACE OF COMMON CNNS

Deep convolutional neural networks (DCNNs) have achieved state of the art performance on a variety of tasks. These high-performing networks require large and diverse training datasets to facilitate generalization when extracting high-level features from low-level data. However, even with the availability of these diverse datasets, DCNNs are not prepared to handle all the data that could be thrown at them. One major challenges DCNNs face is the notion of forced choice. For example, a network trained for image classification is configured to choose from a predefined set of labels with the expectation that any new input image will contain an instance of one of the known objects. Given this expectation it is generally assumed that the network is trained for a particular domain, where domain is defined by the set of known object classes as well as more implicit assumptions that go along with any data collection. For example, some implicit characteristics of the ImageNet dataset domain are that most images are taken outdoors and the object of interest is roughly in the center of the frame. Thus the domain of the network is defined by the training data that is chosen. Which leads to the following key questions: Does a network know the domain it was trained for? and Can a network easily distinguish between in-domain and out-of-domain images? In this thesis it will be shown that for several widely used public datasets and commonly used neural networks, the answer to both questions is yes. The presence of a simple method of differentiating between in-domain and out-of-domain cases has significant implications for work on domain adaptation, transfer learning, and model generalization.

ACKNOWLEDGEMENTS

I would like to thank Professor Ross Beveridge and Professor Bruce Draper for advising me throughout the completion of this project as well as bringing me on as a member of the Computer Vision research group. I would like to thank my committee members Professor Francisco Ortega and Professor Christopher Peterson. I would like to give an additional thanks to Professor Ortega for his support and involvement in the Computer Vision Group. I would also like to thank the Computer Science Department at Colorado State University. I am extremely grateful for everything I was able to learn from the Faculty and Staff during my time at CSU. I would like to thank the members of the Computer Vision Group David White, Jane Wang, Dhruva Patil, Rahul Bangar, Joe Strout, Ben Sattelberg, and Adam Williams for creating a fun work environment and teaching me many things along the way. I would like to thank Elliott Forney for developing this LaTeXtemplate. Finally, I would like to thank my parents Dan and Phyllis and my sister Ellen for being extremely supportive and loving throughout my educational journey.

DEDICATION

I would like to dedicate this thesis to my parents Phyllis and Dan and to my sister Ellen

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

The prevalence and performance of modern computing devices has led to attempts to find computational solutions to many different problems. Many of these problems fall into the category of Artificial Intelligence (AI). At a very basic level an AI solution to a problem tries to make intelligent decisions based on the given state of the perceived *world*. Machine learning, which is an area of AI, is the process of using a collection of data to define a model for making predictions on newly collected data samples. For example, given a dataset of a group of peoples' age, gender, and height it is possible to define a machine learning model that tries to predict the height of an individual based on their age and gender. The process of developing a good machine learning model involves a sequence of steps call training and evaluation. The training part of the process is where the model is created using a subset of the given data, often called the training set. In the evaluation step the quality or performance of the model is evaluated using another subset of the data which is entirely disjoint from the training dataset. This is often called the test set. The test set is extremely important because it allows evaluation on data the model has not seen before. There can be variations depending on the choice of machine learning model, but the training and evaluation process is standard for models.

A subset of problems where machine learning models are applied are classification problems. A classification machine learning model tries to place an data sample into a category. For example, a common classification problem is image classification in which a model tries to place an image into a category based on the object present in that image. The categories are defined based on the categories that are represented in the training set. The output of classification models is discrete. This means that after the model has been trained any new input must be placed in one of the classes defined in the training set. In a sense the model is forced to choose a class. Ideally, the test set and any future data should come from the same classes that the training set represents. However, in practice this is difficult to ensure because the world is not discrete.

Once a classification model has been built and tested then it is ready to be applied to any newly collected data. However, it is not easy to guarantee that this data comes from one of the known classes without some human intervention. Someone could verify that the sample does come from one of the specific classes, but that would defeat the purpose of the classifier. One of the most challenging aspects of developing a machine learning model for solving a real-world problem is the existence of *unknowns*. Even some of the most powerful and influential classification neural networks such as Inception V4 [1] and ResNet V2 152 [2] have no way of dealing with unknowns. There have been many attempts to improve neural networks to deal with the unknown problem. Some methods attempt to measure and improve a network's confidence in a classification [3–7], others have focused on trying to identify out-of-domain images by analyzing the *distance* from known samples [8–10]. Other methods attempt to define one-class models in which a decision boundary is developed that classifies a data sample as an outlier or not an outlier [11, 12]. There has been some success in identifying unknowns through analyzing the intermediate values of neural networks, but the problem remains challenging because there are many variables such as the dataset, the network, what is considered to be an outlier. This work will try to understand some of this underlying behavior when several out-of-domain datasets are compared to the ImageNet 2012 dataset [13].

In order to further understand what an outlier *is* in terms of a neural network two popular neural nets trained on the ImageNet dataset are applied to image datasets that differ from ImageNet by varying degrees. the four datasets used for these experiments are depth [14] which are a collection of hand images taken by a depth sensor and provide examples far from ImageNet samples, anime [15] which consist of anime style drawings and take a step closer towards ImageNet data by having color and greater variation than the depth images have. The art dataset [16] consists of several famous artworks. This dataset presents a more challenging problem than the anime because there is more variation and some of the more realistic paintings will share characteristics with ImageNet. The last dataset is the SUN dataset [17], which is the most similar to ImageNet while

still being different enough to be considered out-of-domain. This dataset is made up of scenes. Examples of each can be seen in figure 1.1.

The basic question this thesis will attempt to answer is: is the there something about the behavior of model activations in a well-trained modern CNN that reveals when an input is out of the domain for which the network was trained? The identification of such a behavior could have valuable implications when applying a pre-trained network to a new domain. In fact, as will be shown in this thesis, there are some properties in the activations of the **feature vector** (the pre-softmax activations) of out-of-domain features which makes them easily distinguishable from in-domain features.

1.1 An Introduction to Neural Networks

This section is intended to introduce neural networks. The first part introduces neural networks and the basics of how they work. Then, the following subsection attempts to describe convolutional neural networks in an intuitive way. The subsection briefly covers the history of neural networks.

1.1.1 The Modern Neural Network

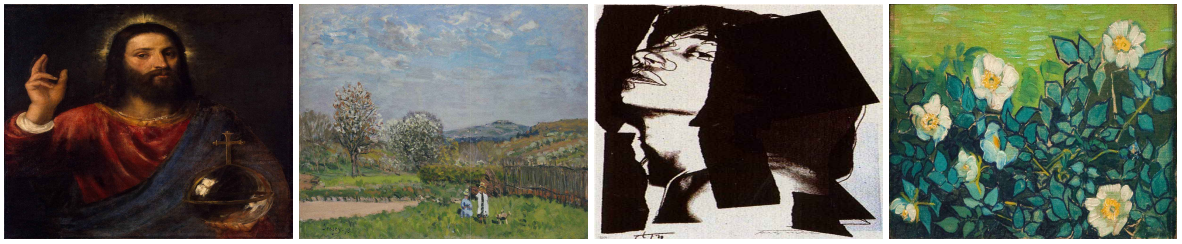
Deep neural networks have become a popular choice for solving modern artificial intelligence and machine learning problems. The popularity of neural networks in modern times is due to their ability to generalize to unseen data and to extract high-level features from low-level data. These characteristics, if harnessed properly, can lead to the development of powerful models. These models are loosely based on the complex array of neurons that exist in the brain and allow animals and humans to learn, hence the name: *Artificial Neural Networks*.

Artificial neural networks consist of units, commonly called neurons or hidden units. These neurons consist of a set of weights and a non-linear function. Figure 1.2 shows a visualization of a single unit in a standard neural network. These units take in a vector of inputs $\mathbf{x}$ of length n. These inputs could be values directly from the dataset or the outputs of other neurons. The input vector will be multiplied (using the dot product) by a weight vector $\mathbf{w}$, also of length n. The

(a) ImageNet

(b) Depth

(c) Anime



(d) Art

(e) SUN

Figure 1.1: Examples from each of the datasets are shown. a) examples from the ImageNet dataset. Images from this dataset consist of natural photographs coming from a wide variety of different classes. b) examples from the Depth dataset. These images consist of hand poses captured by a depth sensor. c) examples from the Anime dataset. These images consist of a wide variety of anime style drawings. d) examples from the Art dataset. These images consist of famous artworks from several artists. e) examples from the SUN dataset. These images consist of natural photographs of scenes.

weight vector consists of weights learned during the training process. In addition to the weights for each input value a bias term b is often included. For this reason, a 1 is tacked on to the input vector to account for the bias term. The computation of $\mathbf{w}^\intercal\mathbf{x}$ produces a single value which will be passed through a non-linear function f to produce the final output y. This non-linear function is often called the *activation function*. By applying a non-linear function it becomes possible for the network to learn non-linearities in the underlying data distribution. These functions are often thought of as on/off switches; the input to the neuron determines if it will fire or not. Depending on the network architecture the unit output y could be passed to another unit or potentially be the output of the network.

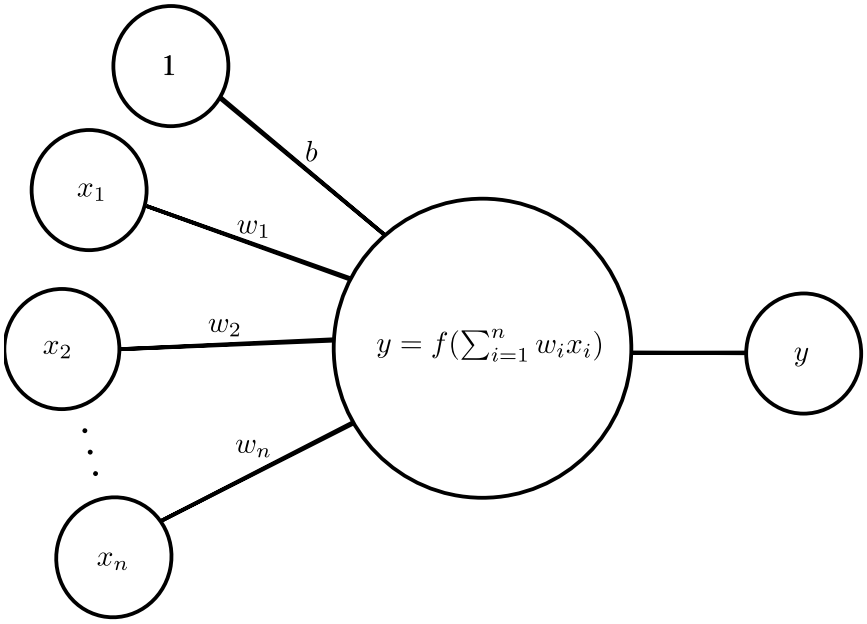


Figure 1.2: An example of a single neural unit in an artificial neural network. Each unit has an input vector $\mathbf{x} = <1, x_1, x_2, ..., x_n>$, where $x_1, ..., x_n$ comes from an initial data sample or the outputs from other units and the 1 is included to account for the bias term b. Each input x_i is multiplied by its respective weight w_i. The weights form a vector $\mathbf{w} = <b, w_1, w_2, ..., w_n>$ which consists of learned weights through the training process. Hence, we are doing the dot product of $\mathbf{w}^\intercal\mathbf{x}$. To produce the output y the result of the dot product is passed through a non-linear function f.

As stated earlier, an activation function roughly plays the role of an on/off switch of whether or not a particular neuron fires based on the input. This idea is based on how biological neurons fire based on the inputs from other neurons. These impulses are then propagated to other neurons.

Hodgkin and Huxley [18] developed a mathematical model to describe the firing of neurons and how the electrical signals move from neuron to neuron. Given that the activation function is supposed to dictate whether the neuron is firing or not, the most basic formulation of such a function is a step function (either 0 or 1). However, modern neural networks employ gradient based optimization. The standard step function is not differentiable at zero, and otherwise the derivative is zero, so this function does not work with gradient based optimization [19]. The sigmoid and hyperbolic tangent functions are popular choices for activation functions because they have a similar shape to the step function while also being continuously differentiable. Other formulations, such as the Rectified Linear Unit (ReLU) $f(x) = \max(0, x)$, have a non-linearity at zero which allows for decision making (the derivative at zero is approximated to allow for gradient optimization) and characterizes the biological neurons in that they cannot produce a value below zero. The use of sigmoidal activation functions in a standard feed-forward neural network can approximate any function arbitrarily well using only one hidden layer [20].

The most common and standard form of neural network is the feed-forward neural network. A visualization of a standard feed-forward neural network can be seen in Figure 1.3. This type of network consists of an input layer with $\mathbf{d}$ input units, l layers of artificial neurons, often called hidden layers, and an output layer. The input layer is were the set of input features are passed into the network. There could be any number of features passed into the network and the features can be categorical or continuous variables. For example, a model that is trying to predict the current price of a vehicle might take into account the age of the vehicle, the make, the model, the number of miles, etc. Each input feature is passed into every unit of the first hidden layer. Then the output of each unit of the first hidden layer is passed to each unit in the second hidden layer and so on. There are a finite number of hidden layers and each hidden layer has a finite number of hidden units. The number of neurons can vary between layers. The final hidden layer passes its outputs into the output layer. The number of units in the output layer depend on the number of dependent variables the model is trying to predict. For example, the model that was trying to predict the current price of the vehicle would have only one output unit, but if the model was trying to predict

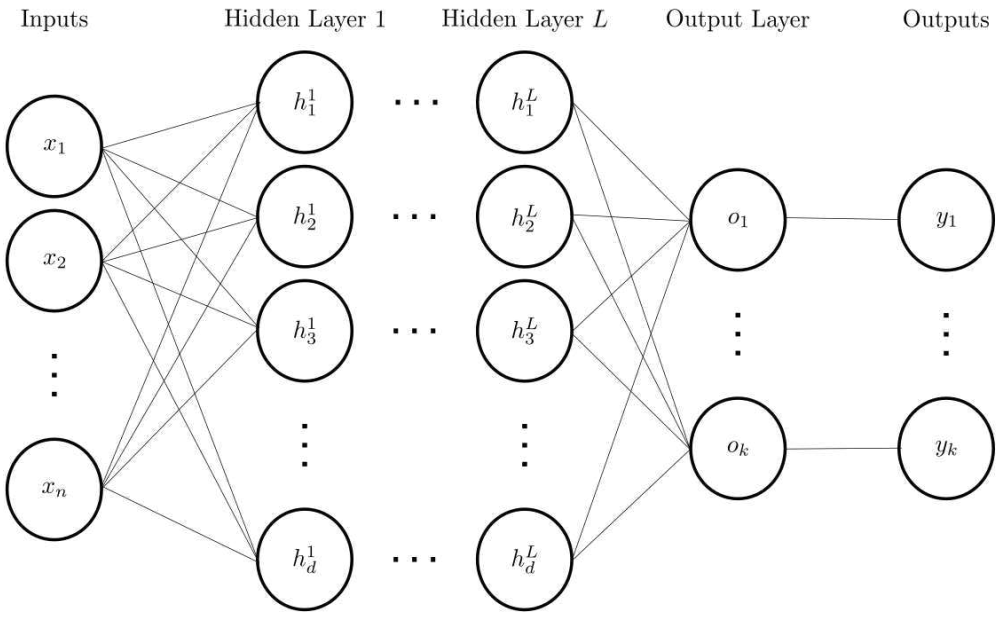


Figure 1.3: An Example of a standard feed-forward neural network. In a standard feed-forward neural network there is an input vector $\mathbf{x} = \{x_1, ..., x_n | n \geq 1\}$ which is fed into the first layer of hidden units $\{h_1^1, ..., h_{d_l}^1\}$. The output of each hidden unit h_i^{l-1} is fed into each hidden unit of the following layer h_i^l. A network has $L \geq 0$ (technically there could be no hidden layers and just a single output layer) hidden layers each having d_l units where $l \in \{1, ..., L\}$. The final hidden layer feeds into the output layer $\{o_1, ..., o_k\}$ which consists of k units where k corresponds with the number of classes in a classification model, or the number of continuous functions that are being modeled. The outputs of the output layer are the model predictions.

price of the vehicle and the amount of money spent on repairs during its lifetime then there would be two output units. The output units are similar to the hidden units, but they lack an activation function. So a complete network consists of inputs which are multiplied by weights in the hidden units and ultimately transformed into a prediction of the true output value. The use of multiple hidden layers lead to the coining of the term *deep neural network*.

Creating a model using neural networks requires training, validation, and lots and lots of data with input and output values. The process of training involves passing a data sample into the network and getting a prediction out the other end. Then, using some error function, such as mean squared error, the difference between the prediction and the true value is calculated. The goal is to minimize this error. By using partial differential equations and the chain rule for differentiation

it becomes easy to determine how weights in the network should be tweaked to reduce the error. This is called gradient optimization. Each weight is only tweaked a small amount because a large step can lead to chaotic behavior in the learning process. The derivative only gives the slope at a select point, so the smaller the step the more likely the slope is a good indication of where to move on the non-linear error surface. Then a different data sample is passed through the network and, again, the weights are updated slightly. With enough time and data the model will converge on weights that minimize the error between the predicted output and the true output.

The validation portion of developing a model involves passing data that was set aside during training through the model and evaluating performance. This set of data is often call the validation or test set and is mutually exclusive from the training set and is not used for training the model. When applying a neural model to a task, the weights that minimize the error on the validation set are usually used. This is because the model with the best performance on unseen data is desirable. Models can overfit to the training data which leads to good performance on training data, but can reduce performance on general data. Thus, model evaluation with unseen data is very important.

1.1.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a special case of the standard feed-forward neural networks which are most commonly used for tasks such as object recognition and classification in images. The modern CNN was formalized in 1989 when LeCun *et al.* [21] was able to fully automate the training process for recognizing hand-written numbers.

Processing an image is difficult because a significant amount of information lies in the spatial orientation of pixels in an image. There is some information in the pixel values themselves, but the key features exist in a pixel's position relative to other pixels. A standard feed-forward neural network can learn how certain pixels may relate to certain other pixels, but this learned relation is not translation invariant. If these related pixel values are shifted to another area of the image the known relation no longer exists and the network will likely fail. In order for neural networks to

be effective at image processing they need to be able to gather information from the relations of pixels within an image and be able to deal with variations in the positioning of important features.

Additionally, a major challenge for image recognition or classification type tasks is the number of input features. The input features for an image are the pixel values and even a relatively small image of 200×200 (some large scale image recognition systems have larger input image sizes than this) will have at least $40{,}000$ input values, and that is assuming a one channel (e.g. grayscale) image. A three channel (e.g. red, green, and blue color channels) image will have $120{,}000$ total input values. Considering that a single unit in the first layer will need a weight for every single input value, the number of weights required for a standard feed-forward network becomes prohibitive. The number of weights required to make an effective network for image classification, especially on tasks with many classes, would require huge amounts of memory to store all of the weights and would be very computationally expensive to train. Even if there was enough memory and computation power to handle this many weights, changing the location of key features in the image can throw off the network. For example, a network could learn to recognize a tiger in the top left quadrant of the frame, but if the tiger is moved to the bottom right of the frame the model weights that correspond to that quadrant know nothing about defining features of a tiger.

CNNs attempt to overcome these challenges by using a combination of convolutional layers and pooling layers. A visualization of the interworkings can be seen in Figure 1.4. Convolutional layers consist of some number of filters. These filters typically are of size 3×3 or 5×5, but the shapes and sizes can vary. Filters consist of values that are multiplied by pixel values and summed together to produce a single value. Because these filters are applied to sets of pixels the spatial orientation of pixels is maintained. Additionally, each filter is passed over the entire image, so a filter can detect features regardless of the location within an image. The use of these filters, also, helps reduce the computational and memory overhead of weights. A convolutional layer with 128 filters that are 5×5 has $3{,}200$ weights in total, in contrast, a standard feed-forward net with five hidden units in the first layer would require $200{,}000$ weights for a 200×200 pixel image. After a filter passes over the image, the resulting values are stored in a new channel so that the spatial

positioning is maintained. Each filter produces a new channel. The new channels are stacked together and used as input for the next layer. A single convolutional layer has a finite positive number of filters. Typically, a CNN will have one or more convolutional layers prior to a pooling layer.

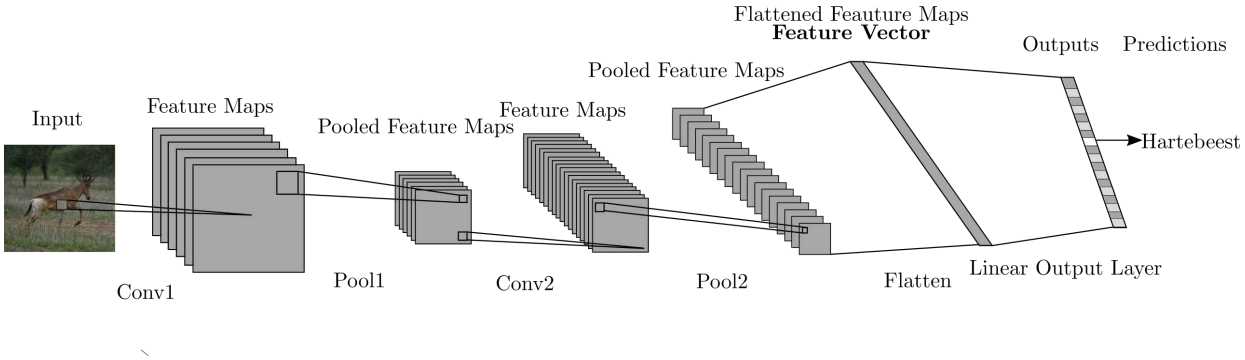


Figure 1.4: An Example of a convolutional neural network. Typically, convolutional networks consist of a pattern of convolutional layers for generating feature maps and pooling layers for subsampling feature maps. Subsampling reduces the overall number of features and helps extract important features from the feature maps. The convolutional and pooling layer pattern can vary significantly depending on the chosen architecture. The convolutional portion of the network is often called the **feature extractor** because it is intended to extract meaningful and high-level features from the image. The output of the feature extractor is flattened into a single vector (called the **feature vector**) which contains values that represent the high-level features from the image. Typically, a single linear output layer is used to predict the class based on the features present in the feature vector. The outputs are normalized using a softmax function and the output with the highest prediction is the predicted class.

