Applying Facial Recognition Techniques to Non-Facial Data

Josh Mandzak
Electrical Engineering and Computer Science Department
University of Tennessee, Knoxville
Knoxville, USA
jmandzak@vols.utk.edu

Doug Lowe
Electrical Engineering and Computer Science Department
University of Tennessee, Knoxville
Knoxville, USA
dlowe13@vols.utk.edu

***Abstract*—Facial recognition has, for the most part, been worked upon enough that it creates practically insurmountably good results. This paper aims to see if the facial recognition strategy that has done so well in the facial recognition world can be applied to things outside of facial recognition. Specifically, semi hard triplet loss will be applied to images of cities in order to see if they have enough similar attributes to faces to be correctly classified. Two evaluations will take place: evaluating the accuracy of a classification, as well as evaluating the accuracy of the ability of the network to classify two images of cities as either matching or different. These networks will be evaluated against traditional convolutional neural networks (CNN) that have already been proven to be suitable for this task.**

I. Introduction

While facial recognition techniques have multiple different facets, the use of semi hard triplet loss will be focused on for this paper. There are two key observations to be made by conducting a few different experiments. The first of these is to evaluate the performance of a model that uses semi hard triplet loss versus a traditional CNN using categorical cross entropy loss with the goal of classifying images as one of ten different cities. Traditional CNNs have already shown that they can accomplish this goal, so the evaluation will focus on how the triplet loss model does in comparison to the CNN. The second goal will be to evaluate zero-shot learning with both models. This will be done by holding out two cities from the original ten, then feeding each model pairs of cities with the output being a binary decision of either same city or different city. This will help answer the question previously discussed of how effective semi hard triplet loss can be on problems not related to facial recognition.

It should be noted that absolute accuracy of all models tested will be relatively low. This is due to multiple factors, including small amounts of training data, along with small models. The goal of this paper is not to improve absolute accuracy as much as possible, rather it is to evaluate the relative accuracy of each model compared to each other when doing the different tasks. Absolute accuracy could be improved with data augmentation, but was not deemed necessary for this paper, as relative accuracy can still be evaluated.

II. Related Work

A. Facial Recognition

Regarding facial recognition techniques, arguably the largest leap in accuracy was the FaceNet paper [3] released by Google. This paper introduced several different aspects of facial recognition still used today, but the main technique introduced by the study that will be used for this paper is semi hard triplet loss. This takes the normal triplet loss function and modifies it slightly to ensure that the loss is always positive by using ”semi-hard” examples for the negative aspect. This allows the model to train more efficiently, as it's not wasting epochs on evaluating triplets that are not at all similar.

B. Geolocation Through Images

Previous work has been conducted on determining the approximate location of a picture through the use of traditional CNNs. This study also utilized a hierarchical structure where they determined multiple things about the image before actually classifying its location. One of these techniques was determining if the picture was taken in the countryside or in the city, which helped them improve their classification accuracy. This approach was taken into consideration for this paper, which is why a data set was chosen that only contained cities rather than any location, as it provides another layer of consistency that any trained model then doesn't have to determine. There have also been more attempts at geolocation that utilize crowd-sourcing specifically in [4], which was used as inspiration for this project.

III. Technical Approach

When looking at this work, the ultimate goal is to be able to see an image of a city and recognize what city it is. There are two primary ways to test this. The first is through a simple classification task. In this case, the ultimate output of any model will be some kind of classification where the results are any one of the cities seen during training. Two models will be trained for this. The first will be a traditional CNN, which follows the pattern of convolutional layer, max pooling, and dropout layer. All dropout layers have a dropout rate of 0.5 and all convolutional layers use padding=same. Once these end, there will be a single dense layer of size 128, followed by a dense layer with the size of the total number of cities to be classified using a softmax activation function. This model will use categorical cross entropy (CCE) loss.

