|  |  |
| --- | --- |
| Name: | Edward Eisenberger |
| ID#: | 1066164 |
| Assignment 5 | |
| Date of Submission: | May 5, 2019 |

Table of Contents

[Part A 3](#_Toc7968021)

[Summary 3](#_Toc7968022)

[Results 3](#_Toc7968023)

[Part B 6](#_Toc7968024)

[Summary 6](#_Toc7968025)

[Results 6](#_Toc7968026)

[Part C 7](#_Toc7968027)

[Summary 7](#_Toc7968028)

[Results 8](#_Toc7968029)

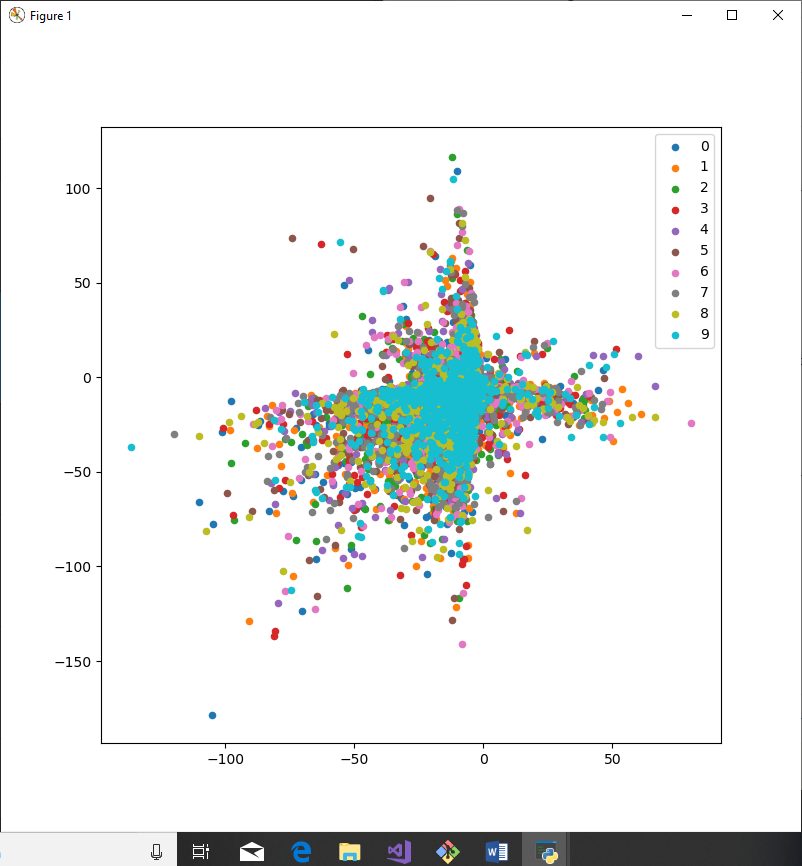
# Part A

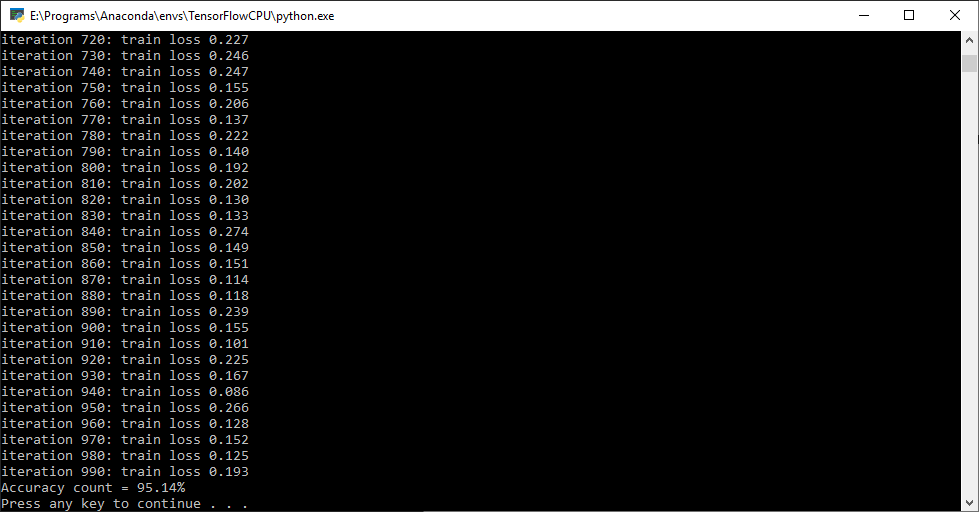
## Summary

Part A of Assignment 07 involved implementing examples of using TensorFlow in python.

## Results

The following images contain outputs from the example Siamese network.





# Part B

## Summary

Part B of Assignment 07 involved studying the Siamese Network using the one-shot learning approach, detailed in <https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf>. This approach was then compared to the contrastive loss function.

The goal of this study was to answer the following:

Can both loss functions be combined to make a better verification network?

## Results

The one-shot learning loss function is:

The contrastive loss function is:

The prediction vector term, p, in compares to the distance term, Dw, in .

The objective of one-shot learning is to drive similar samples to predict 1, while driving dissimilar samples to predict 0. This differs from Contrastive loss, which has the objective of embedding similar inputs close together while embedding dissimilar inputs further apart in a reduced dimensional space.

The loss function could be combined to make a better verification network. The two models could be combined in series such that the embeddings from a trained Contrastive Loss Model are used as the inputs to a one-shot learning model. This approach would allow the one-shot learning model to differentiate on the key features embedded by the Contrastive Loss model.

