UNIVERSITY OF CALIFORNIA
Los Angeles

Multiobjective Deep Learning

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Computer Science

by

Jayanth

2019

ABSTRACT OF THE THESIS

Multiobjective Deep Learning

by

Jayanth
Master of Science in Computer Science
University of California, Los Angeles, 2019
Professor Kai Wei Chang, Chair

Many current challenges in natural language processing and computer vision have to deal with multiple objectives simultaneously. In this article, we study different methods to solve such multi-objective problem for CIFAR-100 and SEMEVAL datasets, and compare with traditional deep learning methods. The multi-output method achieves better results than training a single neural net from scratch with its own model for each objective. Multi-objective deep learning with weights achieves comparable results too.

The thesis of Jayanth is approved.

George Varghese

Quanquan Gu

Kai Wei Chang, Committee Chair

University of California, Los Angeles

2019

To my family . . .

who—among so many other things—

helped me survive my schizophrenic episodes

and are still supporting me through my recovery

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First and foremost, I thank Professor Kai Wei Chang, my advisor, without whose guidance, support, and encouragement this thesis would never have been completed. I am fortunate to have worked with him. It has been a pleasure all along working under his guidance. He consistently steered me in the right the direction whenever I needed it.

I would like to thank Professor George Varghese, who provided me with encouragement and help in completing my degree despite my struggles with physical and mental health. He had been a constant support during my tough period of recovery. I am grateful to him for his guidance, and for serving on my thesis committee.

Next, I would like to thank professor Quanquan Gu for serving on my dissertation committee.

I owe my thanks to my colleagues and lab mates, especially Sudharsan Krish for helping me come up the idea.

I am grateful to my parents, Prabhat Kumar Jaiswal and Neeta Kumari, and my sibling, Rathin Kumar, for their love, support, and encouragement during my last year struggle in graduate school due to mental and physical health challenges.

Thank You,

Jayanth

CHAPTER 1

Introduction

Many current challenges in natural language processing and computer vision have to deal with multiple objectives simultaneously. Multi-objective optimization or classification involve more than one objective function that are to be minimized or maximized simultaneously. In this article, we understand the solution of multiple decision problems on same dataset. In this introductory chapter, we shall first give some examples. Informally, we shall understand such problems as mathematical models of decision problems. The remainder of the thesis is organized as follows. Chapter 2 discusses previous approaches used for multi-objective optimization. Chapter 3 gives detailed explanation about the dataset being used. Chapter 4 then discusses the experiments and the results obtained from it. Chapter 5 focus on discussion and the conclusion respectively.

1.1 Examples

Let us consider the following examples from different fields:

Economics: Predicting consumer's demand for various goods is determined by the process of maximization of the consumer's satisfaction derived from those goods, subject to a constraint based on how much income is available to spend on those goods and on the prices of those goods. The consumer's demand for separate goods are the multiple objective in this use-case.

Vision: Given an image, identify the shape, the color and other such physical properties, followed by finally identifying the object or object category. Shape, color, physical properties, object category etc each can serve as seperate objective. Similarly, given the image,

identifying what categories the image belongs to each level as separate objective, before identifying the exact object, reinforcing from the previous labels and improving the final prediction. For example, CIFAR100 has two separate level of labels - coarse and fine, with fine being one level deeper than coarse.

Natural Language Processing: Predicting how many retweets and likes a news headline will receive on Twitter as two separate objectives. Given a text for sentiment analysis, identifying whether it is subjective or objective, if subjective, whether positive or negative, whether sentiment is implicit or explicit, whether it is irony or not, each serving as seperate objective across different levels.

1.2 Typical Approaches

Typical approaches to deal with multi-objectives are to convert all but one into constraints during problem modeling phase. Another approach is to project it onto R^1 or other lower space by giving them weights as hyperparameter and then, optimizing the weighted sum. Typically, all these approaches simplify the problem consideration, losing information and requirements along the way.

1.3 Literature Survey

Multiobjective optimization is an active area of research. It is an area of multiple criteria decision making that is concerned with mathematical optimization problems involving more than one objective function to be optimized simultaneously. Multi-objective learning has been used successfully across all applications of machine learning, from natural language processing [3] and speech recognition [4] to computer vision [5] and drug discovery [6].

CHAPTER 2

Algorithm

Hard parameter sharing is the most commonly used approach to Multi-Task Learning in neural networks and goes back to [1]. It is generally applied by sharing the hidden layers between all tasks, while keeping several task-specific output layers.

Hard parameter sharing greatly reduces the risk of overfitting. In fact, [2] showed that the risk of overfitting the shared parameters is an order N – where N is the number of tasks – smaller than overfitting the task-specific parameters, i.e. the output layers. This makes sense intuitively: The more tasks we are learning simultaneously, the more our model has to find a representation that captures all of the tasks and the less is our chance of overfitting on our original task.

The algorithm is to decompose the network graph into separate branches. Calculate loss with respect to each branch end and then, use the loss to calculate weights for gradient update. The loss can be used directly as weights as in algorithm 1 or as inversely as harmonic weights as in algorithm 2.

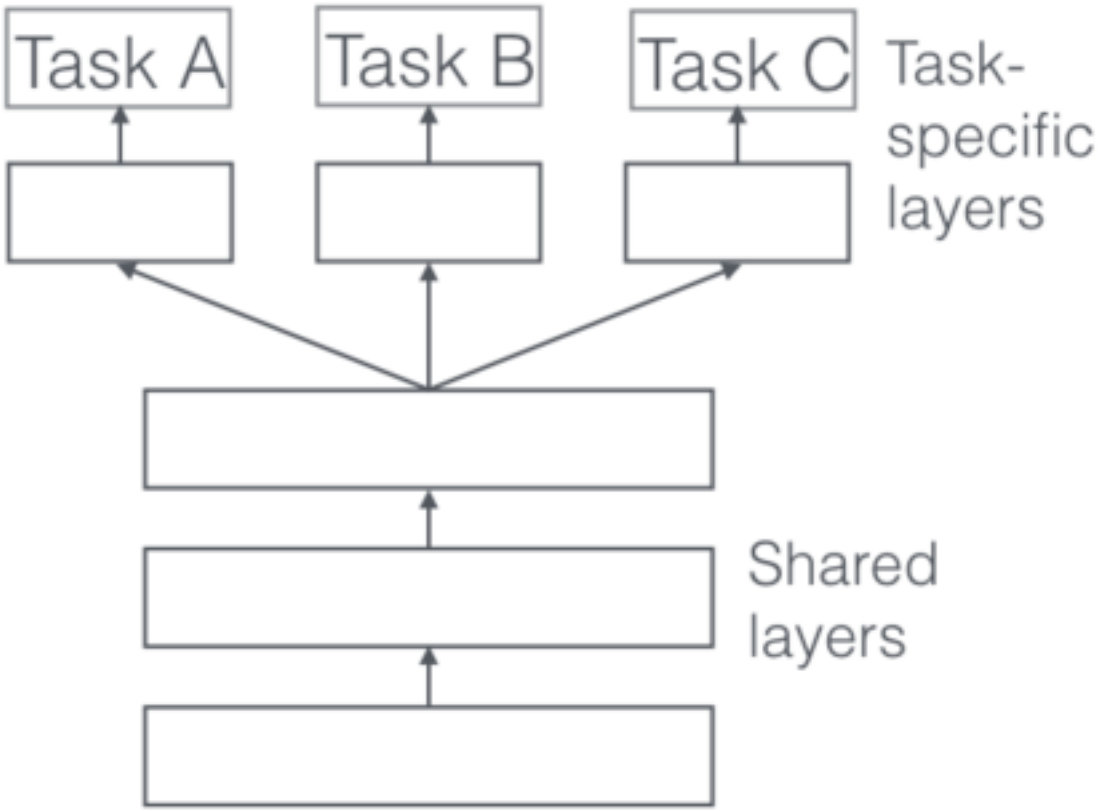

Figure 2.1: Hard parameter sharing for multi-task learning in deep neural networks

define model:

return $branch_1, branch_2 ... branch_n$

$pred_1, pred_2 ... pred_n = model(X)$

for $i \leftarrow 1$ **to** n **do**
| $loss_i = lossfn_i(pred_i, y_i)$

end

for $i \leftarrow 1$ **to** n **do**
| $\alpha_i = \frac{loss_i}{loss_1 + loss_2 + ... loss_n}$

end

for $i \leftarrow 1$ **to** n **do**
| $optimizer_i.step(\alpha_i)$

end

Algorithm 1: Arithmetic weighted update algorithm

define model:

return $branch_1, branch_2 ... branch_n$

$pred_1, pred_2 ... pred_n = model(X)$

for $i \leftarrow 1$ **to** n **do**
| $loss_i = lossfn_i(pred_i, y_i)$

end

for $i \leftarrow 1$ **to** n **do**
| $\frac{1}{\alpha_i} = \frac{\frac{1}{loss_i}}{\frac{1}{loss_1} + \frac{1}{loss_2} + ... \frac{1}{loss_n}}$

end

for $i \leftarrow 1$ **to** n **do**
| $optimizer_i.step(\alpha_i)$

end

Algorithm 2: Harmonic weighted update algorithm

CHAPTER 3

Dataset

The multi-objective deep learning method has been applied to two datasets - one in vision, CIFAR-100 dataset and one in NLP, SEMEVAL Irony dataset.

3.1 CIFAR-100

The CIFAR-100 dataset consists of 60000 32x32 colour images, containing 50000 training images and 10000 test images. The dataset is divided into five training batches and one test batch, each with 10000 images. The training batches contain the images in random order, but some training batches may contain more images from one class than another. This dataset has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. The 100 classes in the CIFAR-100 are grouped into 20 superclasses. Each image comes with a "fine" label (the class to which it belongs) and a "coarse" label (the superclass to which it belongs).

