2021 IFN680 - Assignment Two
Siamese Network

Group Members:

Brendan Wallace-Nash N9993304

Bingqing Bella Qian N10661450

Min-Pu Tsai N9300449

Introduction

One of the key applications of machine learning revolves around similarity measurement. For instance, coefficient is used to measure correlations of two elements; cosine distance and Euclidean distance are used to measure geometric differences. However, these methods are rather powerless when it comes to comparing vectors that have different dimensionality and types. This is where the Siamese Neural Network comes into play (Chicco , 2021).

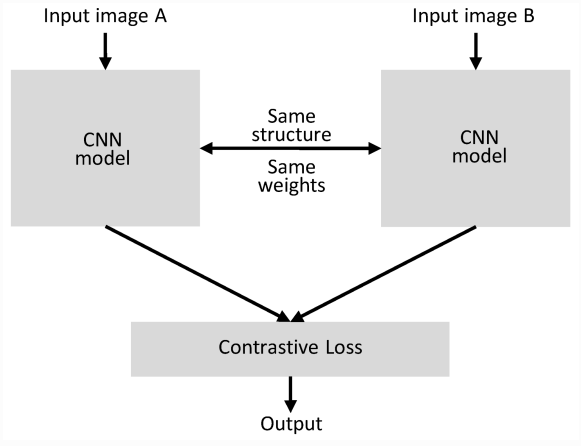


Figure 1 - Example of a Siamese Network

A Siamese Neural Network is a type of neural network that consists of two identical subnetworks. Singh (2020) explained that these two subnetworks share the same configuration with the same parameters and weights and are both capable of learning the hidden representation of an input vector. Two subnetworks work on two different input vectors in parallel and compute comparable output vectors (Chicco , 2021).

Siamese Networks are getting increasingly popular in Deep Learning applications, particularly in verification systems such as face recognition, signature verification, etc. In this report, a Siamese Neural Network was built for the purpose of predicting whether two images correspond to glyphs of the same alphabet using the Omniglot dataset.

Methodology

Siamese Network Architecture

As demonstrated in the graph below, an image x(i) is fed into a sequence of convolutional connected layers and ends up as a feature vector f(x(i)) which is an encoding of the input image x(i) ; similarly, a second image x(j) is fed into the same neural network with the same parameters and a different feature vector f(x(j)) which encodes image x(j) is generated. Since these two encoding vectors are good representations of these two images, we can compare the similarity of two images by comparing the

distance of two encoding vectors. This method of running two identical convolutional networks on two different inputs then comparing them is called a Siamese Network Architecture.

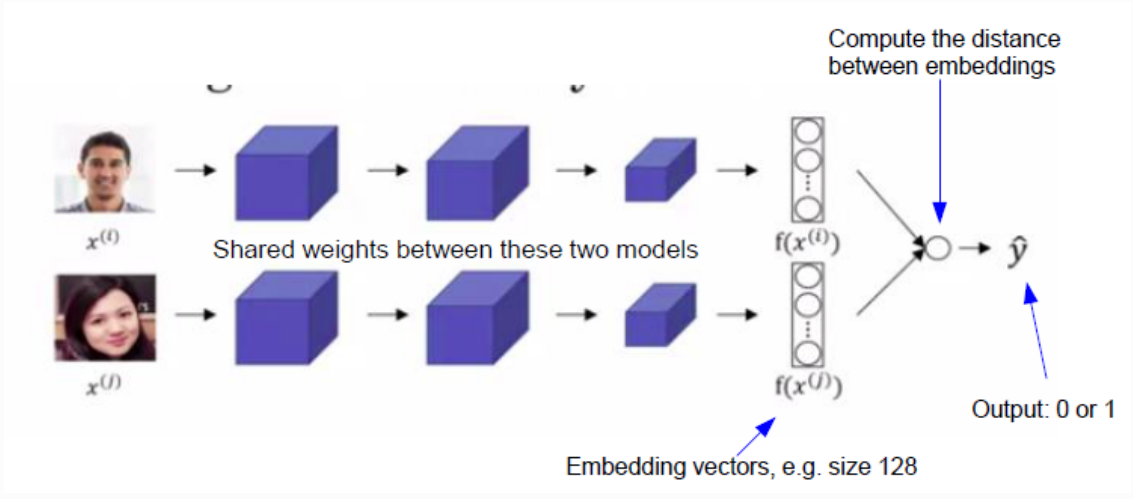


Figure 2 - Example of Siamese Network Architecture (Taigman et al, 2014)

