IFN680 Assignment 2 - Siamese Network

Tan En Hui Ariel, n10497285 | Patrick Choi, n10240501 | Ian Choi, n10421106

Contents

1. **Introduction**
2. **Methodology**
   - 2.1 Experiment Structure
   - 2.2 Process for Data
   - 2.3 Siamese Network Design
   - 2.4 Test Design
3. **Result**
   - 3.1 Contrastive Loss Function
   - 3.2 Triplet Loss Function
   - 3.3 Performance Comparison
4. **Conclusion**
5. **References**

---

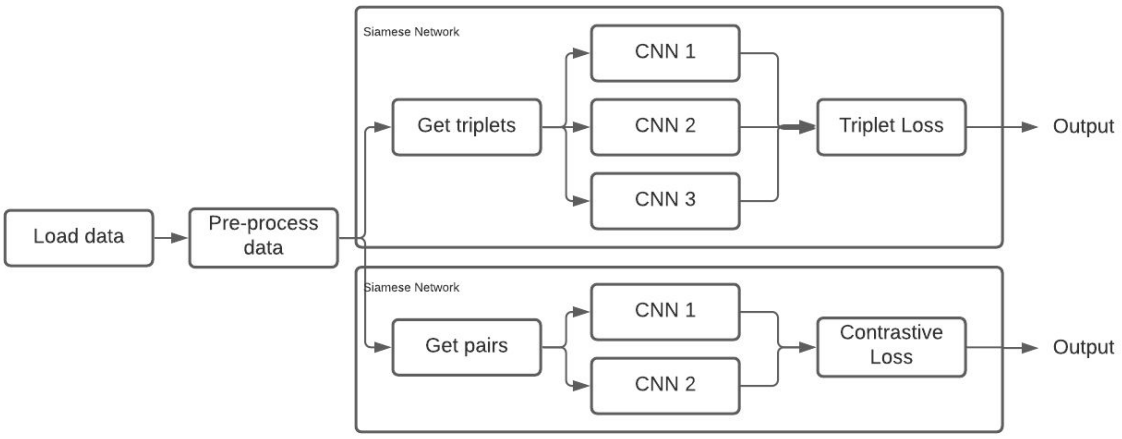
1. Introduction

Siamese networks are neural networks that contain two or more identical subnetwork components (Bromley et al., 1994). In this report, the Siamese network will be using two or more Convolutional Neural Network (CNN) as its subnetwork component. The network will be trained using the Omniglot dataset to predict whether the given inputs belong to the same equivalence class.

2. Methodology

2.1 Experiment Structure

This experiment has been conducted based on the Siamese network in which several CNNs are combined. Figure 1. shows an overview of the experiment structure used to build and train a Siamese network.



<Figure 1. Experiment structure.>

The Experiment has been conducted in the Google Colab environment. The running environment for experiment includes Python version 3.6.9, tensorflow version 2.3.0 and keras version 2.4.3.

2.2 Process for Data

Pre-processing the Omniglot Dataset

The omniglot dataset is loaded and split into training and test datasets. In the *omniglot_dataset* method, the images are converted to grayscale and cast to a float 32 datatype.

```
image = tf.image.rgb_to_grayscale(image)
image = tf.cast(image, tf.float32) / 255.
```

<Code 1. Image processing>

The data is then combined in the *preprocess_omniglot_dataset* method to form four different datasets, one used for training and three for testing.

```
data_train, target_train = x_train, y_train
data_test, target_test = x_test, y_test

data_combine = np.concatenate([x_train, x_test])
target_combine = np.concatenate([y_train, y_test])

return (data_train, target_train), (data_train, target_train), (data_combine,
target_combine), (data_test, target_test)
```

<Code 2. Splitting data into four datasets>

Data used for training were only taken from the ‘train’ split.