Here is the list of classes in the CIFAR-100:

Superclass	Classes
aquatic	mammals beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food	containers bottles, bowls, cans, cups, plates
fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
household electrical devices	clock, computer keyboard, lamp, telephone, television
household furniture	bed, chair, couch, table, wardrobe
insects	bee, beetle, butterfly, caterpillar, cockroach
large carnivores	bear, leopard, lion, tiger, wolf
large man-made outdoor things	bridge, castle, house, road, skyscraper
large natural outdoor scenes	cloud, forest, mountain, plain, sea
large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
medium-sized mammals	fox, porcupine, possum, raccoon, skunk
non-insect invertebrates	crab, lobster, snail, spider, worm
people	baby, boy, girl, man, woman
reptiles	crocodile, dinosaur, lizard, snake, turtle
small mammals	hamster, mouse, rabbit, shrew, squirrel
trees	maple, oak, palm, pine, willow
vehicles 1	bicycle, bus, motorcycle, pickup truck, train
vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

Table 3.1: Superclass and subclass in CIFAR100 dataset

Figure 3.1: CIFAR-100 images

Here is a sample image containing images in the CIFAR-100.

3.2 SEMEVAL Irony

The SEMEVAL2018 task proposes two different subtasks for the automatic detection of irony on Twitter. For the first subtask, participants should determine whether a tweet is ironic or not (by assigning a binary value 0 or 1). For the second subtask, participants are tasked with distinguishing between non-ironic and ironic tweets, the latter of which are subdivided into three categories. More details of both subtasks are described below.

3.2.1 Ironic vs. non-ironic

The first subtask is a two-class (or binary) classification task where the system has to predict whether a tweet is ironic or not. The following sentences present examples of an ironic and non-ironic tweet, respectively.

- I just love when you test my patience!! #not

- Had no sleep and have got school now #not happy

3.2.2 Different types of irony

The second subtask is a multiclass classification task where the system has to predict one out of four labels describing i) verbal irony realized through a polarity contrast, ii) verbal irony without such a polarity contrast (i.e., other verbal irony), iii) descriptions of situational irony, iv) non-irony. A brief description and example for each label are presented below.

3.2.2.1 Verbal irony by means of a polarity contrast

Instances containing an evaluative expression whose polarity (positive, negative) is inverted between the literal and the intended evaluation.

Examples:

- I love waking up with migraines #not :'(
- I really love this year's summer; weeks and weeks of awful weather.

In the above examples, the irony results from a polarity inversion between two evaluations. In sentence 4 for instance, the literal evaluation ("I really love this year's summer") is positive, while the intended one, which is implied by the context ("weeks and weeks of awful weather"), is negative.

3.2.2.2 Other verbal irony

Instances which show no polarity contrast between the literal and the intended evaluation, but are nevertheless ironic.

Examples:

- @someuser Yeah keeping cricket clean, that's what he wants #Sarcasm
- Human brains disappear every day. Some of them have never even appeared. http://t.co/Fb0Aq5Frqs

—#brain #humanbrain #Sarcasm

3.2.2.3 Situational irony

Instances describing situational irony, or situations that fail to meet some expectations. As explained by Shelley (2001), firefighters who have a fire in their kitchen while they are out to answer a fire alarm would be a typically ironic situation.

Examples:

- Most of us didn't focus in the #ADHD lecture. #irony
- Event technology session is having Internet problems. #irony #HSC2024
- Just saw a non-smoking sign in the lobby of a tobacco company #irony

3.2.2.4 Non-ironic

Instances that are clearly not ironic, or which lack context to be sure that they are ironic. Examples:

- And then my sister should be home from college by time I get home from babysitting. And it's payday. THIS IS A GOOD FRIDAY
- Please dont fuck with me when I first wake up #not a morning person!

CHAPTER 4

CIFAR-100 Experiment

For CIFAR-100 dataset, the 20 "coarse" superclass labels serve as first objective while, the 100 "fine" subclass labels serve as the second objective. The aim of the deep learning model is to predict both coarse and fine label simultaneously as separate objectives.

4.1 Motivation for CNN

Convolutional neural nets have an interesting ability to learn spatial relations from raw data without prior feature selection or feature engineering. This end to end learning can be leveraged for classification tasks where the data is image data because it is difficult to make prior assumptions about features for this sort of data set. We designed a custom 4 layered convolutional neural net for the classification task. Please see Table 1 for performance comparisons and Table 3 for architecture details.

4.2 Baseline Single Neural Net architecture

All the architecture was kept common across all the experiment. The baseline architecture was single neural net for each objective separately.

The batch size was kept at 128, the epochs are set at 50, the optimizer is Adam with learning rate of $3e-4$ with loss as categorical cross entropy. The total parameters are 1,255,988 in the single neural net coarse architecture. It achieved training loss of 0.7338 and training accuracy of 75.15% with validation loss of 1.3746 and validation accuracy of 59.26% for coarse predictions.

