On the Robustness of Learned Task Weights in Cross-modal Retrieval

David Nahm
Gerald Friedland, Ed.
Kannan Ramchandran, Ed.

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2020-86
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-86.html

May 28, 2020

Copyright © 2020, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

On the Robustness of Learned Task Weights in Cross-modal Retrieval

by

David Nahm

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Gerald Friedland, Chair
Professor Kannan Ramchandran, Second Reader

Spring 2020

The dissertation of David Nahm, titled On the Robustness of Learned Task Weights in Cross-modal Retrieval, is approved:

Chair _____ Date 5/28/2020

_____ Date 5/28/2020

_____ Date _____

University of California, Berkeley

On the Robustness of Learned Task Weights in Cross-modal Retrieval

Copyright 2020
by
David Nahm

Abstract

On the Robustness of Learned Task Weights in Cross-modal Retrieval

by

David Nahm

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Gerald Friedland, Chair

Multi-task learning has proven to be effective when applied to cross-modal retrieval, the process of making a query in one modality and retrieving relevant results in a different modality. However, the performance of multi-task learning can vary greatly depending on hand-tuned task weights. These task weights determine how much each task in the cross-modal retrieval model contributes to the model's overall loss function and training process. Recent studies have shown that these task weights can be learned during a model's training process. The learned task weights are a measure of the amount of uncertainty in each task, as the model should learn to de-emphasize more uncertain tasks and focus its learning on tasks with less uncertainty. These learned task weights can be particularly useful in the setting of cross-modal retrieval because multimedia datasets commonly contain large amounts of label noise and uncertainty, so task weights should reflect this uncertainty accordingly. However, the behavior of the model with learned task weights could be very unpredictable in the presence of label noise. In this report, we first show how learned task weights react to noisy polynomial data with added label noise, and we compare these results to a multi-task learning model with hand-tuned weights. We also analyze the conditions in which learned task weights are robust to label noise in image classification tasks. We then apply these findings to a cross-modal retrieval model to better understand how factors such as the complexity of the tasks and the modalities being used affect how robust the learned task weight model is to label noise. We find that with high noise datasets, uncertainty-based learned task weights are insufficient when learning task weights and that task complexity must also be considered. Task complexity should also be accounted for in the form of dimensionality reduction or as a learned parameter when learning task weights with noisy data.

Contents

Contents ii

List of Figures iv

List of Tables v

1 Introduction and Motivation 1

2 Related Work 4

3 Background 6
- 3.1 Cross-modal Retrieval . . . 6
- 3.2 Multi-task Learning . . . 9
- 3.3 Learned Task Weights . . . 11

4 Learned Task Weights with Noisy Polynomials 14
- 4.1 Experimental Setup for Noisy Polynomials . . . 14
- 4.2 Experiments with Noisy Polynomials . . . 15

5 Learned Task Weights with Image Datasets 20
- 5.1 Experimental Setup with MNIST Dataset . . . 20

6 Cross-modal Retrieval with Learned Task Weights 26
- 6.1 Datasets . . . 26
- 6.2 Loss Formulation . . . 26
- 6.3 Network Architecture . . . 27
- 6.4 Model Evaluation . . . 29
- 6.5 Results . . . 29

7 Conclusion 32
- 7.1 Lessons Learned . . . 32
- 7.2 Future Work . . . 34

Bibliography **36**

List of Figures

3.1 Triplet loss will encourage the model to pull the positive example closer to the anchor while pushing the negative example away from the anchor [26]. 7
3.2 Learning a common subspace helps group conceptually similar embeddings despite being from different modalities [27]. 8
3.3 Example of a multi-task learning model with soft parameter sharing [28]. 9
3.4 Example of a multi-task learning model with hard parameter sharing [28]. . . . 10

4.1 Network architecture used in the polynomial experiments. 15
4.2 Average minimum validation loss on polynomials of degree 5, 8, and 10 as we increase the amount of noise. The performance of a hand-tuned multi-task learning model is overlaid as a baseline. 16
4.3 Average minimum validation loss on polynomials of degree 5, 8, and 10 for a fixed number of tasks with increasing noise. 18

5.1 Network used to classify MNIST images. 21
5.2 MNIST performance with increased levels of noise and increasing number of tasks. 22
5.3 MNIST performance with increased levels of noise on a model with 3 tasks. . . . 23
5.4 MNIST learning curves for $\sigma_3^2 = 1, 2, 3, 4$ 24

6.1 Sample images and captions from the IAPR TC-12 (left) and MIRFlickr-25K datasets (right) [29] [30]. 27
6.2 Network used to perform cross-modal retrieval. d is the dimensionality of the common subspace and is varied from d = 16, 32, 64. 28

List of Tables

4.1 Example learned task weights for a degree 10 polynomial averaged over 10 random seeds. 17

6.1 MAP scores for IAPR TC-12 and MIRFlickr-25K datasets. d is the dimensionality of the joint embeddings used in the common subspace. 30

Acknowledgments

I would first like to acknowledge Professor Gerald Friedland for his constant support and mentorship throughout this project. I am very appreciative of all of his valuable ideas and feedback, and, most of all, how much he truly cares about the well-being and growth of each of his students. I would also like to thank Professor Kannan Ramchandran for his feedback and insights as a faculty reader. I am also very appreciative of Dr. Jaeyoung Choi not only for taking the time to teach me about the theory and practical aspects of multimedia computing, but also for his guidance and mentorship throughout this project. Last but not least, I would like to acknowledge all of my friends and family for their never-ending support throughout this entire process.

Chapter 1

Introduction and Motivation

Cross-modal retrieval is the process of making a query in one modality and retrieving relevant results from a different modality. Cross-modal retrieval models are commonly trained on multimedia datasets such as YFCC100M which contain multiple modalities such as images, text, and videos [1]. Having to deal with multiple modalities makes cross-modal retrieval a more difficult task than traditional retrieval tasks with one modality. The cross-modal retrieval model must be able to relate data that are similar conceptually regardless of what modality they belong to. For example, a cross-modal retrieval model must be able to learn that an image of a dog and a text description of the dog represent the same concept despite being from different modalities. Cross-modal retrieval has proven to be useful in many different settings dealing with several modalities. Several recent applications of cross-modal retrieval include matching images of faces with audio of voices and searching for videos with audio snippets [2, 3].

Multi-task learning is a specific technique that has improved cross-modal retrieval performance [4]. Multi-task learning models learn by simultaneously learning from several tasks. By doing this, the model should be able to learn more general representations that apply to all of the tasks instead of simply overfitting to one task. This idea of learning more general representations fits naturally with cross-modal retrieval because we are trying to learn general concepts that appear in the data regardless of modality. We can also use the idea of multiple tasks to take advantage of more data in our cross-modal retrieval model. When training our cross-modal retrieval model in a multi-task setting, we train each task with a different dataset. For example, task A of our multi-task learning model would try to perform cross-modal retrieval on dataset A, while task B would try to perform cross-modal retrieval on dataset B. By allowing the model to learn from multiple datasets, we would hope that the model could learn more general features instead of overfitting to one dataset.

While multi-task learning has proven to be useful, it can be hard to obtain the best performance because it is difficult to optimally choose how we weigh the tasks in a multi-task learning model. Prior work has shown that multi-task learning model performance

is particularly sensitive to how tasks are weighted [5]. How we choose to weigh the tasks is crucial because this decides which tasks the model will focus its learning on during the training process. Tasks with smaller task weights contribute less to what the model learns, while tasks with larger task weights are emphasized more in the learning process. In order to hand-tune these task weights, we must perform an expensive and tedious hyperparameter search over all of the possible task weights. Not only is this a time-consuming process, but these hand-tuned task weights also tend to be biased towards whole numbers or to one decimal point. However, if we could learn these task weights while training our model, we would not need to spend time on hyperparameter searches for the optimal task weights and we could also possibly see benefits in model performance. [5] proposes a specific method of learning task weights based on uncertainty that we focus on in this report. The task weights are included as additional learned parameters for the model to learn while being trained. By learning these task weights, the model can learn how to weigh the uncertainty of each task instead of relying on a hyperparameter search to find the best task weights. In addition, the learned task weights are able to achieve a level of granularity that a programmer would not be able to achieve.

In order to apply the idea of learned task weights to cross-modal retrieval, it is important to understand how the learned task weight model will behave under different noisy settings. Many of the multimedia datasets used in cross-modal retrieval have significant amounts of label noise. These datasets have become increasingly large, so extensively verifying the label of each observation is impractical. Since each task uses a different dataset as mentioned earlier, each learned task weight should reflect the amount of noise in the dataset used for that task. With the amount of label noise in mind, we must ask several questions before determining if learned task weights can effectively be used for cross-modal retrieval models trained on these large multimedia datasets. Will learned task weights still be able to be learned effectively in large multimedia datasets with substantial amounts of label noise? At what level of label noise would the model no longer be able to accurately learn how to weigh tasks. Ideally, we would know the limits of a learned task weight model in the presence of label noise and actually understand how the model will behave. This report explores to what extent and the circumstances in which learned task weights in a multi-task learning model are robust to label noise. In particular, we apply a learned task weight model to increasingly difficult problems to ultimately observe how well learned task weights can perform on noisy multimedia datasets. By doing this, we hope to better understand the behavior of learned task weights in the presence of label noise so that we can reliably apply this technique to larger multimedia datasets with more noise.

In this report, we start by first providing background on the topics that are addressed in this project. In particular, we discuss the background behind cross-modal retrieval, multi-task learning, and learned task weights in order to inform the reader of the theory behind these topics that are used in our experiments. We then use a learned task weight model on noisy polynomials of different degrees. We wanted to first start with polynomials for our

initial experiments because the noise was easier to control and we could see general patterns in behavior more clearly. We examine if the learned task weights are still able to reliably weigh the model's tasks as we vary the amount of noise in the tasks. After conducting these experiments with noise polynomials, we then furthered our experiments by applying a learned task weight model to image datasets while varying the amount of random label noise. We wanted to extend our experiments to images because this was a more complex task than predicting polynomials, but not nearly as hard as performing cross-modal retrieval. In both the noisy polynomial and image experiments, we see the learned task weight model perform as well or better than a traditional multi-task learning model with hand-tuned weights up until a certain level of label noise. While the learned task weight model can perform just as well or better, we do observe a tradeoff of longer convergence times when compared to a fixed-weight multi-task learning model. Finally, we train a cross-modal retrieval model using learned task weights. We observe that the learned task weights are still robust to certain levels of label noise in this cross-modal retrieval setting. We find that this level of label noise is dependent on several factors, such as the complexity of the task and the modalities we are dealing with.