TABLE I: CCE Loss Model

Layer Type	Kernel Size	Num_Filters	Stride	Activation
Conv2D	(4,4)	32	N/A	relu
MaxPool	(4,4)	N/A	(4,4)	N/A
SpatialDropout2D	N/A	N/A	N/A	N/A
Conv2D	(4,4)	64	N/A	relu
MaxPool	(4,4)	N/A	(4,4)	N/A
SpatialDropout2D	N/A	N/A	N/A	N/A
Flatten	N/A	N/A	N/A	N/A
Dense	128	N/A	N/A	relu
Dense	num_cities	N/A	N/A	relu

The second model used to accomplish this task will be the one that uses triplet loss, thus testing the facial recognition technique of semi hard triplet loss. The structure of this model will be a replica of the previously described model, however, the last dense layer will be removed, and the last layer will use a lambda layer to perform L2 normalization on the dense layer of size 128 to create an embedding layer. This model will create an embedding for each image. These embeddings created by the training images will then be used to train an SVC with linear splits, thus giving us the necessary classification in order to compare it to the traditional CNN model.

TABLE II: Triplet Loss Model

Layer Type	Kernel Size	Num_Filters	Stride	Activation
Conv2D	(4,4)	32	N/A	relu
MaxPool	(4,4)	N/A	(4,4)	N/A
SpatialDropout2D	N/A	N/A	N/A	N/A
Conv2D	(4,4)	64	N/A	relu
MaxPool	(4,4)	N/A	(4,4)	N/A
SpatialDropout2D	N/A	N/A	N/A	N/A
Flatten	N/A	N/A	N/A	N/A
L2 Normal	128	N/A	N/A	relu

The second task that can be used to test facial recognition techniques on cities will take a somewhat different approach. The traditional CNN will be the exact same; however, when taking output to be tested we will take the output from the dense layer of size 128 as the output rather than the softmax layer output. Similarly, we discard the SVC that was attached to the end of the triplet loss model, and instead simply use the embeddings as the output. In order to test these models, pairs of cities will be fed into each model, both similar cities and different cities. The L2 distance of the outputs of each model will be calculated, and then compared to a constant value we will call d. If the value is less than d, the cities are considered to be the same. If the value is larger than d, then the cities are considered to be different. Pairs can then be fed into each network to classify true positives (same cities), true negatives (different cities), false positives and false negatives.

In addition, this method will be tested specifically on zero shot learning [7], or learning with little to no training data for the city given. In order to achieve this, two cities will be held out while training the model. Once training is completed, pairs will be fed into the model from both of the cities held out. This will demonstrate whether or not the model can identify two images of the same city without having seen that city prior to test time. This is something that facial recognition techniques have already proven to excel at with faces, so this will judge whether or not the same benefits can be seen on cities.

Ultimately, the accuracy of both models will be evaluated and compared. At the conclusion of testing, we should be able to identify two things: which technique is better at direct classification, and which model is better at zero shot learning classification.

IV. Dataset and Implementation

A. Dataset

In order to evaluate both of the tasks described previously, a dataset with multiple cities each with several images is necessary. Multiple datasets were evaluated, but ultimately a dataset containing ten cities [5], each with around 850 images for training, and 100 images for validation/testing. The ten cities contained within the dataset include Amsterdam, Barcelona, Budapest, Instanbul, London, Rome, Stockholm, and Vienna. This dataset was obtained on Kaggle and is publicly available to anyone with a Kaggle account. For both training and testing, all pictures will be kept in color, but reduced in size to 112x112 in order to keep training time down.

B. Implementation

In addition to evaluating two different tasks, multiple models were built, each with unique hyperparameters and training times. When results are displayed, they will always indicate which model produced those results and their unique characteristics. Multiple optimizers were tested, with SGD and Adagrad [6] being the most common of these. Multiple learning rates were tested, but ultimately 0.001 was used in all of the final models. Training was conducted with early stopping in effect, which had a patience of 5 epochs. All models utilized the TensorFlow default batch size of 32. All training was done using a Dell XPS 15 laptop with 32 GB of RAM and an i9 Intel Core processor at 2.5 GHz. Most models were trained somewhere between 40 and 100 epochs, all fully trained in under 30 minutes each.

