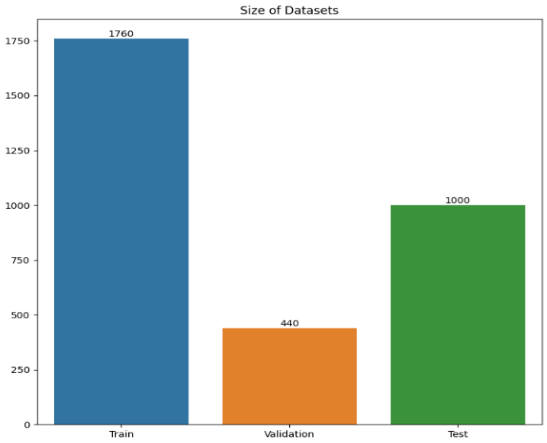
Deep Learning

In this assignment, we constructed a convolutional neural network model to address facial recognition using one-shot classification, drawing inspiration from the "Siamese Neural Networks for One-shot Image Recognition" paper. Our primary objective was to determine whether two facial images correspond to the same individual, for previously unseen images. Throughout the project, we delved into various CNN architectures, aiming to optimize performance and accuracy in this task.

Dataset Analysis:

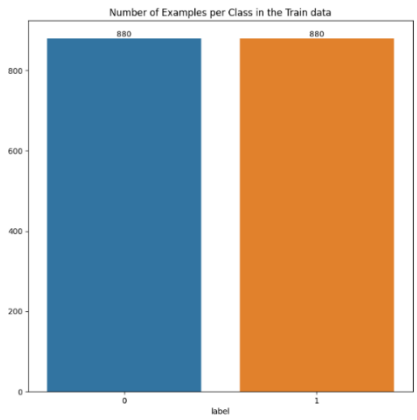


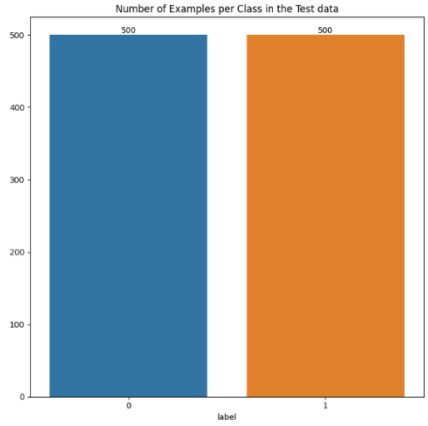
Total number of examples: 3200
Number of examples in the training set: 2200
Validation Set: 20% of the training set – 440 samples

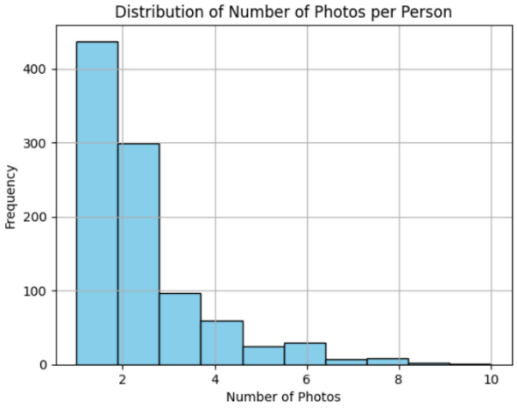
Number of examples in training set after validation: 1760
Number of examples in the test set: 1000

Summary:

	Train	Test	Validation
Class 0	880	500	220
Class 1	880	500	220







Full Experimental Setup –

Now, we will outline the architecture of our final model. Following this, we will delve into the experiments and results that guided our process.

Initialization:

As described in the paper:

Weights – normal distribution with zero mean and standard deviation of 0.01

Bias - normal distribution with 0.5 mean and standard deviation of 0.01

Stopping criterion:

Train the network until there is no improvement on the loos of the validation set for 5 epochs.