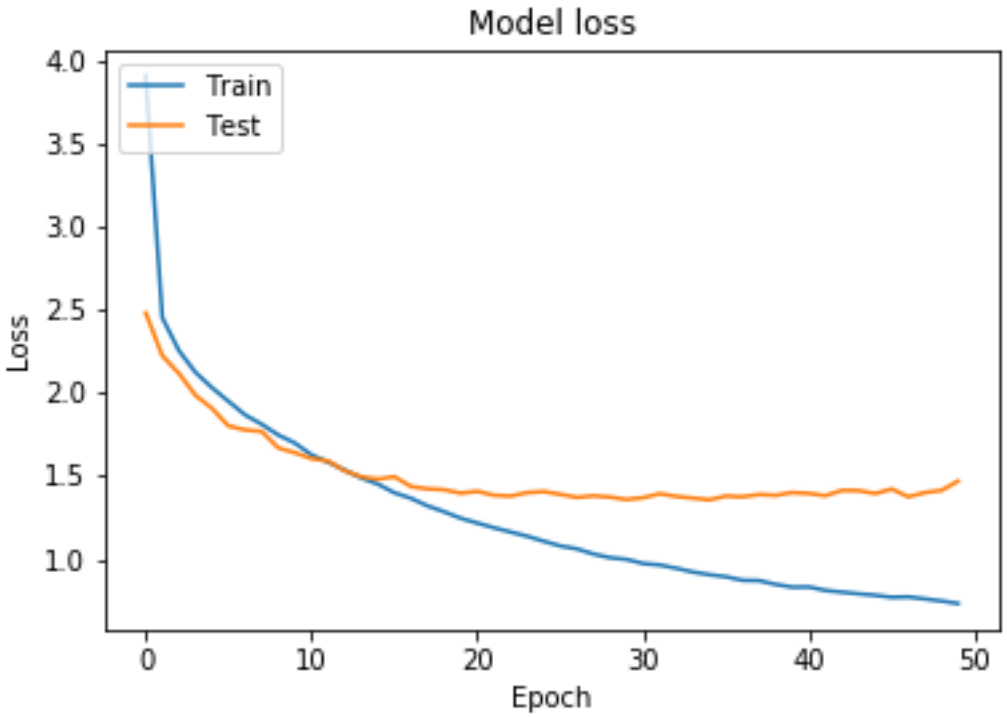

Figure 4.1: Loss of baseline coarse neural net

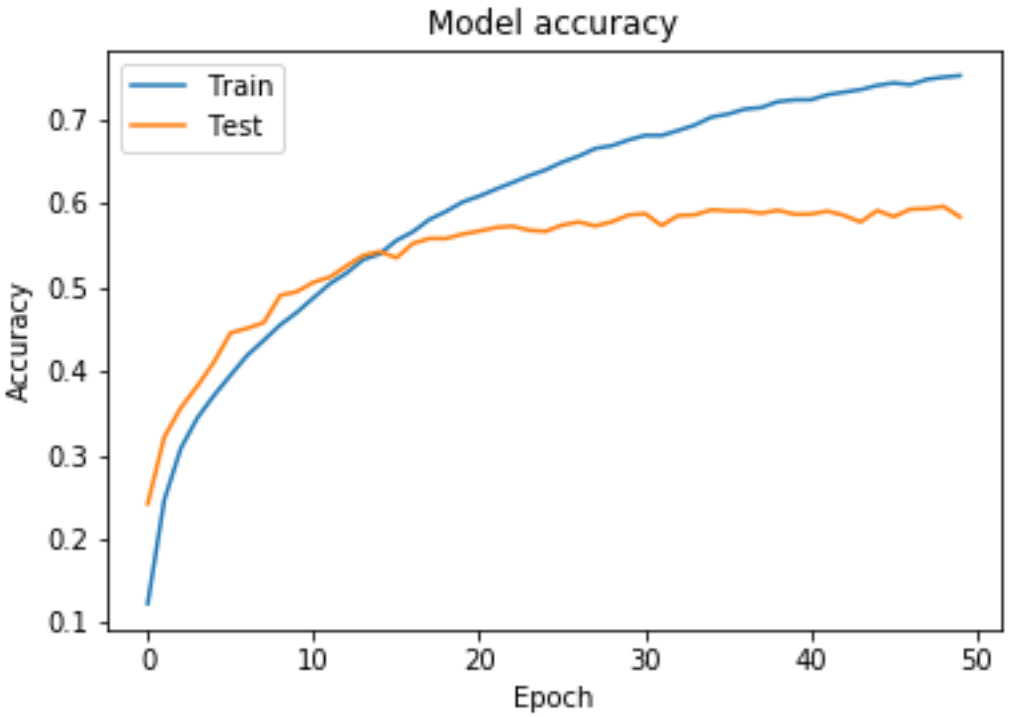

Figure 4.2: Accuracy of baseline coarse neural net

Layer (type)	Output Shape	Parameters
$conv2d_1$ (Conv2D)	(None, 32, 32, 32)	896
$activation_1$ (Activation)	(None, 32, 32, 32)	0
$conv2d_2$ (Conv2D)	(None, 30, 30, 32)	9248
$activation_2$ (Activation)	(None, 30, 30, 32)	0
$maxpooling2d_1$ (MaxPooling2)	(None, 15, 15, 32)	0
$dropout_1$ (Dropout)	(None, 15, 15, 32)	0
$conv2d_3$ (Conv2D)	(None, 15, 15, 64)	18496
$activation_3$ (Activation)	(None, 15, 15, 64)	0
$conv2d_4$ (Conv2D)	(None, 13, 13, 64)	36928
$activation_4$ (Activation)	(None, 13, 13, 64)	0
$maxpooling2d_2$ (MaxPooling2)	(None, 6, 6, 64)	0
$dropout_2$ (Dropout)	(None, 6, 6, 64)	0
$flatten_1$ (Flatten)	(None, 2304)	0
$dense_1$ (Dense)	(None, 512)	1180160
$activation_5$ (Activation)	(None, 512)	0
$dropout_3$ (Dropout)	(None, 512)	0
$dense_2$ (Dense)	(None, 20)	10260
$activation_6$ (Activation)	(None, 20)	0

Table 4.1: Table for single neural network for coarse objective

The total parameters are 1,297,028 in the single neural net fine architecture. They only differ in last layer, where there is a dense layer with 512 input and 100 output for fine, 20 output for coarse respectively. It achieved training loss of 15.957 and training accuracy of 1% with validation loss of 15.957 and validation accuracy of 1% for fine predictions.

The average accuracy of Single net models on both the task was around 30%.

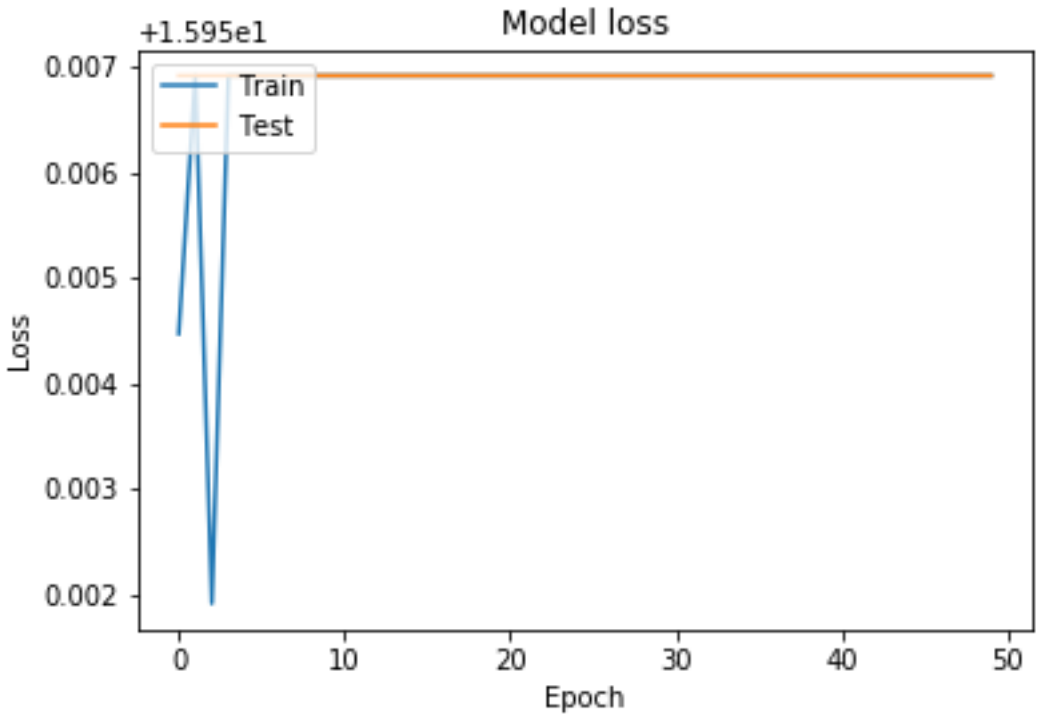

Figure 4.3: Loss of baseline fine neural net

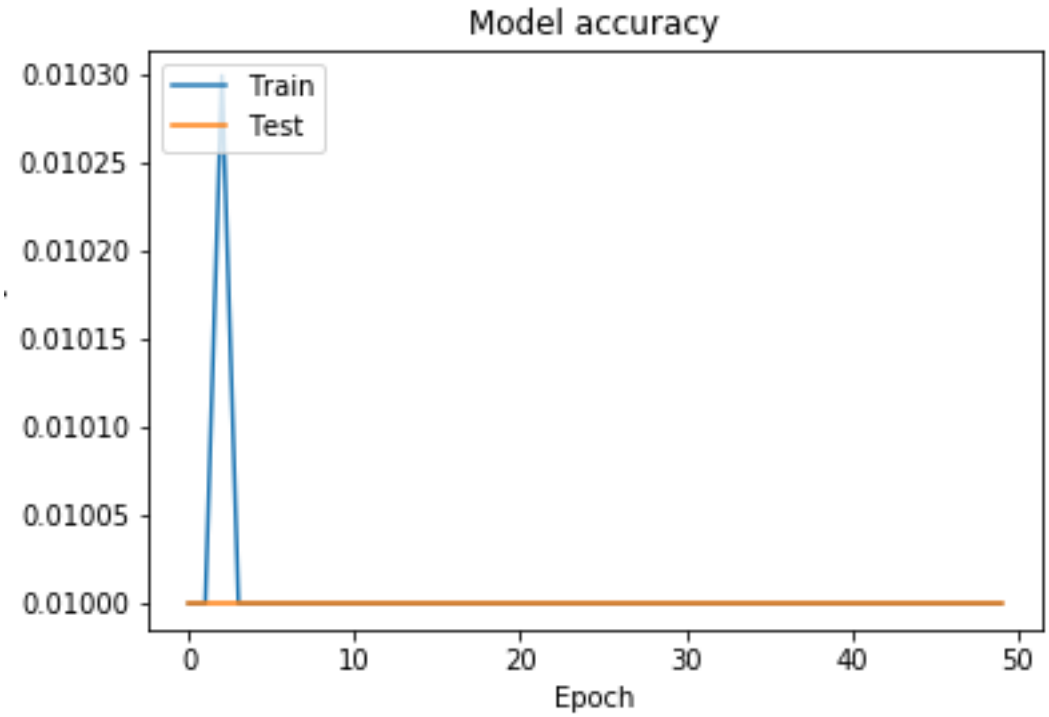

Figure 4.4: Accuracy of baseline fine neural net

Layer (type)	Output Shape	Parameters
$conv2d_1$ (Conv2D)	(None, 32, 32, 32)	896
$activation_1$ (Activation)	(None, 32, 32, 32)	0
$conv2d_2$ (Conv2D)	(None, 30, 30, 32)	9248
$activation_2$ (Activation)	(None, 30, 30, 32)	0
$maxpooling2d_1$ (MaxPooling2)	(None, 15, 15, 32)	0
$dropout_1$ (Dropout)	(None, 15, 15, 32)	0
$conv2d_3$ (Conv2D)	(None, 15, 15, 64)	18496
$activation_3$ (Activation)	(None, 15, 15, 64)	0
$conv2d_4$ (Conv2D)	(None, 13, 13, 64)	36928
$activation_4$ (Activation)	(None, 13, 13, 64)	0
$maxpooling2d_2$ (MaxPooling2)	(None, 6, 6, 64)	0
$dropout_2$ (Dropout)	(None, 6, 6, 64)	0
$flatten_1$ (Flatten)	(None, 2304)	0
$dense_1$ (Dense)	(None, 512)	1180160
$activation_5$ (Activation)	(None, 512)	0
$dropout_3$ (Dropout)	(None, 512)	0
$dense_2$ (Dense)	(None, 100)	51300
$activation_6$ (Activation)	(None, 100)	0

Table 4.2: Table for single neural network for fine objective

4.3 Multi-output Unbranched Neural Net

Next, we tried unbranched neural net for both the task. The penultimate layer gave the fine output, while the last layer gave the coarse output.

The total parameters are 1,299,048 in the above architecture. The layer $dense_2$ followed by $activation_6$ serves as fine output and the layer $dense_3$ followed by $activation_7$ serves as coarse output. It achieved training loss of 18.9527 with validation loss of 18.9527 and

Layer (type)	Output Shape	Parameters
$conv2d_1$ (Conv2D)	(None, 32, 32, 32)	896
$activation_1$ (Activation)	(None, 32, 32, 32)	0
$conv2d_2$ (Conv2D)	(None, 30, 30, 32)	9248
$activation_2$ (Activation)	(None, 30, 30, 32)	0
$maxpooling2d_1$ (MaxPooling2)	(None, 15, 15, 32)	0
$dropout_1$ (Dropout)	(None, 15, 15, 32)	0
$conv2d_3$ (Conv2D)	(None, 15, 15, 64)	18496
$activation_3$ (Activation)	(None, 15, 15, 64)	0
$conv2d_4$ (Conv2D)	(None, 13, 13, 64)	36928
$activation_4$ (Activation)	(None, 13, 13, 64)	0
$maxpooling2d_2$ (MaxPooling2)	(None, 6, 6, 64)	0
$dropout_2$ (Dropout)	(None, 6, 6, 64)	0
$flatten_1$ (Flatten)	(None, 2304)	0
$dense_1$ (Dense)	(None, 512)	1180160
$activation_5$ (Activation)	(None, 512)	0
$dropout_3$ (Dropout)	(None, 512)	0
$dense_2$ (Dense)	(None, 100)	51300
$activation_6$ (Activation)	(None, 100)	0
$dense_3$ (Dense)	(None, 20)	2020
$activation_7$ (Activation)	(None, 20)	0

Table 4.3: Table for single neural network for fine objective

validation accuracy of 5% for coarse, 1% for fine leading to average accuracy of 3%. As we can see, the outputs get stagnant at fixed values of random classification because we used single neural net without branching, leading to low accuracy.