A pooling layer in a CNN is responsible for reducing the total number of input features and can help with the network being invariant to the position of the object within the image. As stated prior, using a standard feed-forward neural network for image recognition tasks leads to a huge number of stored weights which often become prohibitive. Pooling layers typically come after one or more convolutional layers and reduce the number of features by applying some sort of calculation to small subsections of each frame. The small subsection could vary, but is usually 2×2 For example, assume that the previous convolutional layer produced 64 frames of 256×256 features. A 2×2 pooling layer with a stride of two (meaning that the pooling area is shifted over by two pixels for each computation) reduce the features to be 64 frames of 128×128 features. The reduction is typically done by computing a max or an average of the pixels within the pooling frame. Max

pooling (when the max is used for the computation) gives the networks some translation invariant feature detection abilities. By taking the max of the pooling area the strongest feature signal is being passed on, and even if there is a small variation in the feature position, it is likely that the feature will still be picked up and passed on.

A standard CNN consists of a repeating pattern of some number of convolutional layers followed by some number of pooling layers. Usually the number of convolutional filters at each convolutional layer increases as the end of the network is approached. The idea is that many small low-level features such as lines and corners will eventually be combined and point to the presence of high-level features such as the pattern of stripes on a tiger. The end of the network usually has a few, and more recently usually only one, fully connected layers. These layers are intended to take the learned features from the convolutional layers and interpret them to determine the presence of a tiger, for example, based upon the features.

1.1.3 A Brief History

The beginning of neural networks started with theories about the brain, learning, how to mathematically characterize neuron interconnections. McCulloch and Pitts [22] were the first to model the interconnection of biological neural nets and how one neuron influences other neurons. Using the fact that a neuron either fires or it does not, they were able to use propositional logic to model the propagation of information through neurons. The concept of learning in terms of neurological interconnections was formalized by Hebb [23]. Hebb hypothesized that the brain learns and changes over time because the frequent stimulation of one neuron by another neuron will lead to the efficiency and stability of that connection to be improved. This theory is called Hebbian Learning. These theories were important in the development of neural networks.

One of the first computational models that influenced the development of the modern neural network was the perceptron, developed by Rosenblatt [24]. The perceptron is a linear binary classifier that is designed to categorize the output based on the input features. It is a learning model that is trained by weight adjustment based on error of training outputs. A single perceptron is limited

in the problems that can be solved, but the computational power increases when layers of perceptrons are stacked. The idea of multi-layer perceptrons was the first complete construction of neural networks and spurred research in backpropagation and building multi-layer architectures [25, 26].

Research into the multi-layer perceptron was significantly slowed after the publication of *Perceptrons: An introduction to computational geometry* by Minsky and Papert [27]. This book highlighted the strengths of a single perceptron and also pointed out the major limitations in terms of the type of problem it could solve. Although, this book only focused on the limitations of a single perceptron it stunted the research in multi-layer perceptrons for much of the 1980's.

There was a resurgence of interest in the multi-layer perceptron in the mid 2000s due to the success of Hinton *et al.* [28] showing that it was possible to create high-level representations from the input data. Along with the increase in the availability of computing power, it became possible to develop neural network models without the requirement of extremely expensive computing resources.

1.2 Neural Networks and Unknown Inputs

Deep neural networks have achieved state of the art performance when applied to a variety of difficult problems such as image classification, object detection, and natural language processing. The strength of neural networks can be attributed to their ability to generalize. The concept of generalization when presented a machine learning problem is somewhat difficult to characterize. Obviously, a good machine learning model should learn to recognize and, to some degree, give meaning to high-level features or characteristics of the data the model was trained on. But what happens if the model is presented with something unusual? If the model truly generalizes, should it be able to characterize features present in these new data? The field of transfer learning has shown that the application of a well-trained model to a new domain can yield favorable results [29]. This, likely, means that the weights can successfully generalize to a new domain and that the learned weights can potentially extract meaningful features from data that are outside of the domain the model was originally trained for.

The definition of unusual is important in the context of neural networks. In this context an unusual data sample is a sample that is outside of the domain of data that the network was trained on. The definition of a *domain* is a little tricky. For example, consider the ImageNet dataset. This is a large dataset consisting of $1{,}000$ unique classes which vary from volcanoes, to fossils, to several breeds of dog, and so on. Simply because a sample does not come from a class seen during training does not necessarily mean that the sample was out-of-domain. An example of a class that is not contained in the dataset is the okapi. An example of an okapi can be seen in the lower left portion of Figure 1.5. The appearance of an okapi is somewhere between a giraffe, a horse, and a zebra. Is an okapi out of the domain of an ImageNet trained network? Probably not. Especially given that zebra is in the dataset there are enough overlapping features with the okapi that it would fall in the domain of ImageNet even though okapi is not a class within ImageNet. However, the features present in an anime style drawing, for example, would very different from features in any of the ImageNet classes and, thus, could be considered out-of-domain. Another, more practical example concerns self-driving cars and the recognition of street signs. A net that has been trained to recognize street signs could possibly be passed the occasional coffee shop sign. The coffee shop sign is likely very different from typical street signs and would have an unusual high-level representation in feature space. This coffee shop sign would be considered out-of-domain. The definition of domain is tricky because it is related greatly to the expressiveness of the features relative to what the model was trained to recognize and what the model is currently being presented. A model specifically trained to recognize street signs probably will not be able to express a coffee shop sign well, but a model trained on ImageNet could likely express an okapi effectively.

It would be of great value if a neural network could recognize if it has been presented with something unusual. For example, again consider a network that has been trained to recognize street signs and is included in the interworkings of a self-driving car. The performance of that model is extremely important to the safety of those who travel in the vehicle. If, for some reason, the model was passed an image of a sign on a storefront, the ideal behavior of the model would be

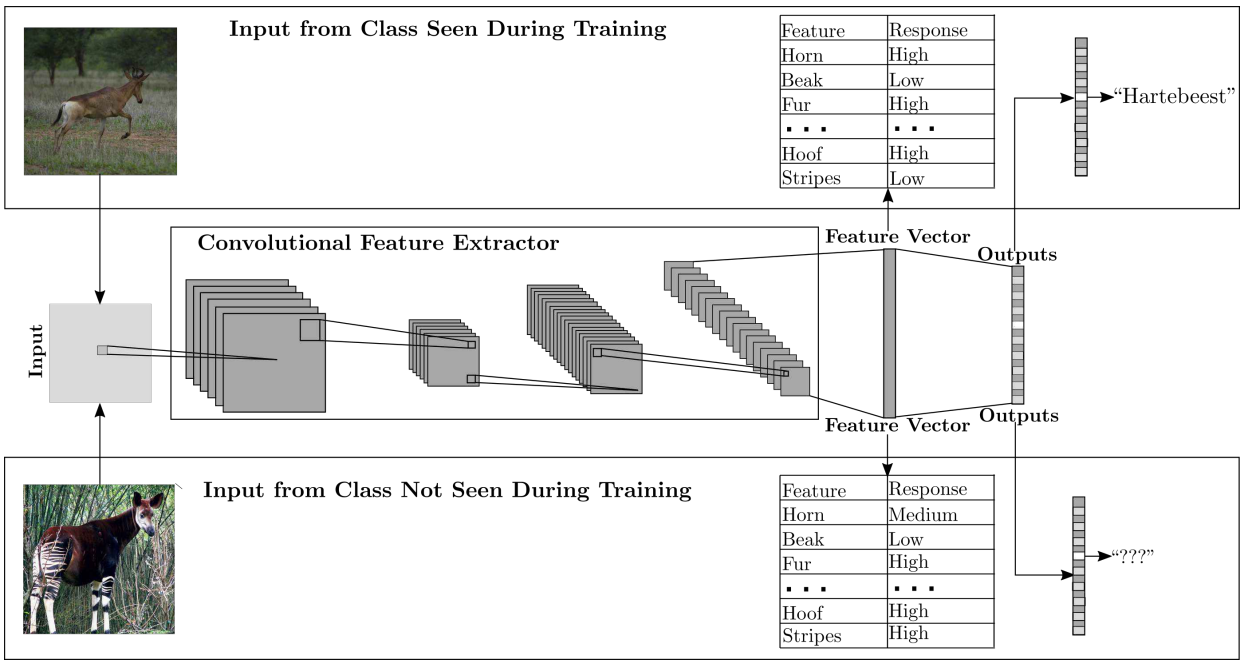


Figure 1.5: A major challenge of modern neural networks is the concept of forced choice. The hartebeest (top) is an image that comes from the ImageNet dataset, so, if the network is trained on the ImageNet dataset, it is expected that the high-level features extracted by the network will contain high responses on features that correspond with the hartebeest and the final linear layer will correctly assign the proper class based on these features. However, the okapi (bottom) is not in ImageNet, so the predicted class will be incorrect and unknown. But what about the extracted features? If the network can extract features from images that are similar to the okapi (such as the hartebeest or the zebra) then it is possible that the extracted feature vector still has meaningful features, but does not know how to interpret them.

to dismiss the image as unimportant. However, modern neural networks do not have the capability of dismissing an input and, thus the storefront would be misclassified as one of the known classes and the self-driving car would respond accordingly. Such a behavior would be unacceptable. In the opposite scenario, the consequences of a street sign being thrown out as unimportant by mistake are extremely dangerous as well. As networks begin to be used in more and more user applications it is imperative that they be able to respond effectively and safely when presented with an unfamiliar input.

The bottom line is that neural networks do not fail gracefully. Although, unfamiliar inputs tend to produce an overall lower confidence when passed through a network it is not uncommon for such an input to produce a high confidence. This makes using the confidence level of a network's prediction as a true measure of *confidence* unreliable, and depending on the consequences of an

incorrect prediction, utterly useless. Several papers have focused in confidence predictions as a method of identifying unusual inputs [3–5]. The aim of this this thesis is to direct this attention to other parts of the network (in this case, one layer back prior to the linear classifier, which is referred to as the **feature vector** or **feature layer**) as a means of recognizing unusual behavior from a particular input. The results shown in this work demonstrate that feature layer outputs encode information which makes recognizing unusual inputs easy and reliable for a set of examples drawn from four common datasets.

In order to justify why there may be some unique and recognizable behavior when a network is presented something novel one might speculate that there may be a general trend followed by the high-level features for unseen data. One might speculate that, during training, a network learns to assign high responses to recognized classes; essentially assigning high value to the features that matter for a given input. With this thought in mind, one would expect that a sample from a known class will show higher responses than one from another domain. However, one might also speculate that the opposite is true, that an unusual image will lead to a higher number features being set to meaningfully active. This idea seems reasonable because the network only knows what it is trained to know. If a network is given something from a domain it does not know then it will try to extract known features from the out-of-domain sample which may have wildly different features leading to higher excitations on average. Overall, the actual difference in excitations of features for an in-domain and an out-of-domain sample is more complicated than what is being speculated on here, but the most important thing is that there has not been much work on the subject and this thesis aims at shedding light in this area.

The results presented in this work are generated using Inception V4 and ResNet V2 152 networks trained on the ImageNet dataset. In a series of experiments the features extracted by these two networks from non-training ImageNet data, depth images of hand pose, anime drawings, famous artworks, and scene images are compared. It is possible to identify, with a reasonable degree of accuracy, which domain a particular image comes from using a very simple classifier. In the first set of experiments ImageNet is compared to each of the out-of-domain datasets. The re-

sults presented show that differentiating between ImageNet features and depth data is a near-trivial problem. Anime and art images show near perfect separability from ImageNet images. The scene images, which are most like ImageNet, are distinguishable over 90% of the time. The second sets of experiments explore the out-of-domain features between the different datasets. Each dataset is nearly perfectly separable from all others in each case showing that the networks at least extract unique features from each out-of-domain dataset. Finally, some preliminary experiments are completed to explore the meaning of this separability. That is, does the separability of these features provide evidence that the feature vector is highly expressive and can be applied to a variety of domains, or does it point to a weakness of the features in that they are separable because the network is unable to accurately express them.

The experiments involve applying a simple two-class Support Vector Machine (SVM) to pairs of datasets to determine how separable they may be. The datasets are split so the SVM is trained on different data than it is evaluated on and the data used in the results was never used in training the SVM. Using this protocol, the results presented in this thesis show that a single two-class SVM solves, to a high degree of accuracy, the in-domain vs. out-of-domain problem for comparisons between datasets. Further, the geometric interpretation of this finding is both compelling and perhaps surprising; there exists a hyperplane in CNN feature space such that in-domain largely falls on one side and out-of-domain on the other.

Chapter 2

Background

This chapter details the related work and necessary background for understanding the motivation behind the work presented in this thesis as well as the approach taken in empirical investigation. Section 2.1 discusses the relationship to transfer learning. Sections 2.2, 2.3 and 2.4 take a deep dive into the literature related to this topic. Much of these sections are based on the literature review done by Roady *et al.* in [30] which provides an excellent review of recent literature. Section 2.5 covers some of the related fields that explore a similar but different topic and are important to mention. Section 2.6 discusses how this work shows that an ImageNet trained feature extractor coupled with a linear SVM may make a decent gating network in a Mixture of Experts model. Section 2.7 discusses how many out-of-domain detectors fair when applied to large datasets. Finally, Section 2.8 discusses some of the background and initial motivations for the bulk of this work.

2.1 Relationship to Transfer Learning

This work is related to transfer learning, but should not be confused with transfer learning. Transfer learning is the process of re-purposing learned weights of a network for application to a new domain or a new application of a previously learned domain. This may involve training a new classifier on the fixed feature extractor or retraining some or all of the convolutional layers for the new purpose. The driving concept behind transfer learning is that a well-trained network has *good* feature representations of the target domain and these representations could be effective when applied to a new domain [31]. The generalization of weights is useful when there is only a small amount of training data available from the new domain [29].

Instead of focusing on the transfer of pre=trained weights to a new application domain, this work focuses on the discrimination between domains based on the high-level features a well-trained network is able to extract. Looking at how easily high-level features of in-domain and out-of-domain features can be discriminated could shed some light on what the network is actually

learning and how well pre-trained weights transfer. In a sense, the goal is not to reapply the weights for a new classification task, rather to analyze the outputs for different datasets and try to understand how the pre-trained weights influence the relative feature mappings.

2.2 Open Space Risk

Open space risk defines the relative risk a model has based on the all the potential data that could be seen at testing time. This concept was defined and formalized by Scheirer, de Rezende Rocha, Sapkota, and Boult [32]. Three key parts of open space risk are the *full space*, *open space*, and the *closed space*. The full space consists of the entire set of samples that a model could encounter. For most applications having a model that can handle all possible samples is impossible, so models are trained on a dataset that encompasses some portion of the full space that still allows it to perform the intended task. This subset of the full space that is used for training defines the closed space which is made up of the area that is near all known data samples. The open space is made up of all possible samples the model could encounter that do not fall in the closed space. Essentially open space is made up of all samples that fall far from known data. Thus open space risk is the ratio of the open space to the full space. Conceptually, open space risk attempts to define the space where a model can be confident in a classification based on where a data sample falls in feature space.

A closed model faces the issue of force choice discussed in Chapter 1. As a very basic example, consider a linear classifier that separates images of dogs and cats. A linear classifier will divide the input space into two half spaces. Regardless of how good the model is, each side of the classifier has a large amount of space that is far away from any of the known training examples. So if an image of a rabbit were passed into the model then it will be either classified as a cat or a dog even if the data sample falls far away from any of the know data samples. Open space risk defines the proportion of the total space that is far enough from the known examples to raise concern about the validity of the input.

Many methods of identifying out-of-domain samples attempt to reduce open space risk by including an unknown or background class. Including an unknown class reduces the open space because it allows for possible classification of a point that falls far from known data samples, reducing the open space risk.

2.3 Out-of-Domain Detection by Analyzing Network Outputs

A common approach to out-of-domain detection is to analyze intermediate features or final outputs (see Figure 1.4 for reference) extracted by a pre-trained neural network. One of the major benefits of this approach is it typically does not require retraining of an existing network to incorporate an out-of-domain detector. That way these methods can easily be applied to existing models without significant architecture change. The underlying concept is that, although an outlier image on its own may be difficult to detect, the unusual behavior will likely surface in the high-level features extracted by the network.

There are several different ways that intermediate features have been used to detect unusual inputs. Thresholding methods, covered in Subsection 2.3.1, look to detect out-of-domain inputs by thresholding on the output confidences of the network [3–7]. Many of these methods try to induce higher confidences for in-domain images. Distance Metrics, covered in Subsection 2.3.2, attempt to define some sort of locale for in-domain inputs and can throw out out-of-domain inputs if they fall outside of this locale [9, 10]. One-class methods (Subsection 2.3.3) attempt to create a decision boundary that separates the in-domain inputs from the out-of-domain inputs [12, 33]. Finally, Subsection 2.3.4 covers the use of extreme value theory to detect out-of-domain images. This approach focuses on the distribution of the extreme values (values at the tail ends of a distribution) of in-domain examples and attempts to make the distinction between in-domain data that falls in the extremes of the distribution and out-of-domain inputs [7].

2.3.1 Thresholding Methods

The most basic method of determining if a sample is outside of the original domain is to threshold on the network outputs. Essentially, if the network is not confident in a prediction then the input is likely from a class that has not been seen before. Hendrycks and Gimpel [3] created a baseline for out-of-domain detection by thresholding on the softmax class scores of the network. The driving concept behind this idea is that an in-domain sample should produce high confidence in the predicted class while the predicted confidences of an out-of-domain sample should be roughly uniform. Thus, by defining a threshold it should be possible to throw out out-of-domain inputs.

In-domain images do tend to produce higher confidences for the predicted class compared to out-of-domain images, but, in many cases, the expected uniform distribution for out-of-domain images is not observed. For example, fooling images, as described in [34], are images that are designed to produce high confidence in network classification. To a human observer, these images look nothing like the images (in this case images from the ImageNet dataset) used in training, but they receive high confidence in network prediction. Additionally, Boult and Bendale [7] showed that slight, nonrandom perturbations to images can totally throw off a network's prediction. This shows that an unfamiliar class has the potential to produce unusual responses when presented to a network.

There have been several methods that have been built off the baseline set by Hendrycks and Gimpel [3]. One example is the Out-of-Distribution detector for Neural networks (ODIN) proposed in [5]. This method applies small perturbations to input images which tends to increase the gap between in-domain and out-of-domain prediction confidences. Additionally, temperature scaling is applied the the network outputs to make thresholding on the softmax scores even more effective. Another approach using thresholding, described in [4], attempts to push out-of-domain confidences to the uniform distribution through additional network training. This method involves training with out-of-domain images and using Kullback-Leibler (KL) divergence to minimize the difference between the distribution of confidences of an out-of-domain image and the uniform distribution. The authors recognize that for most problems there could exist an infinite number of

out-of-domain samples, so they us a Generative Adversarial Network (GAN) for out-of-domain example generation. Ideally, the GAN can learn to generate out-of-domain samples that represent the general out-of-domain data distribution. The proposed GAN is designed to generate samples similar to in-distribution images that come from low density areas of the distribution. This forces the network to be more confident in its prediction in these areas.

Another threshold centered approach, called Deep Open Classification (DOC), proposed in [6], attempts to recognize unknowns in the context of text documents. This method augments the confidences of the network to allow for more effective thresholding. This work is similar to and improves upon the work of Boult and Bendale [7], which will be discussed in a later section. This method applies a one-vs-rest sigmoid classifier instead of the standard softmax. Then by defining a per-class threshold the network can throw out an unusual input if it falls beneath the threshold for all classes. The benefit of this method is it will reduce the open space risk, because it allows for samples that fall far from other known examples to be labeled as unknowns.

2.3.2 Distance Metrics

Another approach to detecting unusual input is to use distance metrics. Distance metric based methods define a locale of in-domain data and if the metric shows that an input is outside of this locale then it can be deemed out-of-domain. Using distance metrics for detecting outliers has been an approach for several years, but Knorr and Ng [35] were the first to formulate methods that showed success on large datasets with more than two dimensions. Since the initial formulation by Knorr and Ng there have been several methods for detecting outliers. Some of these methods include projecting the data to multiple lower dimensional projections and detecting data that falls in a relatively low density area of data in one or more projections as an outlier [36], finding outliers by weighting each point by the sum of its distance from its *k* nearest neighbors and marking the largest weights as outliers [37], efficiently removing top n outliers using randomization and pruning with *k* nearest neighbors [38], and using Voronoi diagrams to improve efficiency of compute *k* nearest

neighbors and marking outliers as points with largest average distance from neighbors [39]. These methods are standalone detectors and do not rely on neural network outputs.

The intuitive approach to distance metrics is to use the Euclidean distance, as it is the true distance of points relative to each other. However, Roady *et al.* [30] notes that these methods do not tend to perform well when data is high dimensional. Another distance approach is to consider probabilistically how far a point p falls from a distribution D, called the Mahalanobis distance. Murphy [8] showed that the use of Mahalanobis distance can be used for detecting outlier features in a convolutional neural network because a network trained with cross-entropy loss approximates a Gaussian discriminant analysis classifier. This means that Mahalanobis distance metrics can be applied to to deep network features to apply class-conditional distance scores to outliers.

Given that Mahalanobis distance metrics can be used for assigning a class-conditional outlier score to deep features Lee *et al.* [9] developed a Mahalanobis distance based metric for identifying when a sample is too far from known samples to be considered in-domain. This method uses the feature outputs from several layers of a CNN to build a class-conditional metric. The metric is computed for each class then combined to create an acceptance score using a linear classifier. A sample is either thrown out as an unknown or the classification is accepted based on the acceptance score.

In an alternative approach, Shalev, Adi, and Keshet [10] developed a distance based method for detecting out-of-domain images that uses word embeddings. Prior to training the model, K word embedding methods are chosen. Each class label is passed through each of the word embedders to produce K embeddings for each class. The model consists of some shared layers which then feed into layers unique to each word embedding. The unique layers of the model try to predict the word embedding of the appropriate output class based on the input data. The class that has the minimum cosine distance between it's corresponding word embeddings and the predicted word embeddings is chosen as the predicted class. Out-of-domain images are detected by thresholding on the sum of squared L_2 norms. When the sum of the squared length of word embeddings is too low an input is labeled out-of-domain.