Chapter 2

Related Work

Since cross-modal retrieval can be used with many different modalities, there are many practical applications of cross-modal retrieval. This includes applications such as video-text retrieval [6], image captioning [7], and retrieving audio samples given a video query [8]. In this report, we focus on a joint representation learning approach to cross-modal retrieval where we attempt to learn a common subspace between all of the modalities. In the common subspace, embeddings representing the same concept should be close together regardless of what modality they come from. This general methodology is commonly used in cross-modal retrieval work [9, 10, 11, 12]. Once we learn this common subspace, we can efficiently perform retrieval between modalities in this new common subspace.

Multi-task learning was first introduced by [13] as a learning strategy to learn more general features from data. Multi-task learning has been used in a variety of different applications, not just limited to multimedia computing. The concept has shown to be effective when applied to a wide range of topics such as computer vision [14], natural language processing [15], and speech recognition [16]. In all of these applications, multi-task learning has proven to be helpful in learning more general representations and avoiding overfitting. While multi-task learning is an established technique that has been used for a while, learning the task weights in multi-task learning is a newer problem with more recent work being done. [5] proposes learning task weights by considering the uncertainty of each task as the model is being trained. This work shows how learned task weights can improve performance on semantic and instance segmentation tasks. This report primarily focuses on the uncertainty-based learned task weights from [5], but there are also other literature on the topic of learned task weights. [17] proposes a dynamic task prioritization approach to learning how to weigh tasks. They use the focal loss function to emphasize more difficult observations while learning. [18] tries to weigh how the model learns from each task by dynamically tuning gradient magnitudes as the model is learning. These techniques to learn task weights all achieve model performance that exceeds or matches the performance of a hyperparameter search such as grid search.

Multi-task learning has been effective in various applications of cross-modal retrieval. [19] uses a multi-task learning framework to perform image annotation, while [20] uses multi-task learning for image-text retrieval. These prior work use multi-task learning with hand-tuned task weights that were found through a hyperparameter search. The work by Choi et al. shows how learned task weights in multi-task learning can improve the performance of cross-modal retrieval systems with text, image, and video modalities [4]. We build on this work by studying how learned task weights in cross-modal retrieval systems react in situations with significant label noise in order to get a better understanding of model behavior.

There has also been significant prior work in learning in the presence of label noise. One of the common approaches to dealing with label noise is modifying the loss function. [21] propose a modified hinge loss to counteract symmetric label noise. There have also been various approaches to dealing with label noise that modify the network architecture or training process. [22] uses a "co-teaching" approach where two networks are trained and communicate to each other about which observations should be utilized in training. By doing this, the two networks can teach each other how to focus on observations that help the model learn. [23] propose modifying the network architecture by adding an additional layer at the end of the network which specifically aims to learn the distribution of the label noise.

Chapter 3

Background

3.1 Cross-modal Retrieval

Cross-modal retrieval is a specific application in multimedia computing where learned task weights can be particularly helpful. The idea behind cross-modal retrieval is to make queries in one modality and retrieve results in a different modality. For example, in a cross-modal retrieval system, one could retrieve relevant video results given an image query or find image results given a text query. While machine learning models have become very good at dealing with single modalities (e.g. finding similar images to a given image query), cross-modal retrieval across several different modalities is a more difficult task. The model must be able to learn how to relate data across modalities which is an added layer of complexity for the model to deal with. Cross-modal retrieval has many practical applications because of its multi-modal nature. Some applications of cross-modal retrieval include matching faces with voices, music retrieval, and image captioning [2] [24] [25]. Cross-modal retrieval is especially relevant in multimedia computing today because of cross-modal retrieval's ability to take advantage of several modalities. Multimedia datasets frequently contain multiple modalities. Image datasets frequently have text captions or user tags and videos contain valuable information in their text descriptions. Cross-modal retrieval provides us with a method to take advantage of this additional data that is stored in different modalities.

Cross-modal retrieval is commonly learned through joint representation learning. In this report, our cross-modal retrieval model uses a joint representation learning approach, so we will focus on this method of learning cross-modal retrieval models. The main difficulty when dealing with multiple different modalities is that each modality most likely has different dimensions and is formatted differently. This makes it difficult to build a model that can take in several modalities that are each structured differently. In order to avoid this issue, our goal is to learn a common subspace where embeddings from each modality can be easily compared to each other. There are two steps in learning this common subspace: intramodal optimization and intermodal optimization.

In intramodal optimization, our goal is to first learn the best embeddings within each modality. We would want the model to first be able to properly group similar concepts together without considering any of the other modalities. For example, after intramodal optimization for the image modality, we would want the model to understand that different images of dogs should be similar (close in Euclidean distance) to each other and that images of computers should be dissimilar (far in Euclidean distance) from those dog images. In order to perform intramodal optimization, we can take advantage of pre-trained networks to embed the raw input. Pre-trained networks are useful for this stage because pre-trained networks, such as pre-trained ResNet-18 or BERT, have already been optimized to numerically represent raw input as embeddings. These pre-trained networks can be fine-tuned as the cross-modal retrieval model is trained or additional layers can be added to the model to improve upon the pre-trained embeddings.

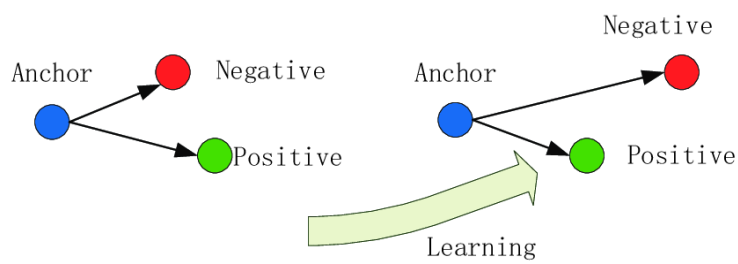


Figure 3.1: Triplet loss will encourage the model to pull the positive example closer to the anchor while pushing the negative example away from the anchor [26].

Once each modality is individually tuned using intramodal optimization, we can perform intermodal optimization to learn across different modalities. We can utilize loss functions such as the triplet loss to allow the model to learn embeddings for each modality in a common subspace. The "triplet" in triplet loss refers to input triplets of an anchor (A), positive example (P), and negative example (N). The positive example is an observation sampled from the same class as the anchor but from a different modality. The negative example is an observation sampled from a different class from the anchor and also from a different modality. The idea behind triplet loss is to pull the anchor and positive example together (despite them being from different modalities) and push the anchor and negative example farther apart. This concept can be visualized in Figure 3.1. Triplet loss is defined as:

$$L(A, P, N) = \sum_{i=1}^{n} \max(\left\|f(A^{(i)}) - f(P^{(i)})\right\|^2 - \left\|f(A^{(i)}) - f(N^{(i)})\right\|^2 + \alpha, 0)$$

where α refers to the margin used in triplet loss, and the size of the training data is n. The

triplet loss will only be non-zero for a particular triplet if

$$\left\|f(A^{(i)}) - f(P^{(i)})\right\|^2 > \left\|f(A^{(i)}) - f(N^{(i)})\right\|^2 - \alpha$$

In other words, the model should only incur a loss from this triplet if the distance between the anchor and the positive example is greater than the distance between the anchor and the negative example with some additional leeway provided by the margin α. By incurring a loss when the anchor is closer to the negative example than the positive example, the model is able to learn and pull the positive example closer to the anchor while pushing the negative example away from the anchor. By doing this, we can learn a common joint embedding space where embeddings are grouped by the concept they represent. As a result, embeddings from separate modalities, but representing the same concept, can still be near each other. This can be observed in Figure 3.2 as it is initially difficult to compare text and images when they are in separate subspaces, but it is easier to compare them in a common subspace.

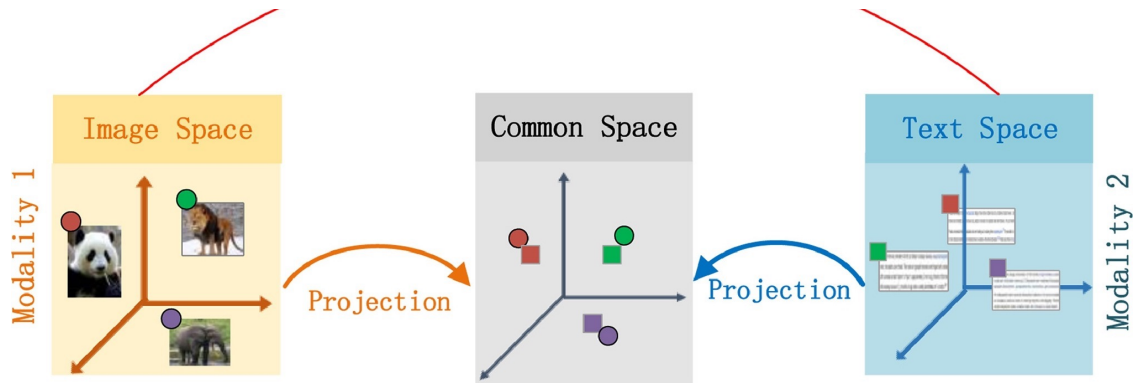


Figure 3.2: Learning a common subspace helps group conceptually similar embeddings despite being from different modalities [27].

Once we have completed this intermodal optimization, conducting queries can easily be done in this new common subspace. New queries can be mapped to this new learned subspace by passing the query through the network and evaluating where this query should lie in the learned subspace. Since embeddings are grouped by concept in this new learned subspace, we can use techniques such as k-Nearest Neighbors (k-NN) to find the closest results for a given query. As a result, retrieving results for a new query can be done efficiently because we would just have to perform one forward pass through the network, then perform k-NN.