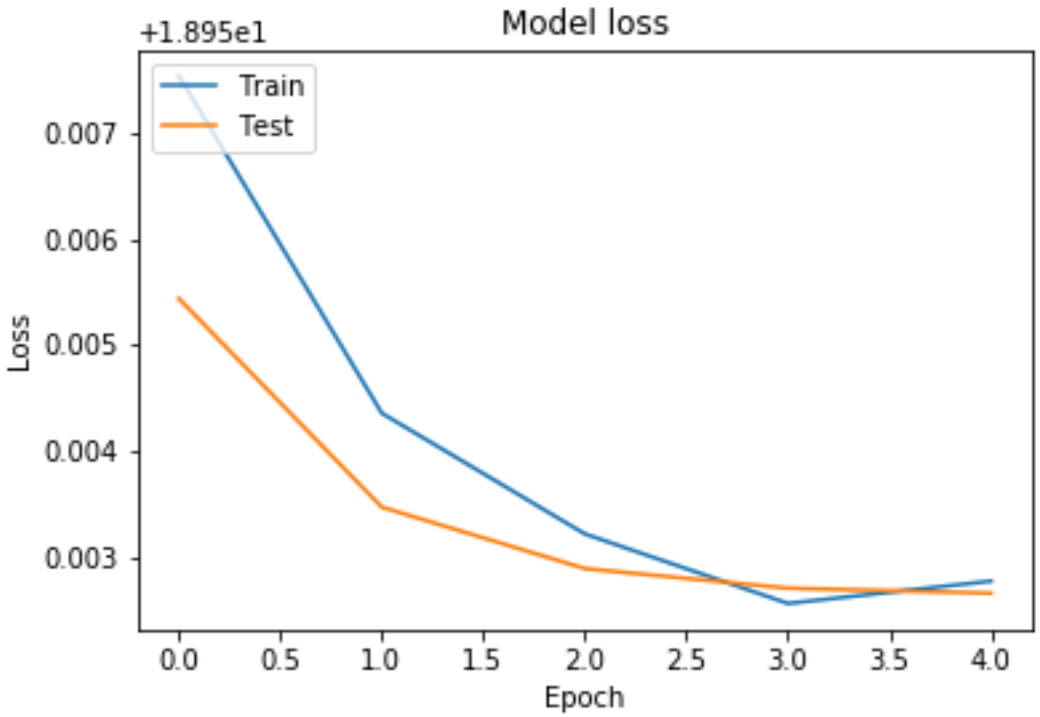

Figure 4.5: Model loss of unbranched neural net

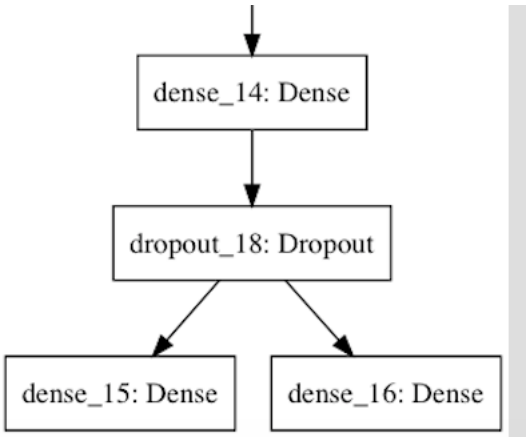

Figure 4.6: Last layers of Multi-output Branched Neural Net

4.4 Multi-output Branched Neural Net

Next, we tried branched neural net for both the task. The penultimate layer branched into two layers, that gave the fine output and the coarse output. The loss gave equal weights to both objective loss with update as arithmetic weighted (page 5).

The total parameters are 1,307,288 in the above architecture. It achieved training loss of 2.2288 with validation loss of 3.4648 and validation accuracy of 59.2% for coarse, 46.13% for fine leading to average accuracy of 52.665%. As we can see, the average accuracy is 52.66% for coarse and fine label based single multi-output neural net model, which is more than single output models by more than 20

4.5 Weighted Multi-output Neural Net

Next, we tried branched neural net for both the task. The penultimate layer branched into two layers, that gave the fine output and the coarse output. The loss gave inverse weights to both objective loss with update as harmonic weighted (page 5).

It achieved training loss of 3.4873 with validation loss of 4.211 and validation accuracy of 42% for coarse, 32% for fine leading to average accuracy of 37% .

