TOWARDS MULTI-MODAL DATA CLASSIFICATION

by

Henry S. Ng

Bachelor of Arts - Computer Science
University of Nevada, Las Vegas
2018

A thesis submitted in partial fulfillment
of the requirements for the

Master of Science - Computer Science

Department of Computer Science
Howard R. Hughes College of Engineering
The Graduate College

University of Nevada, Las Vegas
May 2020

UNLV | GRADUATE COLLEGE

Thesis Approval

The Graduate College
The University of Nevada, Las Vegas

April 30, 2020

This thesis prepared by

Henry S. Ng

entitled

Towards Multi-Modal Data Classification

is approved in partial fulfillment of the requirements for the degree of

Master of Science – Computer Science
Department of Computer Science

Justin Zhan, Ph.D.
Examination Committee Chair

Kathryn Hausbeck Korgan, Ph.D.
Graduate College Dean

Ju-Yeon Jo, Ph.D.
Examination Committee Member

Fatma Nasoz, Ph.D.
Examination Committee Member

Ge Kan, Ph.D.
Graduate College Faculty Representative

Abstract

A feature fusion multi-modal neural network (MMN) is a network that combines different modalities at the feature level to perform a specific task. In this paper, we study the problem of training the fusion procedure for MMN. A recent study has found that training a multi-modal network that incorporates late fusion produces a network that has not learned the proper parameters for feature extraction. These late fusion models perform very well during training but fall short to its single modality counterpart when testing. We hypothesize that jointly trained MMN have weight space that is too large for effective training. To remedy this problem, we design a set of procedures that systematically narrow the search space so that the optimizer would only consider weights that are known to generalize well. As part of our systematic narrowing procedure, we enforce a weight constraint on the weights between the pre-fusion and fusion layers. Due to our given constraints on the network, modern methods cannot optimize our network without breaking our conditions. To remedy the problem, we create a simplex projection module that will be used after applying modern training frameworks. Our module will re-optimize our network such that the weight constraints are enforced. This new framework, which we call Projection Feature Mixture Model outperforms its single modality model as well as standard jointly trained MMN. In this paper, we provide a theoretical analysis to show advantages of utilizing MMN.

Acknowledgements

This thesis becomes a reality with the kind support and help of many individuals. I would like to extend my sincere thanks to all of them.

I would first like to express my gratitude for my adviser, Dr. Justin Zhan for his continual support on my growth as a person, both academically and mentally. He provided me with numerous opportunities, resources, and guidance throughout my master's career and I deeply appreciated it.

Secondly, I would like to express my gratitude for members of my committee, Dr. Jo Ju-Yeon, Dr. Fatma Nasoz, and Dr. Ge Kan for their support and dedicating their time on reviewing my thesis.

I would also like to express my gratitude to Aeren, Carter, Matt, Michael, Shen, and everyone that I worked with in the Big Data Hub. Everyone in the lab have provided valuable assistance and aided my growth as an individual.

Lastly, I would like to express my gratitude for my family and friends, their continuous support for me as a person and giving me motivation to finish what I have started.

Thank You.

Henry S. Ng

University of Nevada, Las Vegas

May 2020

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

To have an accurate perception about a task requires a strong understanding about everything in regards to it. Often when we assess a situation, we as individuals will take precautionary measures, perform a risk assessment, and review all possible outcomes. This is something humans inherently do even with simple tasks. Whenever we cross a road, our brains go through many different scenarios with the provided information. We will check the lights to see if it's our turn to cross. During this time our ears would listen to the speed of the upcoming traffic. We would scan our surroundings while proceeding to cross the street. This is only some of the things we naturally take into consideration. What goes in our mind naturally is multi-modal and it is something we would like to incorporate into machine learning.

Lots of modern-day machine-learning techniques only use one form of modality to perform a specific task. A modality is defined as any form of medium that can be used to represent information. Common modalities are texts, pictures, sounds, and videos. When performing tasks like classification, we usually only feed in one image to the model as input. This process repeats itself until the model is fully trained. But it might be more effective to look at the same image from a different perspective, rather than a different image every time. Many researchers have come to a similar conclusion, which have led to various image pre-processing techniques.

Using this intuition allows us to explore features that are not currently present in the original perspective. The idea of looking at different perspectives allows us to extract more

complementary information, which could enhance our task in learning. This brings us to the idea of introducing different mediums and modalities into our learning process. By introducing different modalities in conjunction with the original, we created something known as complementary features. These complementary features could be crucial information that could lead us to a better training result.

For us to accomplish this task, we first need to establish what constitutes a multi-modality. Following our previous analogy, a modality is a perspective that stores information, and therefore a multi-modality is multiple perspectives that stores information. While having multiple perspectives can be helpful, it is also crucial that our perspectives align with each other to solve the problem. In our case, we are exploring computer vision, specifically the task of classification. Therefore, we might want to adjust our definition of multi-modality a little different from that of natural language processing. For our case, we are creating a multi-modality by using a set of images that are captured under the same environment, with the difference being the type of images. This means all of our data will be captured at the same frame, location, and time, but the medium at which it is being captured will be different. A multi-modal image data set is therefore the same image taken by different cameras, each of which can detect a unique modality.

Now that we have a general overview on what a multi-modality means in our paper, we can dive deeper into our topic of interest. There is a common belief that adding modalities will increase the overall accuracy of the network. This idea makes intuitive sense because if we have more data, we should have a better grasp of the concept. While this holds true in our mind, the results show that multi-modal models often fall into the trap of over-fitting. This shows that having extra data does not always guarantee a more accurate model. It does however ensure that our training results will be good, but those types of results do not always transfer into actual testing and application. Recent research actually suggests that multi-modal neural networks tend to over-fit and actually underperform when compared to its single-modality counterpart.

The experiments in [1] show that a single-modal neural network outperforms multi-modal networks that utilizes late fusion technique. Late fusion is the idea of fusing information at the deep ends of the network. The author that performed this experiment speculated that

late fusion is probably the underlying problem for multi-modal models. However, other techniques such as mid-level fusion by concatenation [2], fusion gating with squeeze, excitation[3], and non-local gate [4] did not make any significant improvement and also failed to solve this issue. Other prominent techniques used for training neural networks such as dropout, pre-training, and early-stoppage all have little to no impact. This problem with MMN persists until today and our paper attempts to address this issue.

We hypothesize that the underlying problem is due to how these multi-modal networks are being trained jointly. When these networks are trained under these specific conditions, the weights for each modality does not learn to extract their corresponding uni-modal feature, but instead focus on minimizing their corresponding loss function. This process could cause a feature dilution, which will ultimately decrease the model's accuracy and ability to generalize. Under these ideas we can see that this process leads to an inflated training accuracy which could cause over-fitting[1]. This process also reduces the quality of the features and their ability to generalize, which results in low testing accuracy.

For this paper, our contribution is a novel training procedure on designing MMN networks. We also utilize a projection module after each step optimizer that is unique to our model. Our network is called the proportional feature mixture model. The key idea behind our network is to provide a reduced search space during the fusion procedure. In order to shrink or narrow our search space, we need to apply weighted constraints on top of the fusion layer. To the best of our knowledge, the proportional feature mixture model is the first neural network model that utilizes a projection module in addition to the weighted sum constraints on a deep neural network. Due to the weighted sum constraints, gradient descent and other modern optimizers cannot find an optimal solution for our network. This is because these optimizers cannot take the weighted sum constraint into consideration during the update, thus having the potential to violate this method. In order to remedy this problem, we propose an approximation technique that can be used in conjunction with gradient descent algorithm. This technique applies a projection to our fusion parameter space and identifies the closest point in the appropriate space.

Chapter 2

Literature Review

2.1 Multi Modal Literature

In the field of multi-modal machine learning, *Baltrusaitis et. al* believes there are five fundamental challenges that are defining this field: representation, translation, alignment, fusion, and lastly co-learning [5]. These challenges serve as the foundation of multi-modal research and should be an objective that one should tackle when addressing a problem within the field. We will begin by giving a brief introduction on each challenge and then expand more on each challenge later in the review. The challenge of representation is determining how to merge separate modalities into a single unified feature. An example of such a challenge can often be found in combining audio with visual data. In *Torfi et. al*, the task of Cross Audio-Visual Matching is a good example that utilizes two very different modalities to accomplish a singular task. The problem aims to solve audio corruption through the usage of images. Their method is a coupled 3D convolutional neural network to map both audio and image modalities into a representation space[6]. Another concern with representation is the proportion between redundancy and complementary. This means we must figure out how similar the modalities must be in order for the fusion to be meaningful while being different enough that it can provide some complementary information. This can be a very difficult task since combining data can either destroy the representation or make it better. It is probably one of the most important aspects since a bad representation could lead to inaccurate results. For any machine learning task, data is everything; therefore if the representation is poor the

output will also be poor.

The second challenge is translation, which is the idea of mapping one modality to the usage of another. This type of problem can often be seen in audiovisual speech recognition tasks. The difficulty behind this is usually due to the subjectivity of the topic and the vast amount of different transformations that can be applied for any given data. We can see this challenge in various audiovisual speech recognition models like *Sato et. al* [7] Multi-modal Speech Recognition Using Cor-relativity Between Modality and *Nakamura et. al* [8] Multi-modal temporal a synchronicity modeling by product HMMs for robust audio visual speech recognition.

The third given challenge in *Baltrusaitis et. al* survey is alignment. The underlying idea behind alignment is to find a direct relation between the two provided modalities. This means that, given a task and multi-modal information, we must figure out how those modalities are related to the given task. This is better illustrated with examples like retrieving a recipe given a cooking video or providing instructions given an instructional video. In *Karpathy et. al* Deep Visual-Semantic Alignments for Generating Image Description, they created a model that takes an image and a caption as input which can then output a description about the object within the image. They use a deep convolutional neural network for the image and a bidirectional recurrent neural network for the sentence caption. By combining these two architectures, this resultant network can generate descriptions about objects within the image[9].

The fourth challenge is fusion, the challenge of combining information from different modalities to make a coherent prediction. This challenge is probably the most intuitive to the reader because it is probably the first thing one would think of when someone mentions multi-modal data. Nonetheless this is still an extremely difficult task and the information from different modalities have different predictive power.

The final challenge mentioned in this survey is co-learning, the idea of transferring information between the modalities. It is hard to get two independent objects to collaborate with each other, and the same goes for modalities. The purpose of co-learning is not for one modality to over shadow another but each to bring out more from another. For the remainder of this section we will go more in-depth about each individual challenge as well

as providing some literature relating to the topic.

To begin we will first take a look at representation. For the field of multi-modal learning there exist two main categories of representation. The first being joint representation and the second being the coordinated representation. The joint representation is performed by merging two uni-modalities into a single representation that is shared between both. This can be conceptualized by having a function that takes any number of modalities as input and outputs a representation that is shared by all. This is in contrast with coordinated representation, which attempts to achieve multi-modal representation by projecting each uni-modality into a multi-modal space. This can be thought of as each individual modality having their own function that projects them into a similar representation [5].

An effective way to perform joint representation is through neural networks. This is often referred to as deep multi-modal learning [10]. With neural networks being state-of-the-art for many uni-modal tasks, many researchers have begun integrating neural networks into the field of multi-modal learning. In a survey on deep multi-modal learning by *Ramachandram et. al*, they provide a general list of positives on using deep multi-modal learning. As mentioned in his survey, deep multi-modal networks can have both uni-modal and multi-modal representation compared to standard multi-modal models which contain only multi-modalities. Furthermore, deep multi-modal networks can be trained end to end, support early, intermediate, and late fusion, and have implicit dimensionality reduction; these are all traits that standard multi-modal models would not have by default.

The coordinated approach can be separated into two separate categories: similarity and structured. The idea behind similarity is to minimize the distance between each modality in a coordinated space. An example of this is Unifying Visual-Semantics Embedding with Multi-modal Neural Language Models which focuses on image caption generation. This model uses a framework that encompasses both an encoder and a decoder. For the encoder, an image-sentence embedding is learned through the use of LSTM. The image features are then learned through a deep convolutional neural network that are projected into the LSTM hidden state. They then do a pairwise loss to minimize the distance and rank the image compared to the description [11]. Another example would be *Kaya et. al*'s Multi-Modal Learning with Generalizable Nonlinear Dimensionality Reduction. *Kaya et. al* devise a

multi-modal supervised representation learning algorithm based on non-linearity dimensionality reduction. They believe that nonlinear embeddings are better than linear embeddings due their flexibility. They believe that this leads to better representation especially due to the difference in data geometries in different modalities. Their experiments show promising results compared to state-of-the-art multi-modal learning methods[12].

Translation is the next challenge and probably one of the key components for multi-modal learning. The ability to use data from one modality to perform a task in a different modality is very powerful. A common study for multi-modal translation such as [13] uses a new technique to synthesize visual speech to text. This technique uses the Hidden Markov Model with dynamic features to generate parameters. Another recent popular study is visual scene description, which encompasses two separate fields: computer vision and NLP. Other researchers had a similar line of thought; *Eitel et. al* proposed an architecture that is used to detect objects with RGB-D images. Their architecture separated the image into two separate processing streams. One modality as RGB and the other modality as depth(D). These modalities are trained across the network at the beginning but are reassembled in the fusion layer during the later stages. In order to tackle their problem with a small amount of data, the authors also introduce a unique multi-stage training methodology. This allowed them to train their network without needing a large dataset. *Eitel et. al* were able to achieve state-of-the-art performance on RGB-D dataset as well as achieving recognition in noisy datasets [14].

Fusion is believably the most recognizable challenge among the five listed above. Despite it being mentioned late on this list, it is the most studied topic in the field of multi-modal learning. The idea behind fusion is very intuitive as we are trying to find a method that accomplishes the goals listed in other challenges. To put it plain and simple, we are trying to integrate information from different domains to perform a singular task, such as classification or regression. The motivation behind multi-modality can be summarized into three key points: an ability to have a more robust prediction, have access to complementary information, and an ability to operate when lacking a source of information [10]. The two main approaches for fusion are model-agnostic and model-based. Model-agnostic approaches do not directly modify the multi-modal architecture, whereas model-based approaches are

directly related to the architecture of the model. Both approaches can be applied to manufacture an appropriate fusion.

Before going into details about model-agnostic approaches, details about the possible fusion structure must be discussed.. There are three different fusion methods in the model-agnostic approach: early, intermediate, and late stage fusion. We will provide a general definition for each method. General findings include some literature supporting the effectiveness of these three fusion methods. Early fusion occurs at the beginning of the network, and the fusion process creates low level features. A simple strategy for early fusion consists of concatenating multi-modal features into one which was done by *Poria et al.*[15]. However, early stage fusion methods lack the ability to extract complementary features. Complementary features are unique features that were derived from the fusion procedure. Flaws of early stage fusion leads to extracting redundant features rather than the sought after complementary features.this issue can be mitigated by using autoencoders as dimensionality reduction [16]. In recent research, *Chen et. al* uses multi-modal data with Deeply-Supervised Shuffling Convolutional Neural Network(DSCNN) for semantic segmentation of aerial view imagery. *Chen et.al* accomplishes this by performing early fusion shuffled upsampled feature maps in true orthophoto and digital surface model data. In addition, they also introduce hand-crafted radiometric and geometric features which are also used in the segmentation task. While early stage fusion was performed, by utilizing multi-scale SCNN, the individual modality can still learn important independent features. This fusion method reduces the computational cost and produces promising results from their experiments. [17].