Number of epochs: Maximum 30 epochs

Loss method: BCELoss

Batch size: 16

Learning Rate: 0.0001

Optimizer: Adam

Batchnorm: True

3) Siamese network:

Reshaped the size of the images from 250x250 to 105x105 (matched to the paper).

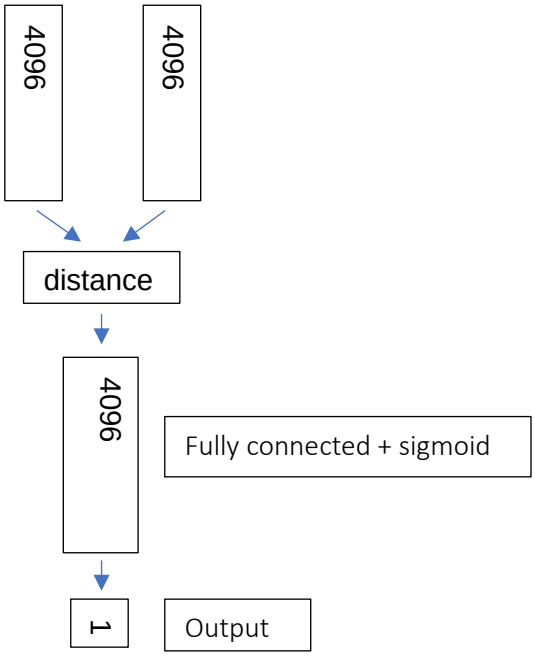
Architecture:

A)

	Dimension In	Filter size	Dimension Out
Conv1 + Relu + batchnorm	105x105x1	10x10x64	96x96x64
Max Pooling 1	96x96x64	2x2	48x48x64
Conv2 + Relu + batchnorm	48x48x64	7x7x128	42x42x128
Max Pooling 2	42x42x128	2x2	21x21x128
Conv3 + Relu + batchnorm	21x21x128	4x4x128	18x18x128
Max Pooling 3	18x18x128	2x2	9x9x128
Conv4 + Relu + batchnorm	9x9x128	4x4x256	6x6x256
Flatten	6x6x256		9216
Fully connected + sigmoid + batchnorm	9216		4096

B)

In the initial phase, Phase A, each image undergoes processing. Subsequently, in Phase B, the distance between the outputs of the two images processed in Phase A is calculated. This distance vector passes through an additional layer of fully connected followed by a sigmoid activation function. In the end, Phase B produces the probability that the two images depict the same person.



Parameters:

The full experimental setup was finalized through the application of grid search, which involved exploring various combinations of hyperparameter values to pinpoint the set that yields the optimal performance for our specific task based on the accuracy of the validation set.
The hyperparameter that we investigated:

Batch sizes: We tried 8 and 16 batch sizes. We investigated those batch sizes because we searched and saw that those sizes are recommended for small datasets.

Learning rate: We tried different values – 0.001, 0.0001, 0.00001. Our objective was to assess the impact of lower learning rates on performance. It's worth noting that the paper we referenced employed learning rates ranging from 0.0001 to 0.1 in their experiments.

Optimizers: We tried Adam and momentum using SGD. In the paper, they used momentum optimizer that is an extension of SGD. Adam is known for its increased robustness compared to SGD, so we chose to include it in our investigation.

Loss Methods: We tried BCELoss and MSELoss. BCELoss is commonly used for binary classification tasks, while MSELoss is suitable for regression problems. Despite MSELoss being designed for regression, we investigated its performance in the context of binary classification for comparison.

Experiments and Performance:

Here we illustrate the results of the performance by changing the parameters. We used the chosen hyperparameters and compared the performance on the validation set when changing their values.