Difficulties with Multimedia Datasets

When discussing the topic of cross-modal retrieval, it is important to think about the data that the models will be trained on. The quality of the cross-modal retrieval results can

depend heavily on the noisiness of the training data. Multimedia datasets tend to include large amounts of label noise and uncertainty because of the scale of these datasets and the difficulties in correctly labeling training data when these datasets are originally created. These datasets can contain millions of images, videos, or audio clips, so ensuring that each training observation is correctly labeled is impractical. As a result, many of these datasets, such as image datasets with captions, rely on user tags or labels which can be very subjective and at times inaccurate. Since there can be significant levels of label noise in these multimedia datasets, it is critical to understand how a model will behave in the presence of label noise before applying various machine learning techniques. If a cross-modal retrieval model cannot be relied upon to be robust to a certain level of label noise, then it becomes very difficult to predict how the model will behave on different multimedia datasets.

3.2 Multi-task Learning

Multi-task learning is an approach that allows a model to learn from multiple tasks simultaneously. In a traditional machine learning setting, the model is solely focused on one task when trying to learn how to solve a problem. In a multi-task learning setting, the model is able to benefit from learning from multiple tasks because information can be shared between tasks. The model can share information through either hard parameter sharing or soft parameter sharing. As depicted in Figure 3.3, soft parameter sharing allows each task to have its own model. This approach differs from simply training k separate models independently (where k is the number of tasks) by enforcing constraints on the parameters of the models. The model parameters can be constrained so that the parameters for each task are close in distance (i.e. L2 norm) to the parameters for each of the other tasks.

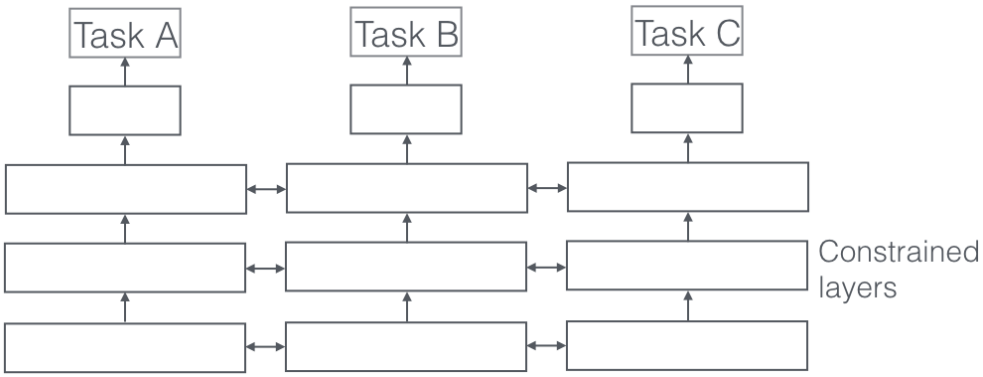


Figure 3.3: Example of a multi-task learning model with soft parameter sharing [28].

Hard parameter sharing differs in that a model with hard parameter sharing has shared layers that remain the same across all tasks. As shown in Figure 3.4, each task uses the same

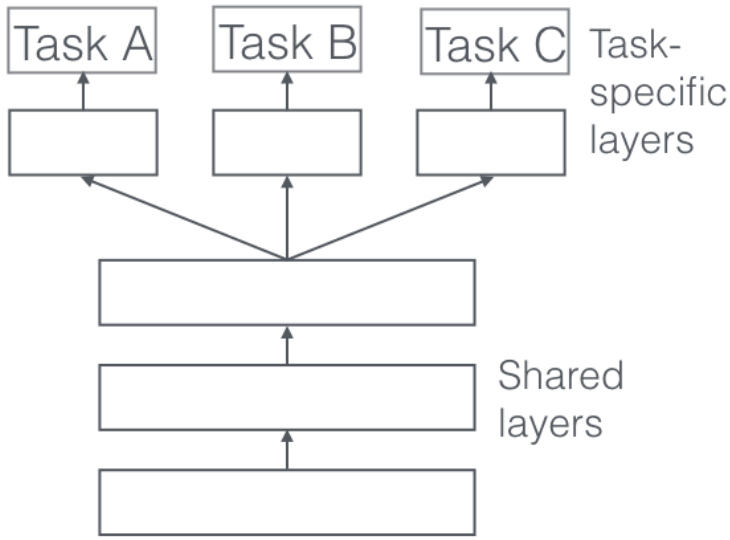


Figure 3.4: Example of a multi-task learning model with hard parameter sharing [28].

shared layers. After the shared layers, each task has a task head that is specific to each task. By doing this, hard parameter sharing can learn a general representation for all of the tasks in the shared layers, then learn more task-specific features in the individual task heads. This project will focus specifically on learned task weights for hard parameter sharing models, and multi-task learning models mentioned from this point on will be assumed to be hard parameter sharing models. In a model with hard parameter sharing, learning is still similar to a traditional single task machine learning model. The loss for the model is computed as a linear combination of each of the individual task losses.

$$L(x) = \sum_{i=1}^{T} \lambda_i L_i(x)$$

T is the total number of tasks used, λ_i is the weight on each task, and $L_i(x)$ is the loss for task head i. Each task weight, λ_i is chosen manually through a hyperparameter search such as grid search. Learning is commonly done by first computing each task loss, summing them, then backpropagating on this summed loss.

One of the main benefits of multi-task learning is its ability to avoid overfitting. By learning from multiple tasks, multi-task learning models are capable of learning more general patterns and representations. The additional tasks can help the model avoid overfitting to noise and generalize better because the model is not able to solely focus on one task. By using multiple tasks when learning, the model is able to take advantage of more data. In doing this, the model can learn a more general feature representation that takes all of the

tasks into account. Instead of simply overfitting to noise that appears in the data for a specific task, the model should be learning more representative features that appear across all or most of the tasks. Learning more general features should not only help avoid overfitting, but also help the model learn features that a single task would not have been able to learn.

Drawbacks of Multi-Task Learning

While multi-task learning has many benefits, it has several drawbacks that can greatly impact model performance. One of the main issues with multi-task learning is the need to manually select and tune task weights. The weight that each task is given in the overall loss function significantly influences what the model learns during training. Prior work has empirically shown how sensitive model behavior can be with regards to slight changes in task weights. For example, [5] has shown that training a multi-task learning model to perform instance and semantic segmentation tasks results in a model that is very sensitive to task weight selection.

In addition to needing hand-tuned task weights, multi-task learning requires manual tuning to find the optimal number of tasks to use for the given problem setting. While adding more tasks can help the model avoid overfitting as discussed earlier, adding too many tasks could confuse the model and cause the model to underfit. It is difficult to know which specific tasks are contributing to the model and which tasks are hindering the model's performance without re-training the model with many different permutations of tasks. Ideally, we would be able to have some idea of the uncertainty of each task so that we could gain a better understanding of how many tasks are actually necessary to get the best model performance.

3.3 Learned Task Weights

Learned task weights can help resolve the issues with multi-task learning that were presented earlier. When we manually select task weights in a multi-task learning problem, we are inherently telling the model which tasks we believe have the most uncertainty. If we give a task a lower weight, the model treats this task as having more uncertainty by emphasizing this task less in the training process. If we give a task a higher weight, the model views this task with more certainty and allows this task to influence the model's learning more in the training process. Instead of requiring hand-tuned task weights, we can learn these task weights while the model is being trained.

Theory

In order to learn task weights, we need to modify the traditional multi-task learning loss function. Most of the following derivation of the modified loss function can be found in

[5]. We present the following derivation in order to justify why we use the loss function that is used throughout this report. We discuss the probabilistic derivation and also the natural intuition behind the loss function that follows. We will show the derivation of the loss function for 2 regression tasks. In this 2 task setting, our likelihood must account for the likelihood of each task occurring. We have input data x, and the output of the ith task head can be represented by $y_i = f(x;\theta_i) + \sigma_i^2$, where $f(x;\theta_i)$ is the output of the model at task i before noise is added to the output. θ_i refers to the parameters of the model to obtain the output at task i. In the case of a neural network, θ_i refers to the weights of the network used to get the output for task i. We can also express the probability of obtaining outputs y_1 and y_2 at tasks 1 and 2 given our network as $P(y_1, y_2|f(x;\theta))$. Then, we can write the likelihood of the multi-task model as:

$$L(x;\theta) = P(y_1, y_2|f(x;\theta))$$

If we assume $\sigma_i^2 \overset{iid}{\sim} N(0, \sigma_i^2)$, then we know the distribution of $y_i|f(x;\theta_i) \sim N(f(x;\theta_i), \sigma_i^2)$ from the definition of y_i stated earlier. Therefore, we can write the likelihood as:

$$\begin{aligned} L(x;\theta) &= P(y_1|f(x;\theta_1)) \cdot P(y_2|f(x;\theta_2)) \\ &= N(f(x;\theta_1)) \cdot N(f(x;\theta_2)) \end{aligned}$$

We can then compute the log likelihood, $l(x;\theta)$:

$$\begin{aligned} l(x;\theta) &= \log(N(f(x;\theta_1)) \cdot N(f(x;\theta_2))) \\ &\propto -\frac{1}{2\sigma_1^2} \|y_1 - f(x;\theta_1)\|^2 - \log(\sigma_1^2) - \frac{1}{2\sigma_2^2} \|y_2 - f(x;\theta_2)\|^2 - \log(\sigma_2^2) \end{aligned}$$

Once we have this log likelihood, we can find the negative log likelihood:

$$-l(x;\theta) = \frac{1}{2\sigma_1^2} \|y_1 - f(x;\theta_1)\|^2 + \log(\sigma_1^2) + \frac{1}{2\sigma_2^2} \|y_2 - f(x;\theta_2)\|^2 + \log(\sigma_2^2)$$

We can then use $-l(x;\theta)$ as the loss function for our model, and the model will try to minimize this negative log likelihood to learn $\sigma_1^2, \sigma_2^2, \theta_1,$ and θ_2. For T tasks, this negative log likelihood can be written more generally as:

$$-l(x;\theta) = \sum_{i=1}^{T} (\frac{1}{2\sigma_i^2} \|y_i - f(x;\theta_i)\|^2 + \log(\sigma_i^2))$$

This loss function intuitively makes sense because if task i has large amounts of noise, σ_i^2, then task i's contribution to the model's loss function will approach zero. The $\log(\sigma_i^2)$ term

helps penalize large values of σ_i^2 in order to prevent the model from simply learning large σ_i^2 to trivially send the loss to zero. From this loss function, we can see how the uncertainty of task i, σ_i^2, is directly related to the weight placed on the ith task's loss function. The loss function for classification tasks can be derived similarly. The only difference arises from the fact that the Softmax function is used to normalize the output of the model into a probability distribution in a classification setting. After deriving this loss function, we can now start applying the idea of learned task weights to different models and datasets.