2.3.3 One-class Methods

Another technique is to define a one-class network. Essentially, defining a model that determines if an input is part of a single class or not. For example, an SVM could be used to define a decision boundary around some portion of the training data. The idea is the SVM would define a space in which all data that fall into that space are classified as in-domain and all other data is considered an outlier. This idea was formalized in [33]. Just like the standard SVM the decision boundary is the maximum margin decision boundary. In this case the decision boundary is set so that it envelops some proportion of the training data. The proportion is set by a hyper-parameter that can be correctly set using cross-validation. Ideally, an out-of-domain sample will be enough of an outlier that it will fall on the other side of the decision boundary. However, this method of outlier detection does not work well when applied to high-dimensional space [11]. This is due to the presence of many irrelevant features hiding anomalies, any sample has the possibility of being considered an anomaly given a selection of a specific subset of features, and finally, in the very high dimensional space, the number of feature subspaces to explore becomes prohibitive. Using feature extractors helps with dimensionality reduction, but feature spaces are still large causing important features to be difficult to distinguish from unimportant ones.

By combining the one-class SVM with a method of dimensionality reduction it is possible to reduce the challenge of detecting outliers in high-dimensions. Erfani, Rajasegarar, Karunasekera, and Leckie [11] developed a method of outlier detection that pairs deep belief networks (DBNs) with one-class SVMs. DBNs are a dimensionality reduction method similar to the first half of an autoencoder. An autoencoder maps data down to a low-dimensional representation then upsamples to try to reproduce the original data. If the autoencoder works well, the encoded features should be a reasonably good low-dimensional representation of the high-dimensional data, otherwise, it would not be able to accurately reproduce the original data. Following a similar idea DBNs produce a low-dimensional representation of original data that should preserve important features and reduce the effect of other noisy features. In this hybrid approach, a Restricted Boltzmann Machine is used to train the DBN in a greedy layerwise fashion. Once the dimensionality of the

data has been reduced, a one-class SVM algorithm is applied to the low-dimensional features to separate the in-domain data from the outliers.

Instead of trying to reduce the dimensionality to only meaningful features, Perera and Patel [12] focused on trying to force the feature extractors to learn meaningful features. The authors wanted to push the features to be more compact, meaning that different images from the same class cluster together in feature space, and more descriptive, meaning each class will have a distinct feature representation from other classes. Two loss functions, one to improve the compactness of features and the other used to improve the descriptiveness of features, were proposed and used to fine-tune an existing feature extractor. Then, using the feature extractor, a small cluster of datapoints in feature space are generated for each class. Finally, by applying some kind of classifier to these clusters it is possible to recognize if an input comes from a specific class or not.

These methods tend to focus on recognizing if an input is part of a single class or part of another class, and not necessarily detecting if something appears to be from an entirely different distribution of data. It is an interesting problem to recognize a single class apart from all other classes, but what happens in the case were a network is trained to recognize a set of classes but may be fed input that does not fall into those classes? This thesis attempts to define a network domain as a single class and checks if there is some consistent pattern in that network domain that allows it to be distinguished from other domains.

One-class approach to out-of-domain detection is the closest approach to the work presented in this thesis. However, there are a few key distinctions. For one, the goal of this thesis is not to define the best one-class classifier for classifying in- and out-of-domain images, but it is to describe an observed behavior. Employing a more sophisticated classifier than a linear SVM could lead to results more comparable to other research related to the one-class method of outlier detection. Here, demonstrating the existence of a separating hyperplane between domains in a high-level feature space is the desired result - not to produce the best decision surface possible. However, it is possible that the presence of a separating hyperplane may have significant implications for one-class methods of outlier detection.

2.3.4 Extreme Value Theory

When considering the high-level features of in- and out-of-domain data, it seems reasonable to expect that outliers would lie at the extremes of the in-domain data. Essentially, if it is possible to separate the in-domain extremes from the out-of-domain images, then it becomes possible to recognize outliers. This concept, called extreme value theory (EVT), is that there would be trends in the tail ends of the data, and thus, an out-of-domain example would look different than the extreme samples of in-domain data. For example, Bendale and Boult [7] used class-conditional Weibull distributions to model the softmax scores of the extremes of each class. In this work they present OpenMax, an alternative to softmax, which will augment the distribution of weights in the pre-softmax layer to improve predictive accuracy and to recognize unseen classes as unknowns. OpenMax is a distance normalization technique that will augment the highest predicted classes in the pre-softmax layer based on the probability that an observation comes from that specific class. An input will be rejected if none of the confidence levels are greater than a pre-specified threshold or the highest predicted class is the *unknown* class. The unknown class confidence level is defined by the sum of each classes predicted confidence multiplied by the probability of not belonging to that class. The authors showed that this method could handle adversarial attacks, unknown classes, and fooling images (an image that is visually meaningless to the human eye but produces high confidence in classification). Bendale and Boult's work in [7] inspired this work and inspired the use of the Weibull distribution for determining when an input is likely not from a known class. However, the use of the Weibull distribution proved to be unnecessary for the purpose of this work. A key difference in this work is it looks at the outputs of the layer prior to the linear transformation rather than the the pre-softmax activations. Presumably, the pre-softmax activations are just unnormalized class predictions, but by looking at the pre-softmax feature vector, it is possible to analyze the high-level features prior to these features being converted into class predictions.

2.4 Training Networks to Recognize Outliers

Another approach to outlier detection is to incorporate detection into model development process. Instead of trying to gather meaningful information about what exactly *is* an outlier from the high-level features of a pre-trained network another common approach is to restructure the network architecture or change the training process to enable better out-of-domain detection. The caveat is, of course, that these methods require training or retraining the network. Training a network is costly, it requires a lot of time, data, and tuning, but there are potential benefits to building the detection into the training of the network. Neural networks are supposed to be able to learn complex relationships between data, so by integrating out-of-domain detection, it is possible that a distinction between in- and out-of-domain data could be learned even if the difference is complex. Subsection 2.4.1 looks into one-vs-rest classifiers as a means of out-of-domain detection. One-vs-rest classifiers allow for an input to not be classified in any class and is therefore considered out-of-domain (e.g. [6]). Methods of training networks to ignore background classes in classification are covered in Subsection 2.4.2. These methods are similar to object detection methods [40, 41] but focus more on identifying if an input comes from a background class rather than selecting regions of interest [4, 42]. Finally, Subsection 2.4.3 covers the use of generative models to learn data distributions for use in detecting abnormalities. By learning the distribution of in-domain or out-of-domain data it is possible to detect when an input is or is not from a particular domain [4, 43].

2.4.1 One-vs-Rest Classifiers

One-vs-rest classifiers try to handle the open set problem by defining decision boundaries that encompass each class. Essentially, each per-class decision boundary is a one-vs-rest classifier that determines what is part of that class and what is not part of that class. Out-of-domain examples can be detected when an input is in the *not part of this class* area for all known classes. Typically, cross-entropy loss paired with softmax is chosen over a one-vs-rest classifier for training a classification neural network. However, cross-entropy loss and softmax only work well for closed-set

problems, essentially forcing an incorrect prediction (with the possibility of a high confidence) if an out-of-domain input is passed into the model [30]. In many real-world applications, it is unrealistic to expect that all incoming data comes from the same closed set that the model was trained for. Additionally, the normalization of the softmax function can lead to high confidences from adversarial and fooling images [7]. The benefit of using a one-vs-rest classifier is, if carefully formulated, it is possible to detect out-of-distribution images that fall into the *rest* category for all of the one-vs-rest classifiers.

The DOC method of outlier detection, initially discussed in Subsection 2.3.1, is an example of a one-vs-rest classifier. The DOC method, presented in [6], replaces the softmax layer with a series of sigmoid functions for each class. Thus creating a one-vs-rest classifier for each class. Each sigmoid function, paired with a specified threshold, determines if an input is from that class or is not. If an input is not predicted as part of any class then it is considered an outlier.

2.4.2 Training Nets to Ignore Background Features

When Training a network on a recognition task that may contain samples from a *background class* (a class that is not part of the initial training domain) it would be useful if the network could learn to devalue features that seem to come from a background class. In essence, training a network to recognize what it does not know is a reasonable approach to handling out-of-domain inputs.

Background class recognition is a major challenge in the field of object detection. Object detection models attempt to extract meaningful (foreground) regions of the image for classification meanwhile ignoring features that are part of the background [40, 41]. These methods work by using a well-trained network (such as ResNet trained on ImageNet) as a feature extractor. Then a Region Proposal Network will evaluate and rate different regions of the extracted features as either background or foreground. Finally, The classifier will classify the regions labeled as foreground. Although the focus is not to explicitly recognize when an input is out-of-domain, there is an implicit understanding of what is and is not a background class. These models show that features

from intermediate neural network layers can hold meaningful information about the relevance of the input.

Methods more directly applied to the task of out-of-domain detection will train the network to not learn meaningful information from out-of-domain examples. Hendrycks, Mazeika, and Dietterich [42] trained a neural network with an additional regularization term in the loss function intended to force the distribution of feature excitations of out-of-domain images to the uniform distribution, making them easier to threshold against. This method is called Outlier Exposure, and, although it requires examples of outliers at training time, the authors show that the method does tend to generalize when unseen images are tested.

One of the key challenges with background class recognition is defining an effective loss function. A method previously discussed in Section 2.3.1, which also applies here, developed a new loss function called *confidence loss* which tries to push network outputs for out-of-domain samples to the uniform distribution and uses a GAN to generate outlier examples [4]. Another loss function called *Objectosphere* loss, tries to make the feature representations of background class samples' Euclidean length shorter [44]. This makes the softmax scores of out-of-domain images more entropic making outlier detection easier.

Even though the regularization of background class features has had some success in making input samples from out-of-domain classes more distinguishable, the vast and broad nature of many modern datasets, such as ImageNet, make defining a true background class difficult [30].

2.4.3 Generative Models

A challenging aspect of detecting out-of-domain images is that it is hard to know the underlying distribution of the in-domain data. Thus, when an out-of-domain image is presented it can be hard to tell if it comes from the in-domain distribution or not. However, there has been a lot of work in the modeling of data distributions in the form of generative adversarial networks [45], variational autoencoders [46], auto-regressive models [47], and invertible latent variable models [48]. By modeling an underlying distribution it becomes possible to test weather or not an input comes

from the underlying distribution. There are two possible methods to derive from this approach: the first is to model the underlying distribution of the in-domain data and the second is to model the distribution of the out-of-domain data.

An example of modeling the underlying distribution of the in-domain data is described by Oza and Patel [43]. This approach involves using three different networks, a feature extractor for gathering key features from the image, a classifier for classifying objects obtained from the feature extractors, and finally a decoder which attempts to reconstruct the input image from the outputs of the feature extractor. The decoder is trained using a reconstruction error. The results of their experiments showed that out-of-domain images tended to have a higher reconstruction error than in-domain images making it possible to distinguish between them. To further improve the accuracy, the authors used EVT to model the tail-ends of the distribution of in-domain reconstruction error to probabilistically determine if an input is in- or out-of-domain. In a counter example, Nalisnick *et al.* [49] argues that these methods may not work as well as expected because little is known about the distribution of background classes. They found that out-of-domain images tend to have higher likelihoods than the in-domain examples. This means that the parameters of the model tend to be a better fit for the out-of-domain images.

The alternate case, where the goal is to model what is considered to be an out-of-domain image has been explored as well. For example, Lee *et al.* [4] used generative adversarial networks to produce general out-of-domain images for background class regularization. Neal *et al.* [50] used generative adversarial networks to generate images similar to the training class and used them for training an additional unknown class in the classifier head. Finally, Yu, Qu, Li, and Guo *et al.* [51] used adversarial sample generation (ASG) to generate out-of-domain samples that are close to the boundary of an in-domain class, but are still distinguishable from the in-domain class. The generated examples are used for training examples in the "unknown" class. Generated samples that are indistinguishable from the known class may be used to pad the number of examples in the known class.

2.5 What this Work is Not

There are a few points to make about what this work is not. First of all, the focus of this work is not to recognize adversarial samples. Adversarial samples are samples designed to make the network fail while retaining characteristics and features of the data [7]. In the case of image recognition, an adversarial image is generated by applying small perturbations to the pixels that almost go unnoticed to the human eye, but result in high confidence in the incorrect class. Adversarial example detection is a related field, and there are some researchers looking at deep features as a means of detection [52]. The method presented in this work is not designed to recognized such minute differences in features, as the focus is on the distribution of feature vectors as a whole across classes. Because the focus of this work is on the feature vector as a whole, it is not unlikely that an adversarial image's feature vector may be consistent with the overall distribution of feature vectors. The aim is not to detect foul players, but simply to reveal a surprising and simple property of feature vectors from common networks.

Also, This work is not an attempt to construct the best possible out-of-distribution classifier. As stated in the previous paragraph the aim is to describe an observed behavior. Although, the methods presented here do tend to differentiate between in-domain and out-of-domain samples with a high success rate, there is an underlying issue with this setup for detecting all out-of-domain images. The issue being that the number of possible out-of-domain samples is infinite and it is unlikely that a simple linear SVM separator between an in-domain dataset and a specific out-of-domain dataset would be reasonably effective at catching a broad range of different domains. Further, the idea of generating a significant sample of all out-of-domain possibilities to create such a detector is infeasible. Again, The main goal is not to be able to detect any out-of-domain sample, but to present an observed behavior that could be useful in creating such a classifier.

2.6 Potential Uses for This Work

The work presented here could be used in a Mixture of Experts (MoE) ensemble model. MoE is a popular method of combining multiple neural networks into one model and has potential to

improve performance in machine learning applications [53]. A MoE model consists of several smaller *expert* networks and one general *gating* network. The gating network is responsible for determining which expert network is best suited for classifying the data sample. Each expert network is trained to be an expert on a local subset of the overall distribution of the data. It is reasonable to speculate that an ImageNet trained network's features could be an effective method of determining which expert to use in an MoE model. As the results show later in Chapters 4 and 5, datasets tend to cluster in feature space and tend to be easily separable (even in the case of similar input data), so it would not be unreasonable to speculate that such separation could be used to identify the proper expert for a given dataset.

2.7 Application to Large Datasets

Much of the above summary of work is based on the summary of out-of-distribution detection methods from [30]. This work provided an excellent and in-depth review of the state-of-the-art of out-of-domain detection methods. Additionally, Roady *et al.* [30] looked into the application of many of the methods discussed prior when applied to a large scale datasets and found that many of the methods do not perform as well when presented more complex datasets. Although, they did see some success, the ability to recognize novel inputs for large-scale datasets remains a difficult problem.

At the time of writing, no prior work on detecting out-of-domain images has specifically focused on the feature vector of an ImageNet trained network. Roady *et al.* [30] produces results for several out-of-domain detectors using the ImageNet dataset, but little work has focused on the high-level features as a means of detection. In particular, the application of a simple linear classifier as a means of detection of out-of-domain images has not been presented in the literature. In this respect the work presented is new.

2.8 Initial Motivations

The initial inspiration for this work came specifically from Bendale and Boult [7] because it showed that the pre-softmax activations tend to have recognizable patterns within each class. This raised the question: is there a recognizable pattern that in-domain images tend to have that makes distinguishing them from out-of-domain images possible. There were two key ideas that were focused on in order to try to find this pattern. The first was to take a step back from the pre-softmax outputs to the outputs of the final convolutional layer (the feature vector) prior to the linear transformation. This is because the pre-softmax activations are effectively just unnormalized confidences for a particular class, but the feature vector represents the presence or lack of presence of certain features within the image. The hope was that there would be a consistent pattern in these features, even across classes. Which leads to the second key idea. Obviously, for a network to be effective, it needs to be able to recognize different features from different inputs, so there would likely not be a consistent pattern, but perhaps there could be consistencies the strength of features as a whole. That is, images will not have the same feature with the highest excitation, but maybe the highest excitation will have a similar value across all in-domain data, and maybe the second highest in one image will have a similar value to all other in-domain images, and so on. The second key idea is that by ordering the features by excitation strength then there may be a resulting pattern that separates in-domain and out-of-domain images.

Ordering features may seem a bit counter-intuitive because all specifics for classes are thrown out, but for this project the class does not matter. The only thing that matters is the domain defined by the training data and how the features of this domain differ from all other domains, so, by ordering features it is hoped to produce an underlying pattern of excitations that separates the in-domain samples from out-of-domain samples.

Figure 2.1 shows that there appears to be a pattern in sorted activations of the feature vector for in-domain images and at least these out of domain datasets. Later sections will delve further into network choices and data set choices, but these initial results provide some interesting information about the excitations of feature vectors.

The top row of Figure 2.1 shows the average sorted feature vector excitations produced by Inception V4 when shown $1{,}000$ images from the ImageNet validation set and $1{,}000$ depth images of hand poses (Left) or $1{,}000$ images of famous artworks (Right). It appears that the Inception network tends to produce higher feature excitations for out-of-domain images than for in-domain images. It is surprising, in particular, that the out-of-domain images tend to produce higher activations than in-domain images for the most excited features. Intuition would suggest that the network, when presented an in-domain image, would produce a few high excitations for the features present in the image and several excitations near zero for the remaining features. Intuition would, also, suggest that an out-of-domain image would likely have fewer feature excitations near zero and the highest excitations less prominent because the network would not be as effective at extracting features that were not learned. Out-of-domain images tend to to have fewer activations near zero in in the intermediate activations, which is to be expected, but having stronger activations in the most prominent features is surprising because it is expected a network would not give a high response to features it was not trained to recognize. Regardless it is apparent that the average ordered activations of in-domain images have a distinct pattern relative to the average ordered activations of out-of-domain images when features are extracted using the Inception network.

The bottom row of Figure 2.1 shows the average sorted feature vector excitations produced by ResNet V4 when shown $1{,}000$ images from the ImageNet validation set and $1{,}000$ depth images of hand poses (Left) or $1{,}000$ images of famous artworks (Right). ResNet feature vectors appear to show a distinctive pattern between the average in-domain feature vector and the average out-of-domain feature vector. In particular, it it appears that an in-domain image tends to have a few high activations with most activations near zero. An out-of-domain feature vector tends to have a similar trend, but the strongest excitations tend to be lower than the strongest excitations of in-domain images, with intermediate excitations typically greater than that of in-domain images. This behavior is intuitive because one would expect that the network would have a few strong responses for features present in the image and several responses near zero when presented an in-domain image, while an out-of-domain image would likely produce stronger intermediate features

and weaker excitations in the strongest features because the network is unfamiliar with the data. Regardless, this network does appear to produce a trend in ordered feature activations of in-domain images at these datasets of out-of-domain images.

This part of the work falls into Chapter 2 because the assumption that a dataset's features would not have a distinctive pattern alone is incorrect. Ordering features does tend to produce a recognizable pattern between in-domain and out-of-domain datasets, but, as it turns out, inducing a pattern by sorting is not necessary. In fact, it is easier to differentiate between an in-domain and an out-of-domain image when the features are not sorted, showing a significantly high success rate use a simple linear classifier. This section is included because it contains some of the motivating ideas than went into the bulk of this work, but did not end up being a major part of the final results.

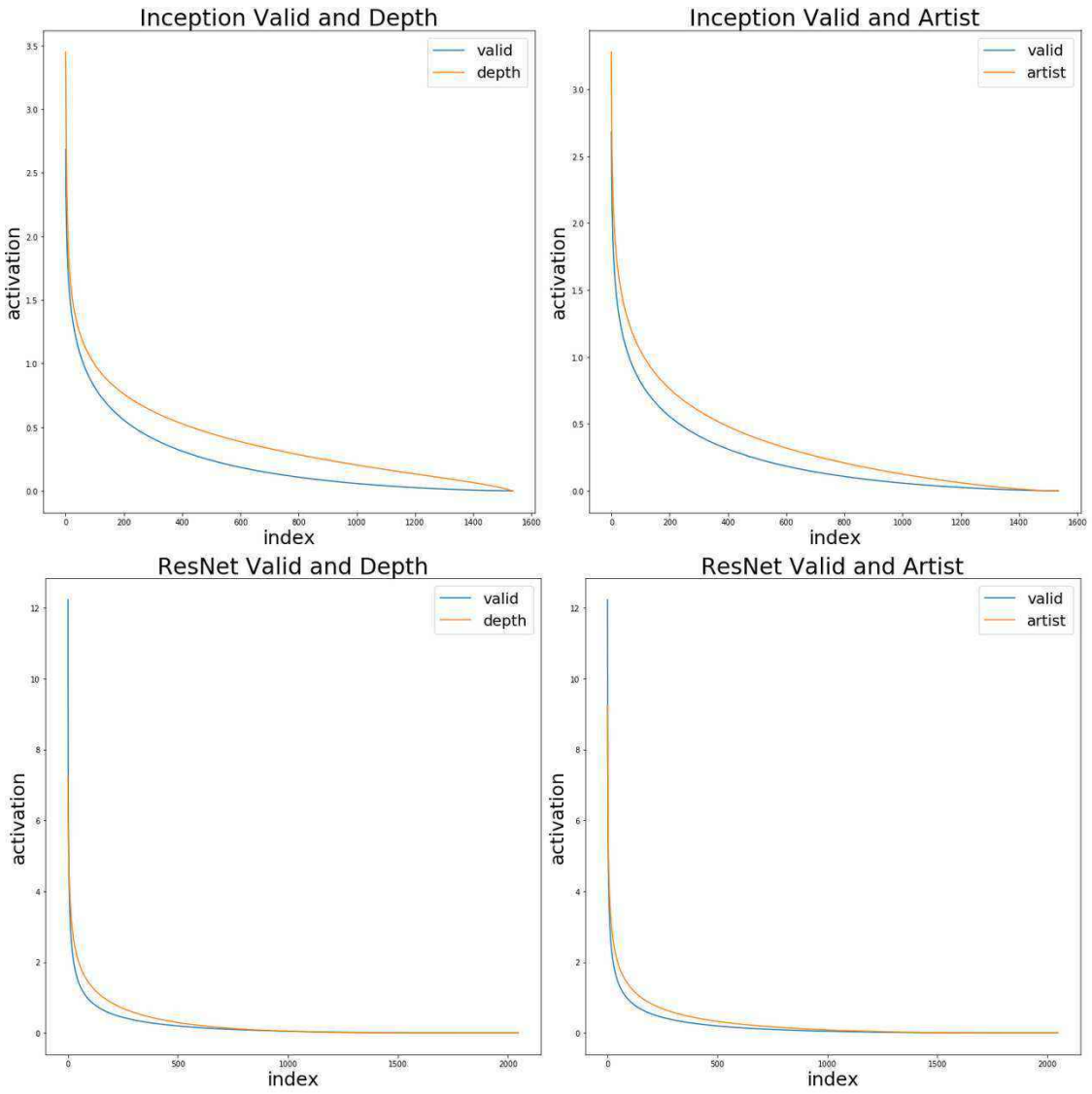


Figure 2.1: The averaged ordered feature excitations from the feature vectors. **Top Left**: Averaged ordered features extracted by Inception V4 from 1,000 examples from the ImageNet validation set and 1,000 examples from a depth dataset with hand depth images. **Top Right**: Averaged ordered features extracted by Inception V4 from 1,000 examples from the ImageNet validation set and 1,000 examples from a dataset of famous artworks. **Bottom Left**: Averaged ordered features extracted by ResNet V2 from 1,000 examples from the ImageNet validation set and 1,000 examples from a depth dataset with hand depth images. **Bottom Right**: Averaged ordered features extracted by ResNet V2 from 1,000 examples from the ImageNet validation set and 1,000 examples from a dataset of famous artworks. These plot show that there is a distinct pattern sorted activations for in-domain and at least these out-of-domain datasets.