The three datasets for testing are:

1. Data from the ‘train’ split
2. Data from both the ‘train’ and ‘test’ split
3. Data from the ‘test’ split

Creating Pairs and Triplets

In testing of the siamese network using contrastive loss function, positive and negative pairs are required as input. The positive pairs are created using images from the same class and negative pairs were created by using images from different classes.

In testing of the network using triplet loss function, three inputs are required. An anchor sample, positive sample and negative sample. Two images of the same class were used for the anchor and positive sample. An image of a different class was used as the negative sample.

2.3 Siamese Network Design

CNN Layers

A Convolutional Neural Network (CNN) can take an input and assign importance to various features of the image and be able to distinguish one from another (Saha, 2018). It uses filters and layers to process the input image in order to get a good prediction.

Layer	**Filter**	**Kernel/pool size**
Conv2D	32	3 x 3
Batch Normalisation		
Conv2D	32	3 x 3
Batch Normalisation		
Max Pooling 2D		2 x 2
Conv2D	64	3 x 3
Batch Normalisation		
Conv2D	64	3 x 3
Batch Normalisation		
Max Pooling 2D		2 x 2
Fully connected layer 1		
Fully connected layer 2		

<Table 1. CNN architecture>

For this experiment, four 2D convolutional layers are used. After each layer, batch normalisation is applied to maintain the output between 1 and 0. Max Pooling layers were used after the second and fourth convolutional layer for downsampling the input. Lastly, the network is connected by two fully connected layers.

Euclidean distance is used to measure the distance between the two outputs from the CNN model with contrastive loss function. RMSprop is used as an optimizer for the model.

```
distance = keras.layers.Lambda(euclidean_distance,
                               output_shape=eucl_dist_output_shape)([output_cnn_1,
                               output_cnn_2])

rms = keras.optimizers.RMSprop()
```

<Code 3. Euclidean distance>

Loss Function

During the training of the Siamese network, two inputs are encoded and their outputs are compared. The network will be using two different loss functions to carry out the comparison.

Contrastive Loss Function
The contrastive loss function operates a pair of inputs received from the model and measures the similarity and distance between them. Positive pairs with small distance between them and negative pairs with distance smaller than the margin *m* will be learned by the network. The *m* value is set at 1 because it performs best in this network.

Triplet Loss Function
Triplet loss function uses three inputs instead of pairs: a baseline sample, positive sample and a negative sample. The function calculates the distance between the positive sample and the baseline sample and the sum of the distance between the negative and baseline samples and margin m. The distance is minimized for the positive samples and maximized for the negative samples. After training, the positive samples will be closer to the baseline and the negative samples further away. The *m* value is set at 0.4 for this function.

2.4 Test Design

In order to test the performance of the Siamese network model generated, three experiments for two different losses, contrastive loss and triplet loss, are conducted. The three experiments used different pairs as input of the model:

- First experiment used the pairs from the set of glyphs from the ‘train’ split.
- Second experiment used the pairs from the set of glyphs from both ‘train’ and ‘test’ splits.
- Third experiment used the pairs from the set of glyphs from the ‘test’ split.

On the other hand, model training is performed using only ‘train’ split.

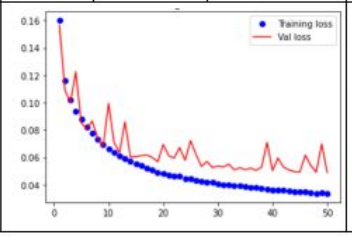
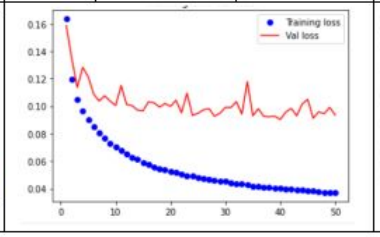
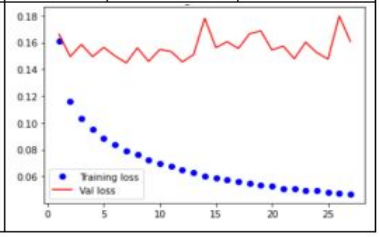
Each experiment followed the performance evaluation method to observe the calculated losses and accuracies of training and validation. Also, a callback function is applied to avoid overfitting of the model, which is set to stop the experiment if the validation loss does not change for twenty epochs.