Loss functions

Contrastive Loss

One way of comparing two images is to compare the encoding vectors of these images. If two images are similar, the encoding vectors are similar and the Euclidean distance between this pair of images is small. On the contrary, if two images are not similar, the encoding vectors are different and the Euclidean distance between this pair of images is large. As shown in the formula below, Y is a binary label (Y=1, positive pair; Y=0, negative pair) ; D is the distance between two input images; Margin is a constant that we can use to enforce a minimum distance between them in order to consider them similar or different (Coursera, 2020).

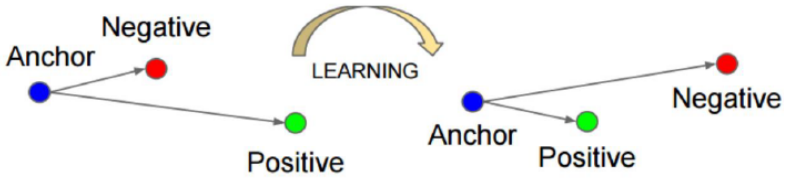
Contrastive Loss - Formula

$$Y * D^2 + (1 - Y) * max(margin - D, 0)^2$$

Figure 3 - Contrastive Loss Formula

Triplet Loss

In order to apply the Triplet Loss function, we first need to be comparing pairs of images. With the anchor (**a**) and positive (**p**) pair, since they share the same identity, we want the encodings to be similar, whereas with the anchor (**a**) and negative (**n**) pair, we want the encodings to be different because they have different identities. In other words, we want the distance between **a** and **p** (D(f(**a**), f(**p**)) to be small and the distance between **a** and **n** (D(f(**a**), f(**n**)) to be rather large, this can be achieved by the Triplet Loss function.



$$\sum_{i}^{N} \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]_+$$

Figure 4 - Triplet Loss Function Demonstration and Formula (Schroff et al, 2015)

The Triplet Loss is defined as:

$L(A, P, N) = \max(\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \text{margin}, 0)$

We expected that:

$|a-p|^2 - |a-n|^2 + \text{margin} \leq 0$

Distance between Anchor and Positive is represent as: $d(a, p) = |a-p|^2$;

Distance between Anchor and Negative is represent as: $d(a, n) = |a-n|^2$;

In other words, we expect d(a, p) minus d(a, n) is less or equal to 0. However, if d(a, p) and d(a, n) both return 0, or are equivalents, that makes no sense. Therefore, the margin value is placed behind the distances.

Model Architecture

The model used for both Contrastive Loss and Triplet Loss used a Siamese Convolutional Neural Network that has two conv2D layers with the first having a filter size of 16 and the next having a size of 32. Both conv2D layers use relu as the activation and have a kernel size of 3x3.

```
Model: "model_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_2 (InputLayer)            [(None, 105, 105, 3) 0
__________________________________________________________________________________________________
input_3 (InputLayer)            [(None, 105, 105, 3) 0
__________________________________________________________________________________________________
model (Functional)              (None, 10)           120438      input_2[0][0]
                                                                 input_3[0][0]
__________________________________________________________________________________________________
lambda (Lambda)                 (None, 1)            0           model[0][0]
                                                                 model[1][0]
__________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 1)            4           lambda[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 1)            2           batch_normalization_2[0][0]
==================================================================================================
Total params: 120,444
Trainable params: 103,508
Non-trainable params: 16,936
__________________________________________________________________________________________________
```