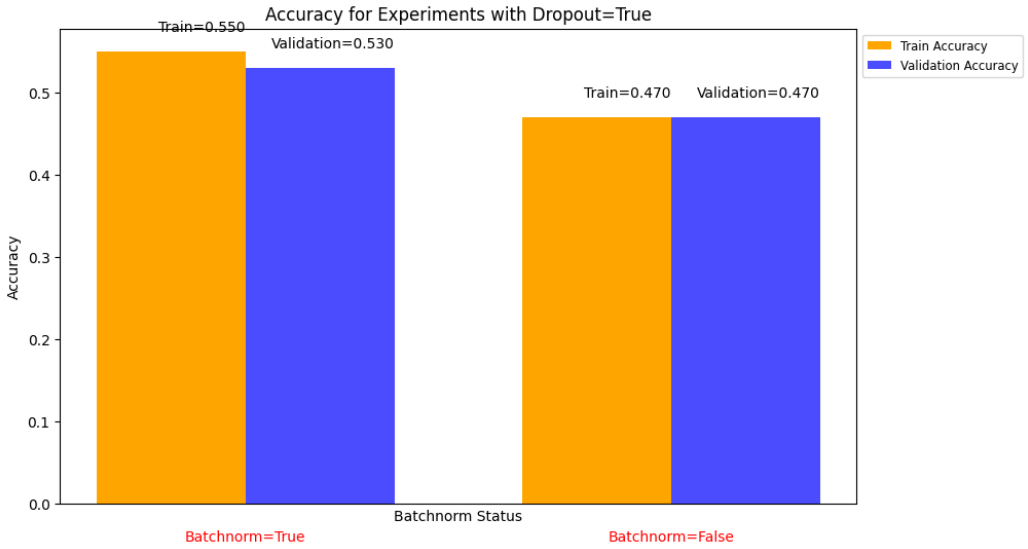
- Experiments' results using – batchnorm and dropout:

Our first experiment was to assess the impact of incorporating dropout and batch normalization directly within the neural network architecture. To achieve this, we conducted four experiments, each exploring different combinations of these techniques alongside the default parameters of the network, so the only impact on the different networks will be from changing the combination of dropout and batchnorm. The default parameters are: batch size – 8, 30 epochs with early stopping of 5 epochs, loss function – BCELoss, learning rate – 0.001, and optimizer – Adam.

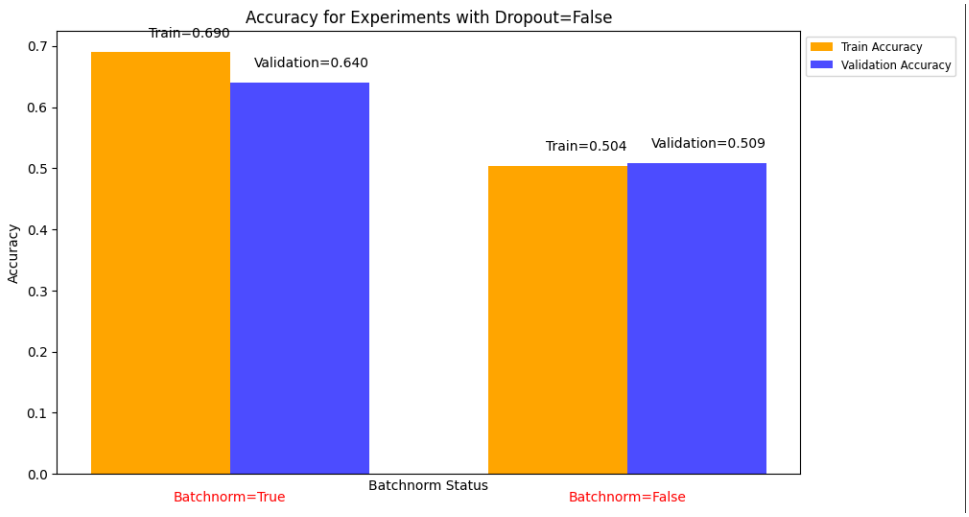
The four combinations:

Batchnorm	Dropout
True	True
True	False
False	True
False	False

When Dropout=True:

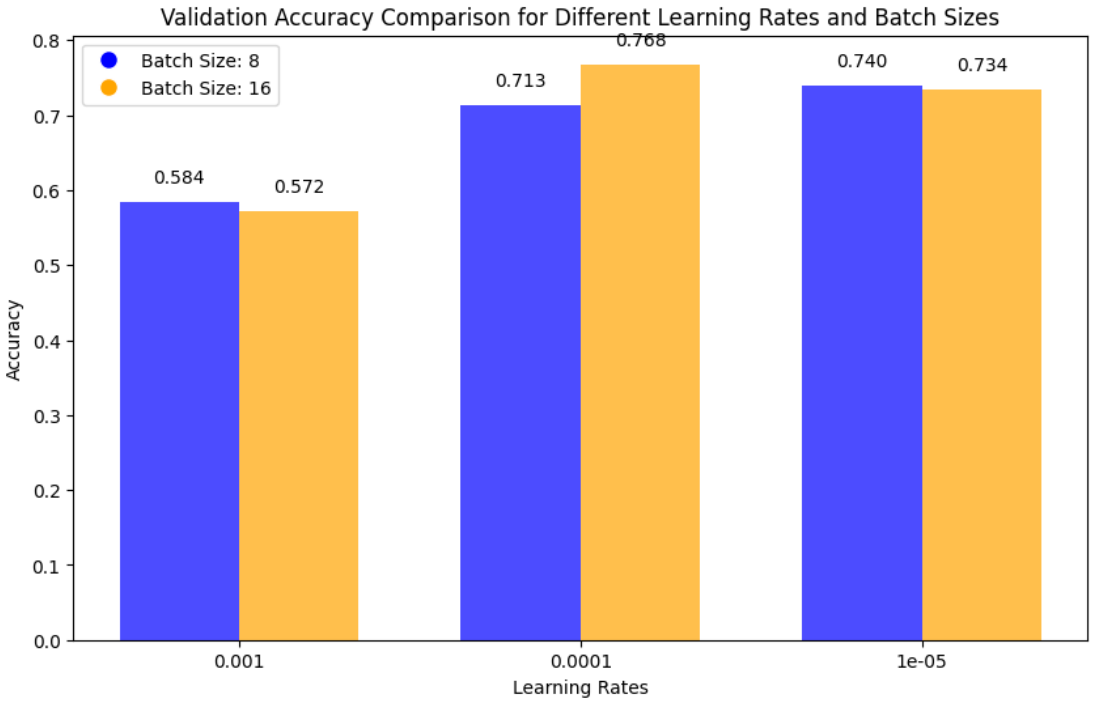


When Dropout=False:



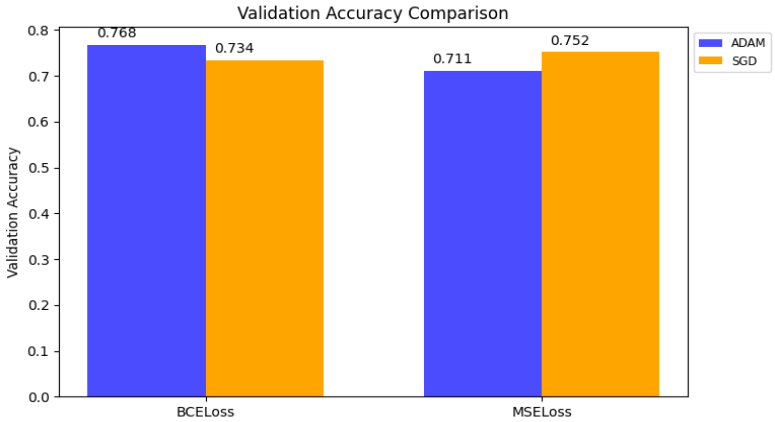
When Dropout was not activated, the performance was better compared to when Dropout was activated. When the batchnorm was activated, the performance was better. Based on these findings, we conclude that batchnorm is a beneficial addition to our neural network architecture, providing improvements in model performance. However, the use of dropout did not exhibit a similar positive impact in our experiments. Therefore, we have opted not to persist with the utilization of dropout in our ongoing model configurations and indeed continue to use batchnorm.