3. Results

3.1 Contrastive Loss Function.

Execution Time & Loss

For three experiments of Siamese Network using contrastive loss function, the training loss, validation loss and the time taken are computed for fifty epochs and displayed in both table and plot chart. The figures below display the overall performance evaluation record.

	Contrastive Test 1			Contrastive Test 2			Contrastive Test 3		
Epoch	Times (s)	Val_loss	Train_loss	Times (s)	Val_loss	Train_loss	Times (s)	Val_loss	Train_loss
1	6	0.1566	0.1601	7	0.1586	0.1635	5	0.1662	0.1613
5	6	0.0807	0.0876	7	0.1213	0.0903	5	0.1563	0.0884
10	6	0.0995	0.0661	7	0.1005	0.0702	5	0.1548	0.0698
15	6	0.0604	0.0550	7	0.0967	0.0588	5	0.1562	0.0587
20	6	0.0693	0.0484	7	0.0999	0.0526	5	0.1544	0.0538
25	6	0.0722	0.0442	7	0.0948	0.0479	5	0.1476	0.0479
30	6	0.0536	0.0405	7	0.0991	0.0454	5		
35	6	0.0508	0.0383	7	0.0931	0.0417	5		
40	6	0.0503	0.0363	7	0.0903	0.0404	5		
45	6	0.0492	0.0348	7	0.1050	0.0385	5		
50	6	0.0490	0.0336	7	0.0936	0.0372	5		
									

<Figure 2. TIme vs loss result using contrastive loss>

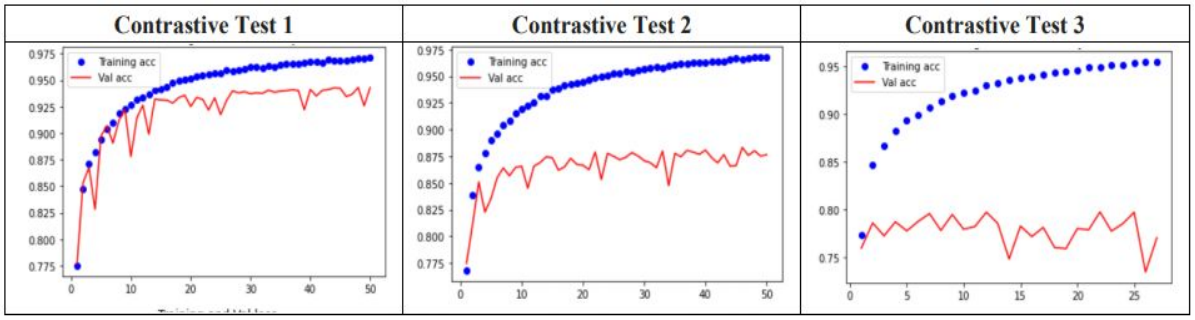
The first experiment indicated as ‘Contrastive Test 1’ took six seconds for each epoch. The average of validation loss is rounded to 0.072 with the minimum value of 0.049 and max value of 0.1566. On the other hand, the average of training loss is rounded to 0.059 with the minimum value of 0.0336 and maximum value of 0.1601. The trend of chart shows decreasing loss for both losses throughout the epochs.

For the second experiment indicated as ‘Contrastive Test 2’, it took seven seconds for each epoch. The average of validation loss is rounded to 0.1048 with the minimum value of 0.0903 and maximum value of 0.1586. Meanwhile, the average of training loss is rounded to 0.0624 with the minimum value of 0.0372 and maximum value of 0.1635. The trend of chart shows decreasing loss for both losses throughout the epochs.