Figure 5 - Summary of the Siamese Model

Training & Evaluation

Implementing Contrastive Loss Function

Here is a screenshot of training done on Siamese Network using the Contrastive Loss function.

```
Epoch 1/20
1482/1482 [==============================] - 149s 99ms/step - loss: 0.2290 - accuracy: 0.6480 - val_loss: 0.2163 - val_accuracy: 0.6810
Epoch 2/20
1482/1482 [==============================] - 138s 93ms/step - loss: 0.2140 - accuracy: 0.6891 - val_loss: 0.2166 - val_accuracy: 0.6810
Epoch 3/20
1482/1482 [==============================] - 137s 93ms/step - loss: 0.2125 - accuracy: 0.6891 - val_loss: 0.2132 - val_accuracy: 0.6810
Epoch 4/20
1482/1482 [==============================] - 137s 92ms/step - loss: 0.2099 - accuracy: 0.6892 - val_loss: 0.2143 - val_accuracy: 0.6812
Epoch 5/20
1482/1482 [==============================] - 136s 92ms/step - loss: 0.2071 - accuracy: 0.6923 - val_loss: 0.2092 - val_accuracy: 0.6903
Epoch 6/20
1482/1482 [==============================] - 135s 91ms/step - loss: 0.2014 - accuracy: 0.6994 - val_loss: 0.2290 - val_accuracy: 0.6230
Epoch 7/20
1482/1482 [==============================] - 136s 91ms/step - loss: 0.1870 - accuracy: 0.7299 - val_loss: 0.1873 - val_accuracy: 0.7357
Epoch 8/20
1482/1482 [==============================] - 136s 92ms/step - loss: 0.1729 - accuracy: 0.7585 - val_loss: 0.1695 - val_accuracy: 0.7596
Epoch 9/20
1482/1482 [==============================] - 136s 92ms/step - loss: 0.1598 - accuracy: 0.7799 - val_loss: 0.1610 - val_accuracy: 0.7665
Epoch 10/20
1482/1482 [==============================] - 136s 92ms/step - loss: 0.1489 - accuracy: 0.7963 - val_loss: 0.1401 - val_accuracy: 0.8156
Epoch 11/20
1482/1482 [==============================] - 138s 93ms/step - loss: 0.1380 - accuracy: 0.8095 - val_loss: 0.1270 - val_accuracy: 0.8273
Epoch 12/20
 282/1482 [====>.........................] - ETA: 1:52 - loss: 0.1286 - accuracy: 0.8251
```

Figure 6 - Siamese Network Training Screenshot (Using Contrastive Loss Function)

As shown in Figure 6 , after running 11 out of the 20 epochs, our model can already achieve a 82% accuracy on the validation set, indicating that 82% of the time, the model is able to correctly determine if two input images correspond to the same alphabet.

Implementing Triplet Loss Function

```
Triplet loss computed with numpy 0.33480813145637517
Triplet loss computed with tensorflow 0.33480808
Train on 5000 samples
Epoch 1/10
5000/5000 [==============================] - 444s 89ms/sample - loss: 1.7742
Epoch 2/10
5000/5000 [==============================] - 441s 88ms/sample - loss: 1.9748
Epoch 3/10
5000/5000 [==============================] - 447s 89ms/sample - loss: 1.9768
Epoch 4/10
5000/5000 [==============================] - 460s 92ms/sample - loss: 1.9768
Epoch 5/10
5000/5000 [==============================] - 450s 90ms/sample - loss: 1.9768
Epoch 6/10
5000/5000 [==============================] - 439s 88ms/sample - loss: 1.9768
Epoch 7/10
5000/5000 [==============================] - 440s 88ms/sample - loss: 1.9768
Epoch 8/10
5000/5000 [==============================] - 446s 89ms/sample - loss: 1.9768
Epoch 9/10
5000/5000 [==============================] - 452s 90ms/sample - loss: 1.9768
Epoch 10/10
5000/5000 [==============================] - 440s 88ms/sample - loss: 1.9768
```

Figure 7 - Siamese Network Training Screenshot (Using Triplet Loss Function)

As shown in Figure 7 , after running 10 epochs, the model has a loss of 1.9768. In the implementation, all three images, A, P and N, went through the same CNN. A and P are from the same language while N is expected to be not.
We adopt positive pairs from pre-processing as Anchors and Positives. The Negatives are randomly selected from the shuffled mix of Anchors and Positives. Yet, it brings out doubt that the Negative may come from the same language of Anchor.
In order to solve the problem, the Negative is expected to go through a test to examine if it was from the same language of Anchor, which may decrease the accuracy of the learning.
Since the positive pairs have closer distance and negative pairs are the opposite, in the end, if we visualise the result, we expect that the glyphs from the same language collocate to the same cluster.