Intermediate stage fusion within a neural network happens with a fusion layer, also known as a shared representation layer. When training a neural network complex representations or features are learned through the hidden layers. These feature representations that are spread throughout the network will be aggregated into the fusion layer. This type of fusion procedure is very flexible but could require a creative design procedure. Depending on how and where the intermediate fusion is implemented, results can vary drastically. *Vielzeuf et. al*[18] designed a multi-modal deep convolution network that works specifically with images. This network was designed under the assumption that each individual modality can be trained by a deep convolutional network. By utilizing uni-modal deep networks for training,

Vielzeuf et. al's model learned the feature mappings for each of the modality. Their model's fusion layer serves as an intermediate fusion and merges the different modalities together, and uses multi-task learning to regularize the modality-specific networks.

Late stage fusion, also known as decision level fusion, is a process that usually happens at the end of the model. The term late stage fusion is misleading, as it is possible to occur in the middle of the model. Object detection and other multi-level detection tasks such as YOLO provide a decision level fusion approach.. However, in most cases the task of the network is usually located towards the end. Unlike early and intermediate fusion, decision level fusion is often applied on the result itself rather than the features. *Tsanousa et. al* devised a multi-modal framework that utilizes late stage fusion to detect human activities. To do this, they have their subjects attach wearable sensors to their body which generate the data for human activity. This framework uses a combination of both early and late fusion classifiers when merging data from multiple sensors. The sensors of choice are accelerometers and gyroscopes. Then a weighted late fusion is applied during the decision [19].

Farahnakian et. al created a deep convolutional neural network to perform intelligent automatic multi-target detection [20]. For their experiment they use RGB and infrared images of the marine environment. They propose a late fusion multi-modal framework that uses RetinaNet as a backend to extract all potential results. Once all the results have been gathered, they use non-maximum suppression to get rid of the redundant results. Their experiments showed that their late stage fusion framework can get a more accurate detection when compared to middle stage fusion and uni-modal frameworks.

Moving away from model-agnostic approaches, we come to the usage of model-based approaches. Model-based approaches focus on the architecture of the model and modifications that are done to the model for a given multi-modal data. Common methods for this are kernel-based, graph-based, and neural-network-based. Since our paper focuses on the usage of neural networks for multi-modal data, our main interest lies with neural-network-based methods. Model-based approaches, much like model-agnostic approaches, are very common in the field of neural networks. *Poria et. al* extracted features in a multi-modal sentiment analysis based on short videos that represent single sentences[21]. *Mao et. al* designed a multi-modal recurrent neural network for generating image captions. This model uses two

different networks: a deep convolutional neural network for images and a recurrent neural network for related image descriptions. This model is largely a recurrent neural network model that includes a multi-modal layer which combines it with the convolutional neural network[22].

Finally, the last challenge for multi-modal learning is co-learning. *Baltrusaitis et. al* describes co-learning as an agent that is used to aid the training process of a multi-modal model. Co-learning is viewed as a helper modality because this process usually occurs during the training stages but has no impact during the testing phase. The authors categorize co-learning into three separate categories depending on how the data is being inputted. These three categories are parallel-data, non-parallel-data, and hybrid-data approaches. The parallel-data approach is where one instance of a modality is directly linked to another instance of a different modality. Parallel data can often be viewed as data that happens during the same time interval. Non-parallel data is the opposite of parallel data where such connection is not directly existing. Hybrid-data is a combination of parallel and non-parallel where not all modalities are directly related, but there is often one modality that serves as a bridge for the model.

Chapter 3

Computer Vision

Computer Vision is a field of study that dates back as early as the 1960s. During this period artificial intelligence was gaining traction due to its success across various fields. Due to its rising success, fundamental goals for research in artificial intelligence began to develop. These goals and challenges include reasoning, knowledge, planning, learning, natural language processing, perception, motion, social intelligence, and general intelligence. While the provided goals for artificial intelligence was very broad, many researchers in the past several decades have made significant contributions to this field. We focus on the sub-field of artificial intelligence also known as computer vision[23].

The field of computer vision addresses one of the primary goals of artificial intelligence: the visual perception of its physical surroundings. As a field, computer vision aims to provide computers with a visual system similar to that of a human. As one of an individual human's primary senses, vision provides us with insight about our physical surroundings. We can detect, identify, and locate information due to our visual understanding. While this task comes as second nature for humans; for computers to mimic such a task can be very difficult. This is because having actual perception about our surroundings is not just one small task, but likely several independent tasks happening simultaneously. The idea of identifying an object, localizing an object, registering an object, and many other factors are needed for our brain to comprehend our physical surroundings. This does not even take into consideration that we have innate metadata about our surroundings such as the development of a natural alignment between objects and surroundings. Because the idea of vision can be viewed

from different angles, different types of studies emerged for computer vision. These studies included image classification, object localization, object detection, and object segmentation. There are likely other kinds of studies, but these are the four major ones that encompasses the majority of computer vision research. For our thesis, we go over image classification since our model will be a multi-modal image classifier. In the appendix, we have a brief overview on object detection.

3.1 Image Classification

Image classification is the task of providing labels to images based on the visual content. For a human, the task of classifying a visual content is a relatively simple task, but for machines this type of work can be very difficult. This is because as humans, we could draw relations between objects and features very quickly and accurately. On top of that, most of us have lived long enough to have a general understanding about our everyday surroundings, thus giving us a huge advantage over machines in perception. Machines, on the other hand, do not have that accumulated experience and would require a large amount of data and training time to create such a foundation. Even with this kind of help, there is no guarantee that the machine will learn the correct representation.

Since having the ability to recognize what you see is the foundation for sight; image classification can be considered the fundamental task for computer vision. Being the very core element of computer vision, one often finds object classification mixed in with other computer vision tasks, such as image localization, object detection, and object segmentation. The topic of image classification has been heavily researched, with techniques such as decision tree, support vector machine, and recently, deep neural networks. Not delving too deeply into the older approaches (pre-neural network), deep neural networks blew away the competition in image classification with AlexNet. Due to AlexNet's success, the interests for neural networks in the field of computer vision have been reignited.

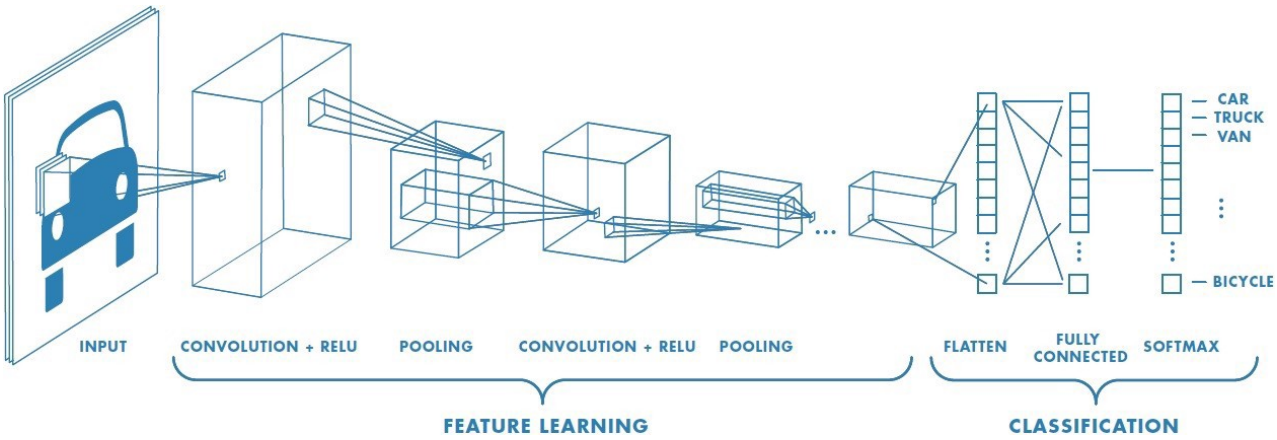

Figure 3.1: Convolutional Neural Network

3.1.1 Convolutional Neural Network

Convolutional Neural Network, often referred to as CNN or ConvNet, is a class of deep neural networks that is most commonly used for visual imagery. The CNN could be interpreted as regularized multi-layer perceptron (MLP), which is also another class of deep neural networks. An MLP is an artificial neural network that is composed of multiple perceptrons. At each layer, each perceptron is connected to every perceptron at the subsequent layer[24]. This type of connection is referred to as fully-connected. The problem with MLPs and fully connected networks is that they are prone to over-fitting. This issue stems from the fact that these types of networks contain a large number of parameters. Because of this, MLP needs to utilize regularization methods to offset the effects of over-fitting when training. CNN avoids this problem by utilizing a strategy known as a receptive field. A receptive field, also known as visual field, locates everything around the given object[25]. In our case, since we are mainly dealing with images, pixels will be that object. With these pixels, the receptive field provides us with the neighborhood of pixels. So, the CNN establishes a connection with every pixel inside the receptive field, rather than the entire image. This makes the CNN less computationally expensive and more in line with the biological visual system [26].

CNN ultimately changed the landscape for computer vision and brought neural networks back into the forefront as the most powerful imagery model. With the performance of AlexNet, a convolutional neural network, at the ImageNet competition of 2012 achieving 15

percent error in classification, creating a 10 percent gap between the runner up; AlexNet demonstrated the effectiveness of neural networks. With AlexNet's impressive performance on image classification, it reignited researchers' interest in neural networks for computer vision. This especially holds true with CNN, establishing itself as core visual interpreter for neural networks. In order to understand our model, we must briefly give an overview on some core components of CNN. The brief overview includes convolution layer, pooling layer, ReLu, and SoftMax. If you are already familiar with these topics you can skip the remainder of this section [27].

3.1.2 Convolution Layer

The convolution layer is the main component for CNN, and it is usually the hidden layer that serves as a feature extractor for the network. The task of feature learning is performed by applying multiple filters that slide across the given input. These filters are similar to those from traditional algorithms, but with one major difference being that they are learnable. These filters learn their parameters during the course of training, and the parameters of these filters are constantly being updated as new inputs are being fed. As these filters slide across the inputs, it performs a point-wise multiplication between the inputs and the filters. Once the filter successfully scans the whole input horizontally and vertically, it produces an object known as a feature map. For CNNs to perform complicated tasks such as object classification and detection, many feature maps must be learned; therefore within each convolution layer there are usually multiple filters.[28].

In a high-level perspective, each filter can be viewed as a feature identifier. Features such as edges, color, and patterns can all be learned by these filters. Because filters from CNNs are not fixed like classical computer vision filters, they are able to derive more meaning from the image. We see that during the early convolutions; the filters learn concepts such as edges, shapes, and corners. In the intermediate layers, filters learn concepts that combine previous features and make eyes, noses, and mouths. During the later stages of the network, nearing the final convolutions, we see those intermediate features combining to create learning concepts such as face, arms, and legs. This process happens because CNN starts out by extracting simple features and then using those simple features to create intermediate

features. Lastly, they use those intermediate features to create final features which can be used to determine the object[29].

By learning a large amount of features we can make more accurate predictions about the target of interest. As humans, we also take a large number of features into consideration when making a decision. Because dogs and wolves are very similar, if we are given only the image of an ear from an unidentified canine we probably can't identify what it is. However, if more parts of the body are given such as teeth or snout, we might be able to make a better prediction. Features serve as the core of classification, and a large CNN can learn many of them.

3.1.3 Pooling Layer

The pooling layer is also another component of CNN. This layer usually performs its task at a specific section of the network; therefore, it is rare to see this layer used at high frequency. Pooling layers' main purpose is to reduce the overall spatial size for the representation. This is important for two major reasons. First, by applying pooling, we can reduce the number parameters, which also reduces the number of computations. Second, it helps the network extract the feature with the most impact. This is important because this will help the network with generalizing features, which is being able to eliminate unnecessary noise that is presented within the image. Pooling is prevalent in removing image transformations and alterations. The pooling layer is applied directly on the feature map, and it returns a feature map of smaller size. The two most common types of pooling are max pooling and average pooling. Maxing pooling extracts the feature with most impact inside the pooling kernel. Average pooling takes the average of all the features within the kernel and returns that average feature back into the pooling layer. Like that of the convolution layer, the pooling layer also slides across the input. Depending on the stride of the pooling filter, the output for the pooling layer can vary. In most cases, pools will usually use a kernel size of two to effectively cut the feature mapping in half[30].

3.1.4 ReLU

ReLu is an activation function that is also known as rectified linear unit. This function could be defined as

$$y = max(0, x)$$

[31].The purpose of an activation function is to provide non-linearity to the input. While there are many activation functions to choose from such as sigmoid, tanh, Maxout and etc, ReLU is usually used for CNNs. This is because ReLU has properties that make it more usable for certain networks. A key reason for choosing ReLU is that it does not saturate at the positive region; that means gradient flow will still have an impact at very high values. Activation functions like sigmoid and tanh do not have this property. It is also more computationally efficient, compared to finding exponential and tanh, making ReLU converge much faster. Also, for the biological aspect, ReLu as an activation function is more plausible than sigmoid / tanh. Because of these properties ReLu and other variations of ReLu tend to be the activation function of choice for neural networks in computer vision [32].

3.1.5 Softmax

The softmax regression is a form of logistic regression that normalizes the input vector to a probability distribution. This is usually used at the end of a classification network. The reason we use softmax rather than logistic regression is because logistic regression can only handle binary classification. That kind of technique is not very effective for image classification whereas in most cases the classification is not binary. Softmax solves this problem by being a multi-class classifier[33]. Converting the output into a probability distribution, we get to see the likelihood that the given input is a certain classification with a certain amount of confidence[34].

$$s(x_i) = e^{x_i} / \sum_{j=1}^{n} e^{x_j}$$

Chapter 4

Theoretical Analysis

Despite all works mentioned in the literature review, to the best of our knowledge, there has not been a rigorous definition of multi-modal fusion networks. We begin by defining a K-N-multi-modal fusion network.

Definition 4.0.1 (K-N-multimodal fusion network)**.** *A neural network is said to be a K-N-multimodal fusion network if $\exists N \in \mathbb{N}$ such that $\forall n \leq N$ the weight matrix at layer n, (denoted by W^n) can be written as a block diagonal matrix, i.e.*

$$W^n = \begin{pmatrix} \boldsymbol{w}_1^n & & \\ & \ddots & \\ & & \boldsymbol{w}_K^n \end{pmatrix} \tag{4.1}$$

Where $w_i^n \in \mathbb{R}^{m_i \times n_i}$.

Figure 4.1 illustrates our definition for a K-N Multi-Modal Network. From the figure we will define K being the total number of modalities within our network and N being the total number of layers. We can then say that if K-N-multi-modal fusion model is modal-agnostic, then it can be decomposed into three separate components. The first component being the pre-fusion layer having the properties mentioned in Definition 4.0.1. The second component is the fusion layer also known as shared representation layer. The shared representation layer uses the outputs from pre-fusion to create a new set of representation. These representation could either be complementary or redundant depending on the result of the training. The third component is the post-fusion layers also known as the decision stage of the network.