- Experiments' results using different learning rates and batch sizes:



Based on these findings, we can see that lower learning rates generally lead to higher validation accuracies across different batch sizes. The highest validation accuracy (0.768) is achieved with batch size 16 and a learning rate of 0.0001.

- Experiments' results using varying optimizers and different loss methods:



The combination that achieved the highest validation accuracy is the Adam optimizer and Binary Crossentropy (BCELoss).

The parameters that lead to the highest performance on the validation set are : batch size – 16, 30 epochs with early stopping of 5 epochs, loss function – BCELoss, learning rate – 0.0001 and optimizer – Adam.

Having identified our optimal model based on its performance on the validation set, we will now assess its performance by running it on the test set.

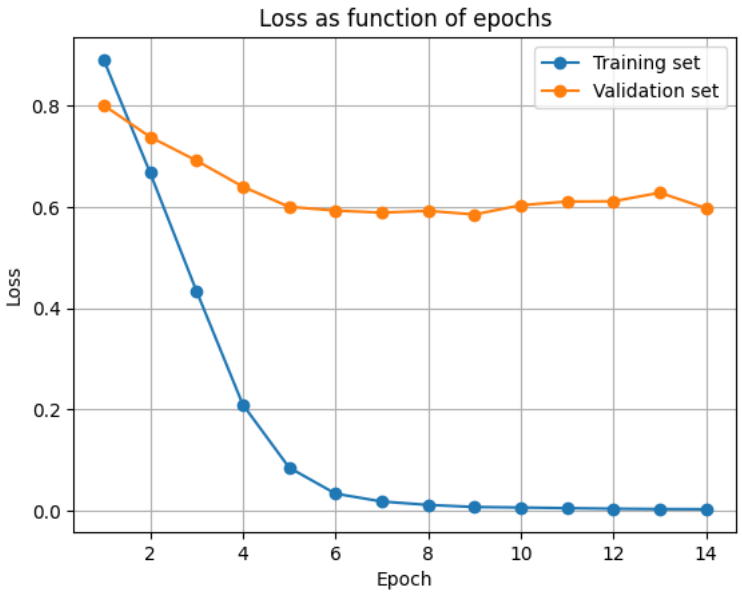
Experimental Results of the chosen model:

	Training set	Validation set	Test set
Final Loss	0.0026	0.555	0.713
Final accuracy	1	0.768	0.711

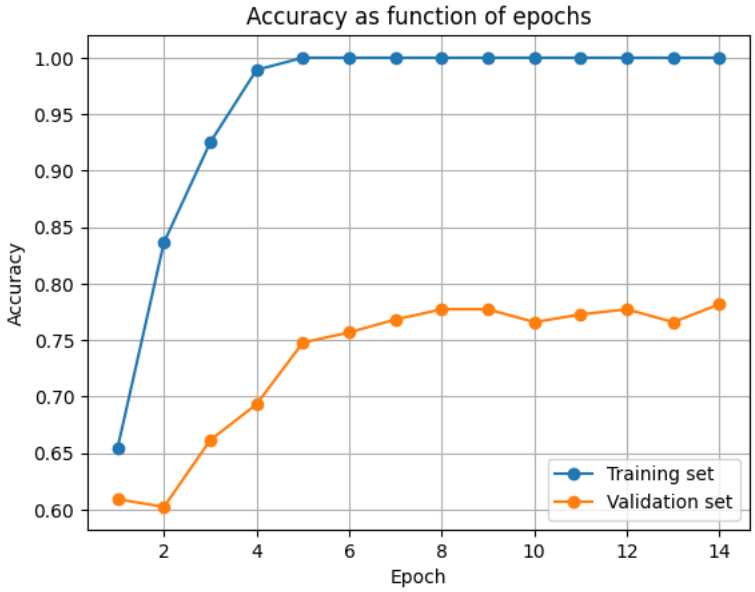
Convergence time: 602.229 seconds

```
convergence time : 602.2299094200134
test loss : 0.71375349259069
test accuracy : 0.711
```

Loss as a function of epochs:



Accuracy as a function of epochs:



We can observe from the graph that overfitting becomes evident on the training set after a few epochs. Given additional time, we could mitigate this issue by expanding the training set. One potential approach is to introduce images with noise.

