Assignment 2 - Report

Overview:

Our network architecture, based on the Siamese Neural Network which described in the article, is designed for one-shot learning tasks. It processes pairs of input images simultaneously, utilizing shared weights to compute a similarity score indicating whether the images belong to the same person.

About the data:

We were provided with two text files for our task: one for training and the other for testing. this files contains the information about the training and test data but not the images.
So, additionally, there's a folder containing all the images data.
Each line in these text files contains either three or four parameters. When a line has three parameters, it means the images belong to the same person. Conversely, if a line has four parameters, it indicates that the images portray different individuals.
The information contained in the text files guided us in organizing and preparing the data for further processing. Afterwards, we exported the processed data into .pickle files for easier loading and accessibility.

The images are greyscale sized 105X105 pixels (105,105,1) which is transformed in PyTorch into (1,105,105).
Since we deal with Siamese network, each “sample” contains two images

Total number of train samples : 2200 .

So that:

- 1100 of them are images of the same person.
- 1100 of them are images of different people.

Total number of test samples : 1100

So that:

- 500 of them are images of the same person.
- 500 of them are images of different people.

We decided to allocate 20% of the training data into the validation set.

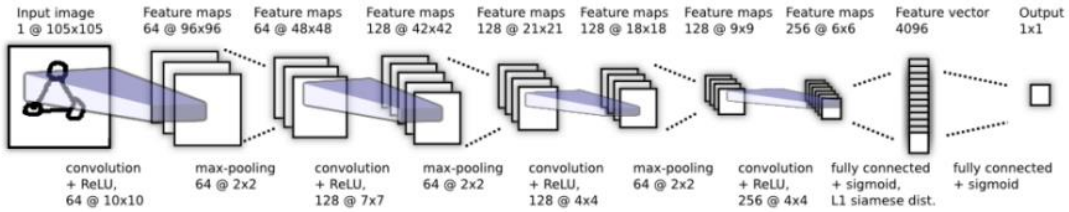
The Network:

The architecture consists of an input layer with each input image having dimensions of 105x105 pixels. Convolutional layers are employed, where the first layer applies 64 filters with a kernel size of 10x10, followed by ReLU activation. Subsequent convolutional layers use filters of sizes 7x7, 4x4, and 4x4, with the number of filters increasing from 64 to 128, and then 256. Each convolutional layer is accompanied by a ReLU activation function.

Pooling layers follow the convolutional layers, with max-pooling operations of size 2x2 to reduce spatial dimensions. Fully connected transition layers after the final convolution, with

the first layer having 256 units and ReLU activation, and the second layer with 4096 units using a sigmoid activation function.

The final output is a single value representing the similarity score, with a sigmoid activation function constraining the score between 0 and 1. This score represents the probability of both images to be the same person.



Experiment setup

Following the methodology in the paper, We performed a wide range of experiments with the following hyperparameters values:

- Epochs: [10,15]
- Batch_size: [16,32]
- Threshold: [0.5,0.6]
- Learning_rate: [0.00003, 0.00005 ,0.0001]
- Weight_decays: [0, 0.0002, 0,0007]
- Dropout: [0,0.2,0.3]
- Min_delta: [0.05,0.1]
- Optimizer: Adam optimizer.
- Loss function: Binary Cross Entropy

These values were chosen due to their known effectiveness in the field of deep learning. The goal was to systematically explore the range of hyperparameters to locate the most optimal model configurations.

We used early stopping monitoring on the validation loss with min_delta and patience 3.

Min_delta - The min_delta parameter specifies the minimum change in the monitored quantity (e.g., validation loss) that must occur to consider it as an improvement. If the change in the monitored quantity between two consecutive epochs is less than min_delta, it is considered insignificant, and the training process may be stopped.

Patience – number of epochs with no improvement after which the training will be stopped.

In all experiments we used batch normalization.