V. Experiments and Results Analysis

A. Direct Classification Task

In order to evaluate this task, there are a few key metrics to examine. The first of these is the loss of each model as they train. Again, both the baseline models and the models using triplet loss were trained with early stopping with a patience of five. This means different models have different number of epochs trained, but ultimately all reach some kind of convergence or stop before they begin overfitting. For this task, four models were trained, two using CCE loss, and two using triplet loss. One of each uses SGD as the optimizer, while the other uses Adagrad. Since the triplet loss model needs a support vector machine to classify cities, there are no accuracy epoch plots for the triplet loss models. Below are the loss epoch graphs for all four models.

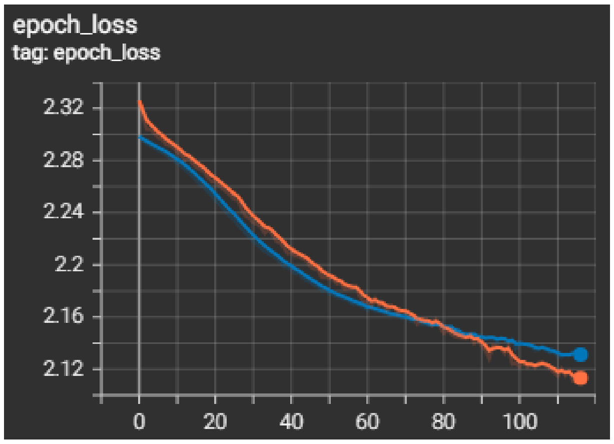

Fig. 1: Model with CCE Loss - SGD Optimizer
Orange: Train - Blue: Validation

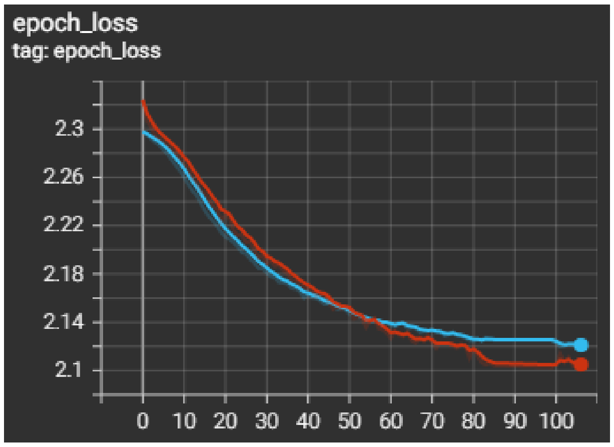

Fig. 2: Model with CCE Loss - Adagrad Optimizer
Red: Train - Blue: Validation

As evident from the loss plots, it can be seen that the models trained with CCE loss train significantly better than the triplet loss models, lowering their loss far more than the triplet loss models. For completeness sake, the accuracy plots of the two CCE trained models are also shown in figures 5 and 6.

Unsurprisingly relative to the loss plots, the accuracy of the two models also see the same pattern with the CCE trained models outperforming the triplet loss models. The results of all accuracy measurements are described in the table below.

Clearly, the models trained with categorical cross entropy loss outperform the triplet loss models for this task. This was not too surprising, as traditional CNNs have already shown to be good at tasks of this type.