Accurate classification:

1) Same person – the model indicated high confidence that the provided images depict the same person (probability of 0.999).

real_label -> 1 and pred_lable -> 1

```
The probability:  0.9997395873069763
```

Label: 1

2) Different people – the model indicated high confidence that the provided images depict different people (probability of 0.999).

real_label -> 0 and pred_lable -> 0

```
The probability:  0.9998626708984375
```

Label: 0

The photo's correct prediction validates the model's proficiency in correctly identifying whether two individuals are the same or different, even without prior exposure to their images.

Misclassifications Prediction

1) Same person - the model indicated low confidence that the provided images do not depict the same person, assigning a probability of 0.011 to this prediction.

real_label -> 1 and pred_lable -> 0

2) Different people – the model indicated high confidence that the provided images depict the same people (probability of 0.995).

real_label -> 0 and pred_lable ->1

We noticed that the model encounters difficulties in consistently recognizing individuals across different facial expressions, particularly variations in smiles seen in the provided examples. These variations seem to affect accurate identification. Additionally, accessories such as scarfs and sunglasses may also impact the model's predictions.

Appendix: Grid search results

optimizer	loss_fn	epochs	early_stopping	batch_size	Learning rate	train_losses	train_accuracy	val_losses	val_accuracy	convergence_time
Adam	BCE	13	TRUE	8	0.0001	0.03	1	0.64	0.71	396.61
Adam	BCE	13	TRUE	8	1e-05	0.02	1	0.52	0.74	379.46
Adam	BCE	5	TRUE	8	0.001	0.87	0.59	0.83	0.58	145.78
Adam	BCE	13	TRUE	16	0.0001	0	1	0.56	0.78	316.35
Adam	BCE	14	TRUE	16	1e-05	0.01	1	0.52	0.73	362.32
Adam	BCE	7	TRUE	16	0.001	0.67	0.66	0.87	0.57	177.44
Adam	MSE	14	TRUE	8	0.0001	0.01	1	0.19	0.73	411.23
Adam	MSE	10	TRUE	8	1e-05	0	1	0.17	0.75	299.19
Adam	MSE	7	TRUE	8	0.001	0.4	0.54	0.37	0.55	197.53
Adam	MSE	6	TRUE	16	0.0001	0.01	1	0.19	0.71	145.55
Adam	MSE	25	TRUE	16	1e-05	0	1	0.18	0.75	607.13
Adam	MSE	20	TRUE	16	0.001	0.2	0.74	0.24	0.65	487.41
SGD	BCE	9	TRUE	8	0.0001	0.07	1	0.56	0.74	238.06
SGD	BCE	23	TRUE	8	1e-05	0.17	1	0.57	0.69	643.74
SGD	BCE	5	TRUE	8	0.001	1.29	0.57	1.92	0.53	133.66
SGD	BCE	11	TRUE	16	0.0001	0.05	1	0.55	0.73	258.8
SGD	BCE	30	FALSE	16	1e-05	0.17	1	0.57	0.7	728.38
SGD	BCE	6	TRUE	16	0.001	0.39	0.86	1.15	0.61	144.64
SGD	MSE	10	TRUE	8	0.0001	0.03	1	0.19	0.71	270.35
SGD	MSE	16	TRUE	8	1e-05	0.11	0.99	0.22	0.66	434.29
SGD	MSE	6	TRUE	8	0.001	0.21	0.87	0.24	0.68	160.67
SGD	MSE	24	TRUE	16	0.0001	0.01	1	0.18	0.75	560.87
SGD	MSE	30	FALSE	16	1e-05	0.08	1	0.21	0.69	705.46
SGD	MSE	15	TRUE	16	0.001	0	1	0.2	0.72	357.64