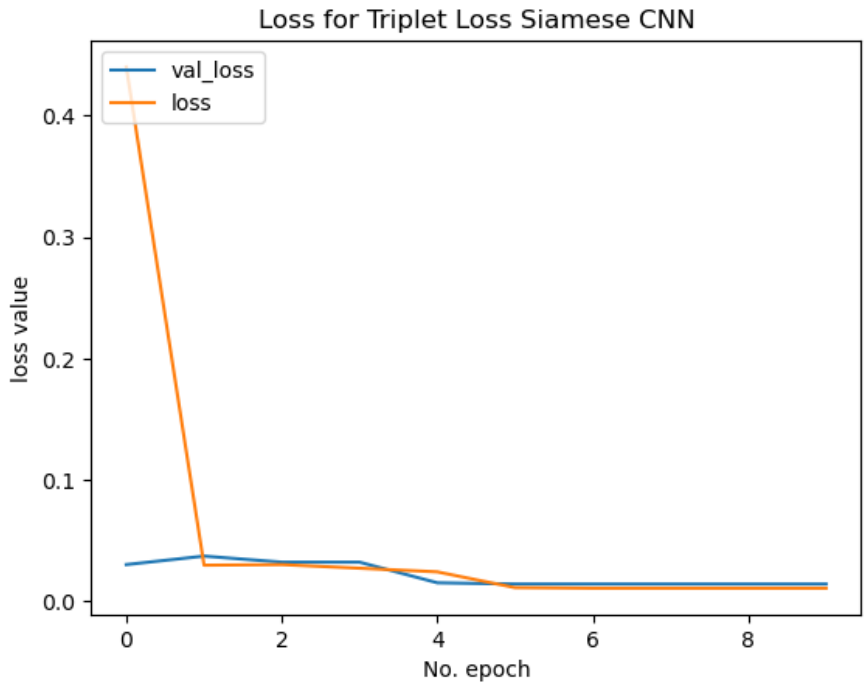


Figure 8 - Validation Loss & Training Loss for Triplet Loss throughout Training History

Figure 8 above shows the validation loss and training loss for our Siamese Network when it is trained with the Triplet Loss function. We can see after 5 epochs, both training loss and validation loss reached plateaus and two lines almost overlap each other. Validation loss is slightly higher which is a sign of minor overfitting. Overall, it appears that training was done fairly well.

Results

To assess the performance of models, Confusion Matrices were generated.

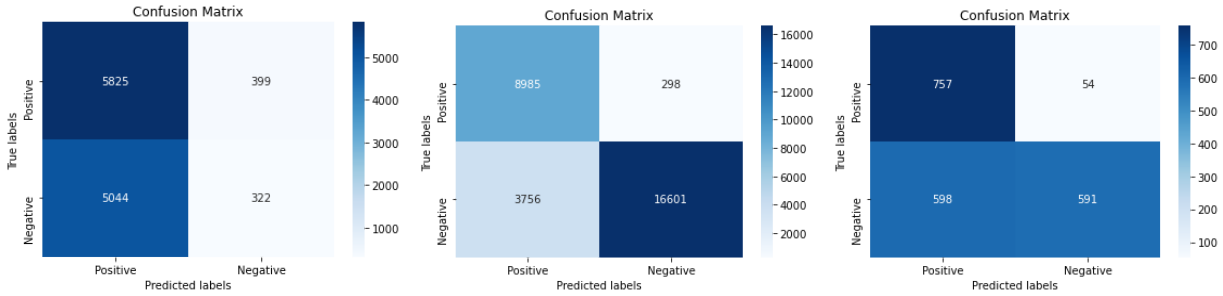


Figure 9 - Confusion Matrices (Contrastive Loss) (L to R: Test Data, Training Data, Test & Training Data)

Network performance when implementing Contrastive Loss function:
On Test Data: Accuracy = 53.04%
On Training Data: Accuracy = 69.45%
On Mix of Test & Training Data: Accuracy = 67.40%