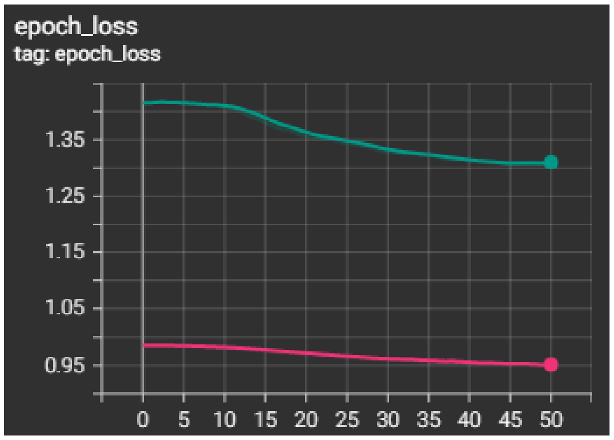

Fig. 3: Model with Triplet Loss - SGD Optimizer
Pink: Train - Green: Validation

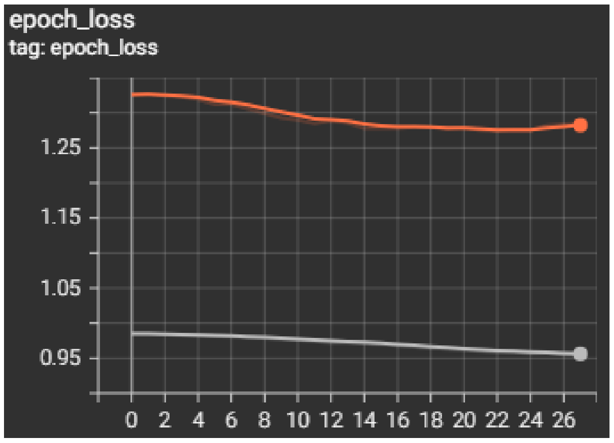

Fig. 4: Model with Triplet Loss - Adagrad Optimizer
Gray: Train - Orange: Validation

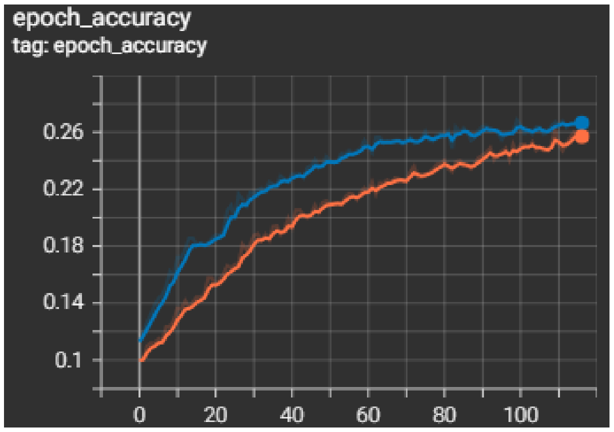

Fig. 5: Model with CCE Loss - SGD Optimizer
Pink: Train - Green: Validation

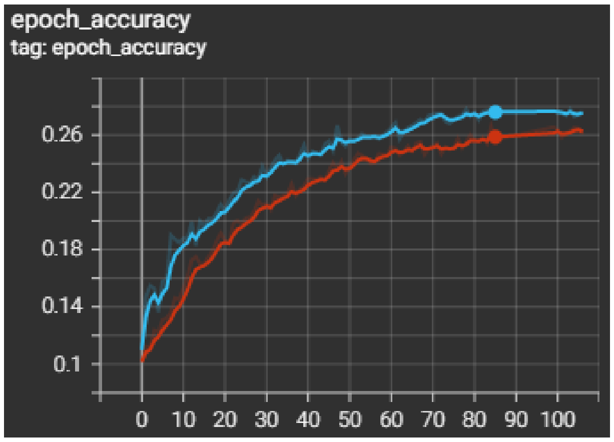

Fig. 6: Model with CCE Loss - Adagrad Optimizer
Gray: Train - Orange: Validation

TABLE III: Classification Accuracy

Model	Accuracy
CCE - SGD	26.79 %
CCE - Adagrad	27.79 %
Triplet - SGD	21.53 %
Triplet - Adagrad	20.18 %

B. *Embedding Matching Task*

This task is evaluated differently than the previous task. Four models will still be trained, however each model will only be trained on eight of the ten cities so zero shot learning can be evaluated. In addition to this, the support vector machine is no longer necessary. Besides these changes, training occurs exactly the same as the previous task and each model has the same structure. However, when generating predictions for the CCE model, output will be taken from the layer directly preceding the softmax layer. This is done so that both the CCE models and the triplet loss models output a size 128 embedding that can be used for evaluation. These models loss curves bear significant resemblance to the previous models loss curves due to the fact that the only difference is less training data, but are shown below nonetheless for completeness sake.