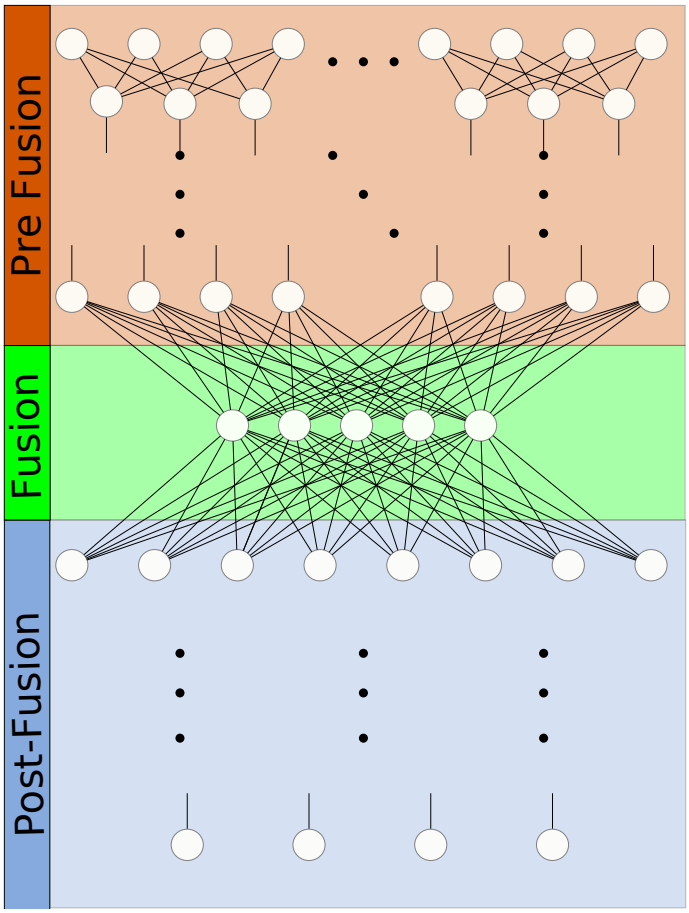

Figure 4.1: Example of K-N-multi-modal

It comprises of all the layers after the fusion layer, and it main focus is to create high level features as well as perform the given task at the end. The greater the value of N, the later fusion in the network happens. A late fusion multi-modal network, we mean a network where N is larger than the number of layers in the post-fusion component.

We want to show that adding modalities does not decrease the approximation power of a network. To this end let $\mathcal{C}$ denote the classification space, $\mathcal{S}$ denote the input space, and $\gamma \in \mathcal{C}$ denote the output of a neural network. We note that the output of a neural network γ with j layers can be written in the following form:

$$\gamma = \boldsymbol{\psi}^j(\boldsymbol{\psi}^{j-1}(...(\boldsymbol{\psi}^1(\vec{S} \cdot \vec{W}^1)...) \cdot \vec{W}^{j-1})$$

where each $\boldsymbol{\psi}^i$ is a nonlinear transformation and $\boldsymbol{S} \in \mathcal{S}$. A neural network φ can then be

thought of as the function

$$\varphi(\boldsymbol{X}) := \boldsymbol{\psi}^j(\boldsymbol{\psi}^{j-1}(...(\boldsymbol{\psi}^1(\vec{X} \cdot \vec{W}^1)...) \cdot \vec{W}^{j-1}) \tag{4.2}$$

Note that the parameters of φ, both hyperparameters and trainable parameters, are finite and therefore can be well ordered. Let f denote the well ordering of the parameters of φ. Denote the set of all realized parameters of φ as P_φ and note that each $p \in P_\varphi$ is an ordered $2j-1$ tuple where for each $k \in [1, 2j-1]_{\mathbb{N}}$, $p(k)$ is an appropriate value for the parameter $f(k)$.

Furthermore $\varphi^{K,N}$ denotes the set of all possible K-N multi-modal fusion neural networks with φ as the neural network in the post-fusion neural network. Finally let $F_\varphi^{K,N} := \{f \in \mathcal{S}^{\mathcal{C}} \mid \exists \mathcal{A} \in \varphi^{K,N}, \exists \alpha \in P_{\mathcal{A}} \text{ s.t. } \mathcal{A}(\alpha) = f\}$ where $\mathcal{A}(\alpha)$ denotes the neural network $\mathcal{A}$ with realized parameters α and $\mathcal{S}^{\mathcal{C}}$ is the set of all functions from $\mathcal{S}$ to $\mathcal{C}$. Then we arrive at the following proposition

Proposition 1. $F_\varphi^{K+1,N} \supset F_\varphi^{K,N}$

Proof. Let $f \in F_\varphi^{K,N}$. Since $f \in F_\varphi^{K,N}$ there exist $\mathcal{A} \in \varphi^{K,N}$ and $\alpha \in P_{\varphi^{K,N}}$ such that $f = \mathcal{A}(\alpha)$. Partition the weight set into the weights in the pre-fusion layer and the post fusion layer so that

$$\alpha = \{W^n\}_{n=1}^N \cup \{\beta\}$$

where $\beta \in P_\varphi$ and $\forall n \leq N$ W^n has the form given by

$$W^n = \begin{pmatrix} \boldsymbol{w}_1^n & & \\ & \ddots & \\ & & \boldsymbol{w}_K^n \end{pmatrix}.$$

Initialize $\mathcal{A}' \in \varphi^{K,N}$ by adding N arbitrary pre-fusion layers to $\mathcal{A}$. We claim there exist $\alpha' \in P_{\mathcal{A}}$ such that $\mathcal{A}'(\alpha') = f$. To see this let

$$\alpha' = \left\{W^{n'}\right\}_{n=1}^N \cup \{\beta\}$$

where for all $n \leq N$, $W^{n'} = diag[\boldsymbol{w}_1^n, ..., \boldsymbol{w}_K^n, \boldsymbol{0}]$ where $\boldsymbol{0}$ is an appropriate sized zero matrix. Then clearly $\mathcal{A}'(\alpha') := f' = f$. ∎

Our main result establishes a method for the user to pre-train a multi-modal network so that when a modality is added, one can guarantee higher accuracy on the training set. Our result does require the existence of an appropriate optimization technique. We define an appropriate optimization technique as a leaning technique that increases the accuracy of a neural network after each training epoch.

Theorem 4.0.2. *If there exist an appropriate optimizer then for any arbitrary training set T we have the following, $\forall f \in F^{K,N}\ \exists f' \in F^{K+1,m}$ such that $acc_T(f') \geq acc_T(f)$. Furthermore, there exists a training procedure that produces such f'.*

Proof. By Proposition 1, there exist $\mathcal{A} \in \varphi^{K+1,N}$ and $\alpha \in P_{\mathcal{A}}$ such that $\mathcal{A}'(\alpha') := f' = f$ In particular, we note that $acc_T(f') = acc_T(f)$. Therefore, by the loop invariant property of the optimizer, after training is complete, we must have $acc_T(f') \geq acc_T(f)$. ∎

Theorem 4.0.2 together with Proposition 1 suggest the following procedure: every time a new modal is added, the pre-existing network should be trained beforehand. Then the weights connecting the new modality to the fusion layer should be preset to zero. If this procedure is followed and an appropriate optimizer is used, then theorem one guarantees the new network will have an accuracy score on training data that is no less than the previous network. This, however, is not enough to guarantee that our network will not over-fit. In order to prevent overfitting, we restrict our parameter space so that our optimizer only considers good parameters. This systematic narrowing procedure of the weight is talked about in detail in the next section.

A number of algorithms have been developed for real-time object detection. These models include the "you only look once" family of networks [35][36][37], Single-shot object detection network [34], Fast R-CNN [38], and Faster R-CNN [39]. These networks are concerned with forward propagation time and accuracy. When we examine these aspects, we observe that the most accurate networks are the slowest and the fastest networks are the least accurate. The Next proposition is concerned with the forwarding propagation time of multimodal neural networks.

Proposition 2. *For a given $\mathcal{A} \in \varphi^{K,N}$, let $O(T)$ denote the runtime for the slowest forward propagation time of the pre-fusion networks and let $O(\varphi)$ denote the runtime for the forward*

propagation time of the post-fusion network. Then the runtime of $\mathcal{A}$ *is bounded by* $O(T+\varphi)$*. In particular, we note that if we attach another pre-fusion network to* $\mathcal{A}$ *to create* $\mathcal{A}' \in \varphi^{K+1,n}$ *and this newly attached network has forward propagation time that is bound by* $O(T)$ *then the forward propagation of* $\mathcal{A}'$ *is bounded by* $O(T+\varphi)$*.*

Proposition 2 suggests that by adding other modalities and parallel processors, to existing object detection algorithms, the accuracy can be increased while the forward propagation time will remain unchanged.

Chapter 5

Proposed Method

5.1 Method

5.1.1 Method Outline

For our proposed method, we will incorporate a variety of techniques that is used in both multi-modal networks as well as computer vision. We will begin by providing a general overview on how multi-modal networks are structured and what each component focuses on. For most multi-modal networks, we could break it down into three distinct sections: pre-fusion, fusion, and post-fusion. The pre-fusion layers are located at the front of the network and are usually in charge of learning low-level features for each of the individual modalities. The fusion layer oversees in taking the low-level features learned during pre-fusion stages and combining them into a new feature. Lastly, the post fusion layer usually takes the output of the fusion layer to either generate higher-level features or perform the given task.

Our model will be K-N multi-modal fusion network that we will call a proportional feature mixture model. To start, the K in K-N multi-modal represents the total number of modalities that our model will have. When training our multi-modal network, our underlying assumption is that the K individual modalities are independent. Since these modalities are independent, we must train our modalities separately in order to keep the property of independence. We achieve this task by first training each uni-modal on its own network. Once each uni-modal network is fully trained, we can incorporate transfer learning for all the pre-fusion layers into our proportional feature mixture model. When applying this procedure,

we must ensure that our outputs share a common dimension. In the case that output feature vectors do not match, we can apply zero padding to fix the dimensionality issues. When creating the fusion layer, we must design it such that it takes the dimensionality of the pre-fusion outputs in mind. Since it is possible for the outputs from the pre-fusion layer to be a volume; we would need to apply a flattening transform to create a single vector.

The purpose of our architecture is to allow our network to learn the concept of mixing corresponding feature vectors. The idea of mixing means that our new feature is the sum of proportion of the corresponding feature vector of each modality. We enforce this idea by implementing a weighted constraint on parameters in the fusion layer. This constraint provided us with a new problem and that is modern optimizers cannot adhere to such constraints. To remedy this issue, we develop a novel projection module known as the simplex projection module.

Let $\mathcal{O}$ denote the total number of nodes used to represent the feature map at any given modality. Then we could also denote the number of nodes within the fusion layer as $\mathcal{O}$. For each node $\mathcal{O}_i$, it will take in K individual inputs, where each modality contributes a feature at its equivalent position. For each node $\mathcal{O}_i$, it outputs a single value that will be denoted as $o \in O$ such that it is the weighted sum of the corresponding output from the K modalities. Mathematically speaking, for all $i \leq |\mathcal{O}|$

$$o^i = \sum_{j \leq K} w_j^i f_j^i$$

where for all $i \leq |\mathcal{O}|$, f_j^i is the ith output node for the jth modality and w_j^i is the weight that connects corresponding nodes. Since each modality is pre-trained, the fusion layer should represent a mixture of each feature map that is learned by each modality. To this end we require that

$$w_j^i \geq 0 \text{ for all } i \leq |\mathcal{O}| \text{ and } j \leq K.$$

and that for all $i \leq |\mathcal{O}|$

$$\sum_{j \leq K} w_j^i = c.$$

Our model can be seen in Figure 5.1.

These constraints imply that the weighted matrix that connects the pre-fusion layer to the post-fusion layer (denoted by $\vec{W}^f$) can always be written in the following form

$$W^n = \begin{pmatrix} w_{1,1} & 0 & \cdots & w_{1,j*\mathcal{O}+1} & 0 & \cdots & & \\ 0 & w_{2,2} & 0 & \cdots & w_{2,j*\mathcal{O}+2,} & 0 & \cdots & \\ \vdots & & & & & & & \\ 0 & \cdots & w_{\mathcal{O},\mathcal{O}} & 0 & \cdots & w_{2,j*\mathcal{O}+\mathcal{O}} & 0 & \cdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 & \cdots \\ \vdots & \ddots & & & & & & \end{pmatrix}$$

.

where $j \leq K-1$ and we've abused some notation, in that $\mathcal{O}$ is used to represent the carnality of $\mathcal{O}$. We multiply by $\mathcal{O}$ since each node in the fusion layer is only connected, via a nonzero weight, to the corresponding node in each of the ith modality. The zeros are fixed to indicate no connection between nodes.

With the structure of our architecture provided, review some of the benefits to having such an architecture. By having a weighted sum constraint of one for parameters between the pre-fusion to fusion layers, we essentially aid the network in creating mixture features. This process should allow the network to optimize the fusion as the training procedure progresses. We also believe that by mixing uni-modal features, we naturally force our network to learn new complementary features rather than redundant features inside the fusion layer. The second benefit is that our architecture allows modalities with higher impact factors to contribute more to our fusion. This will allow our network to put more attention on high impact and less attention on low impact features.

Moving on to the post-fusion layers of our network. There are many potential architectures we could use for the decision portion of our network. This is because the post-fusion layer is heavily dependent on the task of the model. It is important to remember that our model follows the model-agnostic approach, therefore our architecture is not set in stone. In our specific case, we are performing object classification therefore we will use architectures that are related to that field of study. In the case that our task is different like object de-

tection or object segmentation, we would utilize a vastly different architecture for the post fusion-layer.

Since we are performing object classification, for our post fusion layer we will use a series of fully-connected layers. This is likely the most general form of ConvNet, which is often use for image classification. The reason we are utilizing the most basic form is because we would like to test the effectiveness of our multi-modal procedure. By excluding other popular techniques and optimization, we get to the true effectiveness of our method.

5.1.2 Training Procedure

With a general intuition on how our model works, we will now go into a more rigorous and mathematical representation of our model and training process. Let T be an instance of training data. For $\vec{x} \in T$, let $\varphi(\vec{x})$ denote the output of a proportional feature mixture model. Let $L(\vec{x})$ denote the label for $\vec{x}$. Then for a given loss function $\mathcal{D}$ and a weighted assignment set $\vec{W}$, total loss is given by

$$\sum_{\vec{x} \in T} \mathcal{D}(\varphi(\vec{x}), L(\vec{x}) | \vec{W}).$$

The objective function for a proportional feature mixture model is then to minimize the loss function with the given restrictions discussed above, mathematically speaking

$$\min_{\vec{W}} \sum_{\vec{x} \in T} \mathcal{D}(\varphi(\vec{x}), L(\vec{x}) | \vec{W}), \tag{5.1a}$$

$$\text{sb.t} \sum_{i \leq K} w_i^j = c \text{ for all } j \leq \mathcal{O}, \tag{5.1b}$$

$$\text{and } w_j^i \geq 0 \text{ for all } i \leq |\mathcal{O}| \text{ and } j \leq K, \tag{5.1c}$$

Techniques, such as stochastic gradient descent, cannot estimate the solutions to Eq. 5.1a since their updating procedure does not take weighted constraints into account. In order to solve this issue, we needed to project the parameters inside the network after the optimizer. This suggests that we need to add an additional step to our training procedure. From this we created a projection module that is used after stochastic gradient descent. More specifically we project our updated weights into a corresponding simplex.

Mathematically, for a given weight vector $\vec{y} \in \mathbb{R}^K$, we find the vector $\vec{x} \in \mathbb{R}^K$ that satisfies

$$\vec{x} = \min_{\vec{x} \in \mathbb{R}^k} \frac{1}{2} \|\vec{x} - \vec{y}\|^2, \tag{5.2a}$$

$$\text{sb.t } x_i \geq 0 \text{ for all } i \text{ and } \sum_{i \leq K} x_i = c, \tag{5.2b}$$

where $\| \cdot \|$ is the l^2 norm. A complete discussion on solving Equation 5 can be found in [40].