Chapter 3

Approach

Section 3.1 of this chapter will present the datasets used for the experiments in detail. These datasets include the in-domain dataset, ImageNet, and four out-of-domain datasets: depth, anime, art, and SUN. Section 3.2 outlines the two deep CNNs along with a careful definition of the vector of feature activation levels, deemed the feature vector in this work. Section 3.3 will discuss how the feature vectors are harnessed and Section 3.4 will explain the experimental design.

3.1 Datasets

As introduced earlier, the aim of this work is to analyze the behavior of high-level features, i.e. the values in the feature vector a deep CNN learns to construct in order to recognize objects. Specifically, the question taken up here is whether these features naturally expose the distinction between in-domain and out-of-domain samples. To approach this question a working definition of what it means for a dataset to be in- or out-of-domain is needed. For these experiments the chosen in-domain dataset is ImageNet 2012 [13]. The reasons for this choice are the breadth of the dataset and the availability of pre-trained models. The ImageNet dataset consists of 1.2 million training images and $50{,}000$ validation images. Each image falls into one of $1{,}000$ different categories. These categories include several species of dog, multiple instruments, vehicles, and many more. The experiments presented in this work use images from both the validation and the training set.

To define what is considered out-of-domain is a little bit tricky. As stated earlier, an Okapi is likely to not be considered out-of-domain because it will share similar features with classes that are present in ImageNet. In this work out-of-domain is defined as something that has prominent features, but the features are generally different than features present in ImageNet. Some examples would include a dataset of x-ray images or drawings. Both of these examples have prominent features, but the features are not going to be similar to features that exist in ImageNet samples. For the purpose of the experiments four out-of-domain datasets have been carefully selected. Each dataset

is unique and presents features that range from clearly different from ImageNet, i.e. grayscale depth images of hand pose, to datasets with similar but generally different features, i.e. outdoor scenes. See Figure 1.1 for samples of each of the four out-of-domain datasets. The four are presented in detail the following subsections.

3.1.1 Depth

The Elicited Giant Gallery of Naturally Occurring Gestures (EGGNOG) dataset [14] consists of depth images of hand gestures captured by the Microsoft Kinect v2. This dataset is referred to as the *depth dataset*. There are 25 unique hand gestures in the form of one-channel grayscale images with a resolution of 168×168. The shade of gray is dictated by the depth of the hand relative to the camera. The background of each image is whited out. These data are the most visually different from in-domain data because they represent a completely different image modality.

3.1.2 Anime

The second dataset comes from the Danbooru2018 Large-Scale Crowdsourced and Tagged Anime Illustration Dataset [15] which is referred to as the *anime dataset*. This dataset consists of over 3 million anime-style illustrations. These images vary in size and may be a one-channel grayscale or three-channel RGB image. The images have color and have more variation than that of the depth images, but still differ greatly from ImageNet images.

3.1.3 Art

The third out-of-domain dataset consists of about $8{,}000$ artworks from history's most famous artists. These data an be found at [16] and is referred to as the *art dataset*. These artworks cover a broad range of art styles including surrealism, romanticism, post-impressionism, etc. Some of the works have characteristics that are far from that of natural images such as ones in the style of abstractionism and cubism, others contain near photorealistic characteristics and express the composition of a real scene. The artworks are one step closer to ImageNet due to the presence of

more realistic characteristics in some of the works as well as the greater variation in the dataset as a whole that does not exist in the anime and depth images.

3.1.4 SUN

The SUN Database: Scene Categorization benchmark [17] (abbreviated to SUN) consists of 397 classes of photographed scenes including airport terminals, igloos, jail cells, video stores, etc. This dataset steps closer to ImageNet images in terms of visual similarity because they are natural colored photographs. What makes this dataset out-of-domain relative to ImageNet is that what may typically fall in the background of ImageNet imagery is the focus of SUN. One key feature of the SUN dataset is there are a few classes, such as "valley" that overlap with the ImageNet dataset. There are not many of these overlapping classes, so this is still a valid comparison, but some of the challenge related to distinguishing SUN from ImageNet could be related to the overlapping classes.

3.2 Networks

The two deep CNNs used for the experiments are ResNet V2 152 and Inception V4 from [2] and [1] respectively. These two networks were specifically chosen due to their prominence in the literature, the obvious differences in their architecture, and finally the relatively similar overall recognition performance when trained and tested on ImageNet. Both models achieved near state-of-the-art performance when they were initially published and have remained popular choices for experimentation and application.

The ResNet architecture is very deep and narrow with residual shortcut connections. The residual shortcut connections allow for lower level features to be included as inputs to layers deeper in the network. Inception has a wider, but shallower architecture which makes use of convolutional blocks. There are five different block architectures used in Inception V4. Three Inception Block architectures and two Reduction Block architectures. These convolutional blocks consist of multiple dataflow paths. Each path has a unique set of layers of varying shapes and sizes. The results from

each path are concatenated prior to being passed into the next layer and/or block. Despite these distinctions, both networks achieve similar accuracy on ImageNet. At a high level, each network consists of a number of convolutional layers followed by a linear transformation then a softmax layer for classification.

The feature activation levels, or **feature vectors**, are the output of the final convolutional layer just before the linear transformation layer (See Figure 1.4 for an example). Most CNNs designed for an ImageNet style task are capped at the end of the network by a single linear transformation which is essentially a series of perceptrons which are each tuned for a specific class label. This format is true for both ResNet and Inception. The use of a single linear layer for classification forces the bulk of feature extraction and consolidation to occur in the convolutional layers. The number of features in the feature vector can vary between networks while the number of nodes in the final linear layer is directly related to the number of output classes. Inception feature vector encodes $1{,}536$ distinct feature levels while ResNet feature vectors are larger, with $2{,}048$ feature activation levels. In the experiments both CNNs are run within the TensorFlow version 1.14 package. The weights of the networks were trained using ImageNet data. Specifically, the networks are initialized with the pre-trained ImageNet weights provided as part of the TensorFlow-slim Image Classification Library [54].

3.3 Feature Extraction

Images in each of the five datasets highlighted in Figure 1.1 are passed through the two networks trained on ImageNet and the associated feature vectors are retained. As stated previously, the experiments do not require class labels so the class labels have not been retained. Instead it is sufficient to tag each feature vector with a label of the domain from which it came. Ultimately, our experiments will focus on addressing a two class problem; based upon the feature vector derived from an image, is the image from the in-domain class (i.e. ImageNet) or the out-of-domain class (i.e. one of Depth, Anime, Art, or SUN). By being able to effectively separate datasets it is possible to learn about a networks underlying feature representations.

It is possible that the different pre-processing steps used for these datasets could alter the results. To minimize this risk, the exact same ImageNet pre-processing steps were applied to all images used in the study. To be more specific, prior to feature extraction, each image is loaded and decoded (images are stored as .jpg or .png files), a center crop of the image is taken with padding, the pixel values are converted to float, the images are scaled to 299 by 299, and the pixel values are normalized between negative one and one. Grayscale images are treated as 3 distinct channels. This pre-processing approach is adapted from [55].

3.4 Experimental Setup

This Section describes the experimental setup in detail for all of the experiments in this work. Subsection 3.4.1 details the experimental setup of comparing in-domain feature vectors to out-of-domain feature vectors as well as comparing the feature vectors of the different out-of-domain datasets. Subsection 3.4.2 details how feature vectors are compared when the Inception and ResNet weights are fine-tuned to the SUN dataset.

3.4.1 Evaluating Features of Pre-Trained Models

The experimental approach is relatively simple, by design, as there is no need for complicated models to produce significant results. In particular, a simple linear classifier is used to measure the separability of in-domain and out-of-domain datasets. Although simple, linear separability shows a distinct and important underlying behavior. Using a non-linear model may produce better results, however, the goal is not to produce the best model but to understand if there is a significant distinction between in-domain features and out-of-domain features. A linear model will highlight the simple relationship between two datasets, while, a non-linear model has the potential to cloud simple relationships in more complex interpretations of the data. Additionally, linear models have the useful property that the relative distance between the classes becomes easier to visualize. A linear Support Vector Machine (SVM) with a penalty term of `C=0.001` is used to determine the separability of two domains. The same penalty term was used for each dataset because, again, the

performance of the model is not the main focus for this work, but rather to understand if there is a pattern in feature behavior of in-domain and out-of-domain classes. By keeping the penalty term consistent it is easier to compare each dataset comparison to the other dataset comparisons.

For each dataset a subset of $5{,}000$ images were randomly selected without replacement. This choice is intended to make computation easier and, because the images were selected randomly the subset should reasonably represent characteristics of the whole dataset. Most if not all of the classes should be represented in this subset. Feature vectors were extracted and stored for each image.

The linear classifier was trained with a subset of $4{,}000$ feature vectors randomly selected from the ImageNet training feature vectors and a random sample of $4{,}000$ feature vectors from an out-of-domain dataset. The model is then evaluated using $1{,}000$ feature vectors from the ImageNet validation dataset and the remaining $1{,}000$ feature vectors from the out-of-domain dataset. It is very important that the linear model is trained on the ImageNet training set and evaluated on the ImageNet validation set because the performance of the classifier needs to be measured based on the feature vectors from network inputs that the original network was not trained on. In evaluating the linear classifier it is imperative to analyze the difference between in-domain but unseen data (the validation set) and out-of-domain data. This approach for training and evaluating the SVM ensures that the separability of two domains is accurately represented.

The experiments presented in this work differ significantly in approach, networks, and datasets compared to prior work, but to be as robust as possible, the results are presented using similar evaluation metrics to several of the related approaches [3, 5, 10]. The first metric is the Area Under the Receiver Operating Characteristic curve (AUROC). AUROC gives an idea of how well a binary model is able to separate two classes. Secondly, the Area Under the Precision-Recall curve (AUPR) is used as an additional measure. This metric is similar to the AUROC, but tends to show more information in cases where one class has significantly more examples than the other class [3]. The datasets used are selected to be the same size, but this metric is included for the sake of completeness. The AUPR metric is computed for both the in-domain and out-of-domain

samples (designated as AUPR_in and AUPR_out). The final metric used for evaluation is accuracy. Some of the prior work tends to present the True Positive Rate (TPR) at a fixed True Negative Rate (TNR) as opposed to presenting basic model accuracy. However, this extra step in evaluation is not required for these experiments because the datasets are of equal size making standard accuracy a useful evaluation metric.

To quantify repeatability, each metric was computed 30 times and the mean and standard deviation are reported in the resulting tables. This approach is chosen as opposed to K-Fold Cross-validation due to the requirement of fitting the linear classifier on the ImageNet training data and evaluating on the validation data. For each run $4{,}000$ feature vectors are randomly selected from the ImageNet training set, $1{,}000$ vectors are randomly selected from the validation set, and a random shuffle of the out-of-domain vectors are split into $4{,}000$ vectors for fitting the classifier and $1{,}000$ for evaluating the classifier. The data used to evaluate the classifier is exclusive from the data used to train the classifier.

As an additional experiment, the feature vectors of the out-of-domain datasets are also compared to each other in Chapter 5. These experiments retain all of the experimental design aspects previously described, except that instead of using the ImageNet training and validation sets, the two out-of-domain datasets are randomly shuffled and split into $4{,}000$ vectors for training the linear classifier and $1{,}000$ vectors for evaluating the classifier.

To try to better understand why some confusions between datasets occur, some examples of the misclassified images are included for each dataset comparison experiment. Additionally, a table is included to show exactly how many feature vectors are confused on average between two datasets when $1{,}000$ feature vectors from each dataset are used for linear classifier evaluation. These tables bring to light some useful aspects of the dataset comparisons. For example, these tables show if two datasets are confused with each other at the same rate, if ResNet feature vectors are confused at the same rate as Inception feature vectors, and if the confusing feature vectors produced by ResNet are the same as the confusing feature vectors produced by Inception. The values are presented as

whole numbers instead of proportions or percentages because it the results are easier to understand as by image comparisons.

3.4.2 Evaluating Features of Fine-Tuned Models on Sun Dataset

To further understand the behavior of feature vectors of in-domain and out-of-domain images, an additional experiment fine-tunes Inception and ResNet on the SUN dataset using the pre-trained ImageNet weights. This experiment is intended to provide more information about the meaning of the separability of in-domain and out-of-domain datasets. Only the SUN dataset is chosen for these experiments because it is the most similar to ImageNet and because it is the largest labeled dataset of the out-of-domain datasets. This dataset consists of 397 unique classes and $108{,}756$ images.

For these experiments the SUN dataset was split into training and test sets of proportions 0.8 and 0.2 respectively. The training and test sets are exclusive. Each model was initialized with the pre-trained ImageNet weights provided in [54]. The weights of final linear layer for each model were excluded to account for the difference in the number of classes. The pre-processing steps are the same for each model and were described in Section 3.2. A batch size of 16 is used for both models. The classification heads are trained for 10 epochs with a learning rate of 0.001. Then the entire network is tuned to convergence with a learning rate of 0.00001. A dropout rate of 0.5 and a weight decay of 0.0005 are used for tuning. When the model has converged the model weights that produced the best accuracy on the test set are retained. A softmax cross-entropy loss function was used with a gradient descent optimizer.

There is no hyperparameter tuning for these models because the final performance of the model is not extremely important to perform the experiment. It is important that the models are tuned to the SUN dataset, but high performance is not a requirement for analyzing how the SUN feature vectors have changed after tuning the model. It is possible that doing some hyperparameter tuning could slightly improve the accuracy of the model, but likely, if a trend exists, it will be visible without placing extra effort into the development of a high-performing model.

The evaluation of this experiment is similar to the experiments described in Subsection 3.4.1. Evaluation involves selecting $5{,}000$ SUN images randomly from the SUN training dataset and $5{,}000$ from the SUN testing dataset for analyzing feature vector distinctions. Each of these two data subsets are passed through the models with the original ImageNet weights and the models with the fine-tuned SUN weights. The feature vectors are extracted and stored. The final part of this experiment involves training linear SVMs on the original ImageNet features and the tuned SUN features. Comparing the performance of this classifier to another SVM trained on the original ImageNet features and the ImageNet-trained SUN features reveals how feature vectors change after tuning a model for a new domain. The two main points of interest are: does tuning the model to the SUN dataset affect how separable the feature vectors are from the original ImageNet features feature vectors, and do the tuned weights more frequently produce a correct classification if the feature vectors are farther from or closer to the ImageNet feature vectors. The fitting of the linear SVM and the evaluation metrics used are the same as described in Subsection 3.4.1.

Chapter 4

Feature Mappings of In-Domain and Out-of-Domain Data

This chapter will present the results of the experiments comparing the in-domain ImageNet feature vectors and the out-of-domain feature vectors using the ImageNet weights. This chapter will, also, look into what types of images cause confusion, if ResNet and Inception are confused by the same images, and about how many images are confused on average when $1{,}000$ images from each dataset are used for evaluation. The first section, Section 4.1, analyzes how feature vectors of depth data and ImageNet compare. Section 4.2 looks at Art and Anime data feature vectors in comparison to the ImageNet features. Section 4.3 compares features from the SUN dataset to ImageNet. The performance metrics are summarized in Table 4.1. Additionally, Figures 4.1, 4.3, 4.5, and 4.7 allow for visualizing how far the features of the out-of-domain datasets are from the ImageNet features. As a sanity check, Section 4.4 tries to separate the ImageNet dataset from itself.

Even though the feature extractors reduce the overall number of features relative to the initial input image the feature vectors are still high dimensional ($1{,}536$ for Inception and $2{,}048$ for ResNet) making them difficult to visualize. The data is reduced to a one-dimensional space by taking the dot product of the feature vector and the vector normal to the decision hyperplane. By taking the dot product of the feature vectors and the normal to the hyperplane the positive examples map to the positive real numbers and the negative examples map to the negative real numbers. To better visualize, a second dimension is added. This dimension is the total sum of the features in the feature vector. This choice not only improves visualization, but, as can be seen in Figure 2.1 out-of-domain images tend to, on average, have higher activations.

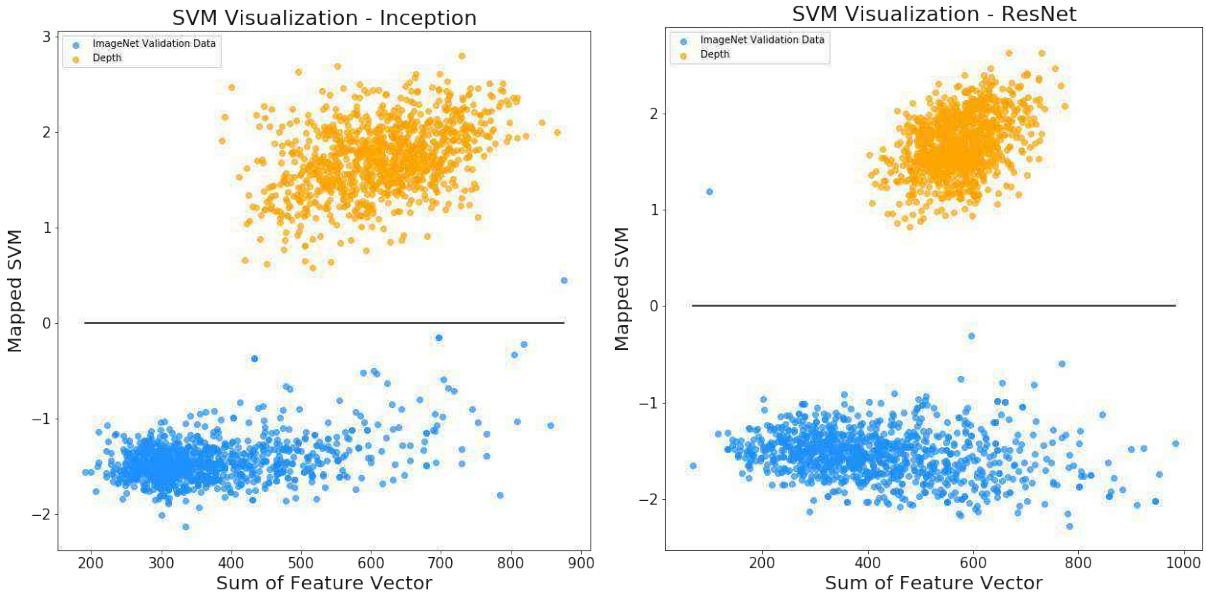
Table 4.1: This table shows the mean and standard deviation of each evaluation metric after the experiment was run 30 times. Each run involves randomly selecting $4{,}000$ of the $5{,}000$ ImageNet training set feature vectors and $4{,}000$ of the $5{,}000$ out-of-domain feature vectors. These metrics are then computed using a randomly selected $1{,}000$ of the $5{,}000$ ImageNet validation feature vectors and the remaining $1{,}000$ out-of-domain feature vectors. These results are presented as mean (standard deviation). Because the experiment was set up so that the number of in-domain samples was the same as the number of out-of-domain samples in evaluation the area under the curve for a random classifier is always 0.5

Network	In-/Out-of-Domain	AUROC	AUPR_In	AUPR_Out	Accuracy (%)
Inception	ImageNet/Depth	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	99.93 (0.05)
	ImageNet/Anime	0.999 (0.000)	0.999 (0.000)	1.000 (0.000)	99.00 (0.24)
	ImageNet/Art	0.997 (0.001)	0.997 (0.001)	0.997 (0.001)	97.43 (0.38)
	ImageNet/SUN	0.962 (0.003)	0.965 (0.004)	0.957 (0.005)	90.33 (0.54)
ResNet	ImageNet/Depth	1.000 (0.001)	1.000 (0.000)	1.000 (0.000)	100.00 (0.00)
	ImageNet/Anime	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	99.49 (0.14)
	ImageNet/Art	0.997 (0.001)	0.997 (0.001)	0.997 (0.001)	97.63 (0.35)
	ImageNet/SUN	0.964 (0.004)	0.969 (0.003)	0.956 (0.006)	90.79 (0.58)

4.1 Depth Data

This section discusses how the feature vectors extracted from the depth data compare to the ImageNet feature vectors. The depth images were captured from a depth sensor and consist of different hand poses. These data are visually far from random noise, but differ greatly from most of the images in ImageNet because they are grayscale, have no background, and are not similar to natural photographs. Figure 4.1 displays the separability of ImageNet and Depth feature vectors by mapping the feature vectors to one dimension and including the sum of features as a second dimension. Additionally, Figure 4.2 shows examples of some images from the depth dataset and ImageNet that are confused by the linear model. Table 4.2 gives the average number of misclassifications by the linear model when the model is trained using $4{,}000$ randomly selected examples from each dataset and evaluated using the remaining $1{,}000$ examples from each dataset.

Figure 4.1 highlights why the inclusion of the sum of features as an additional dimension is valuable in addition to making the results easier to visualize. The overall sum of activations seems to matter as the ImageNet vectors tend to have a lower overall activation. Although not perfect, at least in the case of depth images, the sum of the feature vectors is a reasonably effective method for separating in- and out-of-domain examples. The separation along this axis is not as significant



(a) Inception **(b)** ResNet

Figure 4.1: This figure shows the separability of ImageNet and Depth features. **a.** shows the mappings for Inception and **b.** shows mappings for ResNet. Each feature vector is mapped down to one point by doing a dot product of the feature vector and the normal to the learned hyperplane. This becomes our y-axis. The x-axis is the total sum of activation for each feature vector.

for other domains, but the tendency for out-of-domain images to have a higher overall activation is still present.