Since the models trained with CCE technically still have their softmax layer during training in order to ensure CCE still works correctly, they also still have accuracy plots. These accuracy epoch plots are shown in figures 11 and 12. One interesting thing worth noting is that the accuracy of the validation actually outperforms that of the training data. This seemed odd, but remained true throughout several experiences. Had the model been allowed to overfit this surely would have flipped, but training was stopped before that could occur.

To evaluate zero shot learning, pairs of images are fed into the model. Once two embeddings have been created, the L2 distance between them is calculated. This distance is then compared to a previously set constant value d determined at train time. The value decided on for this constant d for our experiments is 0.5. If the distance computed is less than

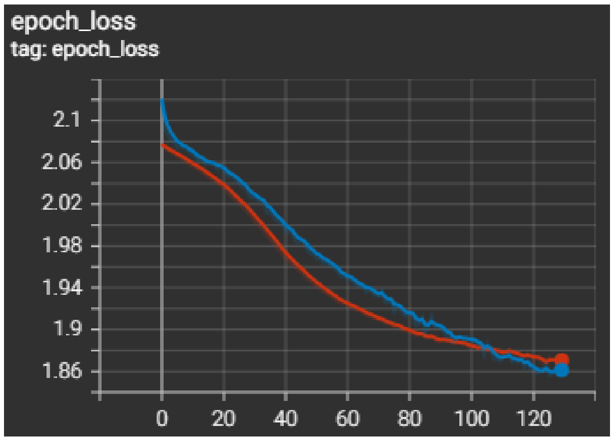

Fig. 7: Model with CCE Loss - SGD Optimizer
Blue: Train - Red: Validation

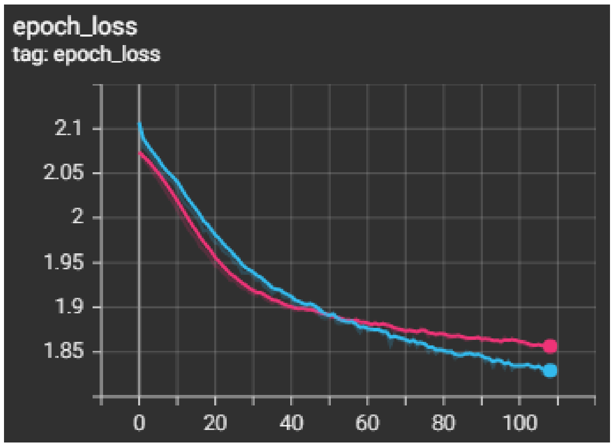

Fig. 8: Model with CCE Loss - Adagrad Optimizer
Blue: Train - Pink: Validation

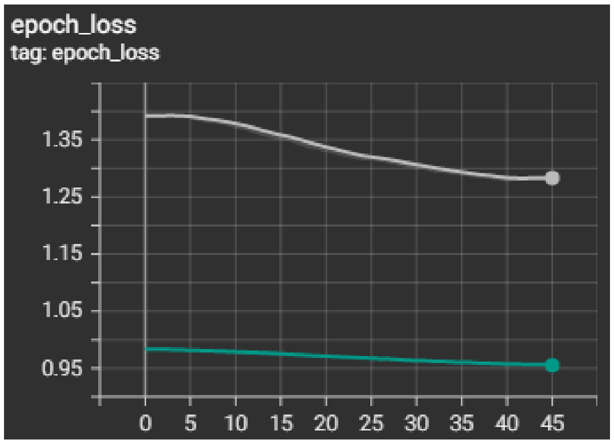

Fig. 9: Model with Triplet Loss - SGD Optimizer
Green: Train - Gray: Validation

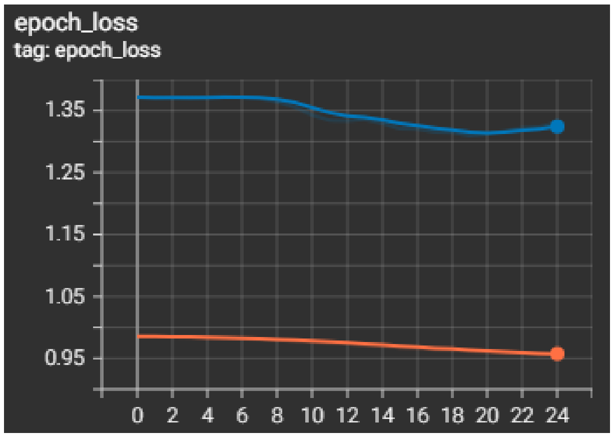

Fig. 10: Model with Triplet Loss - Adagrad Optimizer
Orange: Train - Blue: Validation

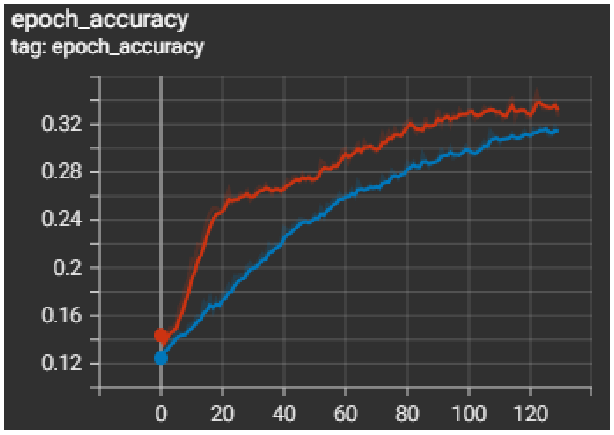

Fig. 11: Model with CCE Loss - SGD Optimizer
Blue: Train - Red: Validation

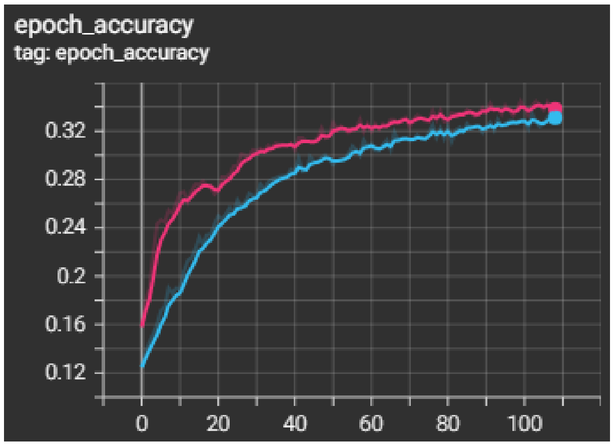

Fig. 12: Model with CCE Loss - Adagrad Optimizer
Blue: Train - Pink: Validation