Lastly, for the third experiment indicated as ‘Contrastive Test 3’, it took five seconds for each epoch. The experiment ended after epoch 27 by the callback function. The average of validation loss is rounded to 0.156 with the minimum value of 0.1476 and maximum value of 0.1662. Meanwhile, the average of training loss is rounded to 0.08 with the minimum value of 0.0479 and maximum value of 0.1613. The trend of the chart shows decrease for training loss but constant for validation loss throughout the epochs.

Accuracy

In addition to the execution time and loss, the training accuracy and validation accuracy for each epoch using contrastive loss are displayed in the following plot chart.



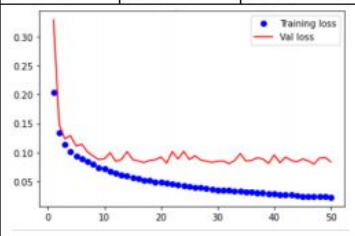
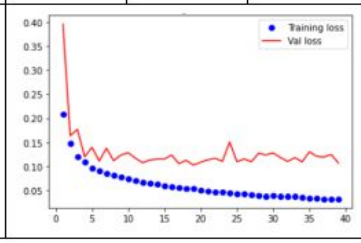
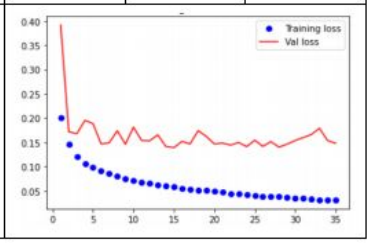
<Figure 3. Accuracy chart using contrastive loss>

The trend for training accuracy showed high performance, with all the experiments recorded higher than 95% accuracy before the expiration. After running 50 epochs, the training accuracies of the final result were 97.12%, 96.72% and 95.41% and the validation accuracies were 94.29%, 87.65% and 77.03% for the three respective experiments. The experiment with test split showed comparatively lower validation accuracy.

3.2 Triplet Loss Function

Execution Time & Loss

For the three experiments of Siamese Network using triplet loss function, the training loss, validation loss and the time taken are computed for fifty epochs and displayed in both table and plot chart. The figures below display the overall performance evaluation record.

	Triplet Test 1			Triplet Test 2			Triplet Test 3		
Epoch	Times (s)	Val loss	Train loss	Times(s)	Val loss	Train loss	Times(s)	Val loss	Train loss
1	4.3	0.3290	0.2037	5.4	0.396	0.2086	4.3	0.392	0.2015
5	4.3	0.1108	0.0935	5.3	0.1398	0.0967	4.3	0.1885	0.0979
10	4.3	0.0887	0.0709	5.3	0.1281	0.0731	4.3	0.1813	0.0717
15	4.3	0.0875	0.056	5.3	0.115	0.0598	4.3	0.139	0.058
20	4.3	0.092	0.0476	5.3	0.1084	0.0506	4.3	0.1466	0.0488
25	4.3	0.0876	0.0409	5.3	0.1095	0.0426	4.3	0.1544	0.0402
30	4.3	0.0845	0.0351	5.3	0.1275	0.0385	4.3	0.1538	0.0353
35	4.3	0.0848	0.0313	5.3	0.1303	0.0342	4.3	0.1481	0.0313
40	4.3	0.0954	0.0283	5.3	0.1067	0.0316			
45	4.3	0.0885	0.0242						
50	4.3	0.0834	0.0221						
									