The separability of the ImageNet and depth features seen in Figure 4.1 is striking. Both networks produce feature vectors that are almost perfectly separable from each other with a linear classifier. The results presented in Table 4.1 show that the feature vectors are highly separable across multiple runs. Both AUPR values and the AUROC value are close to 1.0 indicating that the data is near perfectly separable. The accuracy score backs this up. The average of 30 runs is near perfect with a small standard deviation meaning that these results likely apply in general. An important point to repeat is that all of the data shown in Figure 4.1b is only used for evaluation and not training the linear classifier. Granted, the depth data is very different from most of the ImageNet data, but it is not intuitive that near perfect linear separability would occur in feature space. This means that there is some sort of overall pattern in the high-level features for the ImageNet

domain and the depth domain. Essentially, even though the networks are designed to discriminate, the characteristics of the domain cause some sort clustering across classes.

The ImageNet dataset is pretty broad, so it is not surprising that there are a few examples of images that could be confused with depth images. However, there might be interesting information in those images themselves. Figure 4.2 shows ImageNet images that were confused with depth images by the linear classifier for a random partition of the two datasets. There were no depth images confused for ImageNet images. Figure 4.2d was the only image that was confused by the linear classifier using both Inception and ResNet generated feature vectors. The remaining images in the figure were only confused when the feature vectors were extracted by the Inception network. Figures 4.2b and 4.2d were likely confused for depth images because they both are roughly grayscale, have pretty solid backgrounds, and the main features are somewhat centered. It is a little less clear why Figures 4.2a and 4.2c produce activations that are not easily distinguishable from the depth activations. It is interesting that both of these images contain a statue on top of some sort of structure, but it seems unlikely that there would be a correlation between having a statue and producing depth-like features. Perhaps the confusion occurs because the backgrounds are pretty consistent and the main features are roughly centered. Figure 4.2a is consistently confused for all four of the out-of-domain datasets when the features are extracted by Inception, so it is possible that Inception just has trouble extracting meaningful information from this image.

(a) **(b)** **(c)** **(d)**

Figure 4.2: The four images from the ImageNet dataset that were confused with the depth data for a particular partition of the two datasets when the linear SVM is trained on the feature vectors. Only **d.** was confused by the linear classifier for both Inception and ResNet feature vectors. The remaining three images were only misclassified when the features were extracted by the Inception network.

Table 4.2: This table shows the number of ImageNet and Depth images, on average (with the standard deviation in parentheses), that are confused by the linear classifier for Inception and ResNet. The Inception column is the total number of images that were misclassified when using Inception features. The ResNet column is the total number of images that were misclassified using ResNet features. The Intersection column lists the number of images that were misclassified by both networks.

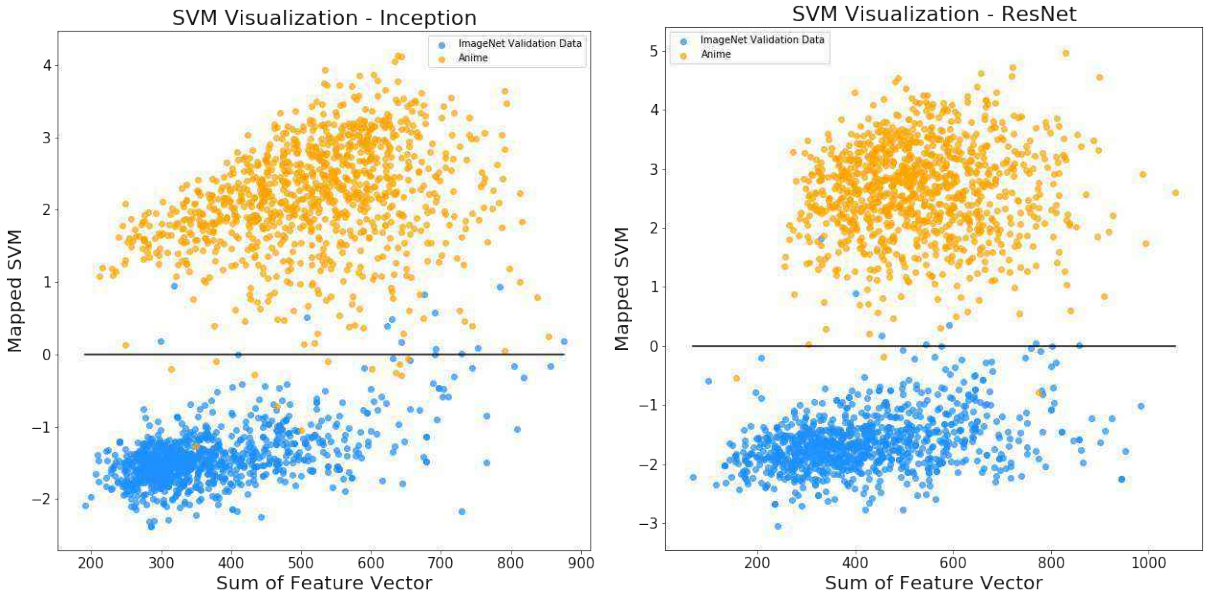
	Inception	Intersection	ResNet	Total
ImageNet	1.7 (0.9)	0.4 (0.5)	0.4 (0.5)	1.7 (0.9)
Depth	0.0 (0.2)	0.0 (0.0)	0.0 (0.0)	0.0 (0.2)
Total	1.7 (0.9)	0.4 (0.5)	0.4 (0.5)	1.7 (0.9)

On a rare occasion, ImageNet images are mistaken for depth images. As Table 4.2 shows, on average, fewer than two ImageNet images with features extracted by Inception are confused with depth images when $1{,}000$ images from each dataset are used to evaluate the linear model. ResNet features from ImageNet are confused even less with fewer than one misclassified ImageNet image per $1{,}000$ evaluations. It is interesting, though, that the only confused features from the ResNet model are also confused when features are extracted by Inception. Depth images are basically never confused with ImageNet images. Depth images do not have much variation, so it would be surprising if they were confused with ImageNet frequently. However, there is so much variation in the ImageNet dataset that a few confusions of ImageNet vectors for depth vectors is likely.

4.2 Art and Anime Data

The problem with depth data is how visually different the depth images are from the average ImageNet image. Out-of-domain images are expected to have a reasonably visible distinction from the ImageNet images, but depth images are grayscale with foreground traits located right in the middle of the frame. Additionally, depth images have a pretty flat background with little variation. For better understanding of features it is important to consider datasets that are still outside of the ImageNet domain but have more variation than the depth images.

The anime and art datasets are employed because they have much more variation than the depth data, but still come from a domain separate from the ImageNet imagery. The anime and art datasets



(a) Inception **(b)** ResNet

Figure 4.3: This figure shows the separability of ImageNet and anime features. **a.** shows the mappings for Inception and **b.** shows mappings for ResNet. Each feature vector is mapped down to one point by doing a dot product of the feature vector and the normal to the learned hyperplane. This becomes our y-axis. The x-axis is the total sum of activation for each feature vector.

can vary greatly, they typically have color, the background contains significant variation in many cases, and the foreground is not necessarily centered in the middle of the frame.

Figure 4.3 shows the separability of the ImageNet and anime features for both Inception (Figure 4.3a) and ResNet (Figure 4.1b) and the performance metrics can be seen in Table 4.1. Perhaps unsurprisingly, by using a dataset with more variation the separation of ImageNet features from anime features is not as striking as with the depth data. However, the distinction between the ImageNet images and the anime images is still surprising. There are only a few examples for both networks where an ImageNet image crosses the boundary and only a few examples of anime images crossing the boundary. The two datasets are almost perfectly separable based on the plots, and the performance metrics back up this claim. The AUROC and both AUPR scores are almost 1.0 on average, and the accuracy is almost perfect with 99.0% accuracy on average for Inception and approximately 99.5% accuracy on average for ResNet. Again, none of the points shown in Figure 4.3 were used to fit the linear model and there was no pre-processing to the features. It is

surprising that even across classes there is a near perfect division of these two datasets in features space using a simple linear classifier.

Looking at the images that produced confusing features in Figure 4.4 it is possible to speculate on why confusions between the ImageNet and anime occur. Because so few images are confused between the ImageNet and anime datasets it is likely that some of the confusing characteristics will be visible. For example, it is likely that Figure 4.4a produces very different features from the typical anime image which probably looks more like Figure 4.4d. Figures 4.4b and 4.4c have painting-style qualities that may lead to the confusion, or possibly the presence of shading adds a three dimensional perspective. The ImageNet images that have confusing features do have some similar characteristics. Images 4.4e and 4.4f are probably somewhat unusual compared to the average ImageNet image especially because there are not many shadows that show three dimensions. The tie in Figure 4.4g has some artistic qualities that could be confused, especially because the background is not busy. Image 4.4h has an illustration, which might produce similar features to the anime images. The anime images that are confused seem to have more three dimensional features or just deviate from what an anime image typically looks like. The ImageNet images that are confused tend to lack a three dimensional perspective or have a colorful illustration or pattern similar to many of the anime images. In total there were 19 feature vectors confused between the two networks.

Table 4.3: This table shows the number of ImageNet and anime images, on average (with the standard deviation in parentheses), that are confused by the linear classifier for Inception and ResNet. The Inception column is the total number of images that were misclassified when using Inception features. The ResNet column is the total number of images that were misclassified using ResNet features. The Intersection column lists the number of images that were misclassified by both networks.

	Inception	Intersection	ResNet	Total
ImageNet	8.8 (2.7)	2.2 (1.2)	4.8 (1.7)	11.4 (3.2)
Anime	10.6 (3.0)	2.8 (1.4)	4.9 (1.8)	12.7 (3.3)
Total	19.4 (3.5)	5.1 (1.7)	9.7 (2.0)	24.1 (3.8)

Table 4.3 shows the counts, on average, of misclassified feature vectors for the anime and ImageNet datasets. When looking at the misclassifications per dataset when $1{,}000$ images from the

Anime

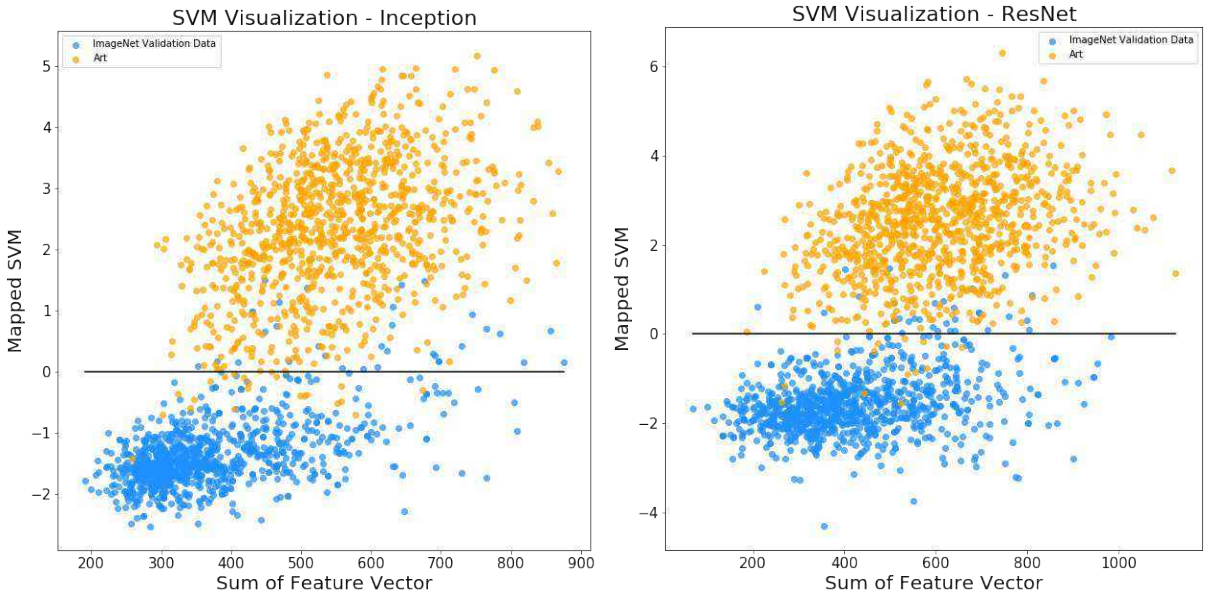
(a) **(b)** **(c)** **(d)**

ImageNet

(e) **(f)** **(g)** **(h)**

Figure 4.4: The top row shows anime images whose features were misclassified as ImageNet features by the linear classifier. The bottom row show ImageNet images whose features were misclassified as anime features by the linear classifier. All four anime images were misclassified by the linear classifier with both Inception and ResNet features. These four anime images were the only four misclassified when features were extracted by ResNet. Anime features from Inception had five additional misclassifications. **e.** was the only ImageNet image misclassified for both Inception and ResNet features. **f.** was only misclassified by ResNet. **g.** and **h.** were only misclassified by Inception.

ImageNet and anime datasets are used to evaluate the linear model, it is interesting, that ResNet features are confused less than half as frequently as Inception features. The reason for this difference is hard to pin down, but one could speculate on a variety of explanations. One reason may be that the ResNet features are more descriptive of the anime dataset than Inception, so the extracted features of anime and ImageNet are more distinct from each other. Alternatively, it is possible that the inverse is true, that ResNet features are less descriptive of the anime dataset meaning that an anime feature vector does not contain meaningful interpretations of the characteristics of an anime image. Thus the well described ImageNet feature vectors would map far from the poorly described anime feature vectors. Either way the linear separability of these two datasets shows that there are some across-class characteristics within a dataset that these networks seem to capture.



(a) Inception **(b)** ResNet

Figure 4.5: This figure shows the separability of ImageNet and Art features. **a.** shows the mappings for Inception and **b.** shows mappings for ResNet. Each feature vector is mapped down to one point by doing a dot product of the feature vector and the normal to the learned hyperplane. This becomes our y-axis. The x-axis is the total sum of activation for each feature vector.

It seems that the anime and ImageNet feature vectors are confused at about the same rate for both networks. This means that there is a reasonable amount of variation in both of the datasets. This is different from the depth data which was never confused with ImageNet. The fact that the two datasets are confused at about the same rate likely means that the anime features are not entirely without information, and in some cases seem to have similar information to ImageNet data.

The art dataset has some similar qualities to the anime dataset, but there are some key differences in the features. In particular the art dataset includes some images that look somewhat similar to photographs which probably does not occur as frequently in the anime art style. Figure 4.5 shows the separability of the art features and the ImageNet features for Inception and ResNet. It appears that this dataset takes one step closer to the ImageNet dataset in terms of features, or is, at least, harder to distinguish. It is interesting to note that the art features do not appear to cluster as well as the previous out-of-domain datasets. Both feature extractors produce features that are not

entirely distinguishable, but the separation of the two datasets is still pretty surprising. Visually it appears that a linear classifier can separate the ImageNet features from the art features with pretty good accuracy. The performance metrics in Table 4.1 show that the art ImageNet separation is near perfect. Both networks produce AUPR scores and an AUROC score that support near perfect separation on average with a low standard deviation. The overall accuracy is about 97.5% for Inception and ResNet features.

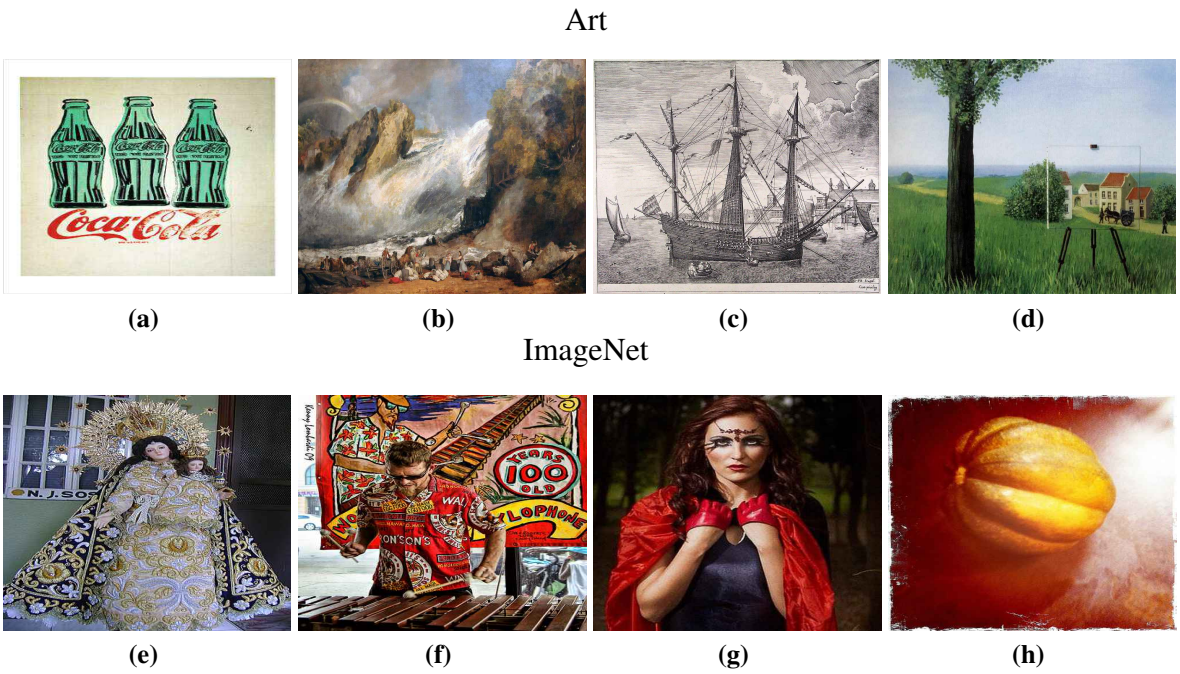


Figure 4.6: The top row shows art images whose features were misclassified as ImageNet features by the linear classifier. The bottom row show ImageNet images whose features were misclassified as art features by the linear classifier. Out of the images **a** and **e** were only misclassified when features were extracted by Inception, **b**, **c**, **f**, and **g** were misclassified when features were extracted by both networks, and images **d** and **h** was only misclassified when features were extracted by ResNet.

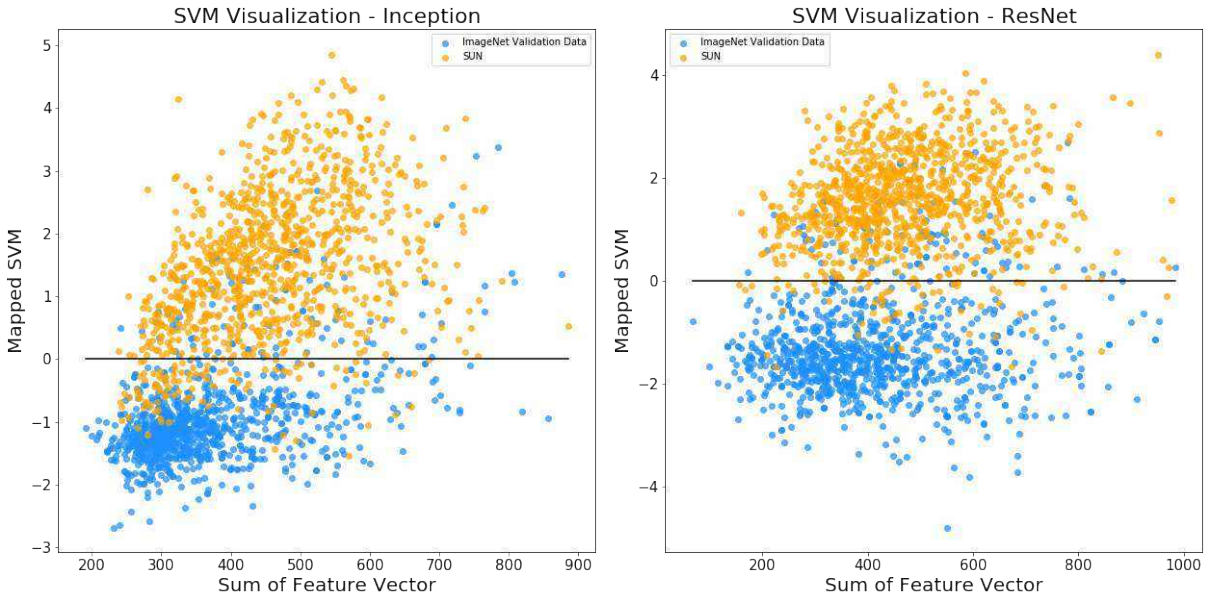
Figure 4.6 shows some of the art and ImageNet images that are confused by the linear model for separating feature vectors. For the most part these linear classifiers are very successful at separating the art and ImageNet datasets, but there are a few cases in which these models seem to have trouble. In particular, it looks like ImageNet includes some paintings in the dataset (Figure 4.6h) which, unsurprisingly cause the classifier to fail. Other than that the ImageNet images that caused the classifier to fail seem to have artistic qualities such as the doll in Image 4.6e or the shirt and

background in Image 4.6f which could throw the model off. Image 4.6g seems to have some image filtering that makes the colors and makeup look more artistic and less realistic. The reason the art images fail is not quite as clear. Images 4.6b and 4.6d seem to have somewhat realistic qualities that could potentially result in features similar to that of an ImageNet image. However 4.6a and 4.6c do not seem to have ImageNet-like qualities. Maybe these images differ from the standard art images enough that the features are just hard to recognize.

Table 4.4: This table shows the number of ImageNet and art images, on average (with the standard deviation in parentheses), that are confused by the linear classifier for Inception and ResNet. The Inception column is the number of images that were misclassified when using Inception features. The ResNet column is the number of images that were misclassified using ResNet features. The Intersection column lists the number of images that were misclassified by both networks.