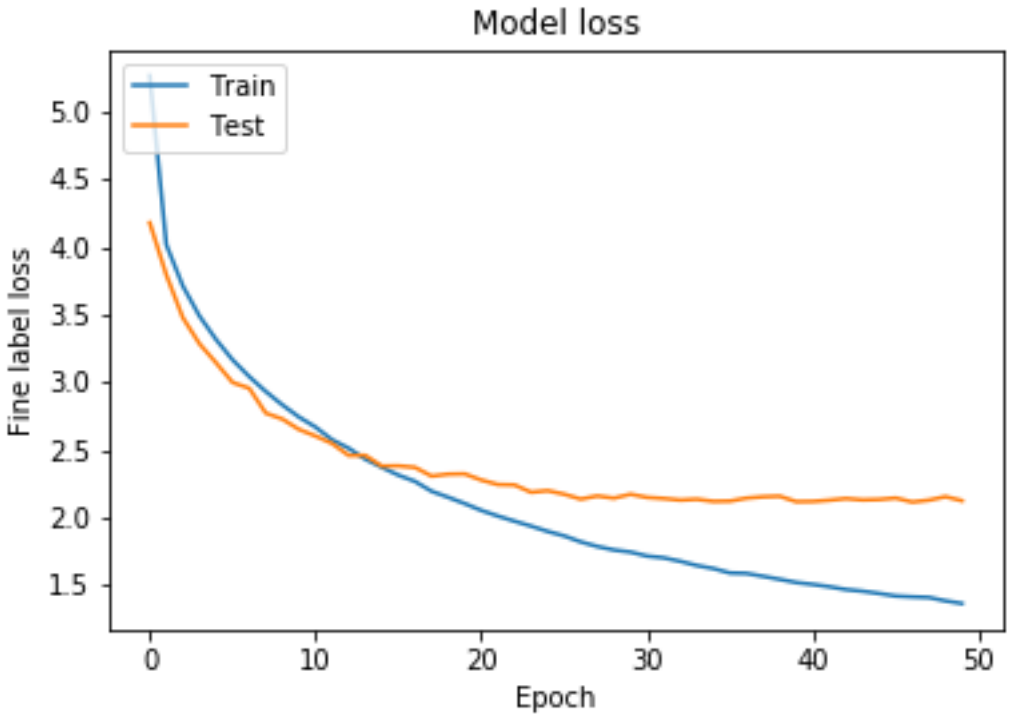

Figure 4.7: Fine loss of multi-output branched neural net

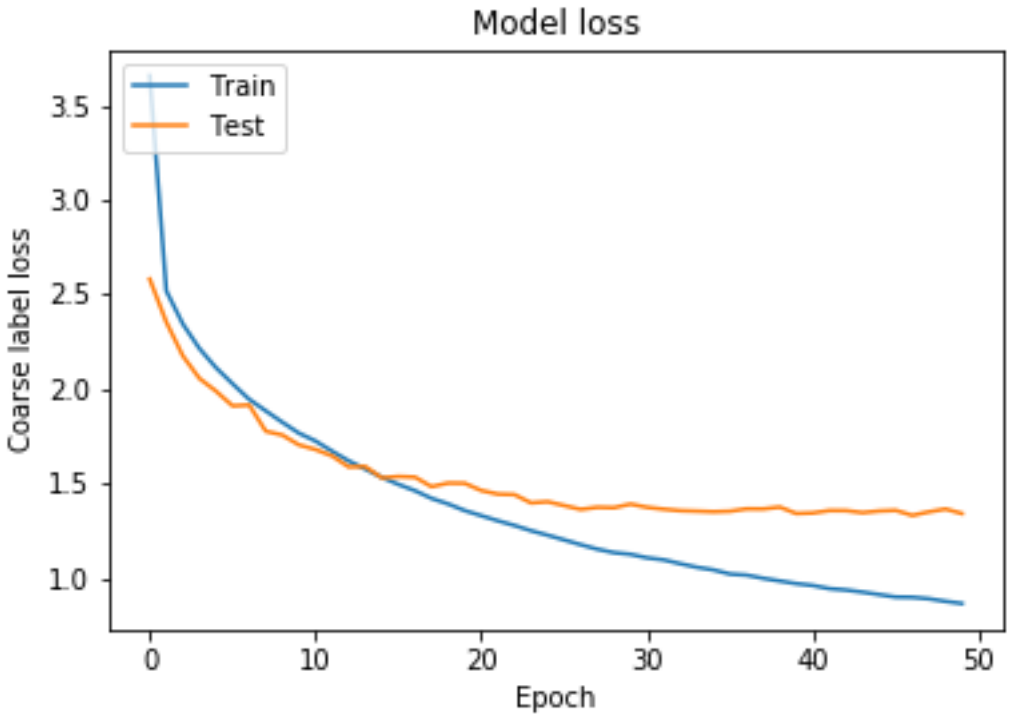

Figure 4.8: Coarse loss of multi-output branched neural net

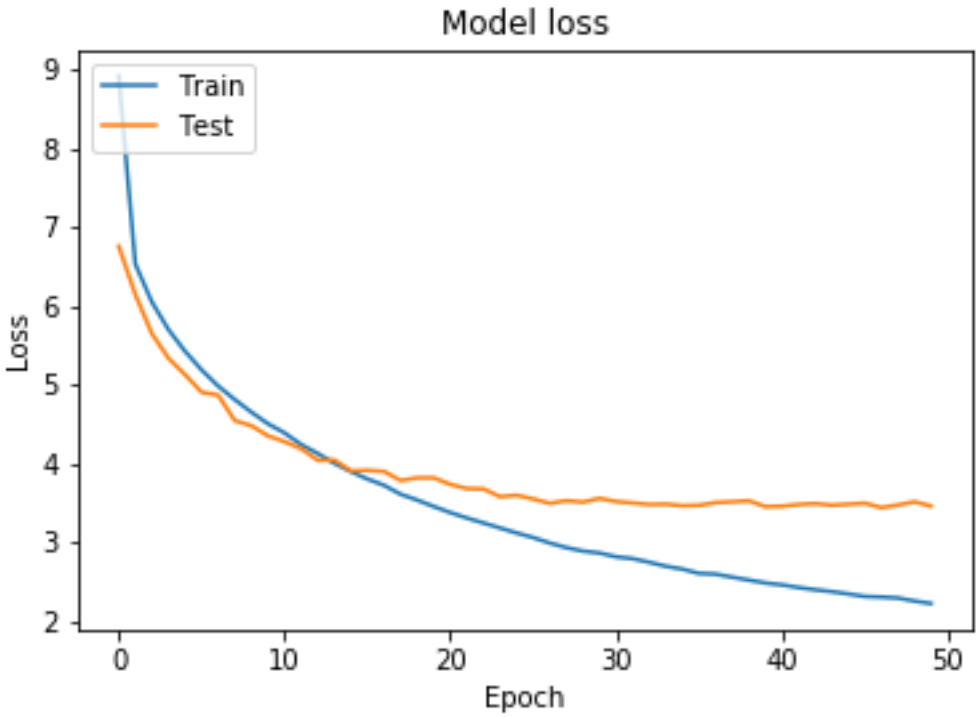

Figure 4.9: Model loss of multi-output branched neural net

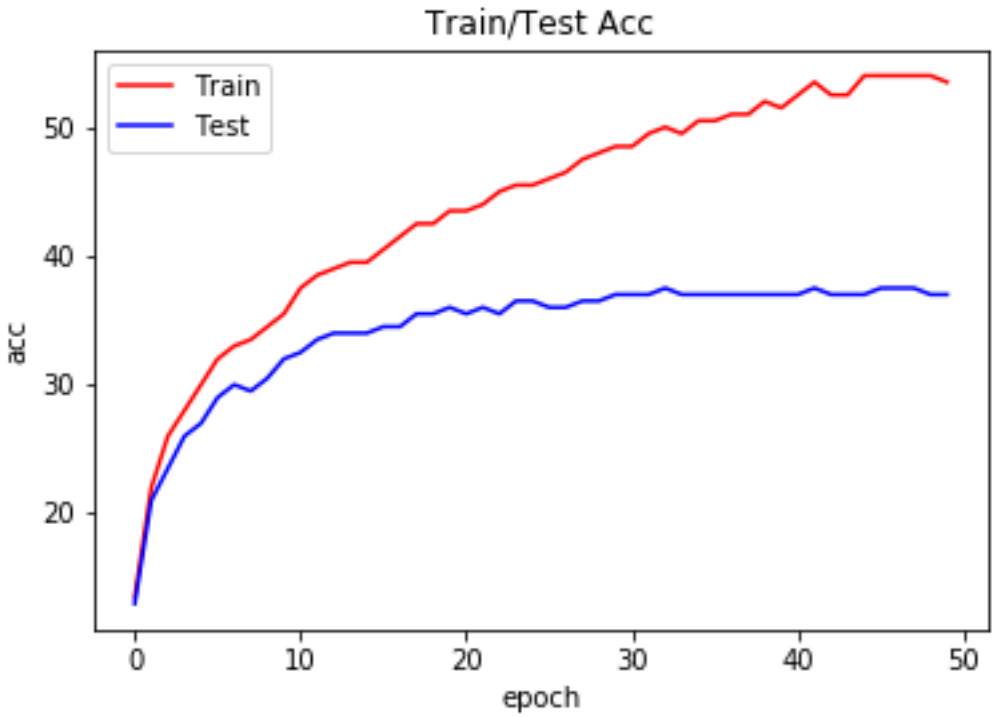

Figure 4.10: Accuracy of weighted multi-output branched neural net

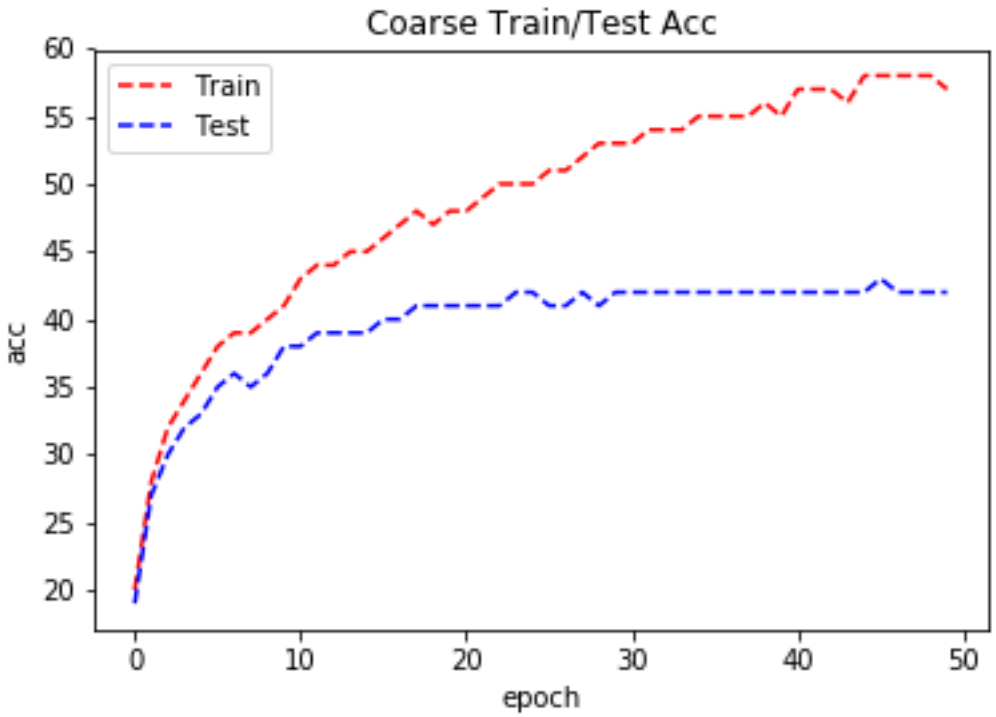

Figure 4.11: Coarse Accuracy of weighted multi-output branched neural net

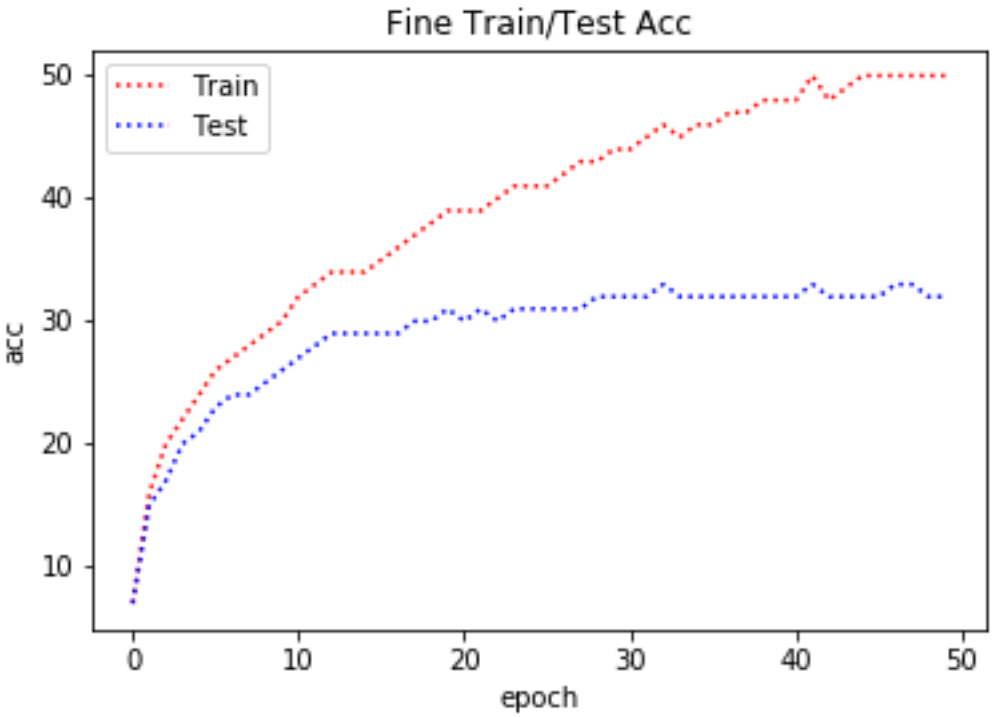

Figure 4.12: Fine Accuracy of weighted multi-output branched neural net

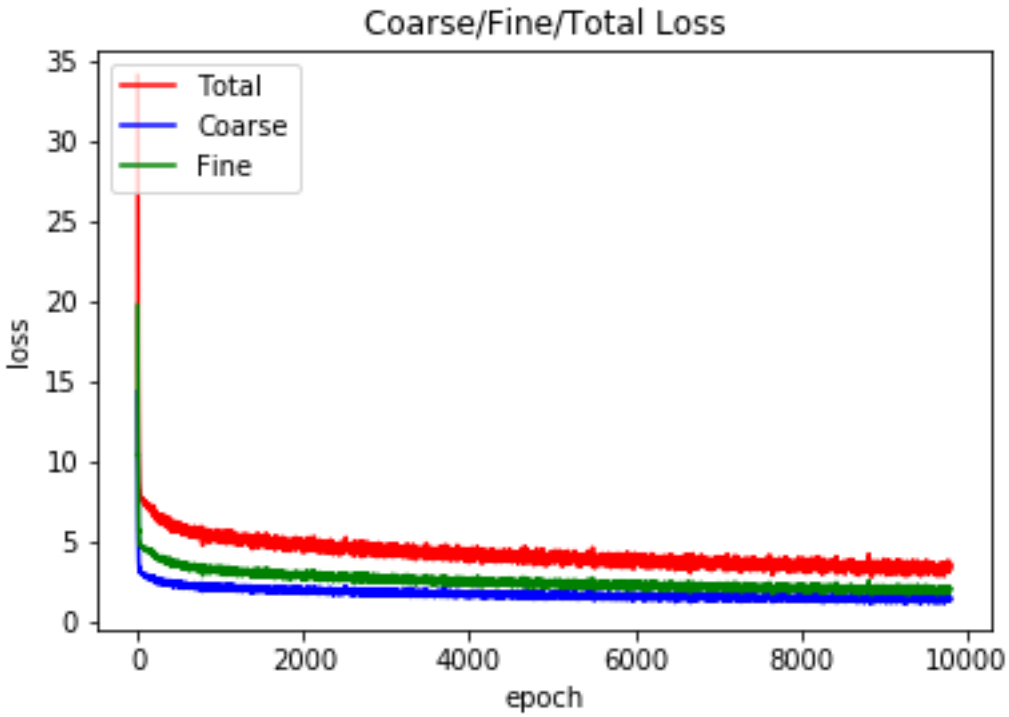

Figure 4.13: Loss of weighted multi-output branched neural net

4.6 Multi-output Resnet

Next, we modified Resnet architecture with depth of 32 to multi-output architechture . The penultimate layer branched into two layers, that gave the fine output and the coarse output. The loss gave equal weights to both objective loss with update as arithmetic weighted (page 5).

The total parameters are 477,368 in the above architecture, with 475,096 trainable parameters and 2,272 non-trainable parameters. It achieved validation loss of 3.108 and validation accuracy of 69.19% for coarse, 53.68% for fine leading to average accuracy of 61.435% .

4.7 Results

Architecture	Loss	Coarse Accuracy	Fine Accuracy	Average Accuracy
Baseline Single net	17.33	59.26%	1%	30.13%
Multi-output Unbranched	18.95	5%	1%	3%
Multi-output branched	3.46	59.2%	46.13%	52.66%
Weighted multi-output	4.211	42%	32%	37%
Multi-output Resnet	3.108	69.19%	53.68%	61.43%

Table 4.4: Table of results

CHAPTER 5

SEMEVAL Irony Experiment

For SEMEVAL dataset, the 2 "coarse" superclass labels - irony vs non-irony serve as first objective while, the 4 "fine" irony subclass labels serve as the second objective. The aim of the deep learning model is to predict both coarse and fine label simultaneously as separate objectives.