We first note that Equation 5.2a has the following form

$$\min_{\vec{x} \in \mathbb{R}^K} f(\vec{x}), \tag{5.3a}$$

$$\text{sb.t } g_i(\vec{x}) \geq 0 \text{ for all } i \leq K, \tag{5.3b}$$

$$\text{and } h(\vec{x}) = c, \tag{5.3c}$$

where $g_i(\vec{x})$ is the ith coordinate projection function from $\mathbb{R}^K$ to $\mathbb{R}$, and $h(\vec{x})$ is the sum of the vector coordinates. In our case, for all i, g_i and h are affine functions so that the sufficient condition, known as linearity constraint qualification, is met and thus the Karush-Kuhn-Tucker(KKT) conditions for optimization can be used. The Lagrangian for our problem is given by

$$\mathcal{L}(\vec{x}, \mu, \boldsymbol{\lambda}) = \frac{1}{2}\|\vec{x} - \vec{y}\|^2 - \mu(\vec{x}'\vec{c} - c) - \vec{\lambda}'\vec{x},$$

where $\vec{c}$ is the vector with c in every component and μ and λ are constants. Our KKT constraints [41] for this problem are given by

$$x_i - y_i - \mu - \lambda_i = 0 \text{ for all } i, \tag{5.4a}$$

$$x_i \geq 0, \tag{5.4b}$$

$$\lambda_i \geq 0, \tag{5.4c}$$

$$x_i \lambda_i = 0, \tag{5.4d}$$

$$\sum_{i=1}^{K} x_i = c. \tag{5.4e}$$

Equations 5.4b, 5.4c, and 5.4d imply that for all $i \leq K$ x_i is given by

$$x_i = \begin{cases} y_i + \mu \text{ if } x_i > 0 \\ 0 \text{ O.W.} \end{cases}$$

Without loss of generality assume that $\vec{y}$ and $\vec{x}$ are ordered. At most K elements of $\vec{x}$ are positive. Let Z denote the number of positive elements of $\vec{x}$. Then by 5.4e we have

$$\begin{aligned} c &= \sum_{i \leq K} x_i \\ &= \sum_{i \leq Z} x_i \\ &= \sum_{i \leq Z} y_i + \mu. \end{aligned}$$

This implies that

$$\mu = \frac{1}{Z} \left(c - \sum_{i \leq Z} y_i \right).$$

This implies that in order to solve Equation 5 we only need to find the correct value of Z of which there are K possibilities. Once Z is known, we can calculate μ and then each component of $\vec{X}$.

The pseudo-code to solve equation Equation 5 can be found in Algorithm 1, and a second proof of its correctness can be found in [42]. The entire training procedure can be found in Algorithm 2.

After each epoch of training the standard gradient descent algorithm is applied to update the weight values. After which each non-zero element in a row of the pre-fusion to fusion layer weight matrix is sent to the projection module. The vector found in the projection module replaces the row element in the pre-fusion to fusion layer weight matrix.

This projection module can also be used to reassign weights when a modality knockout occurs. If $k \leq K$ is the modality that is knocked out then the non-knocked out weights are updated by sending a vector, that has the non-knocked out weights as elements, to the projection module. Since the projection module's runtime is bounded by $O(KlogK)$, this reassigning process is also bounded by $O(KlogK)$.

5.1.3 Review and Algorithms

Recall that we originally hypothesised that jointly trained multi-modal neural networks fail due to the complexity of the weight space. This creates a problem for the optimizer to find a good generalized solution. Our model will try to tackle this problem by using a couple of different methods. By training each modality independently, we can ensure that the weights for each modality are present to extract relevant features. To simplify the search space a weighted constraint is put on the fusion layer. The weighted constraints represent the proportional mixture of the input modalities. Lastly, rather than initializing the weights with random values, we use the values suggested in Theorem 4.0.2. This allows the network to start at a more optimal position, and in a way reduce the search space. Since training methods like that of gradient descent cannot truly estimate the solution to our network; we create a method called the projection module. The projection module takes the weights that do not meet the necessary constraints and modify it in a specific manner. When during training, the parameters within the fusion layer go out the constrained scope, our projection module would update the weights using a simplex projection.

Algorithm 1: Projection Module

Input : Non-zero row elements from pre-fusion to fusion weight matrix $\vec{w}_j$,
Output: Vector $\vec{x}$ satisfying eq. 5
K = $len(\vec{w})$;
Sort $\vec{w}$ to $\vec{y}$ so that $y_1 \geq y_2 \geq \cdots \geq y_K$;
z = NaN ;
for $i = 1$ *till* K **do**
    Calculate $t = y_i + \frac{1}{i}(1 - \sum_{i \leq j} y_i)$;
    if $t > 0$ **then**
        z = i
$\lambda = \frac{1}{z}(1 - \sum_{i \leq z} y_i)$;
for $i = 1$ *to* K **do**
    $x_i = \max\{w_i + \lambda, 0\}$;
return $\vec{x}$

Algorithm 2: Proportional Feature Mixture Model Training

Input : Training set T where each instance of training data is viewed under K modalities; K predefined neural networks with common architecture after some point; pre-fusion hyperparameter; sum constraint C

Output: Weight assignment $\vec{W}$ for the proportional feature mixture model

for *each modality* k **do**
    Train the kth network using the corresponding data in T;

Cut each network at a feature map with common dimensions;

for *each modality* k **do**
    Import weights into pre-fusion kth modality;

Create a fusion layer with the same dimensions of the common feature map;

Create a post-fusion layer with architecture identical to the cut portion of the K networks;

post-fusion weights get weights from the most accurate trained network;

if *pre-fusion == True* **then**
    Train the fusion, post-fusion, and pre-fusion layer using T and implement Algorithm 1 on each fusion layer weight row vector after each weight update.

if *pre-fusion == False* **then**
    Train the fusion, and post-fusion,layer using T and implement Algorithm 1 on each fusion layer weight row vector after each weight update.

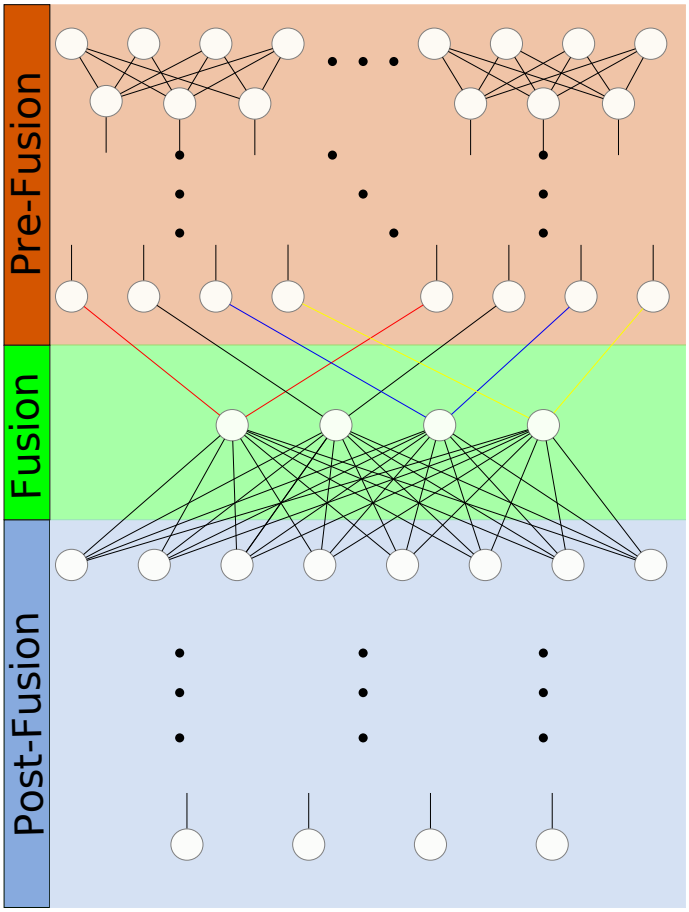

Figure 5.1: Example of a K-N multimodal network once our systematic narrowing procedure as been applied. The colored edges represent the weight that must sum to one.

Chapter 6

Experimental Analysis

6.1 Experimental Setup

For our experimental setup, we decided to divide our experiments into two separate categories. In our first set of experiments, we tested the effectiveness of the proportional feature mixture method compared to various other common multi-modal training methods. This is important since one of our objectives is to tackle the problem of finding an effective way to train and design multi-modal networks. After that, our second set of experiments relates more to the applications of the proportional feature mixture method such as modality knockout.

In the first set of experiments, we tested the effectiveness of our method in the task of object classification. In order for us to perform this experiment, we created and trained multiple models under different conditions. Our list of training conditions are as follows: type of fusion layer, transfer learning, parameter freezing, and proportion constraint. Each of the upcoming models trained under some combination of these four conditions. In the next section, we give an overview of each condition and options that are available.

Experimental Conditions

The first condition is potential types of fusion architectures within our experiments. In our experimental design, the models can have one of two different types of fusion architecture. The first potential option is a fully connected fusion, and the second possible option is

the feature-wise fusion. We begin by creating a fusion architecture that is the exact same size as the output of the pre-fusion layer. This step applies to all fusion architectures for our experiments. For the case of applying a fully connected fusion, every feature from the output of the pre-fusion layer must connect to every feature in the fusion layer. This condition applies to every modality within the network. In the case where we have a feature-wise fusion, every feature connects to its corresponding feature in the fusion layer. The purpose of this condition is to address the problem of fusion. For fusion, there exists many different approaches including model-based and model-agnostic. This condition allows us to test various model-agnostic architectures to see the impact it has on multi-modal fusion.

The second experimental condition is the usage of transfer learning. Our experimental models can either train with transfer learning or train from the ground up. The purpose of including transfer learning as a condition is to test the importance of well-established representations from uni-modalities. This condition addresses the problem of representation for deep multi-modal neural networks. The first step to this process is to train each of the uni-modalities separately for the given task. In our case we train a RGB and infrared image classification neural network. After that we extracted all the weights from the pre-fusion layer and applied the weights that have been extracted into our new multi-modal network.

The third experimental condition is parameter freezing. This condition has two possible options: either freeze parameters or not freeze parameters. The purpose of freezing is to prevent the model from updating its parameters during the training process. For our experiments, parameter freezing does not apply to the whole network, but only at the pre-fusion layers. The purpose of parameter freezing is to prevent our uni-modal representations from being diluted during the training procedure. This condition once again addresses the issue of representation in multi-modal models. From this we can test the importance of the transfer representation compared to the complementary representation learned from the model.

Lastly, the final condition that our experiments take into consideration is the proportional constraint. The proportional constraint is the idea that each uni-modal contribution feature must be proportional. Within our experiments, some of the models use the proportional constraint and some do not. The intuition behind this idea is to reduce the possible search space for the network. This is because for neural networks the search space is often too

complex, and by applying a weighted constraint we can artificially push it into the direction of interest. This final condition tackles the problem of translation in multi-modal. Our experimental condition attempts to address this issue with a novel joint representation.

6.1.1 Experimental Models

Our first multi-modal network is referred to as STANDARD. This network's fusion architecture is fully connected. This network does not use transfer learning therefore it does not contain any single modality representation at the start. The network does not have any of its pre-fusion layers frozen, therefore every parameter within the network is trainable. This allows the network to generate the maximum amount of the complementary representations between the modalities.

Our second multi-modal network is referred to as STANDARDT. This network's fusion architecture is also fully connected. This network unlike the first uses transfer learning, therefore each uni-modal network imports its pre-fusion parameters into the pre-fusion layers of STANDARDT. This network does not have its pre-fusion parameters frozen, therefore it can start with uni-modal representations. We hope that by training this network, that we can retain some of the uni-modal representations as well as develop some new complementary representations.

Our third multi-modal network is referred to as STANDARDTF. This network is very similar to the second network, which means it is fully connected and applied transfer learning. The only difference is that this network also has its pre-fusion parameters frozen, therefore it should retain its uni-modal representations throughout the training procedure. This network generates the least amount of complementary representations, but we believe it should perform as well as the strongest uni-modality.

Our fourth multi-modal network is referred to as FEATURET. This network uses a feature-wise connection for its fusion architecture. This means every feature establishes one connection to the fusion layer. The connection should be in the relative position of the output from the uni-modal network. This network applies transfer learning on its pre-fusion parameters, allowing this network to start with uni-modal representations. The network does not include parameter freezing; therefore, it is possible for this network to develop

complementary representations as well. The purpose of this network is to reduce the total number of parameters in the fusion layer. By applying a feature-wise connection we naturally limit the amount of complementary representations and limit the amount of change to the uni-modal representations.

Our fifth multi-modal network is referred to as PROPORTIONF. This network is vastly different from the four models mentioned previously. That difference is that this network includes the simplex projection module in the fusion architecture. This module enforces a weighted sum constraint in the fusion layer, therefore after each gradient update the module provides a parameter re-adjustment. This network also takes characteristics from FEATURET, where the connections between the pre-fusion layers to the fusion-layer is feature wise. This limits the total amount of connection giving us the same benefits as FEATURET. Additional benefits include less prone to overfitting due to less parameters and less computation needed. This network has its pre-fusion parameters frozen preventing the uni-modal representation from being lost during training. This set of conditions allows the network to simulate the feature mixing effect of uni-modal representations.

Our final multi-modal model is referred to as PROPORTION. This network also uses the simplex projection module at the fusion architecture. The fusion architecture for this network is also feature-wise fusion. The one slight difference between this model compared to PROPORTIONF is that this model does not have its pre-fusion layer weights frozen. By not freezing the weights, we allow the import parameters from transfer learning to update. Therefore, this model gets its uni-modal features from transfer learning and it is also to get complementary representations from regular training. The downside for not freezing the model's parameters is that we cannot guarantee that the uni-modal representation will not be lost after training.

6.1.2 Classification Experiments

With the six models constructed under different conditions, we can finally test the performance of the networks. The networks are designed for image classification with four different modalities. The modalities are as follows: red, green, blue, and infrared. We obtain the red, green, and blue modalities by breaking RGB into its separate components and feeding each

individual modality into its own separate network. We then train four uni-modal convolutional neural networks, one for each individual modality. All the convolutional neural networks have an identical architecture meaning the structure at which they are trained are the same. This structural property is only here for testing purposes, since we do not want to create bias for certain modalities. In practice as long as the created network can fuse at the fusion layer, the projection module can always be included. At this point the pre-fusion layers are connected to the fusion layer of the network and the feature maps should be identical in size for all modality.

Using this experimental setup, we must first define the baseline for our experiments. As mentioned in earlier sections, some recent research has shown that multi-modalities do not perform better than their uni-modal counterparts. Therefore, our baseline are the four uni-modal networks red, green, blue, and infrared. Performing better than uni-modal neural networks is important because there is no reason to use multi-modalities if uni-modalities are in fact better. Our second point of focus for these experiments is to determine the effectiveness of the projection module compared to other multi-modal training methods. The conventional multi-modal training methods are as follows: STANDARD, STANDARDT, and STANDARDTF. In the previous section, we provided a general overview on these three networks. In the next section, we tested whether restricting weights affects the model's performance.