	Inception	Intersection	ResNet	Total
ImageNet	25.3 (4.6)	11.4 (3.0)	24.9 (3.5)	38.8 (4.5)
Art	24.8 (5.3)	9.2 (2.8)	21.4 (4.0)	37.0 (6.2)
Total	50.1 (6.4)	20.6 (3.3)	46.3 (5.2)	75.8 (7.7)

It appears that the confusion of art and ImageNet features is related, at least in part, to the realistic features that occur in some of the artworks and the broadness of ImageNet leads to the inclusion of images that are art themselves or contain art-like qualities. Table 4.4 shows how many images were misclassified on average when $1{,}000$ images of the art and ImageNet datasets were presented to the linear classifiers. Similar to the anime data, the number of art images confused with ImageNet and ImageNet images confused with art is reasonably similar (within one standard deviation of each other). The interesting variation from the results seen with the depth-ImageNet and anime-ImageNet comparisons is that with art there are about the same number of misclassifications with Inception features as there are with ResNet features. It is curious why, with this dataset, Inception and ResNet features seem to be more similar. Inception does have slightly higher performance on the ImageNet dataset than ResNet does, so maybe Inception is slightly more specified for ImageNet and produces weaker features for datasets less like ImageNet. The art dataset probably contains images that are more similar to ImageNet than anime and depth, so Inception may be able to extract more meaningful features from the art dataset. Meanwhile ResNet may produce



(a) Inception **(b)** ResNet

Figure 4.7: This figure shows the separability of ImageNet and SUN features. **a.** shows the mappings for Inception and **b.** shows mappings for ResNet. Each feature vector is mapped down to one point by doing a dot product of the feature vector and the normal to the learned hyperplane. This becomes our y-axis. The x-axis is the total sum of activation for each feature vector.

more general features that allow for decent feature extraction with datasets unlike the dataset it was trained for.

The separability of the anime and art features from the ImageNet features is surprising. These datasets are visually quite different from the ImageNet data in most cases. Even the more photorealistic artworks tend to lack qualities of a natural photograph, but with all the variation in the imagery from these datasets it is surprising that the features, in the end, are so separable from ImageNet features with a linear classifier. It would be reasonable to expect some difference in features, but the presence of linear separability shows that the networks tend to produce similar feature vectors for each image in a domain.

4.3 SUN Data

Given the high rate of separability of the ImageNet features and the art and anime features it is logical to try to push this concept even further. The SUN dataset is very close to the ImageNet

dataset in that it consists entirely of natural photographs, so it is expected that the two datasets will carry many similar characteristics. There are even some overlapping classes. However, the SUN dataset is still considered out-of-domain because the main focus of this dataset is the *scene* which is the background in most ImageNet images. An ImageNet trained network will learn to focus on the foreground while a SUN trained network will learn to focus on the background.

Figure 4.7 shows the separability of the SUN dataset and the ImageNet dataset. Surprisingly there is still a significant amount of separation between the two datasets. The separations are not quite as good as the previous examples, but there are obviously two distinct clusters. The AUPR scores and the AUROC scores are very good ($0.95+$) on average showing that a linear model is pretty effective at separating these two datasets in feature space. The accuracy is just over 90% for both models. Considering the overall similarities of the datasets it is extremely surprising that this separation holds up. It is not unlikely that there is some kind of dataset bias in these datasets [56], for example, many of the SUN images seem to be much larger than the ImageNet images, but to be separable up to 90% accuracy when considering two datasets consisting of natural photographs that span a great breadth of classes and variations. These ImageNet trained models seem to know what dataset they are trained for. These results seem to show that the networks know what ImageNet data looks like and can relatively easily recognize when an image is not from ImageNet even if the image is very similar.

Figure 4.8 shows the the SUN and ImageNet images that were confused. Looking at these images it is not surprising that SUN is more frequently confused with ImageNet as it would be hard for the average person to distinguish which image comes from which dataset. It is not surprising that SUN and ImageNet are harder to distinguish than the other datasets, but the fact that on average both Inception and ResNet SUN features are separable from the ImageNet features with over 90% accuracy is really surprising. This results seems to suggest that there is some underlying dataset characteristic that the networks can pick up on allowing these two very similar datasets to fall in different places in feature space. Perhaps a deep visual analysis into a larger set of images from each dataset would reveal a more clear distinction, but the images presented in Figure 4.8 do

SUN

(a) **(b)** **(c)** **(d)**

ImageNet

(e) **(f)** **(g)** **(h)**

Figure 4.8: The top row shows SUN images whose features were misclassified as ImageNet features by the linear classifier. The bottom row shows ImageNet images whose features were misclassified as SUN features by the linear classifier. The two datasets contain highly similar imagery. Out of the images **a** and **e** were only misclassified when features were extracted by Inception, **b**, **c**, **f**, and **g** were misclassified when features were extracted by both networks, and images **d** and **h** was only misclassified when features were extracted by ResNet.

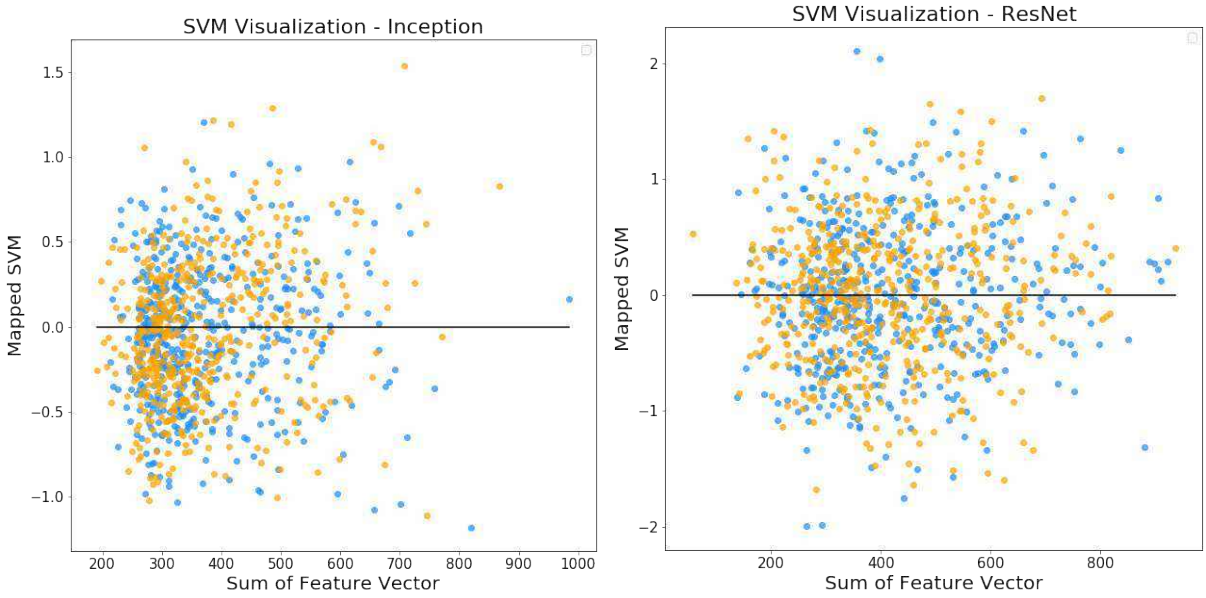
not make any distinction between the two datasets particularly clear (remember that some of the images in Figure 4.8 were successfully classified). It seems that several of the SUN scenes have people in them and are often of indoor locations that probably share many attributes with images in the ImageNet dataset. Additionally, ImageNet images do not seem to lack the scene like qualities present in SUN (Figures 4.8e, 4.8f, 4.8g, and 4.8h.

Table 4.5: This table shows the number of ImageNet and SUN images, on average (with the standard deviation in parentheses), that are confused by the linear classifier for Inception and ResNet. The Inception column is the number of images that were misclassified when using Inception features. The ResNet column is the number of images that were misclassified using ResNet features. The Intersection column lists the number of images that were misclassified by both networks.

	Inception	Intersection	ResNet	Total
ImageNet	89.2 (9.0)	68.7 (9.0)	115.1 (9.6)	135.5 (9.8)
SUN	96.8 (9.8)	37.2 (5.7)	60.7 (6.7)	120.3 (9.9)
Total	186.0 (15.3)	106.0 (12.1)	175.8 (11.8)	255.8 (15.1)

Table 4.5 shows the average number of misclassified features from both networks and the number of misclassifications per class when the images come from ImageNet and the SUN dataset. $1{,}000$ images from both datasets were used to evaluate the linear model, so the per-image comparison is valid. As stated before, the SUN dataset was the most challenging dataset to separate from ImageNet features. This difficulty is due to the similarity of the two datasets. SUN and ImageNet images seem to be misclassified at a pretty similar rate overall. A majority of the misclassified ImageNet feature vectors came from the same input image for both Inception and ResNet. It is likely that these images are similar to the scenes found in SUN meaning both networks produce similar features for these images thus making them similarly hard to distinguish from SUN features. The total number of images misclassified by both networks is higher than the images misclassified by each network independently. This is an interesting variation from previous results because it shows that a majority of Inception features that are hard to classify are challenging with ResNet features as well where as some of the other experiments showed ResNet features tend to be easier to classify. This could mean that both nets are extracting meaningful features from both datasets and because the datasets are similar they are hard to distinguish.

One area of interest shown in Table 4.5 is the images that are exclusively misclassified by Inception and those that are exclusively misclassified by ResNet. In particular, it appears that SUN features extracted by Inception are more likely to be confused with ImageNet features, while, the opposite is true for ResNet, ImageNet features extracted by ResNet are more likely to be confused with SUN. This could be related to Inception having slightly higher performance on ImageNet. If Inception extracts slightly better features from ImageNet it seems reasonable that those features would be more distinct, and SUN features that are misclassified are likely to have the ImageNet level of specialization. Alternatively, ResNet might not be as descriptive of ImageNet, making those features less distinct, but, by being less specialized, it could extract more unique features from the new but similar SUN features. It is surprising that ResNet features from ImageNet, the data the model was trained for, are harder to classify



(a) Inception **(b)** ResNet

Figure 4.9: As a sanity check the separation of two partitions of the validation dataset is measured using the same methods. Fortunately the two partitions are not separable, which helps confirm the validity of the prior results.

4.4 Sanity Check

As a sanity check Figure 4.9 shows the separation of two random partitions of the ImageNet validations dataset. It was of some concern that with such a high dimensional dataset it is possible that any partition of the data could lead to linear separability. However, fortunately it appears that the features from two different validation datasets are not separable. This result helps verify the validity of the previous results, showing that the separability in feature space of the out-of-domain datasets does hold. This raises some interesting questions about what the features mean. The networks seem to know what dataset they were trained for, so how descriptive could these features be for a different domain? Does the separability mean that the features are more or less descriptive?

Chapter 5

Comparison of Out-of-Domain Feature Mappings

It is surprising that all of the out-of-domain datasets covered in Chapter 4 have feature vectors that are so easily separated from the ImageNet features. Both ResNet and Inception, which are architecturally very different networks, seem to be able to recognize when an input is not from the domain that the network was originally trained for. The ease of separating domains in feature space raises some interesting questions about the feature space itself. In particular, one of the driving concepts behind transfer learning is that a well-trained network should have weights that extract meaningful information even when applied to a new domain. However, the fact that the out-of-domain feature vectors are so distinct from the in-domain feature vectors does raise some concern. If the networks are so descriptive it is surprising that a domain as similar as the SUN dataset would produce such distinctly different features. It is possible, though, that the features are descriptive, and the reason that they fall so far from each other is because the valuable features of the domains are very different. So as an additional experiment, to better understand how descriptive the features are, the features of each of the out-of-domain datasets are compared in this chapter.

The benefit of comparing the feature vectors of the out-of-domain datasets is it will help bring to light some information about how descriptive the features of the ImageNet trained networks are. If the features from the out-of-domain datasets cannot be easily distinguished then that would provide evidence that the features really are not that descriptive. The out-of-domain datasets, with possibly the exception of the art and anime datasets, are very different, so if the networks tend to map images from the different datasets to feature vectors that cannot be easily distinguished then it would imply that the features themselves are not that descriptive. However, if the datasets are easily separable in feature space that would provide evidence that the features are at least descriptive enough to differentiate between the datasets.

The following sections discuss out-of-domain feature vectors and how they compare to each other. Section 5.1 discusses the finding depth data and depth features seem to fall far from the other

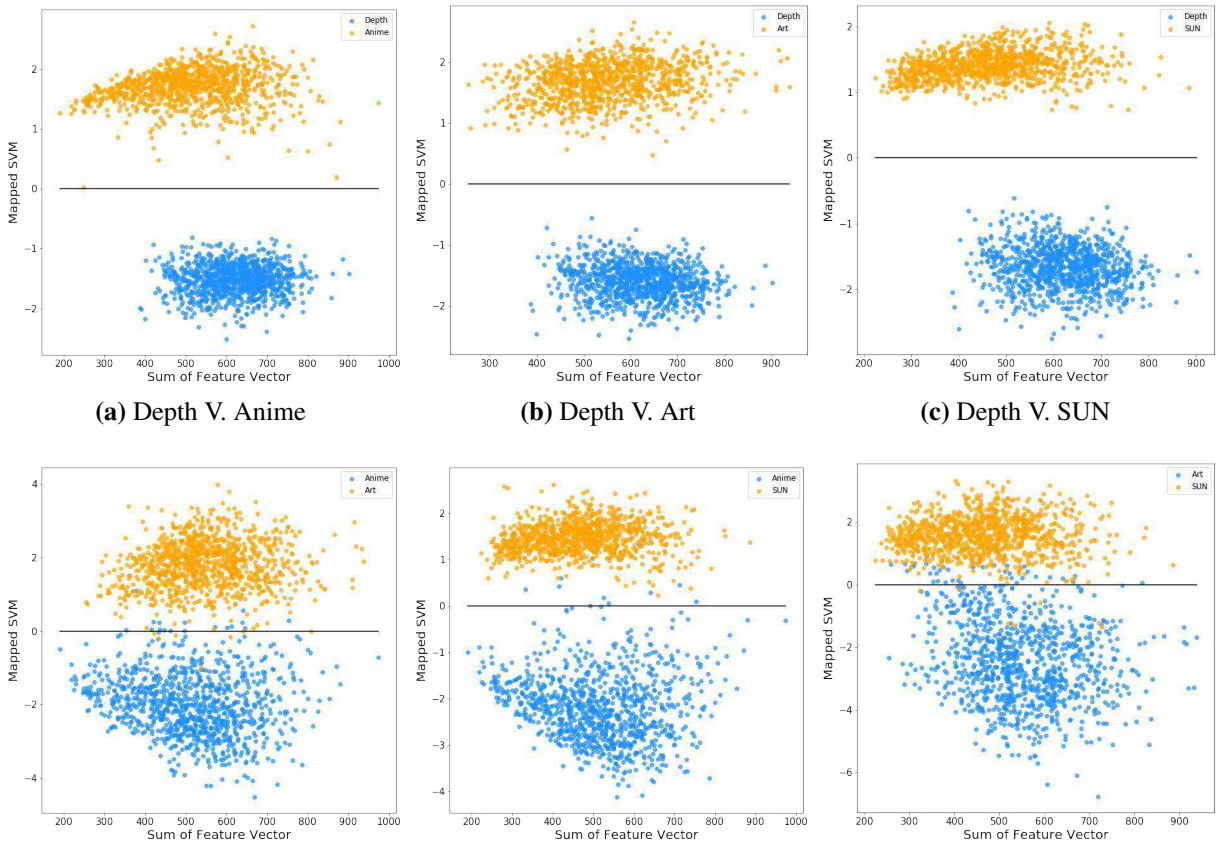
out-of-domain datasets. Section 5.2 looks into how art and anime features tend to differ and looks into what kinds of image characteristics may be the cause of some confusions when distinguishing between datasets. Section 5.3 looks at SUN and anime features, how distinguishable they are, and what images are being confused. Finally, Section 5.4 compares features from the art and SUN datasets and looks at why some images may be confusing.

Table 5.1: This Table shows the performance metrics showing the separability of each of the out-of-domain datasets: Depth, Anime, Art, and SUN when the features are extracted using ImageNet trained Inception and ImageNet trained ResNet. The order of the datasets listed in column one does not matter, it is just important to show the performance metrics for every combination of datasets.

Network	$\text{Domain}_1/\text{Domain}_2$	AUROC	AUPR_In	AUPR_Out	Accuracy (%)
Inception	Depth/Anime	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	99.99 (0.02)
	Depth/Art	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	100.0 (0.00)
	Depth/SUN	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	100.0 (0.00)
	Anime/Art	0.999 (0.000)	0.999 (0.000)	0.999 (0.000)	98.86 (0.18)
	Anime/SUN	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	99.41 (0.19)
	Art/SUN	0.997 (0.001)	0.997 (0.001)	0.997 (0.001)	97.42 (0.38)
ResNet	Depth/Anime	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	100.0 (0.00)
	Depth/Art	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	100.0 (0.00)
	Depth/SUN	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	100.0 (0.00)
	Anime/Art	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	99.53 (0.12)
	Anime/SUN	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	99.84 (0.08)
	Art/SUN	0.999 (0.000)	0.999 (0.000)	0.999 (0.000)	98.39 (0.26)

5.1 Depth Data and the Other Out-of-Domain Datasets

The depth dataset appears to be a somewhat special case in terms of the features the ImageNet trained networks seem to produce. Based on the performance metrics shown in Table 5.1 the depth features are very different from the features of the other out-of-domain datasets when features are extracted by ImageNet trained versions of both networks. In particular, the accuracy shows that both networks produce features that, on average, allow for depth to be perfectly separated from other datasets with only a rare confusion between depth and anime with Inception. The visualizations of the separations for Inception are shown in Figure 5.1 and for ResNet in 5.2. For



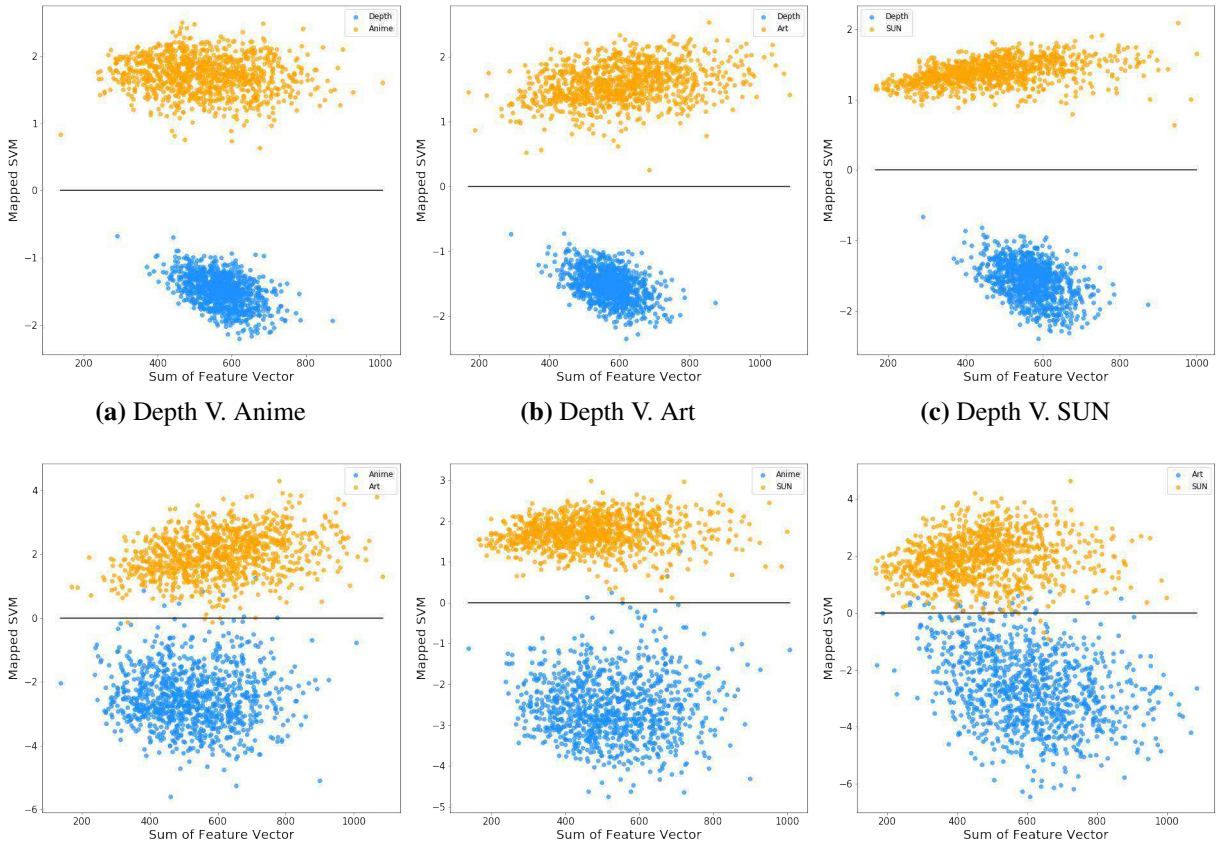
(a) Depth V. Anime **(b)** Depth V. Art **(c)** Depth V. SUN

(d) Anime V. Art **(e)** Anime V. SUN **(f)** Art V. SUN

Figure 5.1: These plots show the separability of each of the out-of-domain datasets when their features were extracted by ImageNet trained Inception. The features are visualized by mapping the features to one dimension by doing the dot product of the feature vector and the normal vector that defines the separating hyperplane in feature space. The additional dimension is the total sum of activation and is included for visualization purposes. Every combination of datasets is presented.

both networks, the depth features seem to map reasonably far away from the other out-of-domain datasets and there really is not any overlap.

The reason for the large degree of separation between the depth dataset and the other out-of-domain datasets could be related to how different the depth data are from the other datasets. The depth images are grayscale, tend to have a flat background, and the foregrounds are mostly centered in the middle of the frame. However, the fact that the depth data are so easily separable is still surprising because this means that the ImageNet trained networks can at least separate the most basic out-of-domain dataset from the other datasets. Based on the results in Chapter 4 it is about as easy to separate the depth data from the other out-of-domain datasets as it is to separate



(a) Depth V. Anime **(b)** Depth V. Art **(c)** Depth V. SUN

(d) Anime V. Art **(e)** Anime V. SUN **(f)** Art V. SUN

Figure 5.2: These plots show the separability of each of the out-of-domain datasets when their features were extracted by ImageNet trained ResNet. The features are visualized by mapping the features to one dimension by doing the dot product of the feature vector and the normal vector that defines the separating hyperplane in feature space. The additional dimension is the total sum of activation and is included for visualization purposes. Every combination of datasets is presented.

depth data from the ImageNet data. It is possible that depth data varies so little and is so different from ImageNet that the networks have trouble identifying anything significant and thus the depth data end up with very different feature making them easier to identify.