this value d, the two images are deemed to be the same city, otherwise they are classified as different. Testing was conducted on the two cities left out of training, where every possible pair was combined and run through all four models. The results can be seen in Table IV.

TABLE IV: Pair Accuracy

Model	Accuracy
CCE - SGD	50.20 %
CCE - Adagrad	50.37 %
Triplet - SGD	50.14 %
Triplet - Adagrad	50.12 %

As expected, the models trained with CCE didn't perform very well, barely managing above a 50% guess rate. This was not surprising as the model wasn't specifically trained to come up with linearly separable embeddings. However, the triplet loss simply couldn't train enough to come up with a way to linearly separate the embeddings. This ultimately marks a failure in the paper's goal of utilizing facial recognition techniques to outperform traditional CNNs in zero shot learning learning. This failure could be due to a multitude of different reasons, some of which will be discussed in the conclusion section of this paper.

VI. Conclusion

A. Accomplishments

Many things were learned and accomplished in this project. At the end of the day, multiple different models were trained and many hyperparameters were studied. Lessons were learned in selecting a good L2 distance to set to determine similarity and differences when evaluating a pair of cities. A lot was also learned in constructing CNN architectures, mainly from testing, failing, and succeeding with several different architectures. Multiple well used architectures were learned about (despite ultimately not being used do to overfitting) including MobileNet and VGG-16. Task 1 can also be considered a success, because despite triplet loss not matching up to the traditional CNN, it did perform better than a random rate of 10%. This allows hope that triplet loss can be effectively used on things other than faces after all.