Chapter 4

Learned Task Weights with Noisy Polynomials

Before experimenting with learned task weights on more complex multimedia datasets, we first wanted to experiment with datasets where we could easily control the level of noise added. The first experiment focused on observing whether learned task weights could result in improved performance on synthetic polynomials. We wanted to first show that a multi-task learning model with learned task weights could perform well on synthetic datasets (polynomials) before experimenting with real image datasets. The source code for these experiments and following experiments can be found at `https://github.com/davidnahm/retrieval-learned-task-weights`.

4.1 Experimental Setup for Noisy Polynomials

The polynomials were of the form:

$$f_i(x) = c_1 x^d + \ldots + c_d x + c_{d+1} + \sigma_i^2$$

The degree d and noise σ_i^2 of the polynomials were varied to observe how the behavior of the learned task weights changed as the degree and noise level of the polynomials changed. The polynomials are generated with 2,500 training points and 200 validation points. A fully-connected network was used to solve this regression problem. The network had one shared layer of size 1x512 and two layers for each individual task head of size 512x512 and 512x1. ReLU activation was used after the first two fully connected layers (of size 1x512 and 512x512). The network architecture can be seen in Figure 4.1 [1]. The ADAM optimizer was used, and we found that a learning rate of 0.1 led to the best results.

[1] We decided on the network architectures used in this report based on our experimentation and prior related work. A simpler network with fewer parameters could be able to obtain equal or better performance, but we decided to leave this as an experiment for future work.

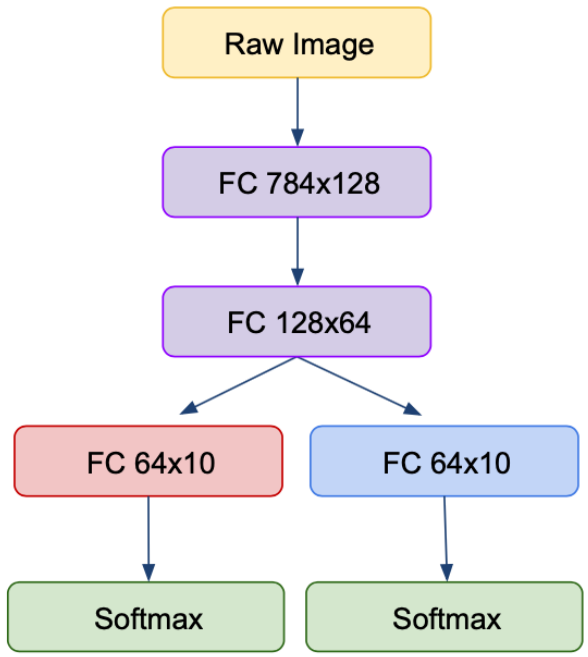


Figure 4.1: Network architecture used in the polynomial experiments.

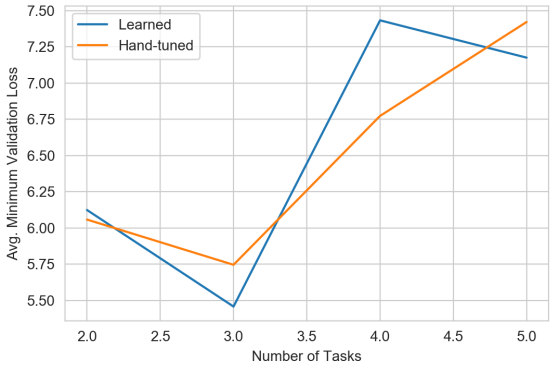
The loss function of the learned task weight model is a linear combination of each individual task loss, weighted by the learned task weights. Mean squared error (MSE) was used as the loss function for each individual task. For the learned task weight model, the network learns the weights of the polynomial as well as the $\log(\sigma_i^2)$. The network attempts to learn the log of the task weight to maintain numerical stability.

$$L(x) = \sum_{i=1}^{T} (\frac{1}{2\sigma_i^2}(\hat{f}(x) - f(x))^2 + \log(\sigma_i^2))$$

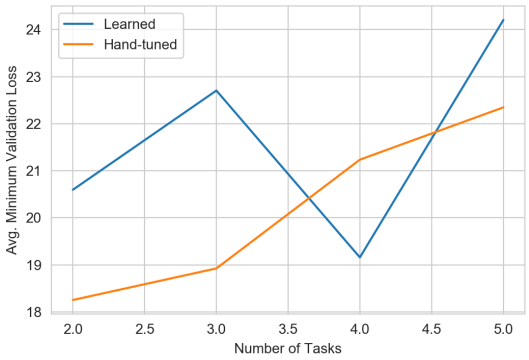
A hyperparameter search was performed in order to find the best hand-tuned task weights for the control multi-task learning model. Each model was trained until convergence, and early stopping was used (with patience = 30 epochs) to prevent the model from overfitting.

4.2 Experiments with Noisy Polynomials

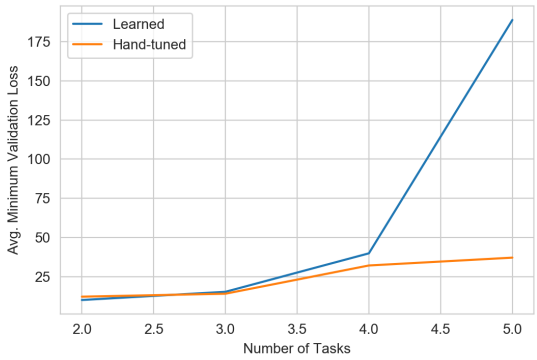
The first set of experiments was performed with varied values of σ^2 for each task. In order to observe how the model performance changes as σ_i^2 changes, we set up two experiments.



(a) Degree 5



(b) Degree 8



(c) Degree 10

Figure 4.2: Average minimum validation loss on polynomials of degree 5, 8, and 10 as we increase the amount of noise. The performance of a hand-tuned multi-task learning model is overlaid as a baseline.

In the first experiment, we start with a two task multi-task learning model with $\sigma_1^2 = 1$ and $\sigma_2^2 = 2$. We train this model and measure the minimum validation loss the model achieves on a holdout set generated from the same polynomial. These minimum validation loss values are then averaged over 10 random seeds. We then add an additional task to the model with $\sigma_3^2 = 3$ and repeat the same process of training and computing the average minimum validation loss. We iteratively set $\sigma_i^2 = i$ and measure the model's performance until $i = 6$. By simultaneously increasing the values of σ_i^2 and increasing the number of tasks, our goal was to observe if the benefits of the added tasks outweighed the drawbacks of increasingly noisy data.

From Figure 4.2, we can observe how model performance changed as the number of tasks

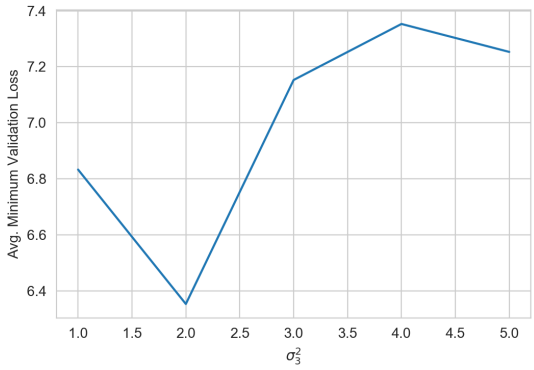
$\hat{\sigma}_1^2$	$\hat{\sigma}_2^2$	$\hat{\sigma}_3^2$	$\hat{\sigma}_4^2$	$\hat{\sigma}_5^2$
1.0015	1.9827	-	-	-
0.9993	2.0231	3.0126	-	-
0.9956	2.0353	3.0242	4.0123	-
0.9622	2.0088	2.9653	3.8892	4.2234

Table 4.1: Example learned task weights for a degree 10 polynomial averaged over 10 random seeds.

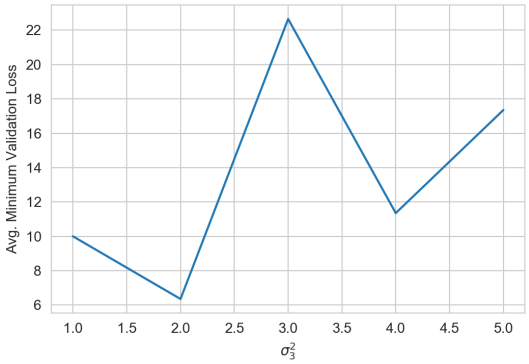
and the amount of noise changed. As the number of tasks increased with increasing amounts of noise, we can see how the model remained relatively robust to the noise from the added tasks for the degree 5 and degree 8 polynomials. For the degree 10 polynomial, the model was robust up to 4 tasks, but then saw a large increase in the validation loss for more tasks. This decrease in performance could have been due to the model requiring more epochs to train. A degree 10 polynomial could also be too complex for the model to handle. The model was not strong enough to learn both the task weights and how to fit the polynomial in the presence of a task with noise $\sigma^2 = 5$. We can see in Table 4.1 that the model was able to identify that the fifth task of the degree 10 polynomial experiment was noisy, but it underestimated the level of noise. For the degree 5 and degree 8 polynomials, the model was still able to improve its performance despite learning from increasingly noisy data. In these cases, the additional data helped the model more than the added noise hurt the model. Ordinarily, the additional data would hurt the model's performance because the added noisy data would simply confuse the model. However, we can see that because the model is able to learn how to properly weight the additional data, it can take advantage of the data without sacrificing model performance.

Our next experiments also varied the values of σ^2 but for a fixed number of tasks. In these experiments, we set the number of tasks to be 3. We also fix $\sigma_1^2 = 1$ and $\sigma_2^2 = 2$. We then vary $\sigma_3^2 = 1, ..., 5$ to observe how the model behaves when only the noise in task 3 is changing. We repeat this process for polynomials of degree 5, 8, and 10. We can observe the results in Figure 4.3. We use the same network architecture that was used in the previous polynomial experiments.