<Figure 4. TIme vs loss result using triplet loss>

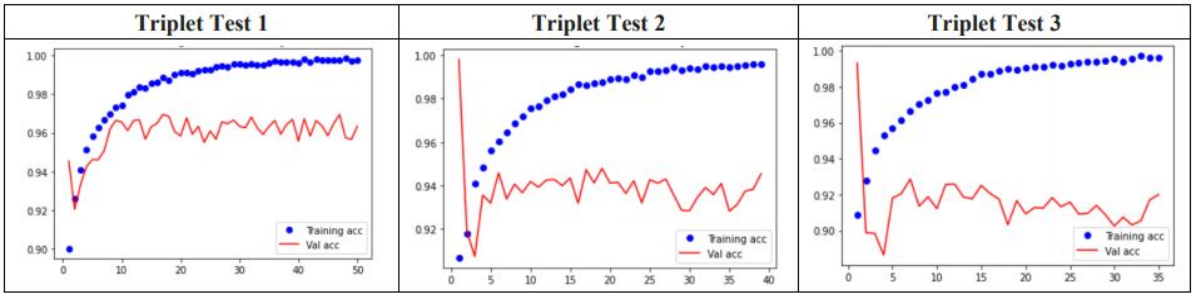
The first experiment indicated as ‘Triplet Test 1’ took 4.3 seconds for each epoch. The average of validation loss is rounded to 0.112 with the minimum value of 0.0834 and max value of 0.329. On the other hand, the average of training loss is rounded to 0.0594 with the minimum value of 0.0221 and maximum value of 0.2037. The trend of chart shows decreasing loss for both losses throughout the epochs.

For the second experiment indicated as ‘Triplet Test 2’, it took 5.3 seconds for each epoch. The experiment ended after epoch 39 by the callback function. The average of validation loss is rounded to 0.1513 with the minimum value of 0.1067 and maximum value of 0.396. Meanwhile, the average of training loss is rounded to 0.0706 with the minimum value of 0.0316 and maximum value of 0.2086. The trend of chart shows decreasing loss for both losses throughout the epochs.

Lastly, for the third experiment indicated as ‘Triplet Test 3’, it took 4.3 seconds for each epoch. The experiment ended after epoch 35 by the callback function. The average of validation loss is rounded to 0.188 with the minimum value of 0.139 and maximum value of 0.392. Meanwhile, the average of training loss is rounded to 0.0731 with the minimum value of 0.0313 and maximum value of 0.2015.

Accuracy

In addition to the execution time and loss, the training accuracy and validation accuracy for each epoch using triplet loss are displayed in the following plot chart.



<Figure 5. Accuracy chart using triplet loss>

The trend for training accuracy showed high performance, with all the experiments recorded higher than 99% accuracy before the expiration. After running 50 epochs, the training accuracies of the final result were 99.77%, 99.58% and 99.64% and the validation accuracies were 96.34%, 94.55% and 92% for the three respective experiments. The experiment with test split showed comparatively lower validation accuracy.

3.3 Performance Comparison

The training accuracy and validation accuracy of both contrastive loss and triplet loss are compared in the figures below.

Training Accuracy (50 epochs)

	Train Split	Both Split	Test Split	Average
Contrastive Loss Function	97.12%	96.72%	95.41%	96.42%
Triplet Loss Function	99.77%	99.58%	99.64%	99.66%

<Figure 6. Training accuracy table for both losses>

Validation Accuracy (50 epochs)

	Train Split	Both Split	Test Split	Average
Contrastive Loss Function	94.29%	87.65%	77.03%	86.33%
Triplet Loss Function	96.34%	94.55%	92.00%	94.30%

<Figure 7. Validation accuracy table for both losses>

4. Conclusion

It was confirmed that the triplet loss function (TLF) showed better performance than the contrastive loss function (CLF) in classification of Omniglot image data using the Siamese network.

TLF showed better accuracy not only in the average, but also in all cases. This is especially evident in Test 3, using only the test split, TLF showed relatively superior performance compared to CLF. In addition, the time and accuracy required for training and validation using TLF is shorter than CLF. Moreover, it was practically confirmed that the predictive model using triplet is very effective in image classification, especially for classifying various types of different images such as the Omniglot dataset from the results of this experiment .

In conclusion, TLF is a better loss function to train a Siamese network for the classification of the Omniglot dataset.

5. References

Bromley, J., Guyon, I., LeCum, Y., Sackinger, E., Shah, R. (1994). Signature Verification using a “Siamese” Time deal Neural Network”

Saha, Sumit. (2018). A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. Retrieved from https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53