5.1 Motivation for recurrent net

Recurrent Neural Networks (RNN) are a powerful and robust type of neural networks and belong to the most promising algorithms out there at the moment because they are the only ones with an internal memory.

Because of their internal memory, RNN's are able to remember important things about the input they received, which enables them to be very precise in predicting what's coming next.

This is the reason why they are the preferred algorithm for sequential data like time series, speech, text, financial data, audio, video, weather and much more because they can form a much deeper understanding of a sequence and its context, compared to other algorithms.

5.2 Baseline Single Neural Net architecture

We created two architectures to model the prediction for SEMEVAL Irony dataset. The first was simple GRU net, which used word embeddings as features.

The batch size was kept at 3, the epochs are set at 50, the optimizer is Adam with

Figure 5.1: GRU Model

learning rate of $1e-4$ with loss as negative log likelihood loss. It achieved training loss of 0 and training accuracy of 100% with validation loss of 4.6688 and validation accuracy of 60.16% for coarse predictions. It achieved training loss of 0.0158 and training accuracy of 99.36% with validation loss of 5.7352 and validation accuracy of 47.66% for fine predictions. The average accuracy for both predictions was 53.91%.

5.3 Multi-output Branched Neural Net

Next, we tried branched neural net for both the task. The penultimate layer branched into two layers, that gave the fine output and the coarse output. The loss gave equal weights to both objective loss with update as arithmetic weighted (page 5).

The batch size was kept at 3, the epochs are set at 50, the optimizer is Adam with learning rate of $1e-4$ with loss as negative log likelihood loss. It achieved training loss of 0 and training accuracy of 100% with validation loss of 10.4498 and validation accuracy of