# Part C

## Summary

Part C of Assignment 07 involved implementing the Triplet Network described in https://arxiv.org/pdf/1412.6622.pdf using TensorFlow. The Triplet Network was implemented as a Deep Convolutional Neural Network (CNN). The CNN consists of 3 layers. The CNN takes in a 784x1 image, reshapes it to 28x28, and feeds the reshaped image into the first the layer.

The first layer takes in a 28x28 image and outputs 32 feature maps via:

1. Applies a 5x5 kernel
2. Applies RELU activation
3. Performs 2x2 max pooling

The output of the first layer is fed into the second layer which outputs 64 feature maps via:

1. Applies a 5x5 kernel
2. Applies RELU activation
3. Performs 2x2 max pooling

The output of the second layer is fed into the third layer which outputs 128 feature maps via:

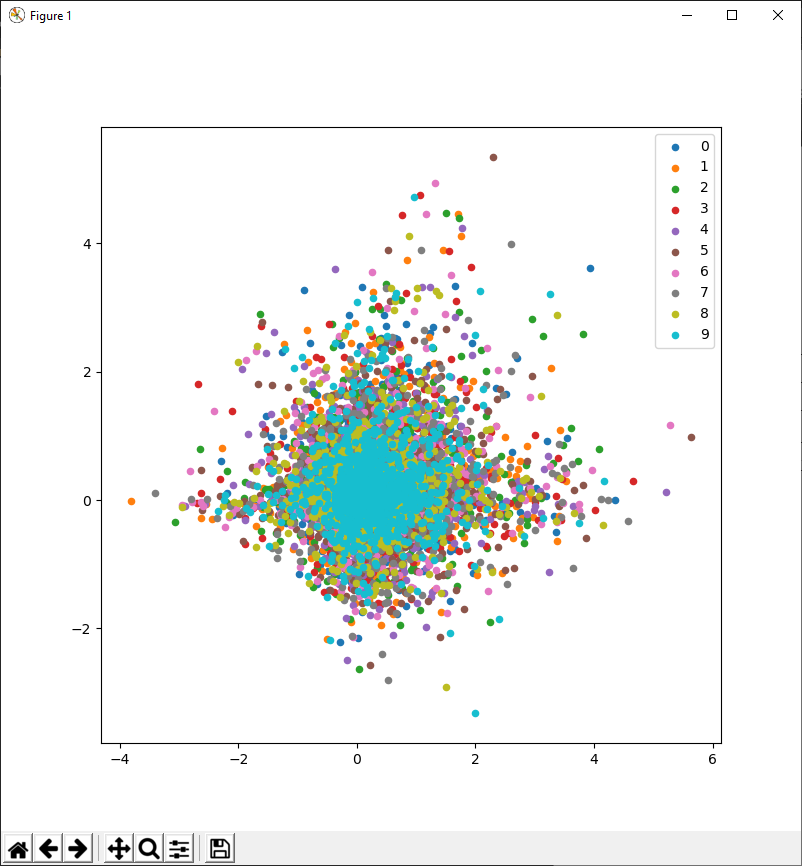
1. Applies a 5x5 kernel
2. Performs 2x2 max pooling

For training, the Triplet Network is given 3 images: , , . is the reference/anchor image. is a randomly selected positive image (an image from the same class as ). is a randomly selected negative image (an image from a class different from ). The Triplet Network runs the CNN once for each image and stores the flattened outputs as . The loss for the Triplet Network is calculated as:

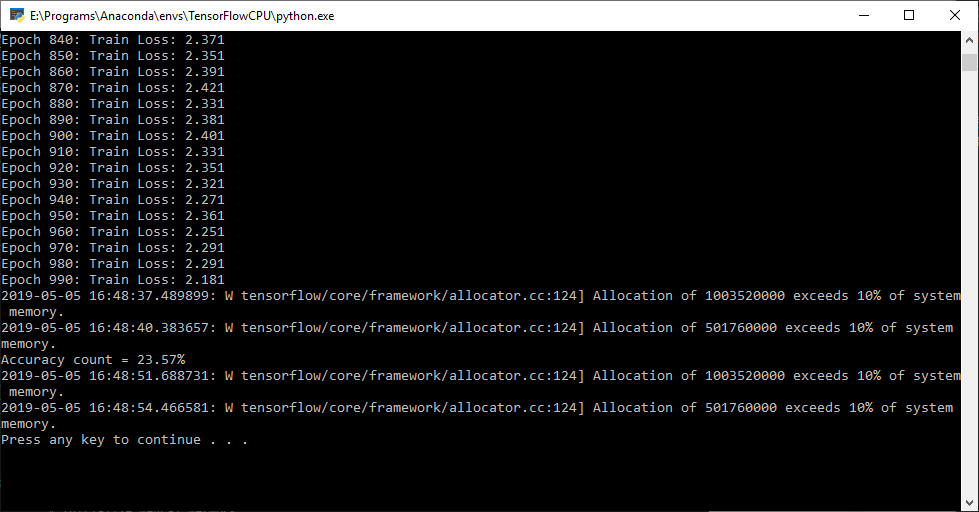
The trained CNN was then connected to a Dense Neural Network classifier. The classifier used the embeddings generated from the CNN as inputs and was trained with a cross entropy loss function.

## Results

The model was trained for 5 epochs, with each epoch running through 160000 permutations of uniformly selected triplet inputs. The following image displays the 2-Dimensional embeddings after the model was trained:



The classifier was trained for 1000 batches of 100 uniformly selected images. The resulting accuracy is shown below:



The embeddings clustered around zero and the low accuracy indicate there is an error either in the CNN or the Triplet Loss Function.