B. Possible Reasons for Failure

All of the successes aside, there were some failures that lessons needed to be learned from, mainly focusing on the lack of a good triplet loss model being trained. There are several possible reasons the triplet loss models could've failed this experiment. The first is data size. When comparing the amount of data in this experiment to the amount of images used in the FaceNet paper, the difference is significant. A single batch size for training in FaceNet was actually larger than an entire city's worth of images from the dataset used in this paper. It was hoped that the dataset size could still be used, but this could be a possible reason for failure.

Another possible reason for poor training could be due to the size of the model. The FaceNet paper trained multiple models, with their smallest model (which was still larger than the model evaluated in this paper) only reaching 51.9% [3]. However, ever single time a larger model was trained on this dataset, it immediately began overfitting, no matter the amount of dropout layers. In order to prevent overfitting even for the extremely small model this project trained, two dropout layers with 0.5 dropout were needed. This inability to construct a large model also probably is a sign of the dataset being too small.

C. Lessons Learned and Future Work

In order to train a model using triplet loss, it is very likely that a large amount of data is necessary, and likely needs to be paired with an above average sized model. For future work to be conducted on this experiment, both of these needs would need to be met to conduct a more thorough study on whether or not triplet loss can be effectively applied to images other than faces. Future work would need to focus on training a successful triplet loss model before training anything else in order to ensure a good comparison can be made.

This paper can not conclusively say whether or not triplet loss can be used effectively on things other than faces, as more work needs to be conducted with the changes mentioned above. However, due to the accuracy of the triplet loss models being above 10% on task 1, there is promise for future success.

VII. Bibliography

References

[1] Muller-Budack, Eric, Kader Pustu-Iren, and Ralph Ewerth. ”Geolocation estimation of photos using a hierarchical model and scene classification.” Proceedings of the European Conference on Computer Vision (ECCV). 2018.

[2] M. Abadi, A. Agarwal, and P. Barham, “TensorFlow Addons Losses: TripletSemiHardLoss,” TensorFlow. 19-Nov-2021.

[3] Florian Schroff, Dmitry Kalenichenko, James Philbin, ”FaceNet: A Unified Embedding for Face Recognition and Clustering,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 815-823

[4] A. Wallén, “GeoGuessr,” geogussr.com, May, 2013. [Online]. Available: https://www.geoguessr.com/. [Accessed April 8, 2022].

[5] Cursiv, “cities dataset 10,” kaggle.com, Aug, 2021. [Online]. Available: https://www.kaggle.com/datasets/cursiv/cities-dataset-10. [Accessed April 8, 2022].

[6] John Duchi, Elad Hazan, Yoram Singer, ”Adaptive Subgradient Methods for Online Learning and Stochastic Optimization”, Journal of Machine Learning Research.

[7] H. Larochelle, D. Erhan, and Y. Bengio, Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, Montreal, Québec , rep., 2008.

VIII. Appendix

A. Code Design

All code can be found it the GitHub repository located at https://github.com/jmandzak/cs525final. All code is custom made, as pretrained or preconstructed models overfitted the data too much. The three main code files are build.py, test_model.py, and test_all_models.py. The build file constructs different models through the use of two main functions, depending on whether a traditional CNN is being built or a triplet loss CNN is being built. The two test scripts evaluate saved models located in the saved_models_h5 directory, with test_model testing a single model you give it on command line, while the test all script simply runs every model in the saved models directory. Libraries used include pandas, numpy, cv2, Tensorflow, and Scikit-learn. These models were used to preprocess data, create the models, and evaluate the models.

B. Workload Distribution

1) Josh Mandzak: Created the network architecture of all models and tested all models. Created all plots and tables in report. Composed final report with the exception of bibliography. Created both testing scripts.

2) Doug Lowe: Found dataset and read in to Pandas dataframe. Found optimal d value for L2 distance. Composed bibliography and in-text citations.