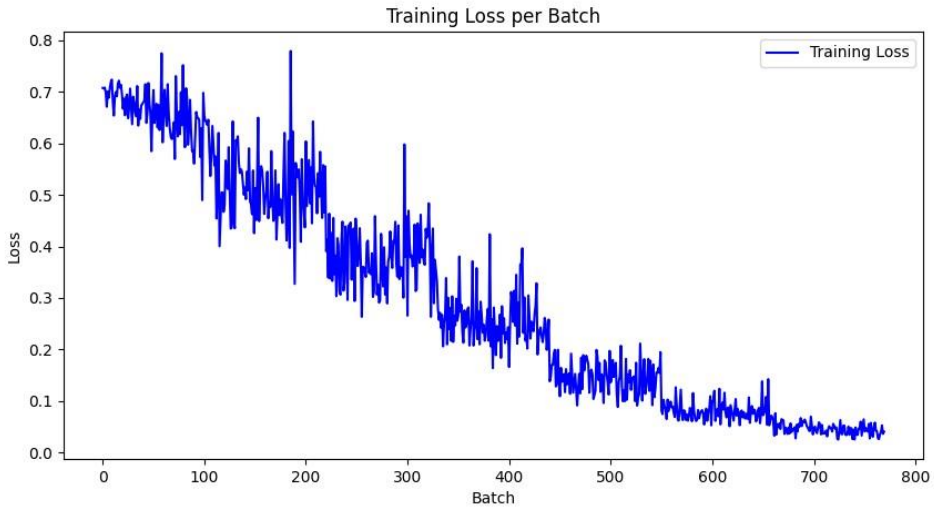
Evaluation

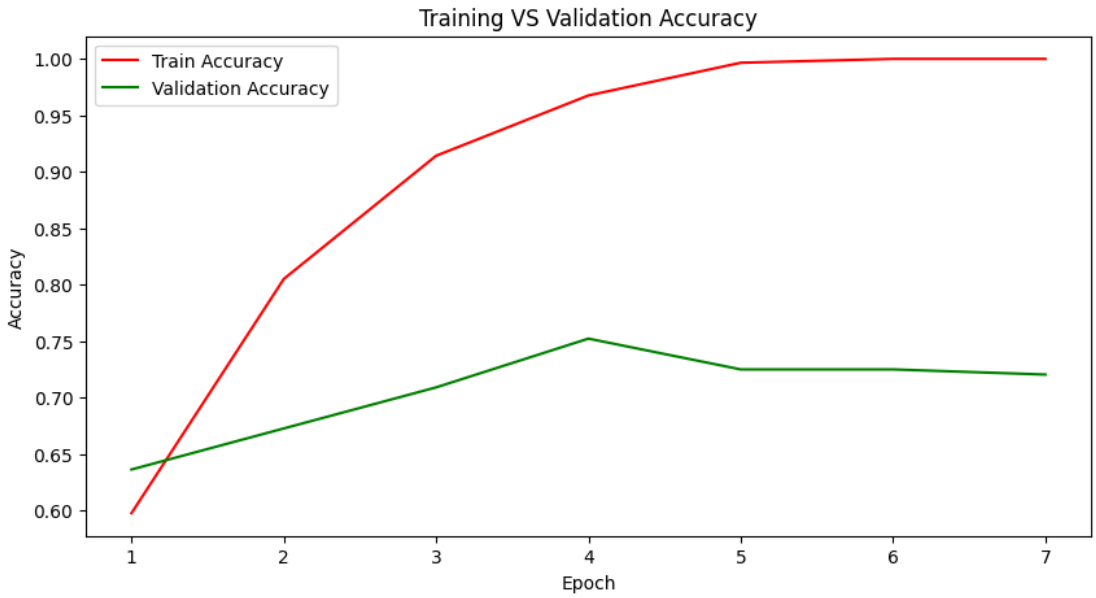
We examined the three models with the highest test accuracy:

Model 1

Hyperparameters:

- Epochs: 15
- Batch_size: 16
- Threshold: 0.5
- Learning_rate: 0.00005
- Weight_decay: 0.0002
- Dropout: 0.2
- Min_delta: 0.05
- Optimizer: Adam





Accuracy:

- Validation set: 0.727
- Test set: 0.726

Total convergence time: 50.52 seconds

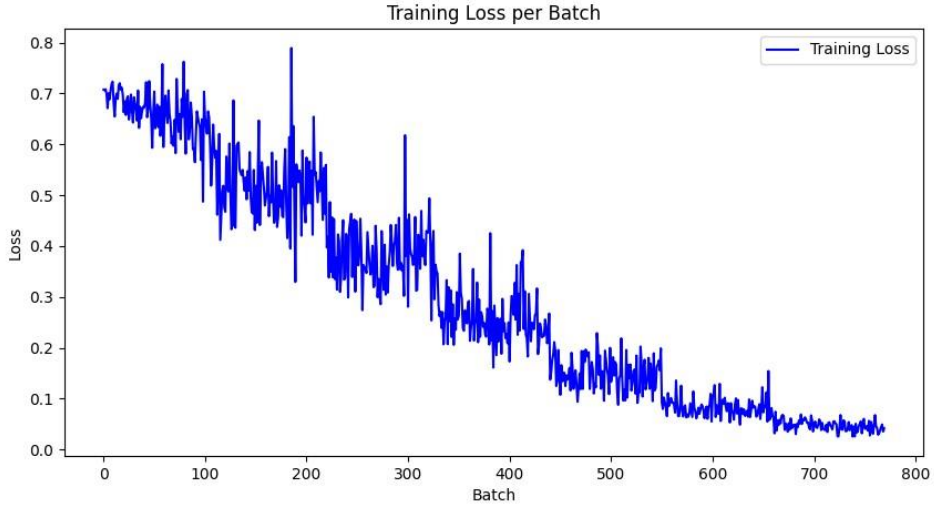
Total epochs: 7

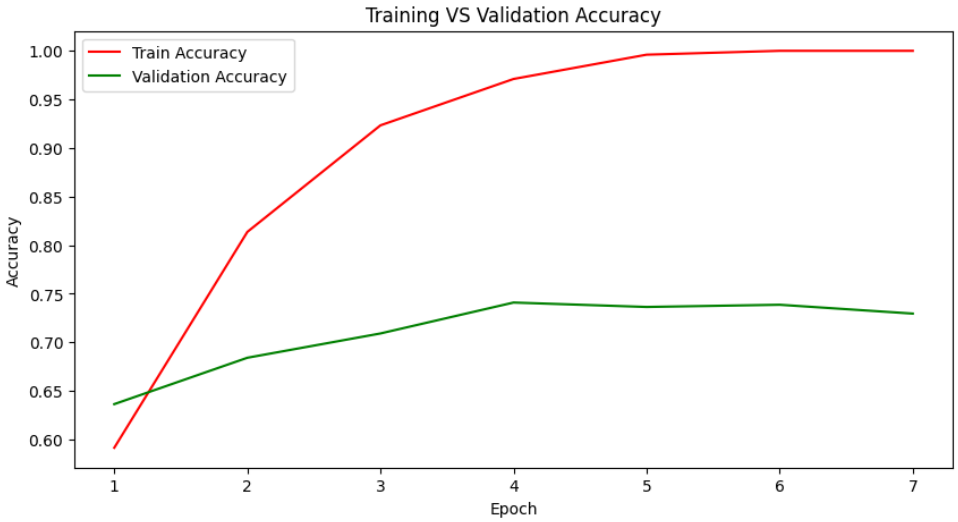
Model 2

Hyperparameters:

- Epochs: 15
- Batch_size: 16
- Threshold: 0.5
- Learning_rate: 0.00005
- Weight_decay: 0.0002
- Dropout: 0.2
- Min_delta: 0.1
- Optimizer: Adam