If we let F^1_{RGB}, F^1_{IRs}, F^2_{STANDARD}, $F^2_{\text{STANDARDT}}$, $F^2_{\text{STANDARDTF}}$, F^2_{FEATURE}, $F^2_{\text{PROJECTION}}$, and $F^2_{\text{PROJECTIONF}}$ denote the set of functions that are approximable by RGB, IRs, standard, standardT, standardTF, feature, projection, and projectionF then we have the following comparisons

$$F^1_{\text{RGBs}} \subset F^2_{\text{PROJECTIONF}} \subset F^2_{\text{PROJECTION}} \subset F^2_{\text{FEATURE}} \subset F^2_{\text{STANDARDTF}} \subset F^2_{\text{STANDARDT}} \subset F^2_{\text{STANDARD}}$$

$$F^1_{\text{IRs}} \subset F^2_{\text{PROJECTIONF}} \subset F^2_{\text{PROJECTION}} \subset F^2_{\text{FEATURE}} \subset F^2_{\text{STANDARDTF}} \subset F^2_{\text{STANDARDT}} \subset F^2_{\text{STANDARD}}$$

F^1_{RGBs}, F^1_{IRs} are not comparable since these are uni-modal networks. Due to the use of feature-wise connection and weighted sum constraint, the projection and projectionF have the lowest approximation power relative to the other multi-modal networks. It should also be noted that the regular standard multi-modal network has the highest approximation

power. Our experiment hopes to show that by applying systematic narrowing, we can guide a network into a better representation.

6.1.3 Proportional Freeze Experiments

For our proportional freeze experiment, we are attempting to train our models with a new hyper-parameter known as partial freezing. Partial freezing is the idea of freezing the network in segments instead of freezing it entirely. For example, if our network contains one hundred parameters, instead of freezing all one hundred, we would only freeze eighty. The purpose of this experiment is to test our hypothesis in the effectiveness of complementary features. Since most of our models use transfer learning, lots of these models have strong uni-modal features. In an attempt to create complementary features, we decide to freeze different portions of our network during training. Depending on the proportion of the freeze, the number of complementary features learned will be different. By freezing ninety percent of the network, only ten percent of the parameters can update.

For us to replicate partial freezing, we begin by training our network. During the training process, we calculate the gradient for every parameter that will be updated during back propagation. With the appropriate gradient matrix, we create a zero-one mask. The zero-one mask will contain a proportion of ones and zeroes that is randomly generated. We will then do an element-wise product between the mask and the gradient matrix, which will zero out some of the gradient values inside the matrix. At this point we will use the new gradient matrix to update our neural network, during the backpropagation.

6.2 Dataset

6.2.1 FLIR Thermal Dataset

For our experiments, we will utilize the FLIR Thermal Dataset for Algorithm Training. This dataset is composed of both thermal and RGB images taken from separate cameras. The thermal and RGB cameras are approximately two inches apart when these images were taken. These images were taken from a driver's perspective on the streets of Santa Barbara,

California. Among all the images taken, about sixty percent of the images were taken during the day and the remaining forty percent were taken during the night. These images were taken between the months of November and May ranging from clear sky to overcast weather. This dataset mentions that there are five classes, but after thorough inspection it really comes down to three major classes. All this information can be found on the dataset website[43]. The FLIR dataset was originally an object detection dataset, so it is not a true fit for the experiment of classification. For us to make a classification dataset out of this we must crop the bounding box coordinates provided by FLIR annotations. After cropping all the bounding boxes from the original images, we obtain around 60,000 training samples and 10,000 testing samples for each modality. Since this dataset was partially crafted by us, there are some potential errors. One major flaw about this dataset is the class imbalance. For our experiments we did not utilize any techniques that fixed the issue of class in balance. The second possible flaw is that the bounding boxes are only provided for the infrared images, but since RGB is very similar to infrared we decided to crop it with the same annotation. Lastly, image classification is usually done with the object fixed directly at the center, as due to the second flaw some images will not be centered correctly. While we did not go through all the images, we inspected enough to say that this is usable. Since we are applying this dataset to all our models the negative impact should affect all the models if any. Before we can feed the dataset into our network, we must do a series of preprocessing to make our training possible and effective. The first important prepossessing we must do is to standardize the size of our input. In our case, we standardize all our images to be the size of 70 by 70. This is needed because we are utilizing a fully connected layer at the end, making our network not size invariant.

6.3 Experimental Result

6.3.1 Multi-Modal Classification

For our experimental results, we can see from the graph that the baseline models Red, Green, and Blue achieve a testing accuracy of about 76 percent. The reason for such low accuracy is most likely due to the flaws of the dataset. As mentioned previously, the RGB dataset

was handmade by us using the infrared annotations with minor adjustments. It is important to also note that the infrared dataset is not perfect, since this dataset was originally made for object detection, therefore even for the infrared images the object might not be totally center like classical classification datasets. As for infrared, we can see that it reaches an accuracy of around 88 percent in testing accuracy. From the results we can see that the best uni-modal model reaches approximately 88 percent given that all these uni-modal models were trained with a standard convolutional neural network. In the case of multi-modalities, we clearly see there is a distinct outlier among the six. That outlier is the standard model, which achieves the same approximate power as the best single modality network. While it is not strictly worse from the graph, we can see that the standard model did not stabilize at 100 epochs. This makes the standard model unreliable compared to the uni-model infrared, thus proving the claim that multi-modal model is not better than uni-model when trained under the wrong conditions. The other five multi-modal models performed much better, and we believe the reason is due to the other five models utilizing transfer learning. We believe there are many reasons why transfer learning makes such a significant impact in multi-modal training. The first being an effective starting point for the task. Regardless of the network, our models are trying to learn image classification and importing weights from previous models help a lot. This is likely the reason why the five models which use transfer learning have less testing accuracy variance compared to standard. The second reason is likely due to the multi-modal having uni-modal representation when transfer learning is applied. When training on a standard model, no uni-modal representations were injected into the model; therefore only complementary features were learned and all uni-modal features were ignored. The second important trend we see from our experimental results is that fully connected networks perform worse than feature-wise connected networks. We suspect that the fully connected networks are probably overfitting due to these networks having a large number of parameters. We can also see this trend in the testing loss where standardT and standardTF test loss is higher than the other multi-modal networks. We expected these results as many neural network literatures show that fully connected networks don't usually perform the best. That is the reason why many of the recent architectures attempt to create networks with less parameters with a more sophisticated connection scheme. From the graphs, we

can also see that models that freeze their pre-fusion layers perform worse than models that do not freeze their pre-fusion layers. We suspect this is most likely due to the fact that models which totally freeze their pre-fusion layers create less complementary features than models that do not. We can see that this result is surprising and not just hard to tell what is the optimal number of uni-modal features, compared to complementary features. In the cases where pre-fusion freezes, we only create complementary features during the fusion and post fusion layers, while the pre-fusion layer remains to be uni-modal features. On the other hand, networks that don't integrate freezing in the pre-fusion layers create more complementary features during the pre-fusion stages. We will delve more into this topic in the next experimental section where we test various freezing ratios. Lastly, from our experiments, we can see models that apply the simplex projection modules perform better than networks which do not have it. This increase in performance is most likely due to our intuition that a weighted sum constraint on the fusion layer creates effective complementary features. By having our fusion layer learn the most effective combination of fusion uni-modal features, we create complementary features that are truly effective in multi-modal domains.

6.3.2 Proportional Freeze Experiments

From our results, we can see that proportional freezing did really have an impact on training our multi-modal network. We attempted this experiment with several different proportions 60:40, 70:30, 80:20, and 90:10. The first value represents the proportion of weights that are not frozen and the second value represents the proportion of weights that are frozen. Our training procedure applies partial freezing for each batch; therefore, each update has a new set of frozen parameters. We hypothesize that this version of proportional freezing has no impact due to the random nature of our selection. Because we randomize our weight selection it is possible for us to update every parameter in our network, therefore converting all our uni-modal features into complementary features. We also believe that given enough training time, this proportional freezing procedure will eventually converge to our normal projection model.

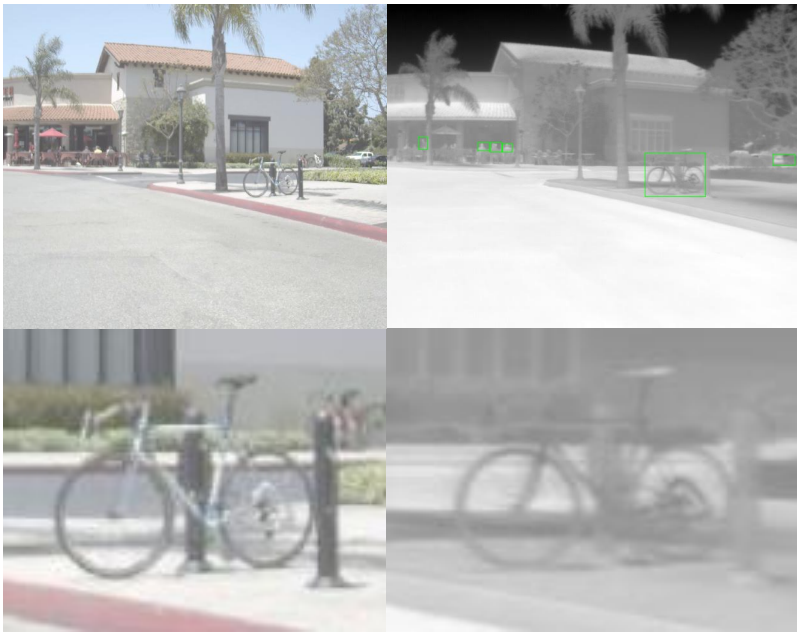

Figure 6.1: The top two squares are a pictures taken from the RBG and IR viewpoint, while the bottom two squares are the extracted images used in our CNN.