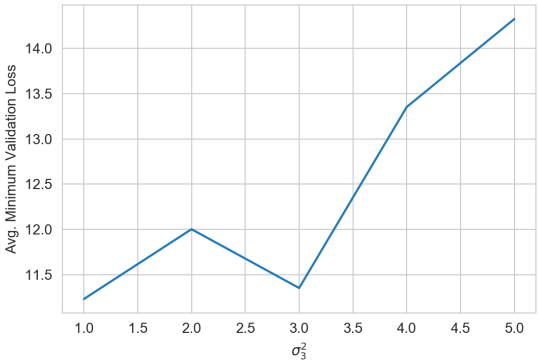
We see in Figure 4.3 that is mostly robust to noise that varies from $\sigma_3^2 = 1, ...5$. This tells us that the model is correctly able to downweight this noisy task while learning, relying instead on tasks 1 and 2 which only have $\sigma_1^2 = \sigma_2^2 = 1$. We do see a slight decline in performance for the degree 8 polynomial with $\sigma_3^2 = 3$, then a return to better performance for $\sigma_3^2 = 4$. This dip in performance could be due to the fact that $\sigma_3^2 = 3$ is large enough to throw off the learning process, but small enough so that the model is still trying to learn from this task. The model could be having more difficulty with moderate noise as opposed



(a) Degree 5



(b) Degree 8



(c) Degree 10

Figure 4.3: Average minimum validation loss on polynomials of degree 5, 8, and 10 for a fixed number of tasks with increasing noise.

to extreme values of noise. However, aside from this one example of a performance dip, the model seems to be fairly robust to an increasingly noisy task. We also see that this is true for polynomials of degrees 5 and 8, but with the degree 10 polynomial, we do start to see a decrease in model performance as we increase the noise. As with our first experiments, this is likely due to the polynomial being more complex and the model being unable to both learn the task weights while fitting the polynomial simultaneously. The complexity of the task at hand affects how much noise the learned task weights are ultimately able to handle and still obtain good model performance.

Takeaways from Polynomial Experiments

From these experiments with noisy polynomials, we can observe that the degrees of the polynomials played a role in how robust the learned task weights are to label noise. While the learned task weights consistently correctly modeled the uncertainty in each task for lower degree polynomials (degree 5 and degree 8), we see that the network's performance declines in the presence of higher noise for a more complex polynomial (degree 10 polynomial). We see that learned task weights are indeed impacted by the complexity of the tasks and that only modeling the uncertainty of each task may not be sufficient for more complex tasks. After conducting these experiments with polynomials and observing these findings with regards to task complexity, we wanted to progress our experiments to image datasets that more closely resembled the multimedia datasets that we encounter in cross-modal retrieval.

Chapter 5

Learned Task Weights with Image Datasets

After observing the benefits of learned task weights on synthetic polynomial datasets, we then wanted to experiment with more complex image datasets. In order to vary the amount of noise and difficulty of this image classification problem, we experimented with randomly sampled label noise to test the robustness of the learned task weights.

5.1 Experimental Setup with MNIST Dataset

The first step in our experimental setup was to determine how to create label noise in our data. We decided to use a randomly sampled label noise approach that takes the following approach. For every observation in the data used for task i, we draw from a $N(0, \sigma_i^2)$ distribution. If the absolute value of that draw is greater than 1.96, then the label of this observation is randomly changed to a different class. The class to be changed to is chosen uniformly from the remaining $n-1$ classes. This results in approximately 5% of labels being changed for $\sigma_i^2 = 1$, approximately 16.5% of labels being changed for $\sigma_i^2 = 2$, approximately 25% of labels being changed for $\sigma_i^2 = 3$, and approximately 32% of labels being changed for $\sigma_i^2 = 4$. Since the true labels for a given class under randomly sampled label noise could be in any of the n classes, randomly sampled label noise was a difficult type of label noise to overcome. Symmetric noise is another common approach to adding label noise to a dataset. We considered using symmetric noise, but we felt that randomly sampled label noise was more representative of the label noise that would occur in real multimedia datasets. Symmetric noise simply flips labels between class i and class $n - i - 1$ with probability p. We felt that label noise in real datasets would be dispersed throughout all n classes and not just be flipped symmetrically.

Our first experiments involved learning from the same dataset with different amounts of random pairwise label noise. For example, to build a 3 task multi-task learning model

trained on the MNIST dataset, we created 3 different versions of the MNIST dataset, each with a different amount of random label noise. Each task is still trying to classify the MNIST dataset, but each task is using a different noisy version of the same dataset. We can then train the model so that each task has the same input data but different noisy labels. The MNIST dataset consists of 60,000 training images and 10,000 testing images of handwritten digits. There are 10 classes in the dataset which represent the digits 0 through 9.

We used a fully-connected network to classify images from the MNIST dataset. The network has two shared layers of size 784x128 and 128x64 and one task-specific layer for each individual task of size 64x10. The network architecture can be seen in Figure 5.1. ReLU activations were used after each layer. In order to train the model, experiments were conducted with both stochastic gradient descent (SGD) and ADAM optimizers, but we found that using SGD resulted in better performance.

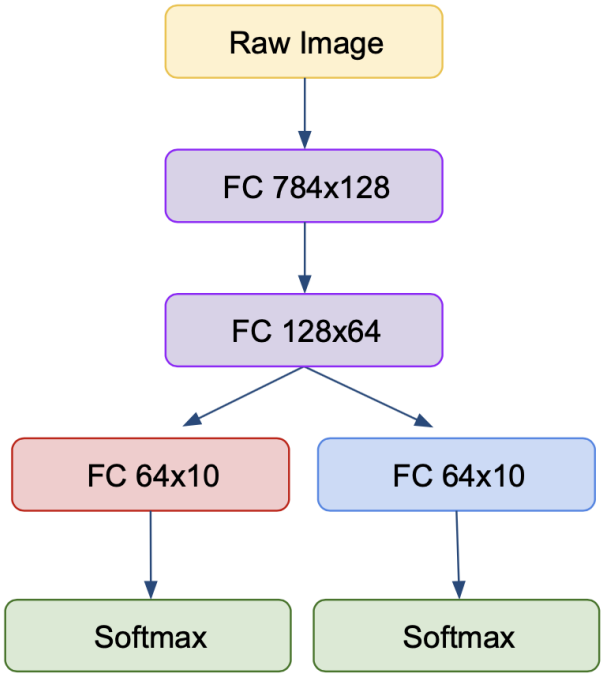


Figure 5.1: Network used to classify MNIST images.

Experiments with MNIST Dataset

In order to make a fair comparison between the learned task weight model and the hand-tuned task weight model, we first compared their performance in a setting where we increased the level of noise with each additional task added. The first configuration was with 2 tasks and with $\sigma_1^2 = 1$ and $\sigma_2^2 = 2$. We trained a model on this two task problem over 10 random seeds and calculated the best test accuracy achieved on the MNIST test dataset for each random seed. We then averaged these best validation accuracies to calculate an average best test accuracy. We then added an additional task with $\sigma_3^2 = 3$ and repeated the training and evaluation process. We continued this setup one more time with $\sigma_4^2 = 4$. By setting up our experiments like this, we could examine how the behavior of the learned task weights and how the model's performance was affected as the amount of label noise and the number of tasks varied. In order to find the best baseline hand-tuned multi-task learning model, we used a grid search over several different combinations of task weights to see which set of task weights performed the best. The results of these MNIST experiments can be seen in Figure 5.2.

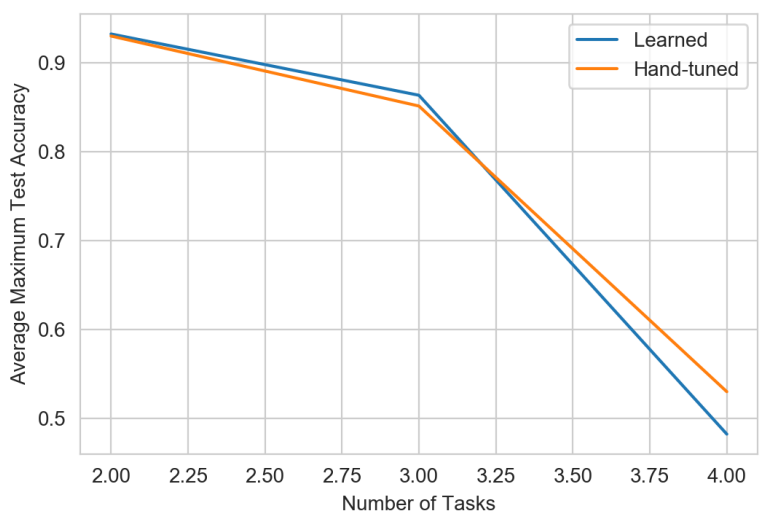


Figure 5.2: MNIST performance with increased levels of noise and increasing number of tasks.

We can see in Figure 5.2 that the learned task weight model is able to perform as well or better than the model with hand-tuned task weights for 2 and 3 task models, but the performance drops off once the 4th task is added. For the model with 3 tasks, we actually see an improvement in performance over the hand-tuned model. We see the dropoff in performance when the 4th task is added because this amount of noise is too much for the learned task

weights to learn effectively. 3 tasks with $\sigma_3^2 = 3$ seems to be the most noise the model can be expected to learn through for these experiments where we are increasing the number of tasks as we increase the noise.

Our next experiments involved fixing the number of tasks at 3 and fixing $\sigma_1^2 = 1$ and $\sigma_2^2 = 1$. We then varied the value of $\sigma_3^2 = 1, ..., 4$. By doing this, we could see how the learned task weights affected the model's performance for a fixed problem size. For each model trained with a different σ_3^2, we record the best test accuracy achieved by the model on the MNIST test images. We repeat this over 10 random seeds and plot the average maximum test accuracy for each value of σ_3^2. We can observe these results in Figure 5.3 as we can see how the learned task weight model performed with varying levels of noise on the MNIST image labels.