Because the depth data is perfectly separable from all other datasets, there very few examples of images that are confused between the two datasets. In particular, across 30 runs there was one depth image confused for anime and one anime image confused for depth (the same images were confused in different runs). Upon further investigation, the anime image that fails is blank and the depth image shows a person's hands with their fingers together. There is a rare case in which Inception features for anime and depth data can become confused, but this occurrence is so rare

(approximately only six misclassifications across thirty model training runs with random partitions of the train and test set for each run) that it is reasonable to consider this classifier to be perfect. This shows that an ImageNet-trained model can, at least, recognize the difference between a very simple out-of-domain-dataset in comparison to other out-of-domain datasets.

5.2 Anime V. Art

The depth data is very different from the other out-of-domain datasets, so it is likely that the features would map far from each other. It would be expected that datasets with more similar features would be much harder to distinguish. For example, one would expect that the art and anime features would be difficult to distinguish because both datasets consist of artworks, just different styles of artwork. However, surprisingly art and anime features are still easily distinguishable. The performance metrics shown in Table 5.1 show that on average the art and anime features are very easily distinguishable with AUPR and AUROC scores close to one and over 98% accuracy for both network architectures. Figures 5.1 and 5.2 show that for both architectures there is very a distinct clustering of both datasets. These details provide evidence that ImageNet trained networks can extract reasonably descriptive features for images from different domains than that for which the network was trained. Otherwise, one would expect that the anime and art images would have more overlap in feature space.

With so few feature vectors being difficult to distinguish it may be useful to look into which images are being confused. Figure 5.3 shows examples from the anime class that were confused with art images and art images that were confused with anime images. Some of the confusions seem to make sense. There do appear to be some characteristics that could understandably cause features that are hard to distinguish. For example, many of the anime images that were confused with art were created in the style of a painting or in a style that deviates from many anime examples where colors are solid and there is very little shading. For example, Figures 5.3a and 5.3c both have style similar to that of the art images. One could speculate that Figure 5.3b was misclassified because it varies greatly from the average anime image. Some of the art images that were misclassified

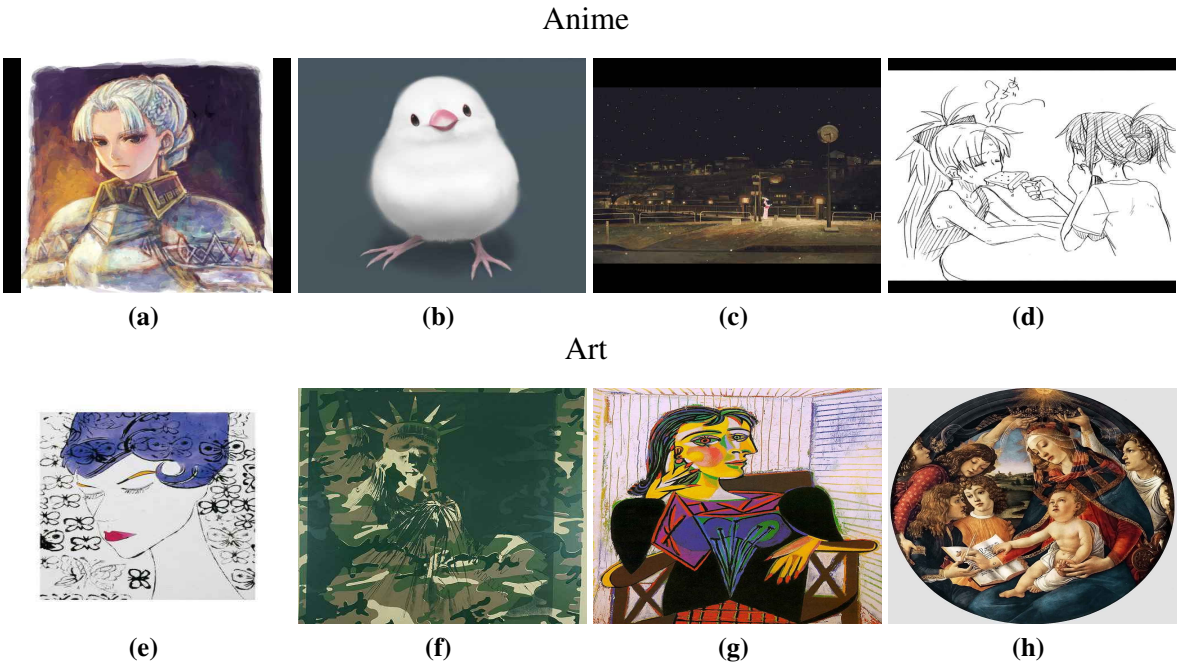


Figure 5.3: The top row shows some examples of anime images that were confused with art images and the bottom row shows some of the art images that were confused with anime images. It appears that the anime images that are confused with art occur because of a blending of styles (e.g. **a** and **c**) or images that vary significantly from much of the other examples in the anime dataset (e.g. **b**). Similar to the anime images, the art images that are confused seem to have more anime style features (e.g. **e**, **f**, **g**). Although, the confusion of some images seems reasonable, there are some images where the reason for confusing in hard to understand (e.g. **h**).

have characteristics similar to that of anime images (e.g. Figures 5.3e, 5.3f, and 5.3g). There are some images, such as Figures 5.3d and 5.3h, where it is difficult to speculate why the features may have been confused. Unfortunately, because much of this information is purely speculation, it is difficult to come to significant conclusions, but the presence of some some visible similarities between an image and the class it is being confused with provides evidence that the features do capture important information.

Table 5.2 shows how many art and anime images on average were misclassified per dataset as well as the images that were misclassified using Inception features, ResNet features, and the images that were misclassified using features from both networks. Note that there are $1{,}000$ examples from both datasets used to test the model, so using proportions is not necessary for these comparisons. The Inception features seem to be more easily misclassified than the ResNet features for these datasets. Additionally, it seems that a decent proportion of of the misclassified ResNet features

Table 5.2: This table shows the number of anime and art images, on average (with the standard deviation in parentheses), that are confused by the linear classifier for Inception and ResNet. The Inception column is the number of images that were misclassified when using Inception features. The ResNet column is the number of images that were misclassified using ResNet features. The Intersection column lists the number of images that were misclassified by both networks.

	Inception	Intersection	ResNet	Total
Anime	14.2 (4.1)	3.8 (2.0)	6.4 (2.6)	16.7 (4.5)
Art	7.4 (2.8)	1.3 (0.9)	3.5 (1.5)	9.6 (3.3)
Total	21.5 (4.3)	5.2 (2.3)	9.9 (2.9)	26.3 (4.8)

were also hard to classify using Inception features. Inception has slightly better performance on ImageNet, so maybe the more frequent misclassifications is because the features are slightly more ImageNet specified. Anime images are more easily mistaken for art images than art images for anime images. This is likely because several of the anime images do look like paintings, but the art images do not usually have anime qualities that are probably present in most of the anime images. About 27 images in total are misclassified by the linear models trained on the feature vectors from the from the neural networks.

5.3 Anime V. SUN

The anime and SUN datasets are not quite as similar as the anime and art dataset. If the trend that datasets tend to map to different locations in feature space continues then one would expect the SUN and anime data sets to be very easily distinguishable. However, just because the art and anime datasets were easily distinguishable does not mean that anime and SUN will be. Because each of the datasets are distinctly different, and the model weights were not trained for either of these datasets, it is important to measure. If two very visually different datasets are not easily distinguishable then it is likely that the reason for confusion is because the model weights cannot extract expressive enough features from the two datasets to allow for significant distinction.

As expected, the anime and SUN datasets are nearly perfectly distinguishable on average based on the performance metrics presented in Table 5.1. The linear models on both feature extractors are over 99% accurate. Figures 5.1 and 5.2 show that there are two distinct clusters in the two

datasets for both networks. The anime and SUN dataset seem to map farther apart than the anime and art datasets. Figure 5.4 shows the anime images that were confused with the SUN images and the single SUN image whose features were confused for an anime image. It seems as though a reasonable number of the anime images confused for SUN do contain scenes (Figures 5.4a, 5.4c, and 5.4d). This is interesting because it seems that Inception and ResNet, with ImageNet weights, can extract similar features from photographs and from drawings that are not particularly realistic. Figure 5.4b is so different from the standard anime images that its attributes are hard to recognize as anime attributes. The one SUN image that was confused with an anime image is shown in Figure 5.4e. This image was only mistaken using Inception features. ResNet features did not cause any SUN images to be confused for anime on this particular run. The presence of a cartoon character statue in the frame may be the reason for this misclassification, however this image does not share many other characteristics with anime drawings. The fact that the anime images that are confused often contains scenes provide evidence that the weights are not only descriptive but can possibly generalize across domains.

Table 5.3: This table shows the number of anime and SUN images, on average (with the standard deviation in parentheses), that are confused by the linear classifier for Inception and ResNet. The Inception column is the number of images that were misclassified when using Inception features. The ResNet column is the number of images that were misclassified using ResNet features. The Intersection column lists the number of images that were misclassified by both networks.

	Inception	Intersection	ResNet	Total
Anime	11.3 (3.3)	2.3 (1.4)	3.2 (1.4)	12.2 (3.7)
SUN	0.9 (0.9)	0.4 (0.5)	0.6 (0.8)	1.1 (1.1)
Total	12.2 (3.3)	2.6 (1.4)	3.8 (1.4)	13.3 (3.5)

Table 5.3 has the average number of misclassifications per dataset when comparing anime and SUN features. The linear models were compared using $1{,}000$ examples from each dataset, so these values can be compared as is. The number of images classified incorrectly using Inception features, classified incorrectly using ResNet features, and images classified incorrectly using features from both networks are presented in the table. The SUN features are very rarely confused with anime features on average, usually only having one misclassified image across both of the models.

Anime

(a) **(b)** **(c)** **(d)**

SUN

(e)

Figure 5.4: The first row shows the anime images that were confused with SUN images and the second row shows the one SUN image that was confused for anime. Interestingly it seems that a reasonable portion of the anime images that were confused with sun are scenes (e.g. **a**, **c**, and **d**). **b.** seems to be a confusing image to extract features from because it has been confused with both the art and SUN datasets and was confused by both models. **e.** has some cartoonish qualities which could explain why this image was confused with anime. **e.** was only misclassified by using Inception features.

Additionally, ResNet is much less likely to produce features that are harder to classify than Inception. For example, on average about 11 of the $1{,}000$ anime feature vectors extracted by Inception are misclassified. For ResNet it is only about 3 misclassified vectors for every $1{,}000$ feature vectors. Also, a majority of misclassified ResNet vectors are misclassified Inception vectors. It seems as though ResNet features tend to be more easily separable, and Inception tends to have trouble with the images ResNet has trouble with. On average only about 16 images total are misclassified between the two networks.

5.4 Art V. SUN

The art and SUN datasets were the hardest to separate from the ImageNet data, so they might be more closely clustered in feature space than the other datasets. Again, the comparison is important because the underlying data distribution of the art and SUN datasets is very different, so if the two

datasets are very difficult to distinguish in feature space then that would provide evidence that the feature extractors cannot effectively extract meaningful features from these datasets.

Comparing the art and SUN datasets produces similar results to the comparison of other out-of-domain datasets. For the most part the art and SUN features seem to cluster separately and are easy to separate using a simple linear classifier. However, art and SUN are slightly harder to distinguish in feature space. The performance metrics presented in Table 5.1 show that the classifier is near perfect on average with the AUROC and both AUPR metrics very close to 1.0 and an accuracy of over 97% using both Inception and ResNet architectures. Looking at Figure 5.1f it can be seen that the SUN and art dataset clusters seem to blend more than the other datasets although they are, for the most part, two separate clusters. Some of the images that were confused between the art and the SUN dataset provide interesting information about the models' ability to generalize and to extract meaningful features. Figure 5.5 shows a some art images that were classified as SUN and some SUN images that were classified as art in feature space. Interestingly, with a few exceptions, many of the art images that were classified as SUN are paintings or drawings of scenes. This is extremely interesting because it implies that the model weights generalize well by being able to produce similar features for painted scenes vs photographed scenes. Additionally, it implies that the model is able to extract meaningful features from both the datasets. Many of the SUN images that are confused with art images have some unusual lighting, coloring, and/or blurring that could cause them to be misclassified. Also, it seems that images that are not in focus could potentially be confused.

Table 5.4 details information on how often, on average, images from the art dataset are classified as SUN images and how often SUN images are classified as art images. $1{,}000$ images from each dataset was used for evaluation, so the number of misclassified images can be effectively compared. Contrary to the previous dataset comparisons, ResNet features are not significantly easier to classify correctly. SUN features from both ResNet and Inception are misclassified at about the same rate. However, ResNet features from the art dataset are much less frequently misclassified. As previously discussed, Inception has slightly better performance on ImageNet, so its

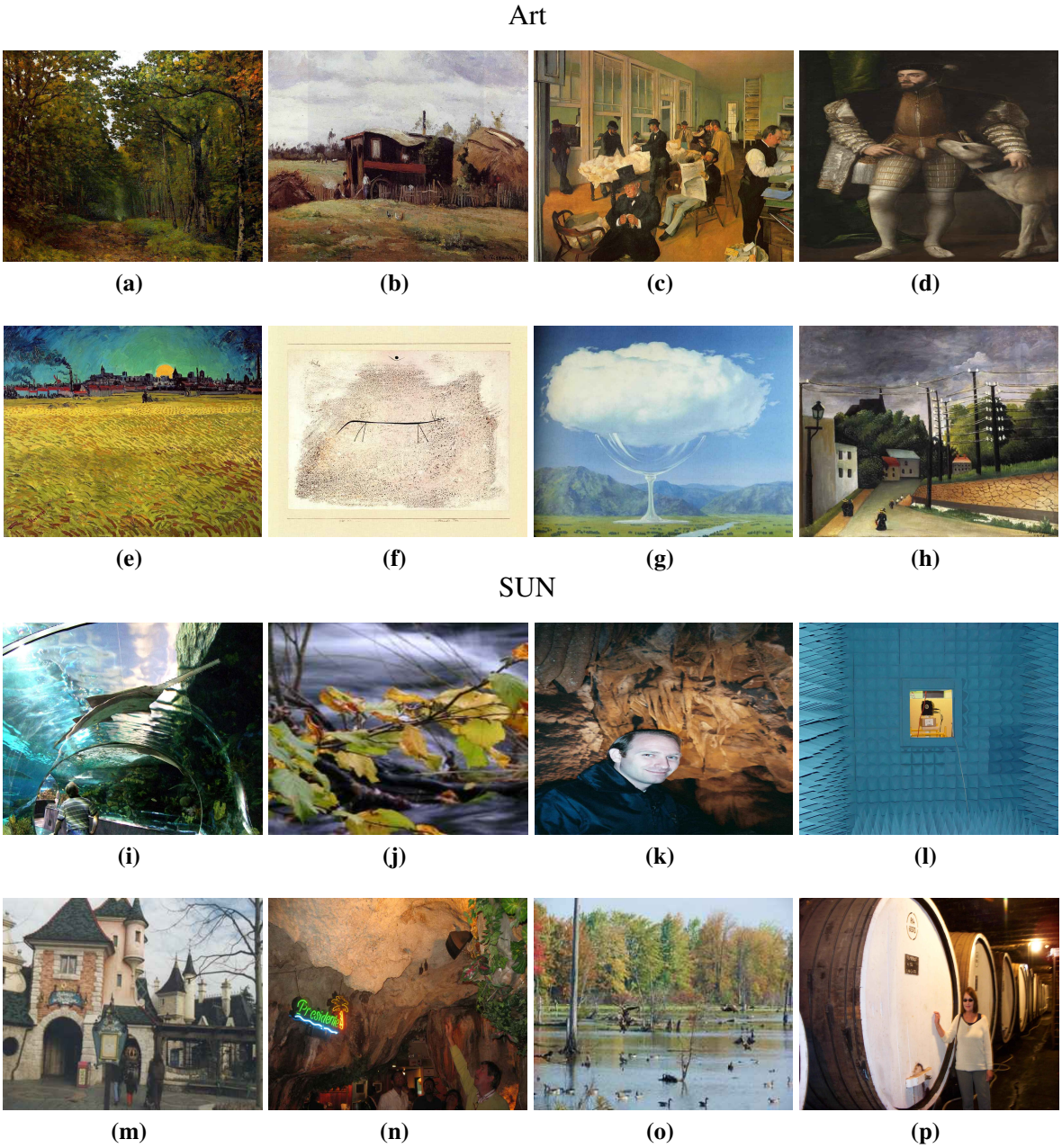


Figure 5.5: The first two rows show art images that were mistaken for SUN images and the remaining two rows are SUN images that were mistaken for art images. Interestingly, many of the art images that were mistaken for SUN are of scenes. There are a few exceptions such as **d.** which is hard to speculate on why this image was classified as SUN. Image **f.** appears pretty visually far from the average SUN image, but it may have been misclassified because it is pretty different from the average art image as well. Some of the SUN images that were confused with art seem to have some unusual lighting effects or colors that seem to have some artistic qualities to them (e.g. **i.**, **j.**, **m.**, **n.**, and **o.**). Others, such as **k.** and **p.** have some defocus blur that may lead to the inability to distinguish between the finer image characteristics that are not usually present in artworks. **l** is probably unusual for a sun image leading to confusion.

weights might not as easily generalize to these new datasets. It would make sense that both ResNet and Inception would have about the same error rate on SUN images because SUN images are the most similar to ImageNet and therefore, the features should be more descriptive of the dataset. ResNet weights may generalize better, making the art feature vectors more easy to separate with a linear classifier. Art images, for both networks, are harder to accurately classify, which could be a result of the art features being more different from the ImageNet features than the SUN features. On average, 49 Inception feature vectors are misclassified and 31 ResNet feature vectors are misclassified.

Table 5.4: This table shows the number of art and SUN images, on average (with the standard deviation in parentheses), that are confused by the linear classifier for Inception and ResNet. The Inception column is the number of images that were misclassified when using Inception features. The ResNet column is the number of images that were misclassified using ResNet features. The Intersection column lists the number of images that were misclassified by both networks.

	Inception	Intersection	ResNet	Total
Art	37.4 (6.0)	11.9 (2.8)	19.8 (3.9)	45.4 (6.7)
SUN	12.5 (3.1)	4.2 (1.7)	11.5 (3.2)	19.9 (4.2)
Total	50.0 (6.6)	16.0 (2.8)	31.3 (5.0)	65.2 (8.0)

Chapter 6

Fine-Tuning and the SUN Features

As a final experiment the ImageNet weights for Inception and ResNet are tuned for the SUN dataset. This experiment will provide information about how the fine-tuning of the weights changes the feature vectors. The specifications of how the models were trained is provided in Section 3.4.2. In particular, there are two major things that this section is looking to explore. The first is related to how the tuned SUN features compare to the ImageNet features prior to tuning the network. This comparison provides information about the quality of features. For example, if the tuned features tend to be closer to the ImageNet features this means that the pre-trained weights might not be as descriptive as initially thought. Essentially meaning that any feature vectors that fall away from the ImageNet features are not descriptive. If the tuned features move away from the ImageNet features it implies that the ImageNet trained feature extractors were originally descriptive and that is why the two dataset have mapped to different clusters in feature space. The second thing to look at is if there is any distinction between the SUN images that were classified correctly in comparison to the SUN images that were not classified correctly. For example, if the SUN images that were classified correctly after fine-tuning seem to trend towards the ImageNet cluster, it could provide evidence that when SUN features are closer to ImageNet features the model can more accurately make predictions. The aim of this section is to try to create an argument for the descriptiveness of high-level features extracted by ImageNet trained networks when applied to a new dataset.

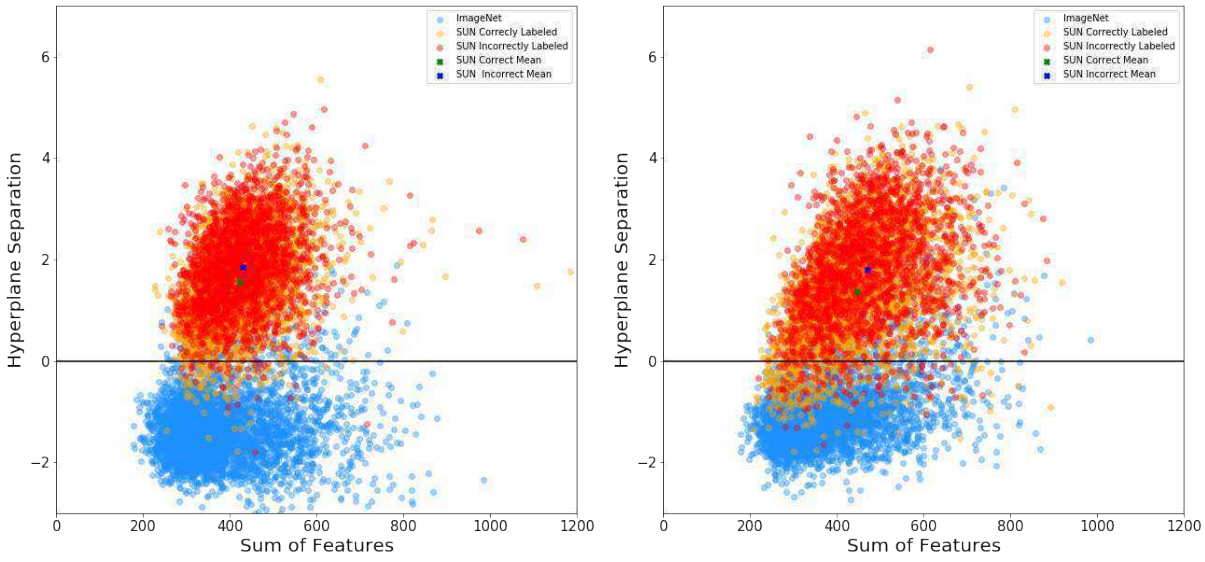
Section 6.1 explores how fine-tuning the ImageNet weights in the Inception model on the SUN dataset changes the high-level features extracted by the model. Section 6.2 explores how fine-tuning the ImageNet weights in the ResNet model on the SUN dataset changes the high-level features extracted by the model. In these experiments the classification accuracy of the fine-tuned models is not extremely important because the focus is to understand underlying trends in how the high-level features change when the model is tuned for a different dataset. By tuning the models

they learn to better classify the SUN dataset, but hyperparameters have not been tuned to produce the best model possible.

6.1 Inception

Table 6.1: This table contains the performance metrics of the SUN dataset compared to the ImageNet dataset when the features are extracted by Inception before and after the Inception weights have been tuned to classify the SUN dataset. Note that the ImageNet features are extracted by the ImageNet weights for both comparisons as to compare how the SUN features change relative to the ImageNet features.