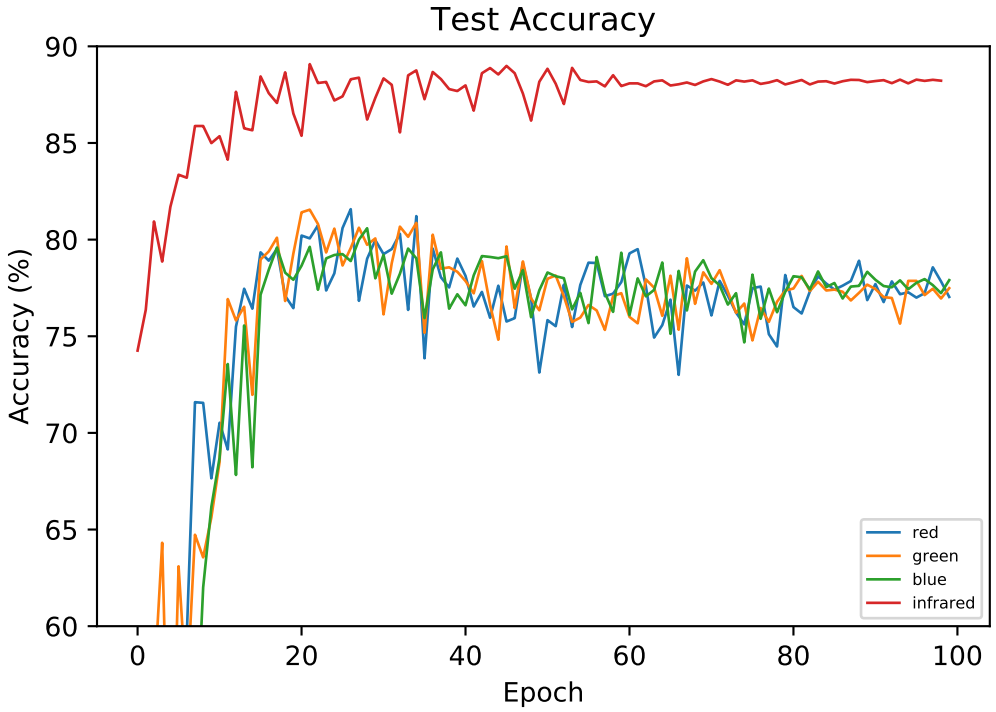

Figure 6.2: Uni-Modal Testing Accuracy

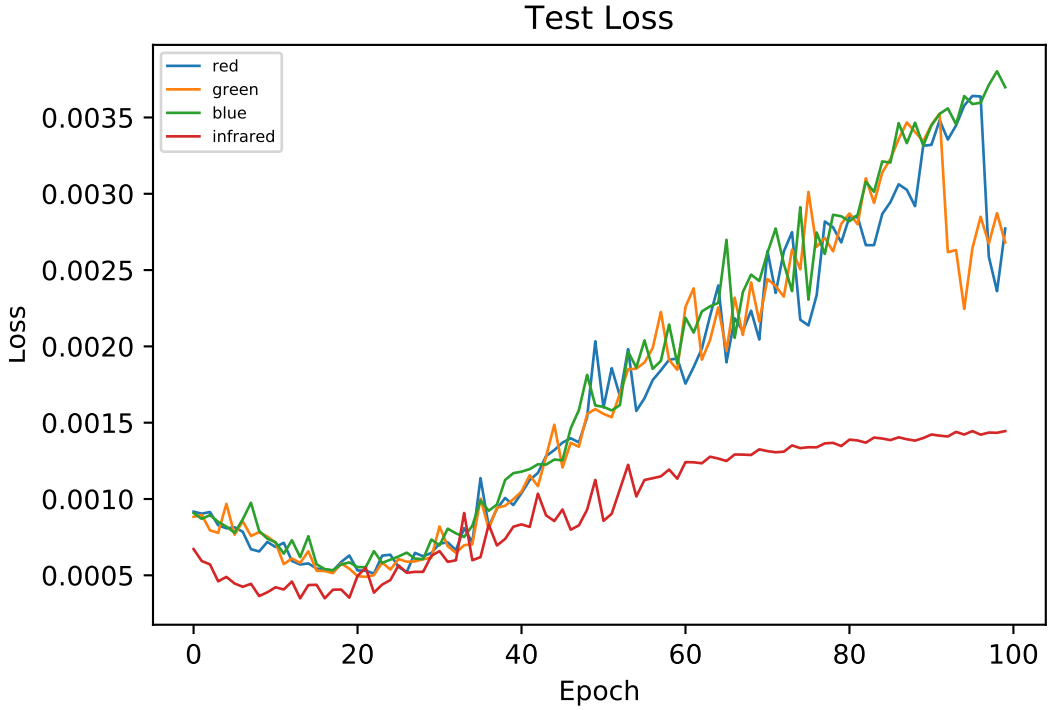

Figure 6.3: Uni-Modal Testing Loss

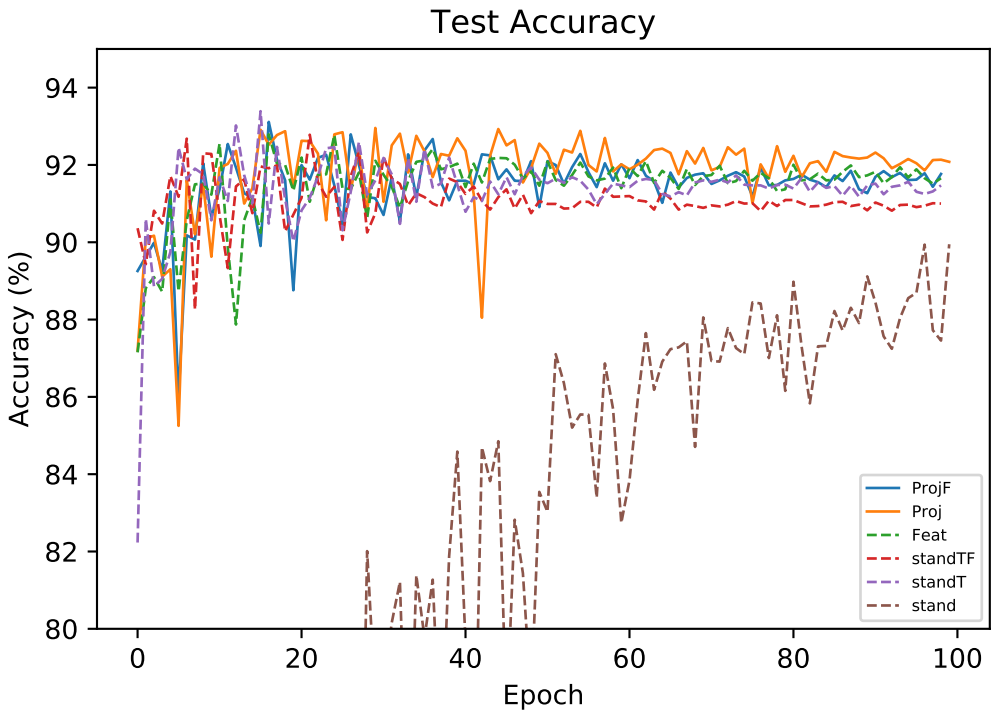

Figure 6.4: Multi-Model Testing Accuracy SGD

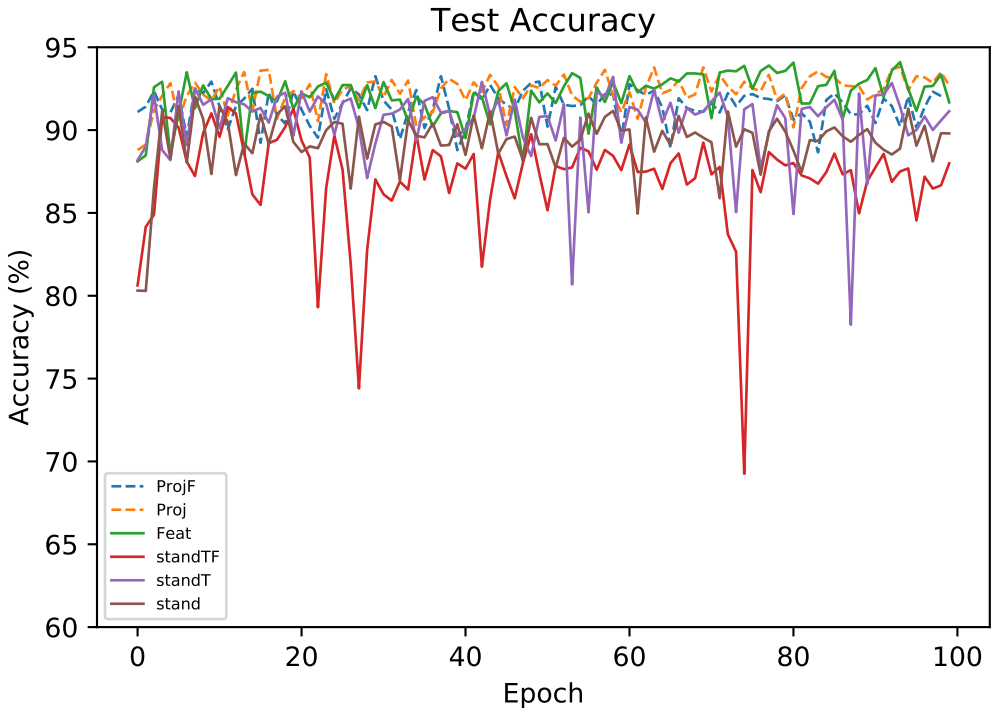

Figure 6.5: Multi-Model Testing Accuracy ADAM

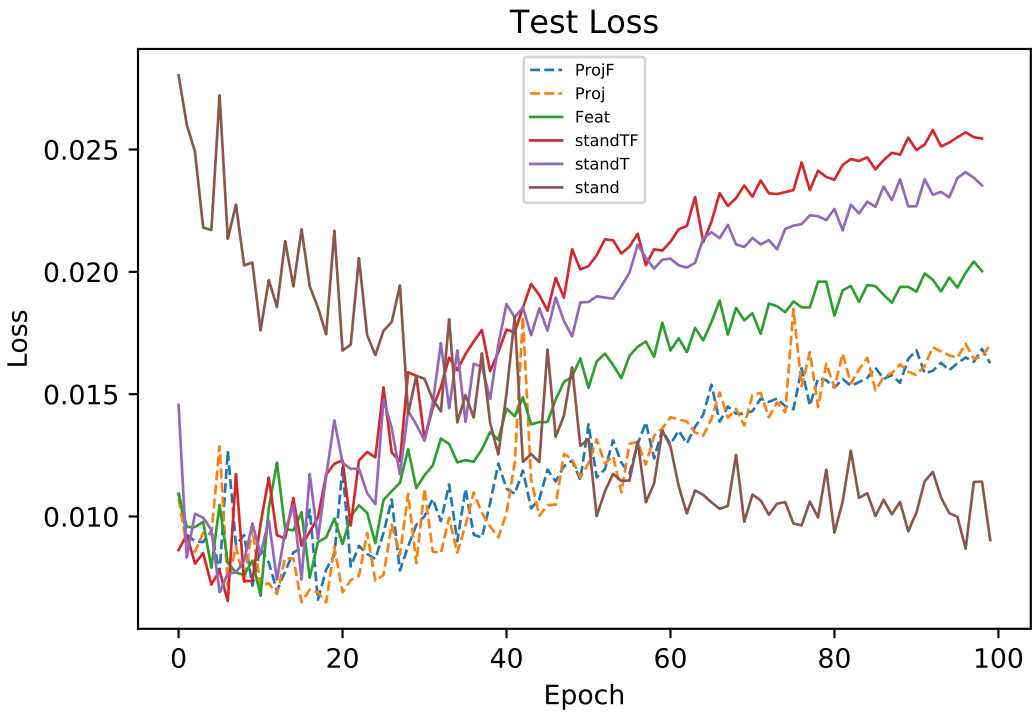

Figure 6.6: Multi-Modal Testing Loss SGD

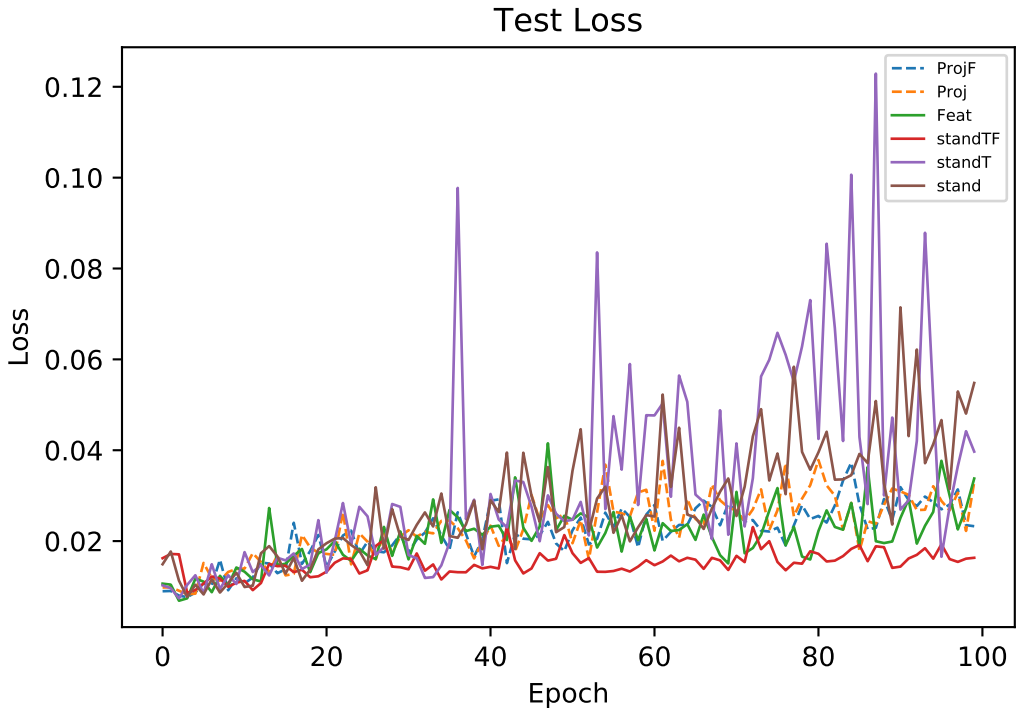

Figure 6.7: Multi-Modal Testing Loss ADAM

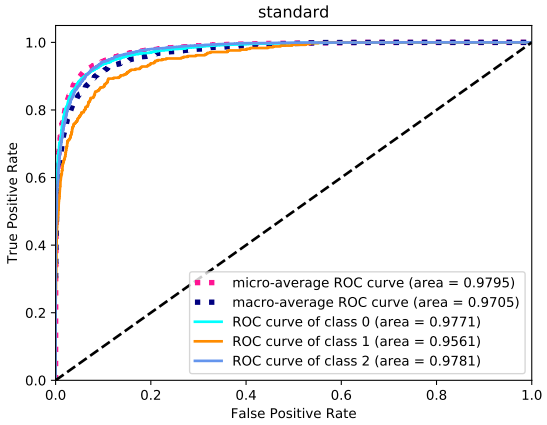

Figure 6.8: ROC Graph - Standard SGD

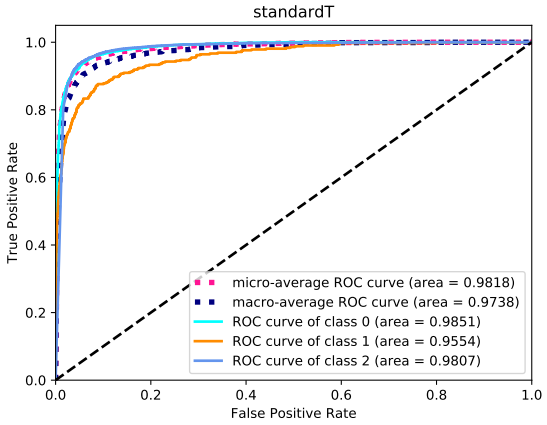

Figure 6.9: ROC Graph - StandardT SGD

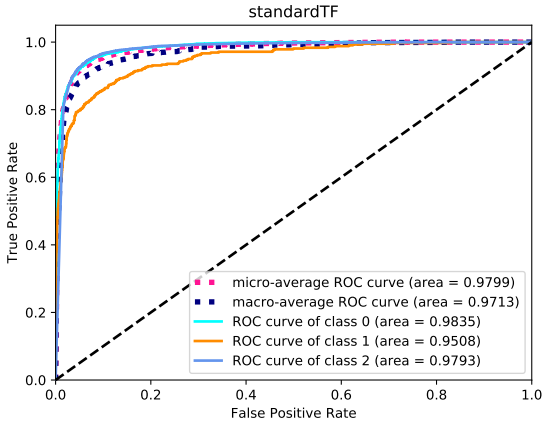

Figure 6.10: ROC Graph - StandardTF SGD

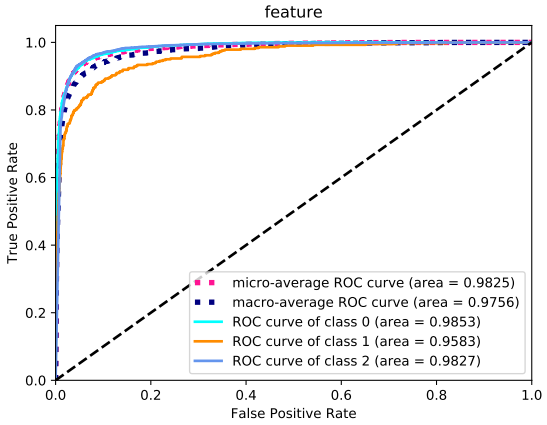

Figure 6.11: ROC Graph - Feature SGD

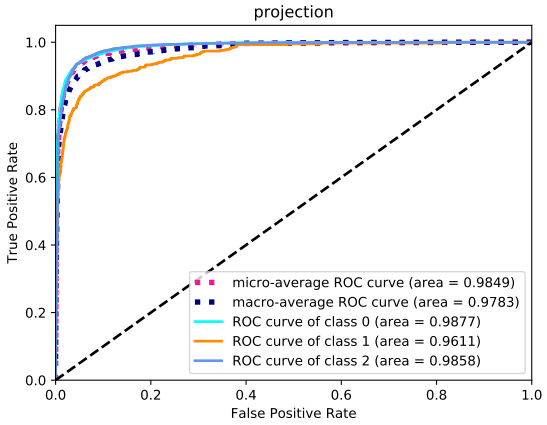

Figure 6.12: ROC Graph - Projection SGD

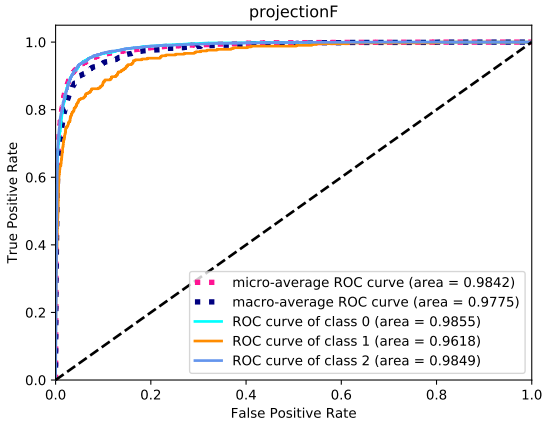

Figure 6.13: ROC Graph - ProjectionF SGD

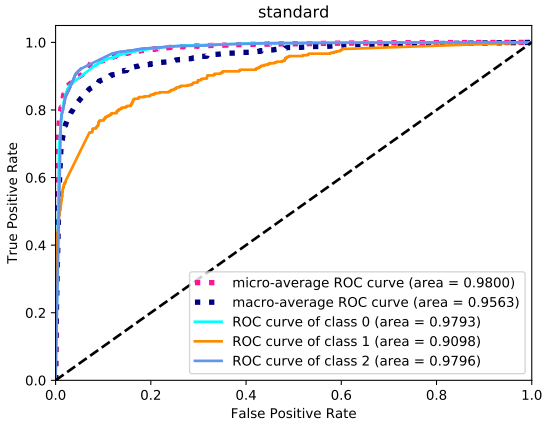

Figure 6.14: ROC Graph - Standard ADAM

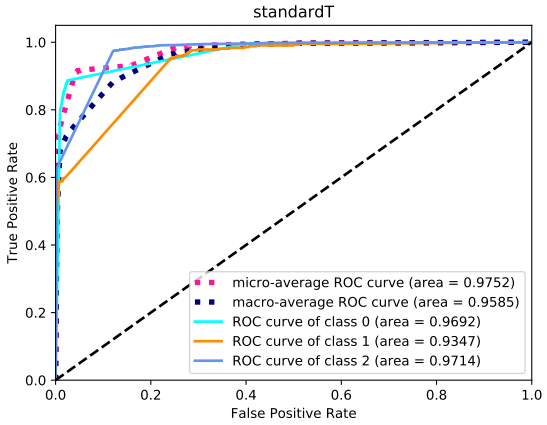

Figure 6.15: ROC Graph - StandardT ADAM

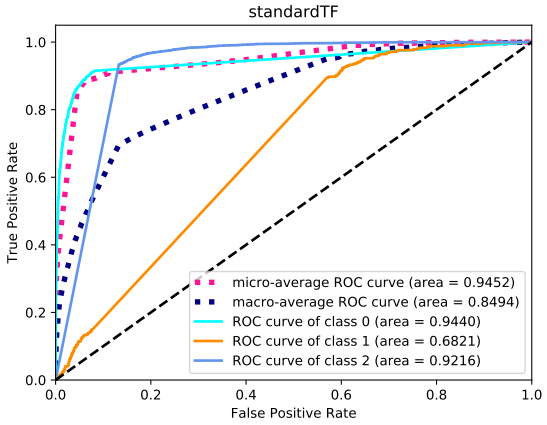

Figure 6.16: ROC Graph - StandardTF ADAM

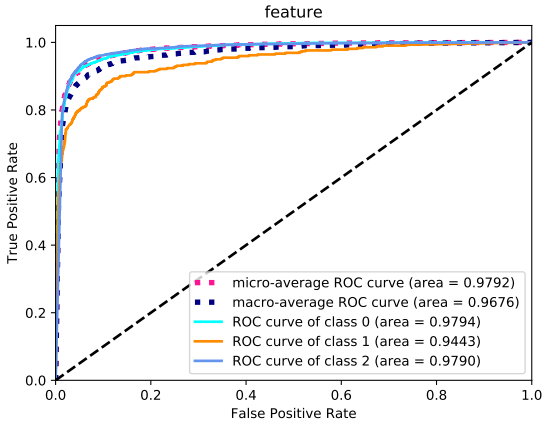

Figure 6.17: ROC Graph - Feature ADAM

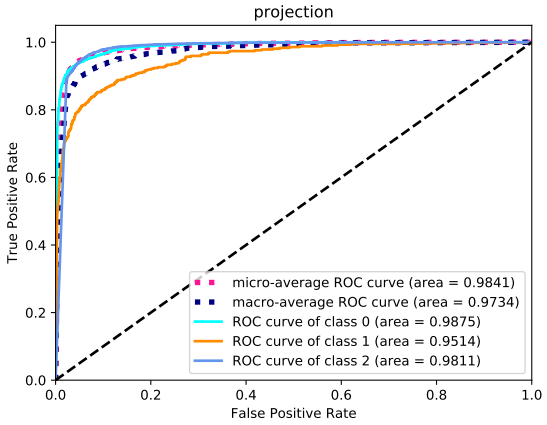

Figure 6.18: ROC Graph - Projection ADAM

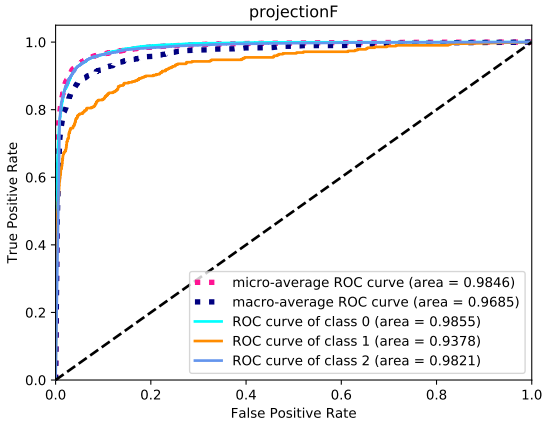

Figure 6.19: ROC Graph - ProjectionF ADAM

Chapter 7

Applications of ProjectionNet

As previously mentioned in the Literature Review Section, the motivation behind multi-modality can be broken down into three key factors. The first key factor is that a multi-modal prediction can potentially be more robust, meaning that multi-modal predictions tend to be stronger. This makes sense intuitively since we have more information, we should be able to more accurately assess the situation. The second key factor is a multi-modality having access to complementary information. There is this train of thought that given an appropriate fusion, multi-modalities should be greater than the sum of its parts. The third and final key factor when using a multi-modal model is the ability to operate even when lacking a source of information. In this chapter, we focus on this final key factor and how our method works in conjunction with this idea.

Not too long ago, few would have imagined modern society to be what it has become today. We went from a time when our cell phones could only make calls; to the present where they are considered minicomputers. We went from a time when the internet was only used for work to now when everything is connected to the internet. As we can see from the trends, our lives are becoming increasingly integrated with technology by the day, and with that comes a lot of data.

In the twenty-first century, data can be considered the new electricity, necessary for everyday life and ubiquitous. Every exchange and interaction gives birth to new data. As people in the modern age, we consume technology every second of our lives, therefore we produce data every second as well. As more and more data is being tracked, stored, and

analyzed there comes the need for multi-modal learning. Being part of a data-driven society, we are presented with challenges that are related to data. For the remainder of this chapter we address some of the potential issues and how our multi-modal framework can prevent such problems.

As we were designing this model, we had a goal in mind, and that was to create a drone that scans its surroundings for potential threats. Unfortunately, when operating such a task there could exist many potential hazards. Such hazards include terrain, weather, or even an opposing enemy. When such a threat is present, it would be naive to believe that our drone can come out of such a situation totally unscathed. In many scenarios, our drone would be damaged and corrupted, which leads to poor feedback to the user. If a multi-modality was built into the drone, problems such as a corrupted modality could exist if any of those events happen.

In cases where our drone has a K modality sensor and at least one modality sensor gets corrupted or eliminated, then we say this problem is a modality knockout. When such an event happens, our multi-modal network has the ability to address such problems. This is due to the fact that our model has the ability to reallocate the weights in a way such that the corrupted modality has no effect on the model. This is all made possible by the usage of a projection module. Our projection module can be used to reassign weights in the event that modality knockout occurs.

The need for multi-modal learning becomes more apparent, as increasing amounts of data is being tracked and stored. This paradigm brings new challenges, and in this section, we address a potential problem and how our model can prevent it. If $k \leq K$ is the modality that is knocked out then the non-knocked-out weights, as elements, are updated by sending a vector to the projection module. Since the projection module is bound above by $O(KlogK)$ this reassigning process is also bounded above by $O(KlogK)$.

The idea of modality knockout is not unique to drones as such problems can exist in any field that uses multi-modalities. In the field of security, technology that uses multi-modal surveillance such as temperature tracker with cameras or even multi-modal authentication can have cases of modality knockout. In the medical field, multi-modal imaging and multi-modal tests can also have a modality knockout. Having a model that can reassign weights

in the cases of modality knockout is beneficial because it provides a necessary safety net for those unforeseeable situations. This can be a valuable application in the modern world since so much of our lives are ingrained with technology. With the growth of the internet, multi-modalities is surely the way of the future, and with that, modality knockout is something that we must pay attention to.

7.1 Knockout

7.1.1 Knockout Experiment

For our knockout experiments we test the effectiveness of each network, by simulating a modality knockout event. To simulate a modality knockout event, we must first train the models that we need to test. These models are trained with parallel data inputs, meaning all modalities are fed in at once. To replicate the event of a modality knockout, we zero out the parameters connecting the knockout modality to the fusion layer. The zero-out procedure prevents the knockout modality from contributing any representation to the fusion layer as well as the overall output. We allow other modalities inputs to continue to forward propagate, and no other parameters are adjusted for the network. When inputting the data, all the different modalities are inputted at the same time. Recall that all the networks being used for these experiments are multi-modal with a total of six different multi-modal networks. Each network is tested with four different modality knockouts, which includes r, g, b, and infrared. By the end of it there should be a total of twenty-four different modality knockout occurrences. For this experiment, we want to see the effectiveness of the simplex projection module as it is implemented into our own models. Recall that the simplex projection module allows the parameters inside the fusion layer to redistribute itself when the weighted-sum constraint is broken. Our hope is that when a modality knockout occurs, the projection module redistributes the parameters in a manner so it always performs as well as the best single modality.

7.1.2 Knockout Experimental Results