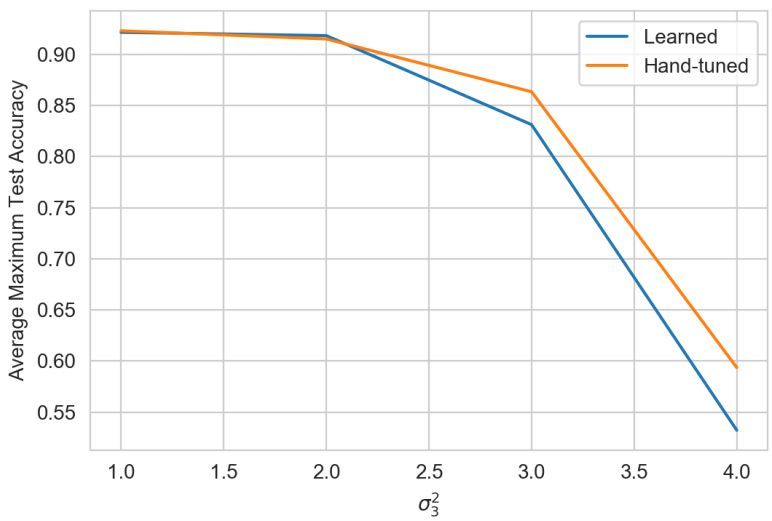
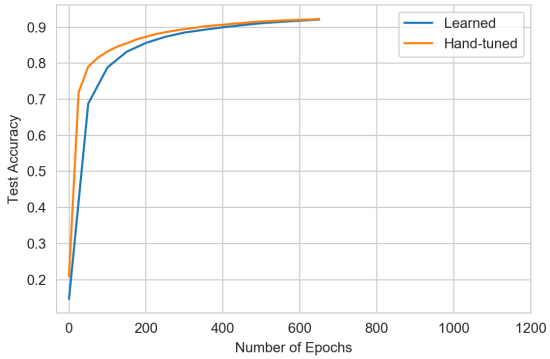
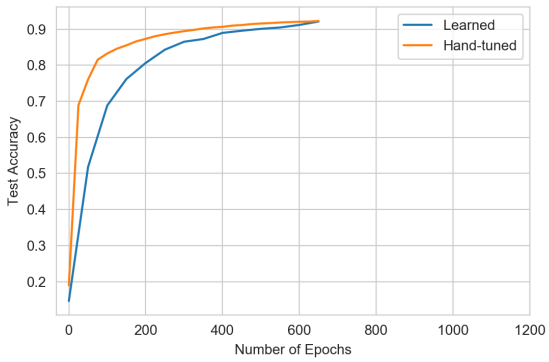


Figure 5.3: MNIST performance with increased levels of noise on a model with 3 tasks.

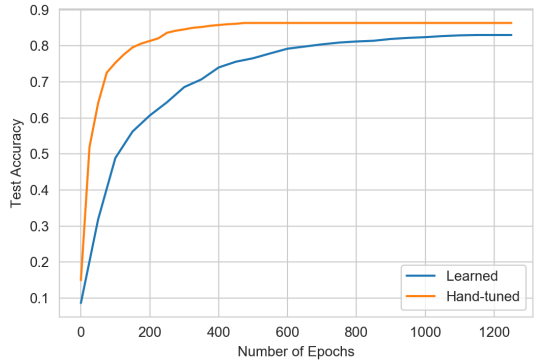
We can see in Figure 5.3 that the learned task weight model is particularly robust to label noise for $\sigma_3^2 = 1$ and $\sigma_3^2 = 2$. In these cases, the model is actually able to slightly outperform the model with hand-tuned task weights because the learned task weight model can learn more specific, non-whole number task weights that would be too granular for a grid search to find. For $\sigma_3^2 = 3$, we see that the learned task weight model starts to dip in performance compared to the hand-tuned task weight model. This is likely due to the longer training times required for the learned task weight model which is discussed below. Once $\sigma_3^2 = 4$, we see a drastic drop in performance in both the learned task weight and hand-tuned task



(a) $\sigma_3^2 = 1$



(b) $\sigma_3^2 = 2$



(c) $\sigma_3^2 = 3$

Figure 5.4: MNIST learning curves for $\sigma_3^2 = 1, 2, 3, 4$

weight model as neither model is able to overcome this much label noise.

While the learned task model performs well, we did observe a tradeoff in the number of epochs the learned task model required to converge. As seen in Figure 5.4, the learned task model learned more gradually. As the noise, σ_3^2, grew larger, the model required more epochs to converge. We can see in Figure 5.4 that for $\sigma_3^2 = 3$, the model required around twice as many epochs to converge as it did for lower values of σ_3^2. One of the reasons for this is that we found that using a lower than usual learning rate resulted in more consistent convergence. With more parameters to learn in the learned task model, the model had to be more careful while learning. Taking large steps with a larger learning rate often resulted in the model overcompensating and adjusting the task weights too much. This led to the task weights being too sensitive to certain batches of data and not being able to recover and approach the optimal task weight.

Takeaways from MNIST Experiments

One of the main takeaways from our MNIST experiments was that the learned task weight model was able to perform as well or better than the hand-tuned model, but only up to $\sigma_2 = 3$. Once the noise surpassed this amount, we saw a clear decline in performance in the learned task weight model as the model was not able to effectively learn the uncertainty in each task. We also noticed in our experiments that lower learning rates were required to get our model to converge for higher levels of noise. With these findings and the findings from our experiments with polynomials, we then moved on to observing how learned task weights behaved in a cross-modal retrieval network trained with real multimedia datasets.

Chapter 6

Cross-modal Retrieval with Learned Task Weights

After experimenting with image datasets, we wanted to further our experimentation with a cross-modal retrieval model using multimedia datasets with more than one modality. In these experiments, we experiment with the image and text modalities to understand how learned task weights will react to label noise. The same methodology of adding noise to our datasets that was used in the MNISt experiments is used for these datasets.

6.1 Datasets

We experimented with two datasets to observe the performance of the learned task weights on a cross-modal retrieval model. One of the datasets we experimented with was IAPR TC-12 [29]. IAPR TC-12 consists of 20,000 natural images with 255 different classes. Each image also has a text caption, which is in the form of a sentence. We also experimented with the MIRFlickr-25K dataset [30]. The MIRFlickr-25K dataset contains 25,000 images from Flickr with 20 classes. This dataset also has captions in the form of user tags that describe briefly what is in the image. Sample data from IAPR TC-12 and MIRFlickr-25K can be seen in Figure 6.1.

6.2 Loss Formulation

Triplet loss is commonly used to train cross-modal retrieval models because it can be interpreted naturally when trying to learn joint embeddings. Triplet loss is defined as:

$$L(A, P, N) = \sum_{i=1}^{n} \max(\left\|f(A^{(i)}) - f(P^{(i)})\right\|^2 - \left\|f(A^{(i)}) - f(N^{(i)})\right\|^2 + \alpha, 0)$$

We set the margin to be $\alpha = 1$. For observation i in a dataset of size n, triplet loss takes three inputs: an anchor $A^{(i)}$, a positive example $P^{(i)}$, and a negative example $N^{(i)}$. In our

(a) "two light brown church towers with pinnacles on all four sides"

(b) "building, city, sky"

Figure 6.1: Sample images and captions from the IAPR TC-12 (left) and MIRFlickr-25K datasets (right) [29] [30].

experiments where we are dealing with image and text modalities, our triplets would consist of (anchor image, positive text, negative text) and (anchor text, positive image, negative image). The positive example is randomly sampled from the same class as the anchor but from the opposing modality the anchor is from. In order to apply the learned task weights to a cross-modal retrieval model with triplet loss, we can use triplet loss as the loss function for each task. Therefore, the loss function for a multi-task learning model where each task uses triplet loss can be written as:

$$
\begin{aligned}
L(A,P,N) &= \sum_{i=1}^{T}\left(\frac{1}{2\sigma_i^2}L_i(A,P,N) + \log(\sigma_i^2)\right) \\
&= \sum_{i=1}^{T}\frac{1}{2\sigma_i^2}\Big[\sum_{j=1}^{n}\max\left(\left\|f(A^{(j)};\theta_i) - f(P^{(j)};\theta_i)\right\|^2 - \left\|f(A^{(j)};\theta_i) - f(N^{(j)};\theta_i)\right\|^2 + \alpha, 0\right)\Big] + \\
&\qquad \log(\sigma_i^2)
\end{aligned}
$$

6.3 Network Architecture

Our network consists of two sub-networks: an "image network" and a "text network". The image network takes in raw images as input and learns embeddings for these images. Similarly, the text network takes in the raw captions as input and learns embeddings for the text. We used pre-trained networks to embed the input images and text from the datasets. For images, we used a pre-trained ResNet-18 model (trained on the ImageNet dataset). This pre-trained ResNet model embeds the raw input images as 512-dimensional embeddings. For the text input, we used a pre-trained, open-source BERT model from [31]. Each cap-

tion is passed through the pre-trained BERT model and embedded as a 768-dimensional embedding. Once we have these image and text embeddings, we can project these image embeddings onto a common subspace where embeddings from different modalities can easily be compared. This projection is learned for image embeddings using a fully connected shared layer of size 512x128 followed by ReLU activation. This is then followed by a layer of 128xd for each task head, where d is the dimensionality of the learned joint embeddings in the common subspace. The projection for text embeddings is learned using a fully connected shared layer of size 768x128 followed by ReLU activation. This is then followed by a layer of 128xd for each task head. We varied d to see how our model performance changes as we increase the dimensionality of the joint embeddings. The full network architecture can be seen in Figure 6.2. An ADAM optimizer with a learning rate of 0.00001 was used to train the model.