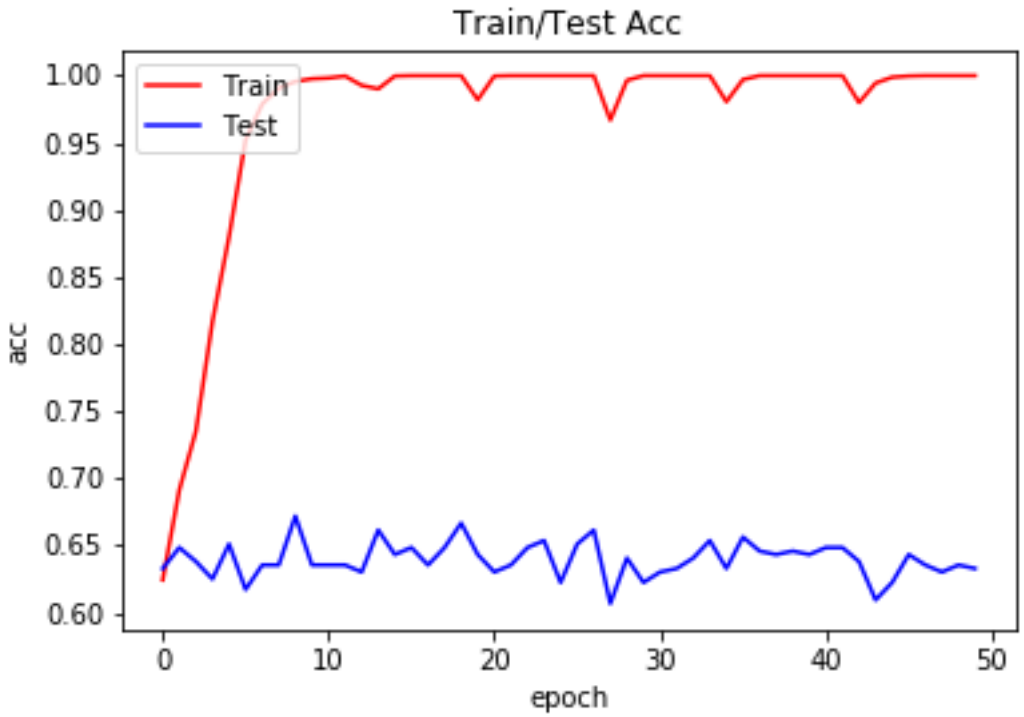

Figure 5.2: Accuracy of baseline coarse neural net

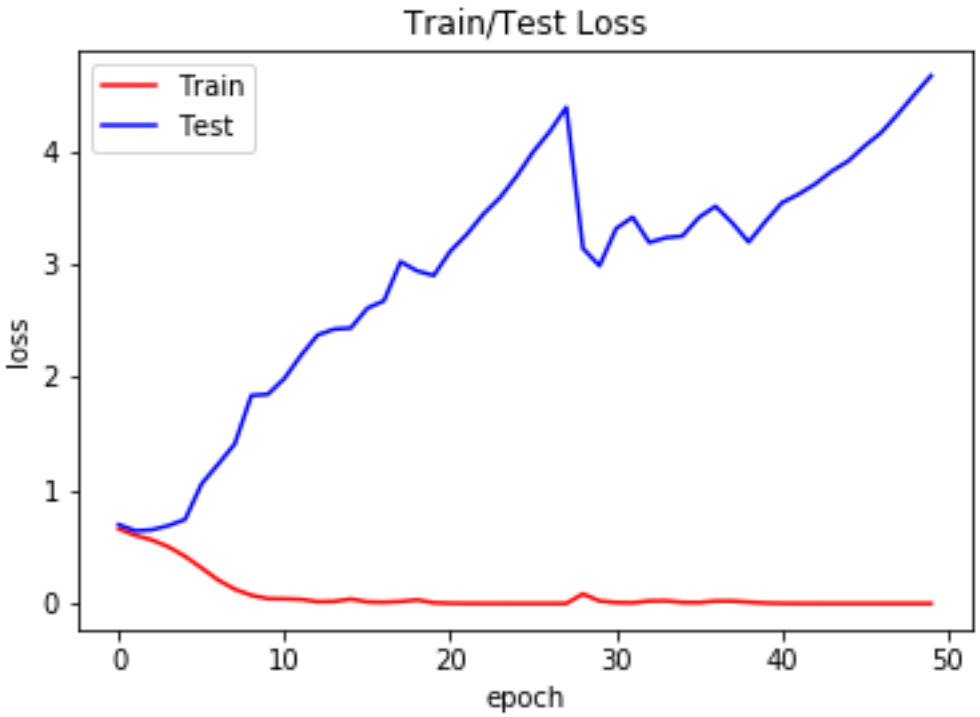

Figure 5.3: Loss of baseline coarse neural net

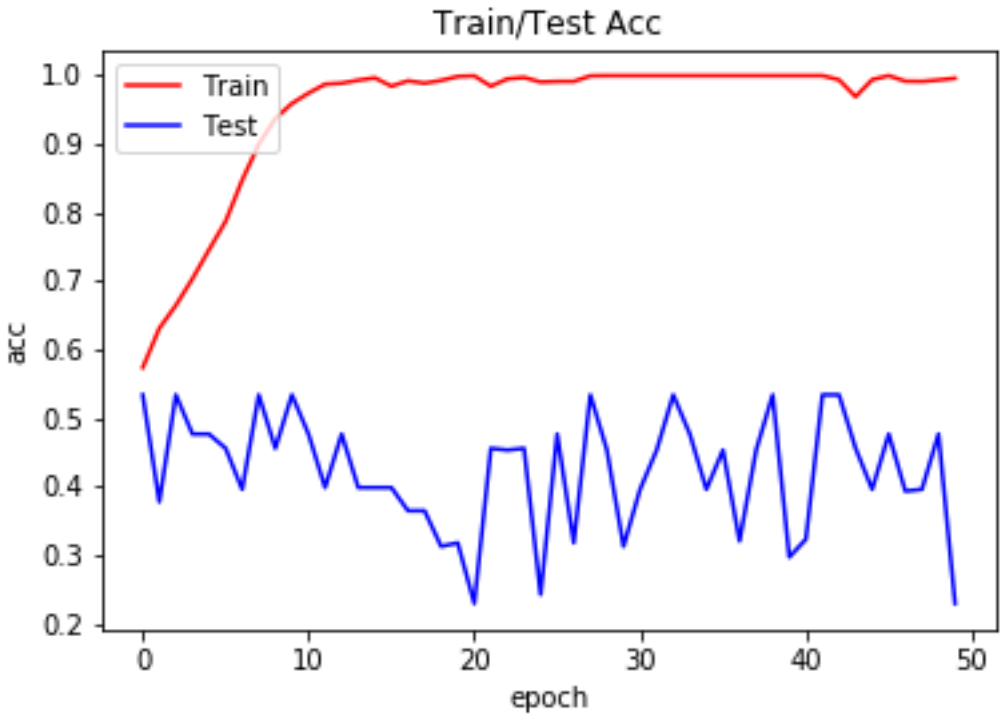

Figure 5.4: Accuracy of baseline fine neural net

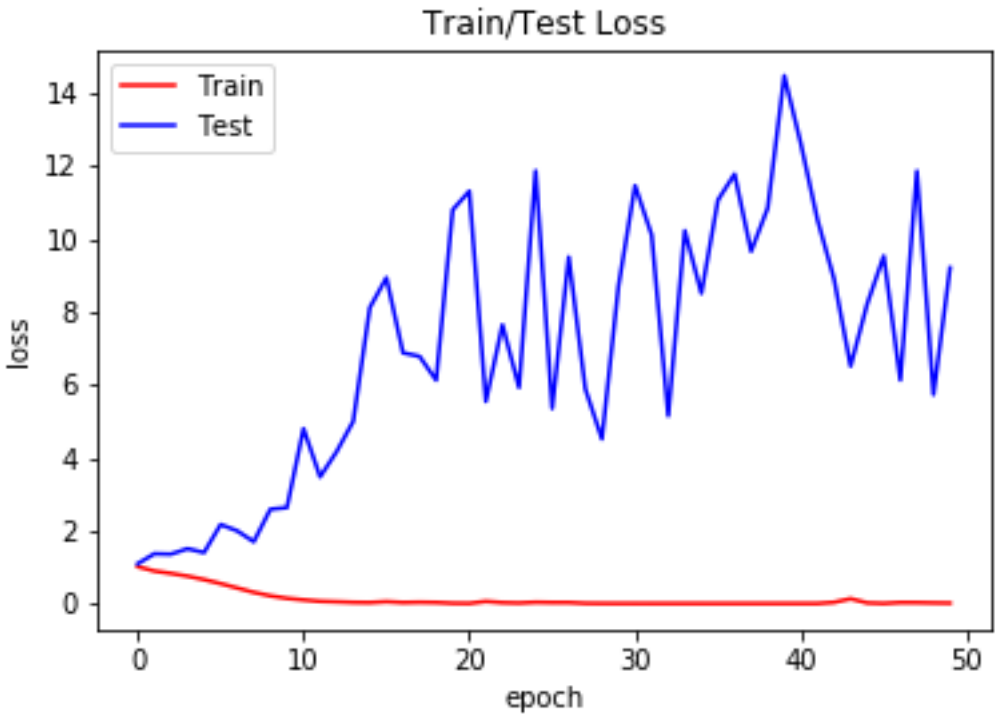

Figure 5.5: Loss of baseline fine neural net

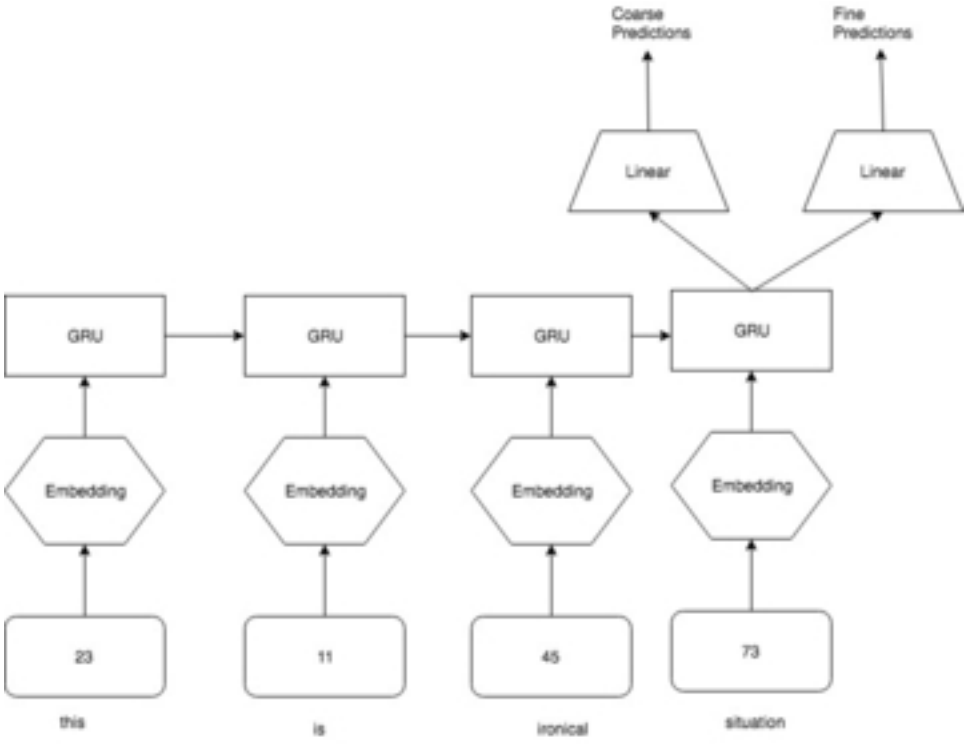

Figure 5.6: Multi-output Branched GRU Neural Net

56.64% for both predictions on average.

5.4 Weighted Multi-output Neural Net

Next, we tried branched neural net for both the task. The penultimate layer branched into two layers, that gave the fine output and the coarse output. The loss gave inverse weights to both objective loss with update as harmonic weighted (page 5).

The batch size was kept at 3, the epochs are set at 50, the optimizer is Adam with learning rate of $1e-4$ with loss as negative log likelihood loss. It achieved training loss of 0.0022 and training accuracy of 100% with validation loss of 5.8202 and validation accuracy of 61.59% for both predictions on average.