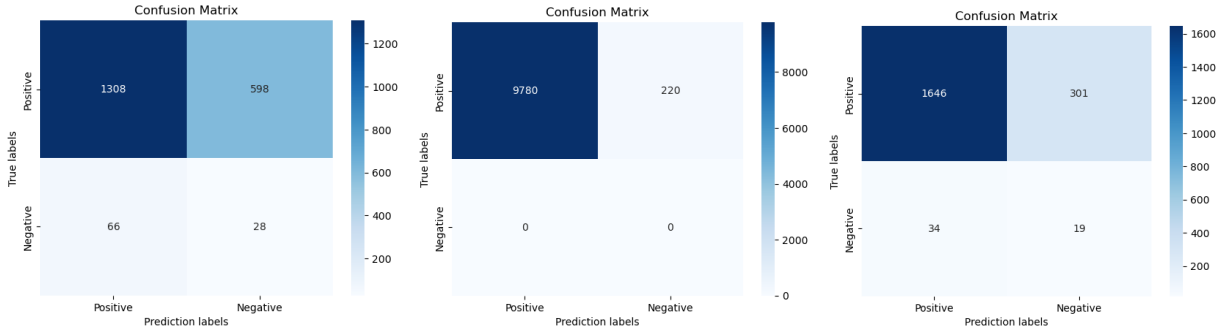


Figure 10 - Confusion Matrices (Triplet Loss) (L to R: Test Data, Training Data, Test & Training Data)

Network performance when implementing Triplet Loss function:
On Test Data: Accuracy = 66.80% *
On Training Data: Accuracy = 97.80%
On Mix of Test & Training Data: Accuracy = 83.25%

*Noting that our model shuffles pairs/batches of images each time it runs. It randomly selects 10,000 pairs out of 8 millions. As a result, accuracies vary each time the model is run. If the 10,000 selected pairs happen to contain pairs of images that are rather similar, the accuracy on test data will appear higher, and vice versa.

Discussion

According to our experiment and results, the model trained with Triplet loss outperforms the Contrastive Loss model. By analysing the confusion matrices from the section above, it is clear that the model trained with the Triplet loss function has significantly higher accuracies. This might be because

Contrastive loss only updates the weights to either minimize or maximize the similarity but triple loss conducts a more thorough process by not only pushing the negative image further but also pulling the positive image closer to the anchor image.

Conclusion

In this experiment, we implemented and trained a Siamese Network using *tensorflow* and *keras*. We trained our Siamese Network on the Omniglot dataset.
Using the Contrastive Loss function, our network accepts a pair of input images and then attempts to determine if these two images correspond to glyphs of the same alphabet. For instance, if two images that do not correspond to the same alphabet are fed into the network, then our Siamese Network would report low similarity between the two images, which implies that they indeed correspond to two different alphabets, and vice versa. Our model appears to be stable and fairly accurate. When tested on pairs from the set of glyphs from the training split, it achieved a 69.45% accuracy. It also achieved a 67.40% accuracy on training & test split and a 53.04% on the test split.

Using triplet loss function, our network accepts 3 input images and makes predictions. one pair is an anchor image and positive (two images correspond to the same alphabet), the other is anchor and negative (two images correspond to different alphabets). The model has a 97.80% accuracy when tested on pairs from the set of glyphs from the training split; a 83.25% accuracy on the test split and a 66.80% accuracy when test on training & test split.

Reference

Chicco, D. (2021). Siamese neural networks: An overview. *Artificial Neural Networks*, 73-94.

Coursera. (2020). Contrastive Loss [Video file]. https://www.coursera.org/lecture/custom-models-layers-loss-functions-with-tensorflow/contrastive-loss-iGjXg

Singh, P. (2020, September 19). *Introduction To Siamese Networks.* Medium. https://medium.com/analytics-vidhya/a-friendly-introduction-to-siamese-networks-283f31bf38cd

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 815-823).

Taigman, Y., Yang, M., Ranzato, M. & Wolf, L. Deepface: closing the gap to human-level performance in face verification. In *Proc. Conference on Computer Vision and Pattern Recognition* 1701–1708 (2014).