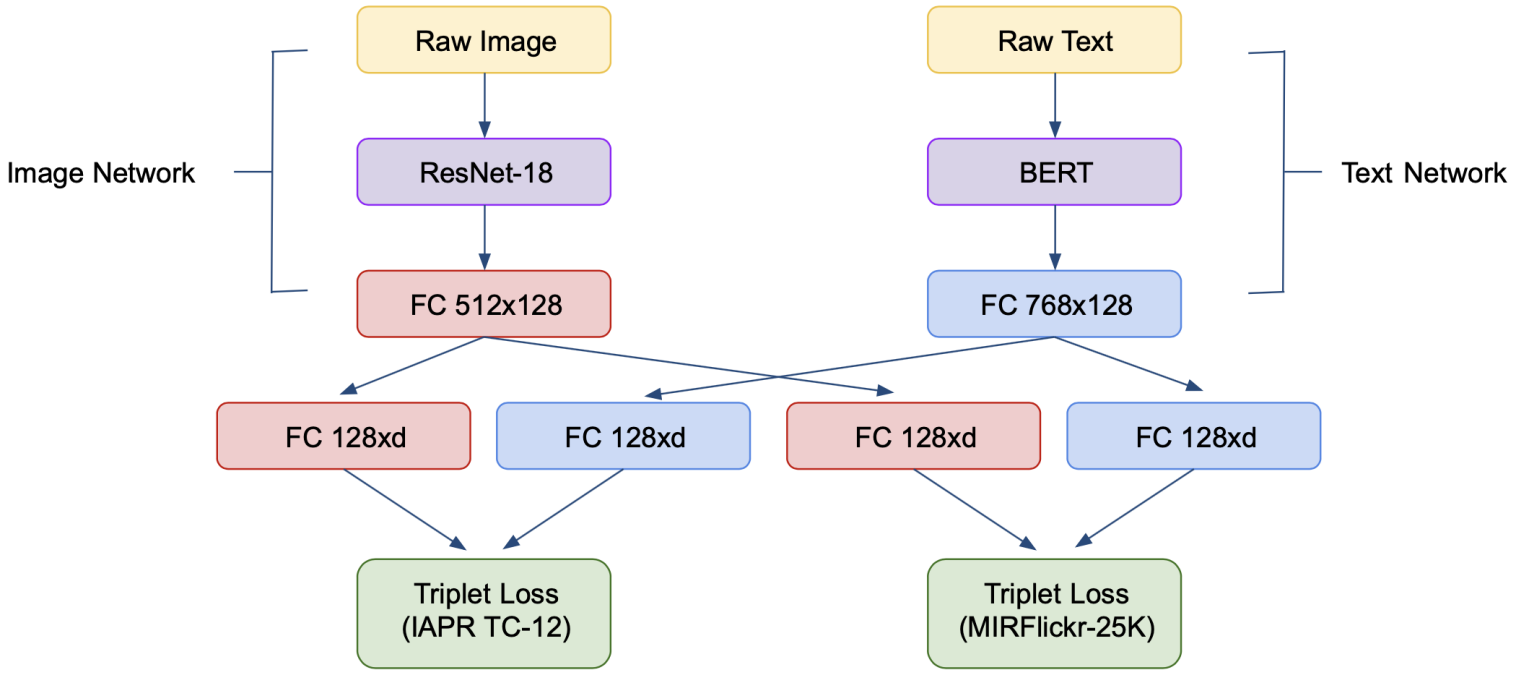


Figure 6.2: Network used to perform cross-modal retrieval. d is the dimensionality of the common subspace and is varied from d = 16, 32, 64.

6.4 Model Evaluation

In order to evaluate the performance of the cross-modal retrieval model, we use Mean Average Precision (MAP). MAP is defined as the mean of the average precision scores for each query:

$$\text{MAP} = \sum_{q=1}^{Q} \frac{\text{AveP}(q)}{Q}$$

where Q is the total number of queries and AveP(q) is the average precision given query q. The average precision for a given query can be computed as:

$$\text{AveP}(q) = \sum_{i=1}^{K} p(i) \cdot r(i)$$

$p(i)$ is the precision of the top i ranked retrieved items. $r(i)$ measures the relevance of the ith ranked retrieved item — $r(i) = 1$ if the ith ranked retrieved item is relevant to the current query (if they share the same label) and $r(i) = 0$ otherwise.

6.5 Results

In order to measure the performance of the model in the presence of label noise, we fixed the amount of label noise for one task and varied the amount of label noise for the other task. The amount of label noise added to the IAPR TC-12 data is σ^2_{IAPR}, and the amount of label noise added to the MIRFlickr-25K data is σ^2_{MIR}. To measure how well the task using IAPR TC-12 data performed with added noise, we fixed $\sigma^2_{\text{MIR}} = 1$ and varied $\sigma^2_{\text{IAPR}} = 0, ..., 3$. Similarly, to measure how well the task using MIRFlickr-25K data performed with added noise, we fixed $\sigma^2_{\text{IAPR}} = 1$ and varied $\sigma^2_{\text{MIR}} = 0, ..., 3$. We then trained our cross-modal retrieval model for each pair of σ^2_{IAPR} and σ^2_{MIR} and computed the MAP for each model. These MAP values are recorded in Table 6.1. By feeding the model with one slightly noisy dataset and one dataset with increasing noise, we hope to see if the model can learn which dataset to weigh more in its learning process. The model should emphasize its learning on the dataset with fewer swapped labels, and it should adjust its task weights accordingly.

There are several observations we can make from the model performance seen in Table 6.1. The first observation we can see is that the model still performed well through significant amounts of label noise up to $\sigma^2_{\text{IAPR}} = 2$ for IAPR TC-12 on both image-to-text and text-to-image retrieval. Without any prior knowledge of the noise in each task, the model was still able to use the learned task weights to properly weigh the tasks. We saw this was true in the cases of synthetic polynomials and image datasets, but these results on IAPR TC-12 and MIRFlickr-25K show how those results can be extended to more complex multimedia

IAPR TC-12					
	d	$\sigma^2_{\text{IAPR}} = 0$	$\sigma^2_{\text{IAPR}} = 1$	$\sigma^2_{\text{IAPR}} = 2$	$\sigma^2_{\text{IAPR}} = 3$
Image $\rightarrow$ Text	16	0.4732	0.4563	0.4135	0.3541
	32	0.4892	0.4704	0.4380	0.3597
	64	0.4934	0.4756	0.4439	0.3612
Text $\rightarrow$ Image	16	0.4877	0.4493	0.4135	0.3254
	32	0.4962	0.4721	0.4306	0.3245
	64	0.5029	0.4753	0.4379	0.3292
MIRFlickr-25K					
	d	$\sigma^2_{\text{MIR}} = 0$	$\sigma^2_{\text{MIR}} = 1$	$\sigma^2_{\text{MIR}} = 2$	$\sigma^2_{\text{MIR}} = 3$
Image $\rightarrow$ Text	16	0.6836	0.6472	0.6120	0.5178
	32	0.6914	0.6731	0.6379	0.5192
	64	0.6982	0.6790	0.6411	0.5194
Text $\rightarrow$ Image	16	0.7268	0.6737	0.6004	0.4873
	32	0.7311	0.6905	0.6184	0.4898
	64	0.7384	0.6964	0.6233	0.4884

Table 6.1: MAP scores for IAPR TC-12 and MIRFlickr-25K datasets. d is the dimensionality of the joint embeddings used in the common subspace.

datasets. We see slight dropoffs in model performance as the noise initially increases, but the model is still able to learn well. We do see that MIRFlickr-25K performance seems to be affected negatively slightly more by the added noise. This is likely because the MIRFlickr-25K dataset is a noisier dataset than IAPR TC-12, so it will be less robust to added noise. We can also see how text-to-image retrieval performance in MIRFlickr-25K falls faster than any of the other tasks. This was likely due to the fact that the raw input text itself already contained noise that the model could not account for. The captions in the MIRFlickr-25K dataset are noisier than the IAPR TC-12 captions because they are user captions from Flickr and contain noise such as spelling mistakes or possibly irrelevant user tags. As a result, the modality we are looking at can have a large effect on whether the learned task weights will be robust to label noise. Some modalities, such as text, tend to have more noise on the raw input that must be taken into account when using learned task weights.

In general, we also see that for larger joint embeddings ($d = 32$ and $d = 64$), the model can perform better in the presence of noise. When noise is added to the model with $d = 16$, we see the model shows higher drops in performance than with $d = 32$ or $d = 64$. These differences can primarily be seen with $\sigma^2 = 1$ and $\sigma^2 = 2$. Once $\sigma^2 = 3$, the noise is large enough that all of the joint embedding sizes seem to perform equally. The higher dimensionality of the joint embedding space likely leads to better results because the model is able to learn more features to differentiate different concepts. Performing cross-modal

retrieval with larger embeddings is a less complex problem to solve because the embeddings are able to capture more of the features in the input images and text. As a result, based on our results from polynomial and image datasets, we would expect the learned task weights to perform better on the less complex task to solve.

Takeaways from Cross-modal Retrieval Experiments

There are several main conclusions we can take away from these cross-modal retrieval experiments. One takeaway is that while uncertainty-based task weights performed well with low levels of noise, only learning task weights based on uncertainty is not sufficient for higher levels of noise. We saw large dropoffs in performance once the label noise in a dataset grew too large, suggesting that we need to consider more than just the uncertainty of each task for high levels of noise. We also see that the modality used when performing retrieval matters when looking at the robustness of the learned task weights. We saw that text suffered more from added label noise than images did. With these findings and our takeaways from our experiments with polynomials and images, we can present some actionable conclusions on learning task weights for cross-modal retrieval with noisy datasets.

Chapter 7

Conclusion

7.1 Lessons Learned

There are several main lessons that were learned through our experiments with learned task weights on noisy datasets. One of the main lessons is that we saw how the complexity of the problem being solved affected how robust the learned task weights are to added noise. This is most clearly displayed in our experiments with polynomials where we compare the performance of the learned task weight model with polynomials of increasing degree. In our polynomial experiments, we started seeing a dropoff in performance for the degree 10 polynomial when large amounts of noise were added to the data. For the same amount of noise, the less complex polynomials (degree 5 and degree 8) did not see the same dropoff in performance. When we added smaller amounts of noise, we did not see the degree of the polynomial affect the robustness of the learned task weights. Our experiments show that while learning task weights solely based on the amount of uncertainty in each task may be effective in lower noise settings, the complexity of the tasks must also be considered in higher noise settings. There are several ways that this could actually be implemented and incorporated into the learning process. One possible direction is to decrease the dimensionality of our input data through different dimensionality reduction techniques. By reducing the dimensionality of our data, we can control the complexity of each task before using our uncertainty-based learned task weights. As a result of handling the complexity issue as part of a pre-processing step, we can then focus solely on learning the uncertainty of each task because we have already controlled the complexity of each task. Alternatively, instead of dealing with complexity as a pre-processing step, we could also deal with complexity as part of the learning process. Our work suggests that learning uncertainty and complexity simultaneously would be a reasonable step towards provide more robustness for learned task weights when using high noise datasets. Whether we deal with task complexity as a pre-processing step or in the form of learned parameters, it is important that cross-modal retrieval models using learned task weights take this into account. Multimedia datasets commonly contain large amounts of noise, so dealing with task complexity is crucial if we want to be confident

in the behavior of a cross-modal retrieval model with learned task weights.