5.5 Results

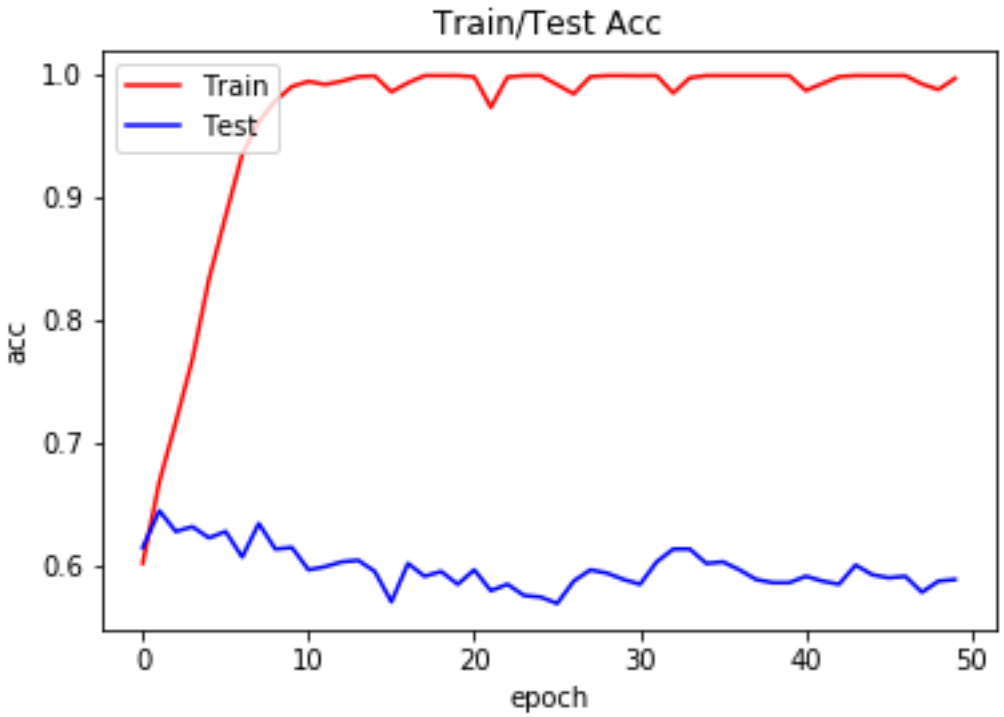

Figure 5.7: Accuracy of GRU multi-output neural net

Figure 5.8: Loss of GRU multi-output neural net

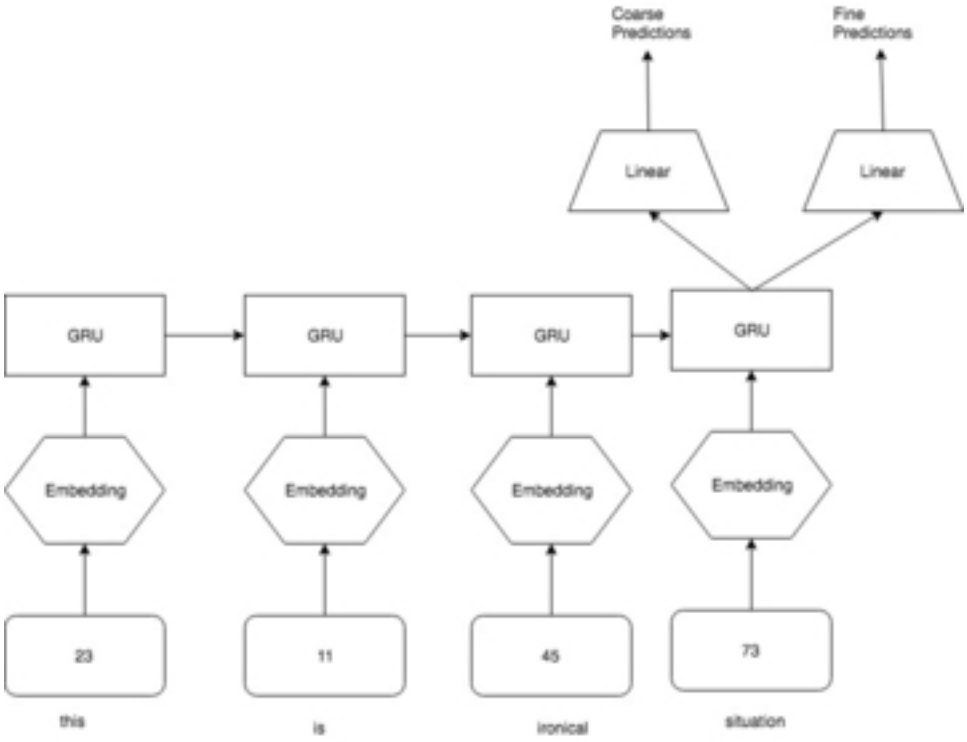

Figure 5.9: Multi-output Branched GRU Neural Net

Architecture	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
Baseline Single net	0.0158	99.68%	10.404	53.91%
Multi-output branched	0	100%	10.4498	56.64%
Weighted multi-output	0.0022	100%	5.8202	61.59%

Table 5.1: Table of results

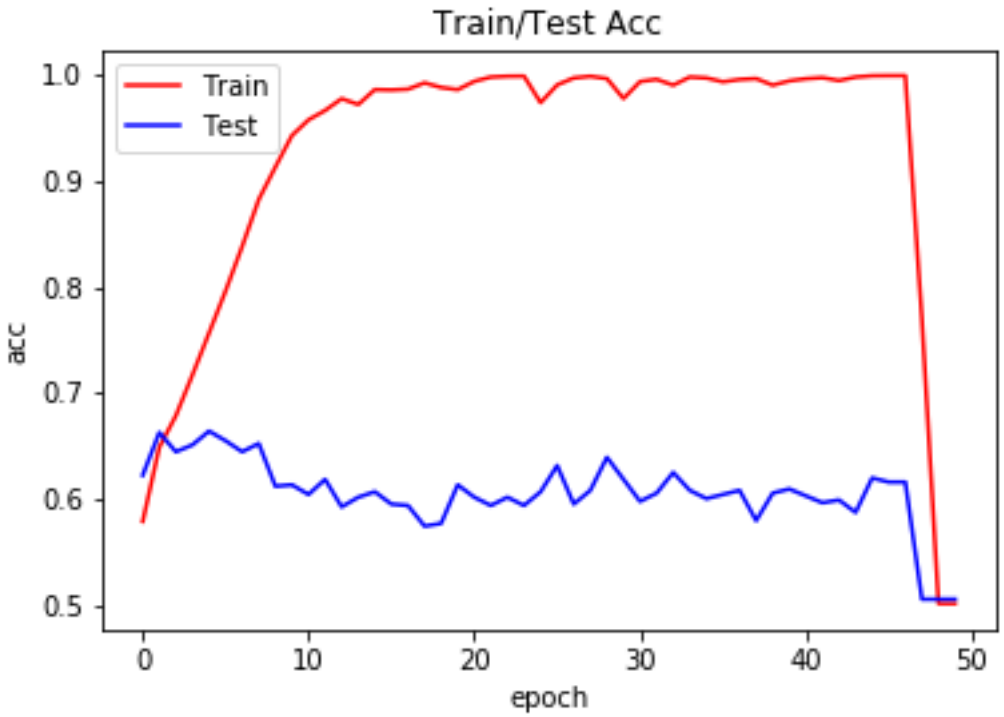

Figure 5.10: Accuracy of GRU weighted multi-output neural net

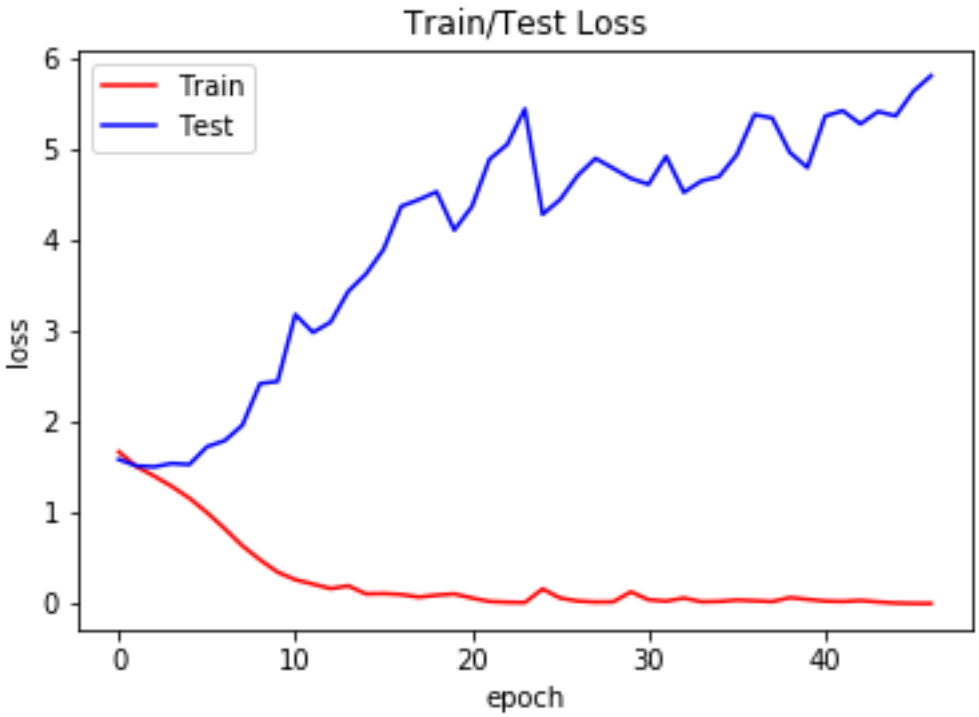

Figure 5.11: Loss of GRU weighted multi-output neural net

REFERENCES

[1] Caruana, R. *"Multitask learning: A knowledge-based source of inductive bias."* Proceedings of the Tenth International Conference on Machine Learning. 1993.

[2] Baxter, J. (1997). *"A Bayesian/information theoretic model of learning to learn via multiple task sampling."* Machine Learning, 28, 7–39.

[3] Collobert, R., & Weston, J. (2008). *"A unified architecture for natural language processing."* Proceedings of the 25th International Conference on Machine Learning - ICML '08, 20(1), 160–167.

[4] Deng, L., Hinton, G. E., & Kingsbury, B. (2013). *"New types of deep neural network learning for speech recognition and related applications: An overview."* 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 8599–8603.

[5] Girshick, R. (2015). *"Fast R-CNN."* In Proceedings of the IEEE International Conference on Computer Vision (pp. 1440–1448).

[6] Ramsundar, B., Kearnes, S., Riley, P., Webster, D., Konerding, D., & Pande, V. (2015). *"Massively Multitask Networks for Drug Discovery."*