Training VS Validation Loss
Train Loss
Validation Loss
Loss
Epoch
0.1
0.2
0.3
0.4
0.5
0.6
1
2
3
4
5
6
7


Training Loss per Batch
Training Loss
Loss
Batch
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
100
200
300
400
500
600
700
800


Training VS Validation Accuracy
Train Accuracy
Validation Accuracy
Accuracy
Epoch
0.60
0.65
0.70
0.75
0.80
0.85
0.90
0.95
1.00
1
2
3
4
5
6
7

Accuracy:

- Validation set: 0.72
- Test set: 0.725

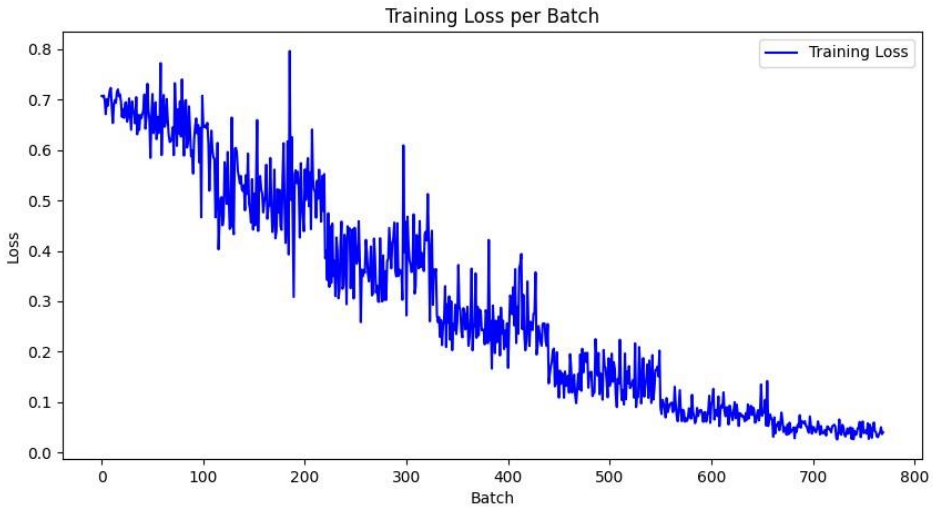
Total convergence time: 50.44 seconds

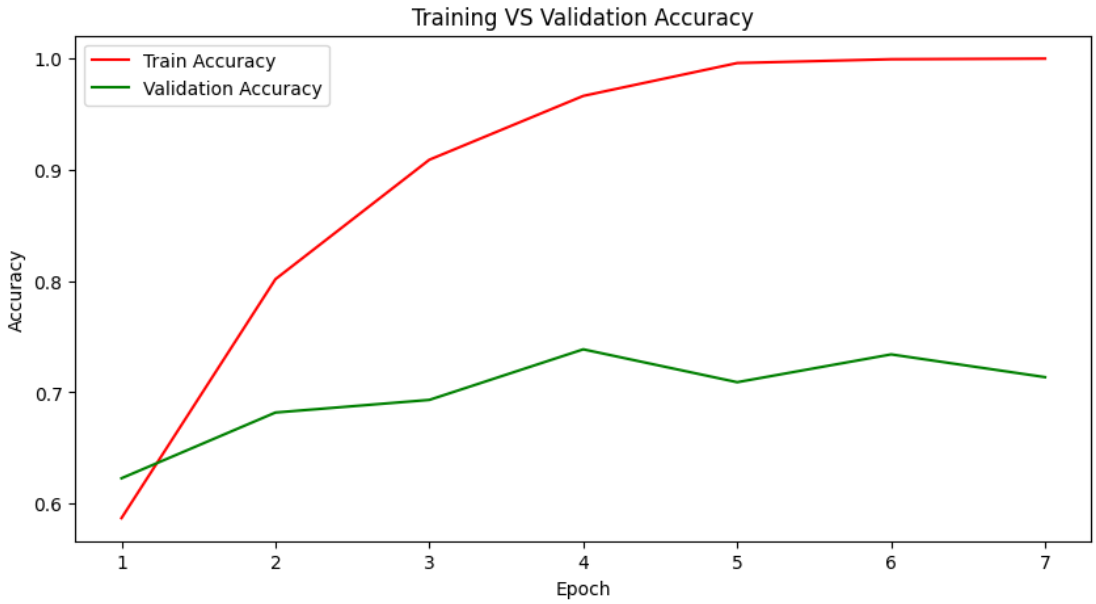
Total epochs: 7

Model #3

Hyperparameters:

- Epochs: 15
- Batch_size: 16
- Threshold: 0.5
- Learning_rate: 0.00005
- Weight_decay: 0
- Dropout: 0.2
- Min_delta: 0.05
- Optimizer: Adam





Accuracy:

- Validation set: 0.725
- Test set: 0.724

Total convergence time: 49.05 seconds

Total epochs: 7

Models comparison:

After analyzing the results, it's clear they share a significant resemblance. One notable aspect is that all experiments stopped early, usually after 7 or 8 epochs, meeting the early stopping criterion. Additionally, smaller batch sizes consistently lead to better outcomes.

On the other hand, Comparing the three models, we observe that Model 1 achieved the highest accuracy on both the validation and test sets.However, Model 2 had slightly lower accuracy compared to Model 1, possibly because of the higher value of min delta.
Model 3 performed the least in terms of accuracy, which might be influenced by the weight decay of 0.

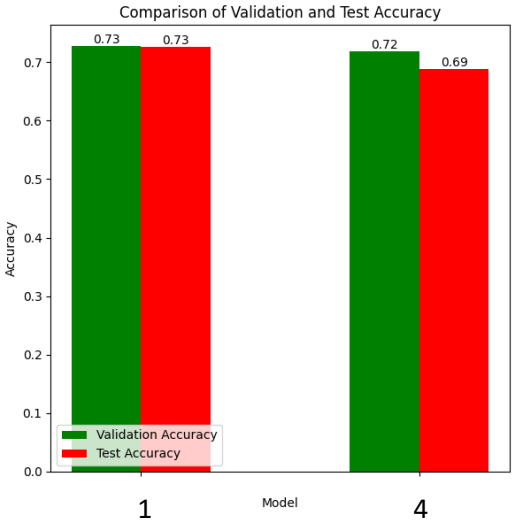
When examining the training-validation loss plots of the three models, despite a very low training loss which approaches to zero the validation loss remains very high compare to the training loss. When epochs is increasing the validation loss is decreasing well until at some points it stabilizes and even increasing a bit. In order to prevent this situation we tried different configurations of regularization and modified the weight decay and dropout parameters. Those models were the ones that gave the results while the other confugurations didn’t paid of in terms of general performance.

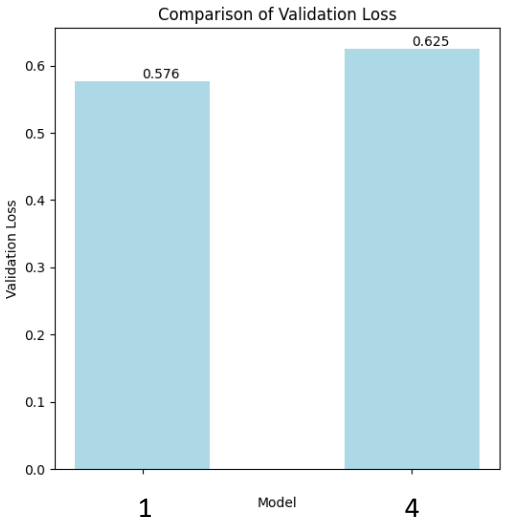
Let's examine two models: one of the best-performing models and one that is comparatively worse. We'll refer to the best model as Model 1 and the other, model 4 as follows:

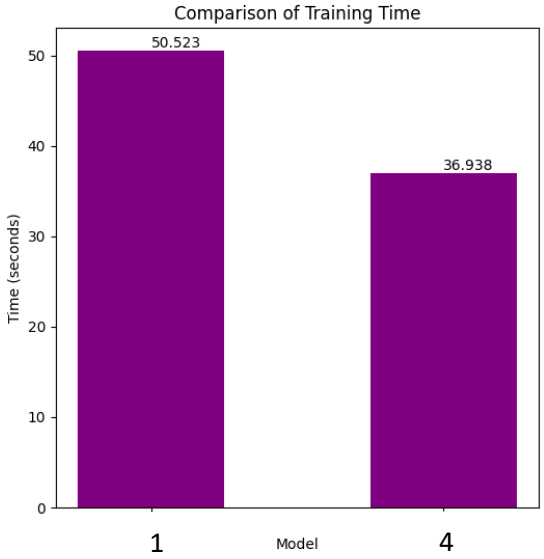
Model 4:

Hyperparameters:

- Epochs: 15
- Batch_size: 32
- Threshold: 0.6
- Learning_rate: 0.0001
- Weight_decay: 0
- Dropout: 0
- Min_delta: 0.05
- Optimizer: Adam







Time convergence - Model 4 has a larger batch size (32) compared to Model 1 (16) and a larger learning rate.
Generally, larger batch sizes can lead to faster convergence, but may sacrifice model generalizability. Since the total number of batches is larger with 16 batch, it takes more time for that model to converge. As for the learning rate, a higher learning may lead to faster convergence as well.

Accuracy and validation loss - Model 1 has a lower threshold (0.5) compared to Model 4 (0.6). This difference in thresholds may impact accuracy due to how the models classify the samples. Model 1, with its lower threshold, may classify them with less confidence, but on the other hand those samples appears to be classified correctly.
In addition, model 1 has a weight decay (0.0002) and dropout of 0.2, while Model 4 has no weight decay and no droput. Weight decay prevents overfitting by penalizing large weights in the model while the dropout does it by randomly dropping neurons (in training) just before

our last layer. The presence of weight decay and dropout in Model 1 may contribute to its slightly better performance in terms of validation and test accuracies.

Comparison

In this part we used our best model (model number 1) in order to analyze accurate classifications and misclassifications and understand why the model was wrong or right.

Example of TP: Both images are of the same person, and the model correctly identifies it as such.

True Positive - correct classification of same person

Probability: 0.976

Example of TN: The images are different from each other, and the model correctly identifies it as such.

True Negative - correct classification of different people

Probability: 0.023

Example of FP: The images are different from each other, but the model wrongly identifies it as the same person

False Positive - wrong classification of different people

Probability: 0.606

We believe this error stems from the similarities shared among individuals depicted in the images, encompassing factors such as gender and race. These resemblances likely posed a challenge for the model in discerning distinct enough features among their faces to accurately classify them as separate individuals.

Example of FN: Both images are of the same person**,** but the model wrongly identifies it as different person.

False Negative - wrong classification of same person

Probability: 0.361

We attribute this misclassification to several factors: variations in the person facial expression, with a smile presents in one image but not the other and the eyebrowsat both looks differently. In addition his hair in left one looks a bit shorther and even darker than the hair in the right image. The difference in background, with one image being brighter and the other is darker.
As a result, we think that the model encountered difficulty in recognizing sufficient similarities between the images to correctly classify them as the same person.

Conclusions

In general, the network's performance almost reached the 0.73 accuracy, which we consider as not very well. We believe this can be attributed to its initial design and the inability to generalize the problem of classify human faces. This architecture which was primarily designed towards less complicated problems than facial identification.
Due to the complexity of facial recognition tasks, which includes facial expressions, phisycal changes (such as beard, mustach) and different camera angles, the network likely struggled to achieve desired levels of accuracy.
As shown in previous sections, the high difference in validation loss compare to validation loss supports our argument that the network suffers a difficuly to deal with this task, even when regularizations techniques was used.