Our experiments with cross-modal retrieval suggest that the robustness of learned task weights in high noise settings also varies based on the modalities being used. In our experiments, we saw that text-to-image retrieval suffered more from increased label noise than image-to-text retrieval. This was likely due to the fact that the text modality tends to have more noise in its raw input than the image modality, adding to the complexity of learning how to embed text data. This decrease in performance in the more complex task further shows how task complexity should be taken into account when learning task weights. There are several possible solutions to this that we addressed earlier. One way to alleviate this problem is to perform stronger dimensionality reduction on more complex modalities as a pre-processing step. For example, if we knew the text modality is more complex and is more difficult to embed than the image modality, then we could reduce the dimensionality of the raw text input more than the dimensionality of the raw image input. By doing this, we would hope to balance out the complexity between the two modalities so that we would only have to use uncertainty-based learned task weights in our training process. In addition, we could try to learn the complexity of each modality as part of our training process to account for the differences in modality.

In addition to our takeaways regarding the complexity of each task, we also learned some useful guidelines for training learned task weight networks. We commonly saw that the learned task weight model required lower than average learning rates to converge in high noise settings. While we expected the training time to be slightly longer due to the increased number of parameters the model needed to learn, we did not expect the model to require lower than average learning rate sizes. From our experiments, we observed that using lower learning rates allowed the model to learn more gradually and provided better results empirically.

If we can confidently use learned task weights in multimedia computing, there would be many possible benefits. One clear benefit is that learned task weights would simply reduce the amount of time and effort spent on model selection with multi-task learning models. Additionally, learned task weights are not prone to the same biases that programmers have when trying to select task weights manually. A programmer would never be expected to select task weights to the granularity and specificity that learned task weights could learn. Programmers would be naturally biased towards choosing task weights that are whole numbers when the optimal weights likely are not whole numbers. Learned task weights also give us a natural insight into how the model is learning and which tasks it is focusing its learning on. The learned task weights provide an easily interpreted measure of how much uncertainty each task contains. While there are many benefits of learned task weights, it is important to understand their behavior and how they will react to learning in noisy environments. As discussed in this report, there are several primary factors that affect the robustness of the learned task weights to large amounts of label noise, such as the complexity of the tasks and

the modalities being used.

7.2 Future Work

There are several areas of future work that we would like to pursue. One main area of future work is exploring other forms of learned task weights. We saw in this report that only considering the uncertainty of each task has its limitations in high noise settings. There are several alternatives to learning task weights that could result in better performance in the presence of large amounts of noise. One alternative is to consider task difficulty in addition to task uncertainty when learning task weights. This would allow the model to understand the uncertainty in each task while also taking the difficulty of each task into consideration when learning how to weigh each task. We could also further extend this idea by increasing the granularity of our learned task weights to the observation level. Our current approach only focuses on weighing each task as a whole, but future work could build on this by also learning how to weigh the uncertainty and difficulty of classifying each observation. This would add an additional level of knowledge that the model can use to learn how to weigh each task.

Another area of future work to explore is extending the number of modalities that we are dealing with. This work focused on image-to-text and text-to-image retrieval, but it would be interesting to examine the behavior of the learned task weights on other modalities such as video or audio. All of these modalities vary in how well they are labeled, resulting in different performance for each modality. These modalities also vary in how noisy their respective inputs are. For example, text datasets might contain substantial spelling mistakes or slang terms that a model might struggle to learn how to deal with. Audio datasets could contain input noise in the form of background noise or multiple voices speaking simultaneously.

Finally, we would like to study how large of a network we really need to effectively learn task weights. In this report, we based our network sizes on prior work and our own experimentation, and we did not make varying the size of the networks a priority in our analysis. A smaller network could be able to learn task weights that are more robust to label noise. For example, we could ask how many neurons it would actually take to properly learn task weights. We could then examine the behavior of networks of varying sizes to see if smaller networks are able to learn task weights properly and if they learn better task weights. In addition to possibly providing better performance and more accurate task weights, this would help decrease the amount of time required to train our models. As discussed earlier, one of the main motivations for using learned task weights is the reduced amount of time spent on hyperparameter searches. By learning networks with smaller networks, we would most likely see the added benefit of spending less time training the model which can be of

great use when training models on the large multimedia datasets that are commonly seen in cross-modal retrieval.

Bibliography

[1] Bart Thomee et al. "YFCC100M: The New Data in Multimedia Research". In: (2015). DOI: 10.1145/2812802. eprint: arXiv:1503.01817.

[2] Chuyuan Xiong et al. *Voice-Face Cross-modal Matching and Retrieval: A Benchmark*. 2019. eprint: arXiv:1911.09338.

[3] Amanda Duarte et al. "Temporal-aware Cross-modal Embeddings for Video and Audio Retrieval". In: *NIPS 2017 Women in Machine Learning Workshop (WiML)*. NIPS 2017 Women in Machine Learning Workshop. Long Beach, CA, USA: NIPS 2017 Women in Machine Learning Workshop, Dec. 2017.

[4] J. Choi et al. "From Intra-Modal to Inter-Modal Space: Multi-task Learning of Shared Representations for Cross-Modal Retrieval". In: *2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM)*. 2019, pp. 1–10.

[5] Alex Kendall, Yarin Gal, and Roberto Cipolla. *Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics*. 2017. eprint: arXiv:1705.07115.

[6] Niluthpol Chowdhury Mithun et al. "Learning joint embedding with multimodal cues for cross-modal video-text retrieval". In: *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*. 2018, pp. 19–27.

[7] Jiuxiang Gu et al. *Look, Imagine and Match: Improving Textual-Visual Cross-Modal Retrieval with Generative Models*. 2017. eprint: arXiv:1711.06420.

[8] Didac Surís et al. *Cross-modal Embeddings for Video and Audio Retrieval*. 2018. eprint: arXiv:1801.02200.

[9] K. Wang et al. "Joint Feature Selection and Subspace Learning for Cross-Modal Retrieval". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.10 (2016), pp. 2010–2023.

[10] Yunchao Gong et al. *A Multi-View Embedding Space for Modeling Internet Images, Tags, and their Semantics*. 2012. eprint: arXiv:1212.4522.

[11] A. Sharma et al. "Generalized Multiview Analysis: A discriminative latent space". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pp. 2160–2167.

[12] Karl Ni et al. *Sampled Image Tagging and Retrieval Methods on User Generated Content*. 2016. eprint: `arXiv:1611.06962`.

[13] Rich Caruana. "Multitask Learning". In: *Machine Learning* 28 (July 1997). DOI: `10.1023/A:1007379606734`.

[14] R. Girshick. "Fast R-CNN". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1440–1448.

[15] Minh-Thang Luong et al. *Multi-task Sequence to Sequence Learning*. 2015. eprint: `arXiv:1511.06114`.

[16] Gueorgui Pironkov, Stephane Dupont, and Thierry Dutoit. "Multi-task learning for speech recognition: an overview." In: *ESANN*. 2016.

[17] Michelle Guo et al. "Dynamic Task Prioritization for Multitask Learning". In: *ECCV* (2018), pp. 282–299.

[18] Zhao Chen et al. "GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks". In: *CoRR* abs/1711.02257 (2017). arXiv: `1711.02257`. URL: `http://arxiv.org/abs/1711.02257`.

[19] Liang Xie et al. "A Cross-Modal Multi-Task Learning Framework for Image Annotation". In: *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. CIKM '14. Shanghai, China: Association for Computing Machinery, 2014, pp. 431–440. ISBN: 9781450325981. DOI: `10.1145/2661829.2662023`. URL: `https://doi.org/10.1145/2661829.2662023`.

[20] Junyu Luo et al. "Cross-Modal Image-Text Retrieval with Multitask Learning". In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. CIKM '19. Beijing, China: Association for Computing Machinery, 2019, pp. 2309–2312. ISBN: 9781450369763. DOI: `10.1145/3357384.3358104`. URL: `https://doi.org/10.1145/3357384.3358104`.

[21] Brendan van Rooyen, Aditya Krishna Menon, and Robert C. Williamson. *Learning with Symmetric Label Noise: The Importance of Being Unhinged*. 2015. eprint: `arXiv:1505.07634`.

[22] Bo Han et al. *Co-teaching: Robust Training of Deep Neural Networks with Extremely Noisy Labels*. 2018. eprint: `arXiv:1804.06872`.

[23] Sainbayar Sukhbaatar et al. *Training Convolutional Networks with Noisy Labels*. 2014. eprint: `arXiv:1406.2080`.

[24] Yi Yu et al. "Deep cross-modal correlation learning for audio and lyrics in music retrieval". In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 15.1 (2019), pp. 1–16.

[25] Qiaolin Xia et al. *XGPT: Cross-modal Generative Pre-Training for Image Captioning*. 2020. eprint: `arXiv:2003.01473`.

[26] Xinghua Dai et al. "Bilinear CNN Model for Fine-Grained Classification Based on Subcategory-Similarity Measurement". In: *Applied Sciences* 9 (Jan. 2019), p. 301. DOI: 10.3390/app9020301.

[27] Meixiang Xu et al. "Subspace learning by kernel dependence maximization for cross-modal retrieval". In: *Neurocomputing* 309 (2018), pp. 94–105. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2018.04.073. URL: http://www.sciencedirect.com/science/article/pii/S0925231218305307.

[28] Sebastian Ruder. *An Overview of Multi-Task Learning in Deep Neural Networks*. 2017. eprint: arXiv:1706.05098.

[29] Michael Grubinger et al. "The IAPR TC12 Benchmark: A New Evaluation Resource for Visual Information Systems". In: *Workshop Ontoimage* (Oct. 2006).

[30] Mark J. Huiskes and Michael S. Lew. "The MIR Flickr Retrieval Evaluation". In: *MIR '08: Proceedings of the 2008 ACM International Conference on Multimedia Information Retrieval*. Vancouver, Canada: ACM, 2008.

[31] Thomas Wolf et al. "HuggingFace's Transformers: State-of-the-art Natural Language Processing". In: *ArXiv* abs/1910.03771 (2019).