Based on the results from our knockout experiments, we can see that the models that use the projection module have the highest testing accuracy in the event that modality knockout occurs. From the table, we can see that the standard training procedure scores the lowest out of the six multi-modal models. It also has a surprisingly low score in the case where the red modal gets knockout, achieving a test accuracy of sixty-nine percent. We believe this rare occurrence is due to the lack of transfer learning, since every other model seems to reach a more appropriate testing accuracy. Besides the standard multi-modal model, every other models' performance matches our expectations. For standardT, standardTF, and feature, these models reach a testing accuracy between the ranges of eighty-six to ninety percent in the case where red, green, or blue modality gets knockout. This is to be expected because when we applied transfer learning to the models, infrared had the best performance which makes losing red, green, and blue less impactful. Since the more significant modality is still intact with the network, it can still derive enough significant information to make a reasonable prediction. Since infrared is the most significant modal for our dataset, we decided to improve the effectiveness of our network by dropping infrared as well. Once again, the results put the standard model at the bottom with a testing accuracy of fifty percent. Our other models such as standardT and standardTF, achieve a testing accuracy between the range of seventy-seven to seventy-nine percent, while features receive an accuracy of seventy-two percent. From these results, we can see the effectiveness of multi-modal networks and its robustness to knockouts. If we observe the results for our uni-modal experiments, we can see that red, green, and blue average a testing accuracy of around seventy-eight percent. We can also observe that infrared average testing accuracy is around eight-nine to ninety percent. These results are directly related to our results from knockout if we compare our table with the graph. We can see in the event that red, green, or blue gets knocked out then the testing accuracy converges to infrared. This also holds true if infrared gets knocked out. From our table, we can also see the results for multi-modal networks that utilize projection modules. We can see for the most part that models that use the projection module perform better than models that don't. On average there is around a three percent increase in testing

	KO R	KO G	KO B	KO IR	KO AVG
STANDARD	69.6326	89.0622	81.5580	50.7107	72.7409
STANDARDT	86.4500	86.9761	89.5883	79.1489	85.5408
STANDARDTF	86.0439	88.3791	89.5052	77.5798	85.3770
FEATURE	89.3021	90.0221	89.3944	71.9217	85.1601
PROJECTION	91.7389	91.2774	87.9638	81.4103	88.0976
PROJECTIONF	91.5820	90.5482	88.0930	81.6503	87.9684

Table 7.1: Knockout Experiments SGD

	KO R	KO G	KO B	KO IR	KO AVG
STANDARD	89.9852	89.9852	87.8992	46.0310	78.4752
STANDARDT	88.4807	91.1944	89.9575	65.5990	83.8079
STANDARDTF	86.4592	87.2161	86.5423	66.8358	81.7634
FEATURE	90.9082	91.6559	87.3639	69.6603	84.8971
PROJECTION	92.7543	92.7451	90.4283	81.1334	89.2653
PROJECTIONF	92.2928	92.0158	89.5883	73.1585	86.7638

Table 7.2: Knockout Experiments ADAM

accuracy. It is important to take notice that the projection module performs better than its uni-modal counterparts. While other multi-modal networks testing accuracy converges to the non-missing modalities, the models that use projection modules exceed that performance. This result is most likely due to how projection models are being trained. Our speculation is that our training procedure allowed our network to learn a small number of complementary features, thus allowing it to perform better than its uni-modal counterparts. While on average the projection model performs better, in cases where blue gets knockout, projection performs the worst. We don't really have any idea why this happens, but the results are off by one percent. From our findings, we can see that multi-modal networks are naturally resilient to modality knockout. By providing our projection module, our model is performing better than other multi-modal training methods even in the case of a modality knockout.

Chapter 8

Conclusion and Future Works

We have presented a novel training framework for deep fusion multi-modal networks. Our proposed framework uses a narrowing procedure that only allows the optimizer to consider good solutions. The narrowing procedure incorporated many different multi-modal learning and machine learning techniques. We address various challenges within the field of multi-modalities by creating the projection feature mixture model. For representation we attempt to summarize multi-modal data using a joint representation by projecting both RGB and infrared into the same space. We tackle the issue of fusion by extending the projection module into the fusion layer. We integrate late stage fusion into our network. Lastly, we utilize transfer learning and experimented with partial freezing for co-learning.

We also created a projection module that works in conjunction with our framework. The module allows our network to continue using modern techniques such as stochastic gradient descent, since it can project the updated weights into the appropriate space. Besides developing a training framework, we also delve into how multi-modalities can be used in the event of a modality knockout. We also show that the network which uses our framework with projection module performs better than baseline in both classification and knockout.

Finally, for future works, we would like to explore different forms of fusion techniques in the coordinated space by applying our current projection module into other neural networks such as recurrent neural networks. We would also like to test our projection module on non-parallel data and see what kind of results it can achieve.

Appendix A

Object Detection

A.1 Object Detection

Object Detection is the study of locating objects inside a visual content such as pictures and videos[44]. Unlike object classification this task is not as intuitive, that is because object detection needs to accomplish two different tasks: identify the object as well as the location of the object. This creates a lot more work because now we must also store the coordinates for the given objects. In order for an object to be considered successfully detected, the classification and bound region must meet some criteria. This is because even if we locate the object correctly yet lack confidence on what the object is, it would still be a poor model. This applies to the other way around as well. So there has been much research done on object detection, but as of right now most algorithms utilize CNN. Some of the more advanced object detection models are faster RCNN [45] [46], Single Shot Detection (SSD)[47], and You Only Look Once (YOLO). Because our model utilizes YOLO as a base network, we will briefly overview some important features on YOLO.

A.2 YOLOv3

YOLOv3 is a real-time object detection algorithm that is part of the YOLO family tree. It is the third iteration of YOLO. Compared to the previous two versions; YOLOv3 has improved in accuracy as well as detecting small objects. It is important to take notice that YOLOv3 is a fully convolutional neural network, meaning that every layer inside the network is a

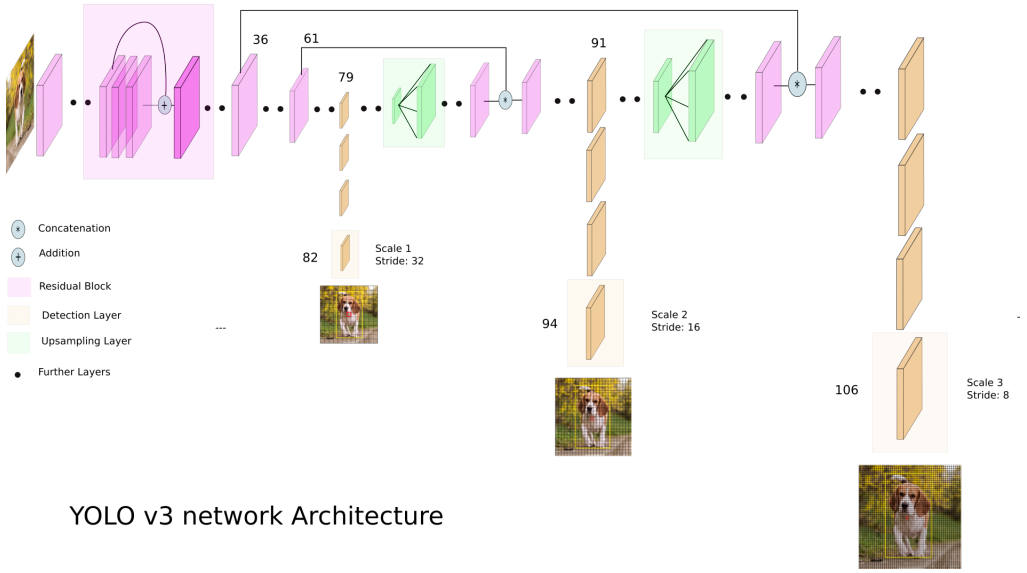

Figure A.1: YOLOv3

convolution layer. This allows the network to be size-invariant with its input. (It should be noted that we use tiny-yolo for our base model. This model includes a pooling layer, but it still holds all the same properties of YOLOv3.) YOLOv3 also makes detection at different scales, which allows it to focus on detecting objects of different sizes. The addition of detection in different scales vastly improves YOLOv3 performance in detection of small objects. YOLO also adopted an idea from ResNet known as a residual block. This allowed the network to feed features from earlier stages to the later stages of the network. For large networks, learning good features at the deeper layers is very difficult and the residual block solves that problem. This allows the network to supplement features to the end of the network. For the rest of the YOLOv3 section we will go over some key components and techniques that utilize this model.