	AUROC	AUPR_In	AUPR_Out	Accuracy (%)
Before Tuning	0.966 (0.003)	0.970 (0.003)	0.962 (0.004)	90.71 (0.55)
After Tuning	0.991 (0.001)	0.992 (0.001)	0.990 (0.002)	95.79 (0.40)



(a) Tuned Model **(b)** Before Tuning

Figure 6.1: This figure shows the separability of the SUN data when the Inception model weights have been tuned to the new dataset (**a**) and before the weights were tuned (**b**). The ImageNet feature vectors are extracted by ImageNet weights in both plots because the comparison to the original ImageNet features before and after tuning to the SUN data is desired. Note that although the plots are on the same scale the dividing hyperplane could be drastically different making the ordering of points in the y-axis incomparable. The red points are SUN data that were incorrectly classified after tuning the model and the orange points were correctly classified. In **b** the classifier head was trained to classify the SUN data based on the outputs of the Inception feature extractor. The green point marks the average of the SUN data that was correctly classified and the blue point marks the average of the SUN data that was incorrectly classified.

So far Inception features have seemed to be less easily dividable than ResNet features when the extracted features come from datasets that are further from ImageNet. However, as the out-of-domain dataset gets closer to ImageNet, Inception seems to be better suited for feature extraction, at least when the task is dataset classification. As speculated previously, this could occur because Inception has slightly higher performance on ImageNet and, therefore, could be more specialized for ImageNet type features making images from very different datasets more challenging. By tuning Inception features for the SUN dataset it may be possible to better understand how specializing the network for a new dataset affects how the feature vectors of in- and out-of-domain datasets map in feature space.

The most notable result is that, by fine-tuning the model using the SUN data, the feature vectors of SUN and ImageNet become significantly easier to separate. Figure 6.1 shows the separability of the SUN features compared to the ImageNet features before and after tuning the model weights. Figure 6.1a shows how separable the datasets are after tuning and Figure 6.1b shows how separable the datasets are prior to tuning the model to the SUN dataset An important thing to note is, although the plots are on the same scale, the dividing hyperplane that defines the y-axis is not likely to be the same for the tuned and un-tuned weights. Points cannot be directly compared in the y dimension because the y-axis is likely oriented differently in feature space after the model is tuned. Overall, after tuning, the two datasets appear to map into more distinct clusters. This could mean that as an ImageNet trained model is tuned with new data to a new task the feature vectors produced by the new data tend to move further away from the original ImageNet feature vectors. This provides evidence that the reason for dataset separation in feature space is because the ImageNet trained networks are capable of describing characteristics of a wide variety of datasets and thus will map them to different locations within the feature space. However, considering how similar the SUN dataset is to the ImageNet dataset it is surprising that features become more separate with training. The SUN feature cluster appears to be slightly more tightly clustered in both dimensions after tuning. This backs up the belief that an out-of-domain dataset tends to have a higher overall activation because the total activations of the tuned model are more tightly clustered and on average

lower. The center of the cluster in the y dimension is about the same, but there appears to be fewer feature vectors that are at the extremes of the cluster.

Table 6.1 lists the performance metrics of dataset separability of SUN and ImageNet in features space before and after the weights were tuned for the SUN dataset. Again, the ImageNet feature vectors were extracted by ImageNet weights for both comparisons because the goal is to see how much the SUN features change relative to the original ImageNet features. The performance metrics show that, on average, the tuned SUN features are easier to distinguish from ImageNet than the SUN features prior to tuning. The AUROC and AUPR scores increase from around 0.97 to 0.99 and the accuracy increases by over 5%. Considering that the accuracy before fine tuning was over 90% a five percent increase is significant.

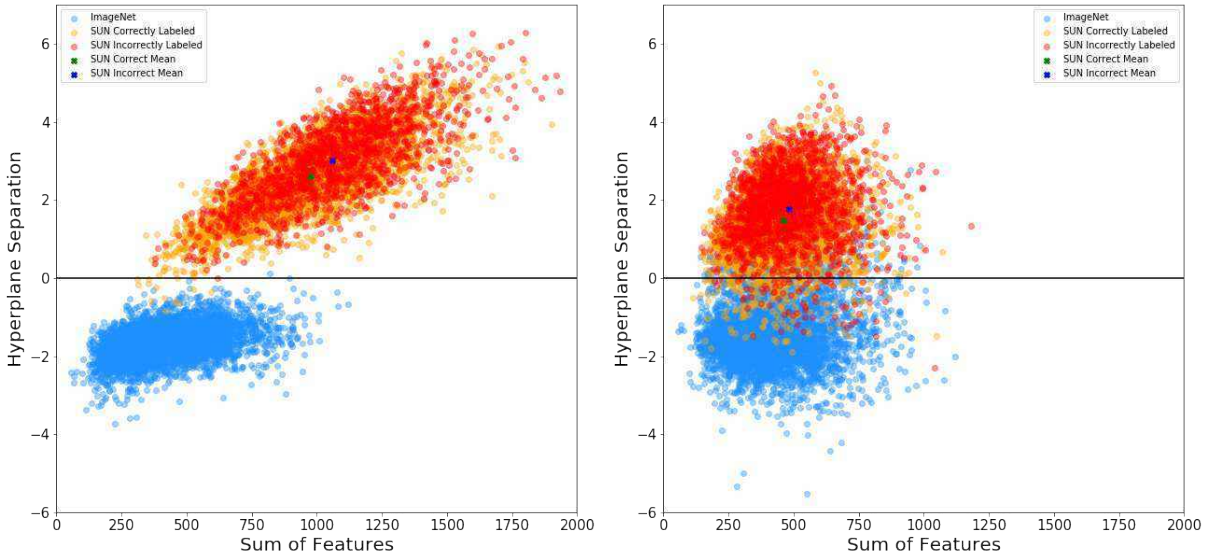
The orange points in plots 6.1 are the SUN images that were correctly classified and the red points are the ones that were incorrectly classified. In Figure 6.1a the entire model has been tuned to the SUN dataset, and in Figure 6.1b the feature extractor weights are from the ImageNet-trained model and only the classifier trained on the SUN dataset. The blue points in Figure 6.1 are the average of the incorrectly classified SUN images and the green points are the average of the correctly classified SUN Images for the tuned model. These two points are pretty close, so it seems that there is not much of a difference between the feature vectors of correctly and incorrectly classified data. It is somewhat interesting that the correctly classified average tends to be slightly closer to the ImageNet cluster, but it would require deeper evaluation to determine if this detail is significant.

Overall, by tuning the Inception weights for SUN, the SUN feature vectors tend to fall farther from the original ImageNet vectors. This is a surprising result, especially with SUN data, because they are similar to ImageNet data. It seems to provide evidence that the reason for the separability to begin with is because the features are descriptive enough of the two datasets that they do not overlap very much. By training the weights they become more descriptive of the SUN dataset and thus increase the gap between the SUN and ImageNet datasets in feature space.

6.2 ResNet

Table 6.2: This table contains the performance metrics of the SUN dataset compared to the ImageNet dataset when the features are extracted by ResNet before and after the ResNet weights have been tuned to classify the SUN dataset. Note that the ImageNet features are extracted by the ImageNet weights for both comparisons as to compare how the SUN features change relative to the ImageNet features.

	AUROC	AUPR_In	AUPR_Out	Accuracy (%)
Before Tuning	0.965 (0.003)	0.970 (0.003)	0.959 (0.005)	91.24 (0.67)
After Tuning	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	99.84 (0.07)



(a) Tuned Model

(b) Before Tuning

Figure 6.2: This figure shows the separability of the SUN data when the ResNet weights have been tuned to the new dataset (**a**) and before the weights were tuned (**b**). The ImageNet feature vectors are extracted by ImageNet weights in both plots because the comparison to the original ImageNet features before and after training is desired. Note that although the plots are on the same scale the dividing hyperplane could be drastically different making the ordering of points in the y-plane incomparable. The red points are SUN data that were incorrectly classified after tuning the model and the orange points were correctly classified. In **b** only the classifier head was trained to classify the SUN data from the outputs of the ResNet trained on ImageNet feature extractor. The green point marks the average of the SUN data that was correctly classified and the blue point marks the average of the SUN data that was incorrectly classified.

Based on the results from previous sections, ResNet seems to be able to produce out-of-domain-features that are more easily distinguishable from the ImageNet features than Inception. Especially

with anime and depth data, which are the furthest from ImageNet, it seems that ResNet may be able to better extract distinguishable features. The caveat is this does not necessarily mean that the features are better. It is possible that the features extracted for these datasets are particularly meaningless making them very different from the ImageNet dataset, but maybe the performance is better because the features are descriptive of the datasets and are thus easier to distinguish. Tuning ResNet weights for the SUN dataset could help clarify if the reason for out-of-domain features to be easily separable is a result of the ImageNet-trained ResNet weights being general and able to extract meaningful features from different datasets.

Similar to Inception, tuning the ResNet weights for the SUN dataset makes it easier to separate the SUN dataset from the ImageNet dataset. Figure 6.2 shows the separability of the SUN and ImageNet datasets before (Figure 6.2b) and after (Figure 6.2a) the ResNet weights have been fine-tuned for the SUN data. The difference in feature vectors before and after training is much more significant with ResNet. Without tuning, the SUN and ImageNet dataset do not overlap too much, but if all the points were the same color it would be difficult to tell that there are two different clusters. However, after the weights have been tuned, there are very obviously two different clusters. There is a significant change to the shape of the SUN features. In the x dimension, the total sum of feature activations became more spread out and are larger on average after the model has been fine-tuned. This is interesting because it is contrary to the prior assumption that out-of-domain images tend to produce higher overall feature activations. The total variation of the SUN features in the y dimension does not seem to have change much, but the cluster itself is further away from the ImageNet features. It is surprising that the tuned ResNet weights seem to have a pretty different trend compared to the Inception weights. It would be interesting to further try to understand what is changing.

Table 6.2 lists the performance metrics of the separability of the SUN dataset when compared to the ImageNet dataset when features are extracted using ResNet before and after the model weights have been tuned on SUN data. The ImageNet features are only being extracted by ImageNet weights because the aim is to compare the pre- and post-trained SUN features to the original

ImageNet features. The performance metrics show that, on average, it is much easier to separate the SUN and ImageNet features after the model has been fine-tuned to the SUN data. Interestingly, the difference is much greater for ResNet than for Inception. The AUPR and AUROC scores all increased from about 0.965 to a perfect score of 1.0. This means that by fine-tuning the model the SUN and ImageNet features become basically perfectly separable in Feature space. The accuracy increases from about 91% to nearly 100%. These results seem to provide evidence that the ImageNet model weights are initially descriptive of the SUN dataset because they map to a different area of feature space, and by fine-tuning the model it becomes more capable of extracting meaningful features from the SUN data moving the cluster further from the ImageNet features.

The blue points in Figure 6.2 represents the mean of incorrectly classified SUN images and the green points represent the mean of correctly classified SUN images. Note that the classification head was trained using the ImageNet weights to produce SUN classifications in Figure 6.2b. Similar to Inception, the means of correct and incorrect classifications are pretty close together, providing some evidence that there is not a relationship between having features close to ImageNet features and producing a correct classification. However, like Inception, it is interesting that the correctly classified Images have mean features closer to ImageNet features. It would require a more complete analysis to understand if there is anything significant here.

Overall it seems that tuning ResNet to SUN makes SUN and ImageNet more easily separable. It is hard to say if this would happen for another dataset and further experimentation with other datasets would be needed to make a more concrete claim about how tuning a model changes the high-level features. This seems to imply that the weights more specialized for SUN produce feature vectors that are further from ImageNet features relative to the SUN features prior to the model being tuned. This result is surprising because the SUN and ImageNet datasets have very similar features, so one would not expect tuning an ImageNet model for SUN to result in the datasets being further apart in feature space, but it seems that by tuning the model becomes more descriptive of SUN by moving away from the ImageNet weights.

Chapter 7

Conclusions

This chapter will wrap up the previous experiments and open up a discussion about how this work could be taken further. Section 7.1 opens up a discussion about the experiments and what important details should be taken away. Section 7.2 will discuss some areas that may be interesting to explore in the future.

7.1 Discussion

The main focus of this work is to present the new finding that datasets tend to be linearly separable from each other in feature space when the features are extracted by an ImageNet trained model and to speculate on what this separation suggests about the generality and expressiveness of model weights.

First and foremost, these results show that ImageNet trained models tend to map classes into clusters in feature space. The fact that there appears to be an across-class pattern is an interesting result in itself. Additionally, the results show that these clusters do not tend to overlap very much and are separable with relatively high accuracy using a simple linear classifier. This finding could have big implications in the computer vision world because it raises some interesting questions about how descriptive features actually are when models are applied to a new dataset. It also provides a way to define a domain. Once a set of data becomes easily separable from another set of data in feature space then it is likely part of a different domain. Beyond that, the fact the out-of-domain datasets are easily separable in feature space seems to show that ImageNet trained weights are capable of recognizing differences in other domains.

Although, as it is, the linear separation in feature space would not make a great out-of-domain recognizer because it requires examples of out-of-domain data. It shows that there could be some characteristics of the in-domain high-level features that cause them to lend themselves to a poten-

tially successful out-of-domain detector. This would require understanding the underlying pattern that exists in the in-domain features, which could be possible with more analysis.

Given the highly separable datasets in ImageNet trained features spaces it becomes interesting to speculate on why this separation seems to occur. Unfortunately, these results tend to be more qualitative, so speculation is about the extent of analysis. By finding that the out-of-domain weights are separable from ImageNet and from each other, it provides evidence that the ImageNet weights maybe reasonable descriptive of out-of-domain datasets. It is possible that the depth and anime data only happen to be separable because they are so different from ImageNet. However, because Section 5.4 showed that art features commonly mistaken for SUN tend to have scenes and SUN features mistaken for art have some unusual lighting and color effects making them look more artistic suggests that the ImageNet trained weights are, in fact, reasonably descriptive of these two datasets.

The final experiment in Chapter 6 showed that by tuning weights the out-of-domain features tend to move further away from ImageNet features even if the dataset is similar. This seems to provide evidence that the reason datasets are easily separable when models use ImageNet weights is because the weights are descriptive of that dataset. By fine-tuning the weights become more descriptive and thus the feature vectors move farther away from the ImageNet feature vectors. To some degree, this result is concerning, because visually, SUN and ImageNet are very similar. Similar enough that one would expect them to produce similar feature vectors. The fact that these datasets tend to produce different enough features in feature space to be easily classified is surprising.

7.2 Future Work

It would be interesting to try to take this finding and build upon it to create a general out-of-domain detector. The linear classifiers trained in this work are effective at separating two known datasets, but would likely not work for other out-of-domain datasets. The fact that there appears to

be a pattern in in-domain features it seems like it would be possible to harness this information to make a general detector.

The separability of in- and out-of-domain datasets shown in Chapters 4 and 5 make it seem like an ImageNet trained feature extractor paired with SVMs could act as an effective pre-trained gating model for a Mixture-of-Experts model. A brief overview of the Mixture of Experts concept was presented in Section 2.6. It may be more effective to train a gating network, but it seems as though one could use an ImageNet feature extractor to place an input into a particular category, and the expert at that category could make a classification based on the input weights.

Chapter 5 showed that each out-of-domain dataset tends to map separately in feature space. It would be interesting to look further into where these datasets fall relative to ImageNet in feature space. Using the normal to the decision hyperplane, it would be possible to determine the relative direction of the in-domain and each out-of-domain datasets relative to each other. This might be an interesting way to measure dataset similarity in terms of what the model is trained to recognize. Additionally, it could show that datasets that look very different to the human eye may fall close to each other in feature space.

Finally, these experiments were confined to two networks and only one in-domain dataset. By using different and more current models as well as different datasets it would be possible to test whether the trends shown here generalize beyond ResNet and Inception.

Bibliography

[1] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. *CoRR*, abs/1603.05027, 2016.

[3] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *CoRR*, abs/1610.02136, 2016.

[4] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *International Conference on Learning Representations*, 2018.

[5] Shiyu Liang, Yixuan Li, and R. Srikant. Principled detection of out-of-distribution examples in neural networks. *CoRR*, abs/1706.02690, 2017.

[6] Lei Shu, Hu Xu, and Bing Liu. DOC: deep open classification of text documents. *CoRR*, abs/1709.08716, 2017.

[7] Abhijit Bendale and Terrance Boult. Towards open set deep networks. *arXiv preprint arXiv:1511.06233*.

[8] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[9] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks, 2018.

[10] Gabi Shalev, Yossi Adi, and Joseph Keshet. Out-of-distribution detection using multiple semantic label representations. *Conference on Neural Information Processing Systems*, (3), 2019.

[11] Sarah M. Erfani, Sutharshan Rajasegarar, Shanika Karunasekera, and Christopher Leckie. High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning. *Pattern Recogn.*, 58(C):121–134, October 2016.

[12] Pramuditha Perera and Vishal M. Patel. Learning deep features for one-class classification. *CoRR*, abs/1801.05365, 2018.

[13] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[14] Isaac Wang, Pradyumna Narayana, Dhruva Patil, Gururaj Mulay, Rahul Bangar, Bruce Draper, Ross Beveridge, and Jaime Ruiz. EGGNOG: A continuous, multi-modal data set of naturally occurring gestures with ground truth labels. In *12th IEEE International Conference on Automatic Face & Gesture Recognition*, 2017.

[15] Anonymous, Danbooru community, Gwern Branwen, and Aaron Gokaslan. Danbooru2018: A large-scale crowdsourced and tagged anime illustration dataset. https://www.gwern.net/Danbooru2018, January 2019. 2019-05-28.

[16] Icaro. Best artworks of all time: Collection of paintings of the 50 most influential artists of all time, 2019.

[17] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3485 – 3492, June 2010.

[18] A. L. Hodgkin and A.F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, 117:500–544, 1952.

[19] A. Žilinskas. *Interfaces*, 36(6):613–615, 2006.

[20] G. Cybenko. Approximation by superpositions of sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 12 1989.

[21] 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.

[22] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 12 1943.

[23] Donald Olding Hebb. *The organization of behavior: a neuropsychological theory*. J. Wiley; Chapman & Hall, 1949.

[24] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386, 1958.

[25] Paul J. Werbos. Applications of advances in nonlinear sensitivity analysis. In R. F. Drenick and F. Kozin, editors, *System Modeling and Optimization*, pages 762–770, Berlin, Heidelberg, 1982. Springer Berlin Heidelberg.

[26] Alekseĭ Grigorevich Ivakhnenko, Valentin Grigorevich Lapa, and R. N. Mcdonough. Cybernetics and forecasting techniques. 1967.

[27] Marvin Minsky and Seymour A Papert. *Perceptrons: An introduction to computational geometry*. MIT press, 2017.

[28] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[29] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. *CoRR*, abs/1808.01974, 2018.

[30] Ryne Roady, Tyler L. Hayes, Ronald Kemker, Ayesha Gonzales, and Christopher Kanan. Are out-of-distribution detection methods effective on large-scale datasets?, 2019.

[31] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, Oct 2010.

[32] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, and T. E. Boult. Toward open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1757–1772, July 2013.

[33] Bernhard Schölkopf, Robert Williamson, Alex Smola, John Shawe-Taylor, and John Platt. Support vector method for novelty detection. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, page 582–588, Cambridge, MA, USA, 1999. MIT Press.

[34] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *CoRR*, abs/1412.1897, 2014.

[35] Edwin M Knorr and Raymond T Ng. Algorithms for mining distance-based outliers in large datasets. In *VLDB*, volume 98, pages 392–403. Citeseer, 1998.

[36] Charu C Aggarwal and Philip S Yu. Outlier detection for high dimensional data. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pages 37–46, 2001.

[37] Fabrizio Angiulli and Clara Pizzuti. Fast outlier detection in high dimensional spaces. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 15–27. Springer, 2002.

[38] Stephen D Bay and Mark Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 29–38, 2003.

[39] Wen Qin and Jilin Qu. Vod: A novel outlier detection algorithm based on voronoi diagram. In *2010 WASE International Conference on Information Engineering*, volume 2, pages 40–42. IEEE, 2010.

[40] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[41] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[42] Dan Hendrycks, Mantas Mazeika, and Thomas G. Dietterich. Deep anomaly detection with outlier exposure. *CoRR*, abs/1812.04606, 2018.

[43] Poojan Oza and Vishal M. Patel. Deep cnn-based multi-task learning for open-set recognition. *CoRR*, abs/1903.03161, 2019.

[44] Akshay Raj Dhamija, Manuel Günther, and Terrance E. Boult. Reducing network agnostophobia. *CoRR*, abs/1811.04110, 2018.

[45] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[46] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

[47] Aäron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C. Cobo, Florian Stimberg, Norman Casagrande, Dominik Grewe, Seb Noury, Sander Dieleman, Erich Elsen, Nal Kalchbrenner, Heiga Zen, Alex Graves, Helen King, Tom Walters, Dan Belov, and Demis Hassabis. Parallel wavenet: Fast high-fidelity speech synthesis. *CoRR*, abs/1711.10433, 2017.

[48] Esteban G Tabak and Cristina V Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013.

[49] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know?, 2018.

[50] Lawrence Neal, Matthew Olson, Xiaoli Fern, Weng-Keen Wong, and Fuxin Li. Open set learning with counterfactual images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 613–628, 2018.

[51] Yang Yu, Wei-Yang Qu, Nan Li, and Zimin Guo. Open-category classification by adversarial sample generation. *CoRR*, abs/1705.08722, 2017.

[52] Jiajun Lu, Theerasit Issaranon, and David A. Forsyth. Safetynet: Detecting and rejecting adversarial examples robustly. *CoRR*, abs/1704.00103, 2017.

[53] Saeed Masoudnia and Reza Ebrahimpour. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42(2):275–293, Aug 2014.

[54] Nathan Silberman and Sergio Guadarrama. Tensorflow-slim image classification model library. , 2016.

[55] Alexey Kurakin. https://github.com/tensorflow/models/blob/b8bfb196fa05735a832c0257f2894f1706927407/research/adversarial_logit_pairing/datasets/imagenet_input.py, July 2018.

[56] Antonio Torralba, Alexei A Efros, et al. Unbiased look at dataset bias. In *CVPR*, volume 1, page 7. Citeseer, 2011.