A.2.1 Anchors

When detecting objects within a visual content such as pictures or videos, we usually identify the detection by drawing a box around the detected object. These boxes are often referred to as bounding boxes in object detection. In order to draw these boxes around the objects,

we are required to find the coordinates of the object relative to the visual content. In the past, there were many different techniques in detecting and drawing these bounding boxes. A common technique would be the sliding window with spatial pyramids. This is a technique that is very computationally heavy due to the large amount of input data it can generate for a given visual content. This happens because given a single image it will create an input for each iteration of the sliding window, on top of that it will repeat this procedure for each level of the spatial pyramid. Since objects can take on various sizes, a special pyramid is often needed for older object detection algorithms. Another prominent technique that brought forth a lot of different research is the region proposal method. The region proposal aims to solve a numerous amount of computation that is done by regulating sliding window methods. Like the sliding window technique, regional proposals input different parts of the visual content into a classifier. The main difference and contribution are that regional proposals only provide inputs where the likelihood of an object being there is high. This idea allowed models to eliminate unnecessary inputs that often ramp up computation time for older object detection techniques. While regional proposals had a serious impact on reducing the total number of computations, it is still not the most effective method to date. Popular object detection such as YOLO and SSD utilizes anchors, which could be viewed a pre-determine bounding boxes. Anchors are usually defined before the model and are spread through the visual content. It is also important to note that while there can be different variations of boxes, this is often done by initializing anchor boxes with different aspect ratios. By having anchor boxes with different aspect ratios, it allows the model to detect multiple objects within the same area and different anchors can specialize at detecting different objects. When using anchor boxes, we usually try to find the relative offset rather than the absolute position. In theory, learning the absolute position of a bounding box should be the same, but in practice neural networks struggle with this, thus leading to a vanishing gradient problem.

A.2.2 Intersection over Union

Intersection over Union is a technique that is very prevalent in the field of object detection. The purpose of Intersection over Union is to determine how relevant the predicted bounding

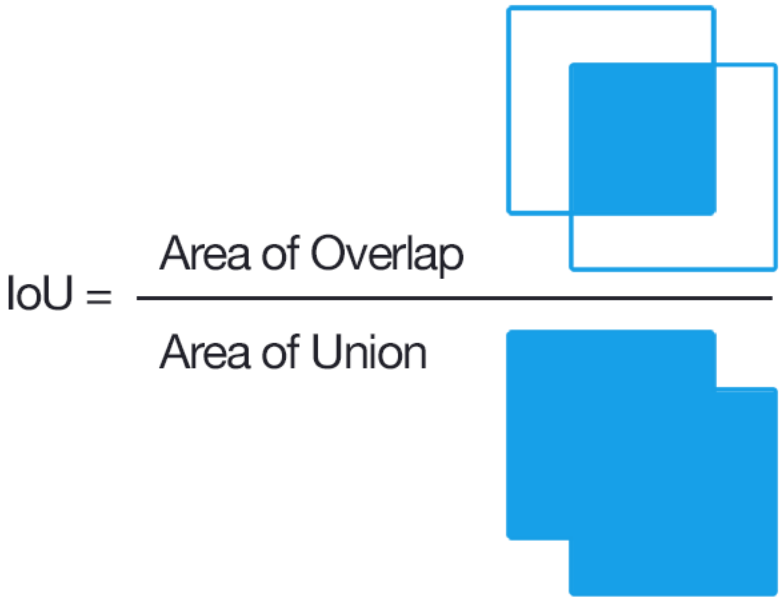

Figure A.2: IoU

box is to the given visual content. When training for object detection, the labels for the bounding boxes are usually given in a coordinate system. The object detection model learns the relative location of these coordinates and provides a bounding box whenever it detects an object. However, when going through a detection layer it is possible for the model to make predictions that are incorrect and intersection over union solves this problem. In order to perform intersection over union, we first need to find the area of the bounding boxes and then use these areas to determine union and intersection. The intersection of the area is just the area shared between the predicted and the targeted. It could be solved by intersection = predict - (target - predict). The union is the overall area of these two bounding boxes, and this could be solved by union = predict + target - intersection. With these two values we IoU by intersection / union, which will give us a value between 0 and 1. At this point it is up to the designer to determine a cut of point for also known as a threshold. The threshold is used to determine if the prediction is good or bad. In most cases the threshold is at least 0.5, therefore any boxes with a Iou less than 0.5 will be discarded. It is important to recognize that IoU is used to filter bounding boxes that have no chance in detecting an object. [48]

A.2.3 Non-Maximum Suppression

While performing object detection, our network will produce many bounding boxes, some of which is irrelevant to our detection. To put this into perspective, YOLOv3 usually divides its images into some n x n grid, where each cell within the grid contains several anchors (a). So, when an image goes through a YOLOv3 object detection algorithm, it will produce n x n x a bounding boxes across the image. As we saw in the previous section, intersection over union removes boxes that are not relevant to our prediction. Since intersection over union is heavily reliant on the threshold parameter, depending on the threshold we still have many bounding boxes. As mentioned IoU only filters out boxes that cannot be a detection, so we are still left with many boxes that can be a detection. To solve this problem, we need to use a technique called Non-Maximum Suppression. Non-Maximum Suppression works by finding the bounding box with the highest confidence score and setting it as the main detector. It then uses IoU to remove all extra bounding boxes that have a high intersection with the main detector. This is because the main detector is objectively the best bounding box due to its confidence score, and a high IoU indicates that these other boxes are detecting the same object, therefore NMS removes all boxes that are highly intersected. [49]

Appendix B

Transfer Learning

B.1 Overview

Transfer learning is a technique that is often used in the field of deep learning. The core aspect behind transfer learning is to leverage a well established model as the foundation for a new learning process. This approach adopted its idea how humans learn tasks from prior knowledge if applicable. For example, let us say there is an individual who learned to play basketball at a very early age, and that individual decided to switch sports later in life. It would be strange to believe that the individual needs to start from scratch because these two sports have transferable skills. While these two sports are not identically the same; there are still enough similarities such that his previous experiences can have an impact. Techniques such as running, passing, and teamwork are all fundamental skills that are needed for both sports. This is the fundamental idea behind transfer learning, and in later sections we will go over the benefits and types of transfer learning.

B.2 Benefits

Applying transfer learning provides many advantages to the model. One important key advantage is the amount of training time. A deep neural network that takes advantage of transfer learning vastly reduces its training time. This is because transfer learning provides the model with powerful features, therefore the network will require less time to converge. Another benefit is reducing the amount of training data needed. This is extremely important

because the inherent weakness of neural networks are small datasets. With transfer learning the issue of a small dataset can be ignored. As we can see, transfer learning allows us to avoid common pitfalls that plagues neural networks, especially multi-modal neural networks. This is because multi-modal data is extremely rare to begin with, especially parallelized data, therefore transfer learning can help us avoid such problems. Lastly, transfer learning aids the model in generalizing. Since the transfer parameters are trained on an entirely different dataset or possibly a different task; models that utilizes transfer learning will be less specific compared to models that do not. [50]. This is important since generalized models often have better features allowing them to work with a wider range of datasets.

B.3 Types of Transfer Learning

Within the procedure of transfer learning there are also different types of transfer learning. There is domain adaptation which trains the model under the same task but performs on a different domain. There is multitask learning where we transfer knowledge between related tasks. There is Zero-Shot learning where the model will try to solve a task that it is not exposed to. There are more types of transfer learning aside from the aforementioned brief previews.

Bibliography

[1] W. Wang, D. Tran, and M. Feiszli, "What makes training multi-modal networks hard?" *arXiv preprint arXiv:1905.12681*, 2019.

[2] A. Owens and A. A. Efros, "Audio-visual scene analysis with self-supervised multisensory features," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 631–648.

[3] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[4] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7794–7803.

[5] T. Baltrušaitis, C. Ahuja, and L.-P. Morency, "Multimodal machine learning: A survey and taxonomy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 2, pp. 423–443, 2018.

[6] A. Torfi, S. M. Iranmanesh, N. Nasrabadi, and J. Dawson, "3d convolutional neural networks for cross audio-visual matching recognition," *IEEE Access*, vol. 5, pp. 22 081–22 091, 2017.

[7] Y. Sato and N. Hamada, "Multi-modal speech recognition using correlativity between modality," in *2010 International Symposium on Intelligent Signal Processing and Communication Systems*, 2010, pp. 1–4.

[8] S. Nakamura, K. Kumatani, and S. Tamura, "Multi-modal temporal asynchronicity modeling by product hmms for robust audio-visual speech recognition," in *Proceedings. Fourth IEEE International Conference on Multimodal Interfaces*, 2002, pp. 305–309.

[9] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 664–676, 2017.

[10] D. Ramachandram and G. W. Taylor, "Deep multimodal learning: A survey on recent advances and trends," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 96–108, 2017.

[11] R. Kiros, R. Salakhutdinov, and R. S. Zemel, "Unifying visual-semantic embeddings with multimodal neural language models." *CoRR*, vol. abs/1411.2539, 2014. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1411.htmlKirosSZ14

[12] S. Kaya and E. Vural, "Multi-modal learning with generalizable nonlinear dimensionality reduction," in *2019 IEEE International Conference on Image Processing (ICIP)*, 2019, pp. 2139–2143.

[13] T. Y, T. Y, M. Y, J. Y, and K. Tokuda, "Text-to-visual speech synthesis based on parameter generation from hmm," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 6, 08 1998.

[14] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard, "Multimodal deep learning for robust rgb-d object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 681–687.

[15] S. Poria, E. Cambria, and A. Gelbukh, "Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 2539–2544. [Online]. Available: https://www.aclweb.org/anthology/D15-1303

[16] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006. [Online]. Available: http://www.ncbi.nlm.nih.gov/sites/entrez?db=pubmeduid=16873662cmd=showdetailviewindexed=g

[17] K. Chen, M. Weinmann, X. Sun, M. Yan, S. Hinz, and B. Jutzi, "Semantic segmentation of aerial imagery via multi-scale shuffling convolutional neural networks with deep supervision," vol. IV-1, pp. 29–36, 09 2018.

[18] V. Vielzeuf, A. Lechervy, S. Pateux, and F. Jurie, "Centralnet: a multilayer approach for multimodal fusion," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 0–0.

[19] A. Tsanousa, G. Meditskos, S. Vrochidis, and I. Kompatsiaris, "A weighted late fusion framework for recognizing human activity from wearable sensors," in *2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA)*, 2019, pp. 1–8.

[20] F. Farahnakian, J. Poikonen, M. Laurinen, and J. Heikkonen, "Deep convolutional neural network-based fusion of rgb and ir images in marine environment," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 21–26.

[21] S. Poria, E. Cambria, and A. Gelbukh, "Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis," 09 2015.

[22] J. Mao, W. Xu, Y. Yang, J. Wang, and A. L. Yuille, "Deep captioning with multimodal recurrent neural networks (m-rnn)," *CoRR*, vol. abs/1412.6632, 2014.

[23] WIKI. (2019) History of artificial intelligence. [Online]. Available: https://en.wikipedia.org/wiki/Convolutional$_neural_network$

[24] M. W. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences," *Atmospheric environment*, vol. 32, no. 14-15, pp. 2627–2636, 1998.

[25] B. Mehlig, "Artificial neural networks," *CoRR*, vol. abs/1901.05639, 2019. [Online]. Available: http://arxiv.org/abs/1901.05639

[26] WIKI. (2019) Convolutional neural network. [Online]. Available: https://en.wikipedia.org/wiki/History$_of_artificial_intelligence$

[27] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Neural Information Processing Systems*, vol. 25, 01 2012.

[28] P. Sidike, M. Z. Alom, T. Taha, and V. Asari, "The history began from alexnet: A comprehensive survey on deep learning approaches," 11 2018.

[29] Pytorch. (2019) Training a classifier. [Online]. Available: https://pytorch.org/tutorials/beginner/blitz/cifar10$_tutorial.html$

[30] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," 01 2010, pp. 92–101.

[31] WIKI. (2019) Rectifier (neural networks). [Online]. Available: https://en.wikipedia.org/wiki/Rectifier$_(neural_networks)$

[32] A. F. Agarap, "Deep learning using rectified linear units (relu)," 2018, cite arxiv:1803.08375Comment: 7 pages, 11 figures, 9 tables. [Online]. Available: http://arxiv.org/abs/1803.08375

[33] WIKI. (2019) Softmax function. [Online]. Available: https://en.wikipedia.org/wiki/Softmax$_function$

[34] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[35] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[36] ——, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[37] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[38] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[39] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[40] W. Wang and M. A. Carreira-Perpinán, "Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application," *arXiv preprint arXiv:1309.1541*, 2013.

[41] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[42] S. Shalev-Shwartz and Y. Singer, "Efficient learning of label ranking by soft projections onto polyhedra," *Journal of Machine Learning Research*, vol. 7, no. Jul, pp. 1567–1599, 2006.

[43] FLIR. (2019) Free flir thermal dataset for algorithm training. [Online]. Available: https://www.flir.com/oem/adas/adas-dataset-form/

[44] Matlab. (2019) Object detection. [Online]. Available: https://www.mathworks.com/discovery/object-detection.html

[45] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013. [Online]. Available: http://arxiv.org/abs/1311.2524

[46] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015. [Online]. Available: http://arxiv.org/abs/1506.01497

[47] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015. [Online]. Available: http://arxiv.org/abs/1512.02325

[48] S. H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. D. Reid, and S. Savarese, "Generalized intersection over union: A metric and A loss for bounding box regression," *CoRR*, vol. abs/1902.09630, 2019. [Online]. Available: http://arxiv.org/abs/1902.09630

[49] J. H. Hosang, R. Benenson, and B. Schiele, "Learning non-maximum suppression," *CoRR*, vol. abs/1705.02950, 2017. [Online]. Available: http://arxiv.org/abs/1705.02950

[50] K. Weiss, T. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big Data*, vol. 3, 12 2016.

Curriculum Vitae

Graduate College
University of Nevada, Las Vegas

Henry Ng
henryng.shiu@gmail.com

Degrees:

Bachelor of Art in Computer Science 2018
University of Nevada Las Vegas

Thesis Title: Towards Multi-Modal Data Classification

Thesis Examination Committee:

Chairperson, Dr. Justin Zhan, Ph.D.
Committee Member, Dr. Ju-Yeon Jo, Ph.D.
Committee Member, Dr. Fatma Nasoz, Ph.D.
Graduate Faculty Representative, Dr. Ge Kan, Ph.D.