Deep Learning HW2 – Report

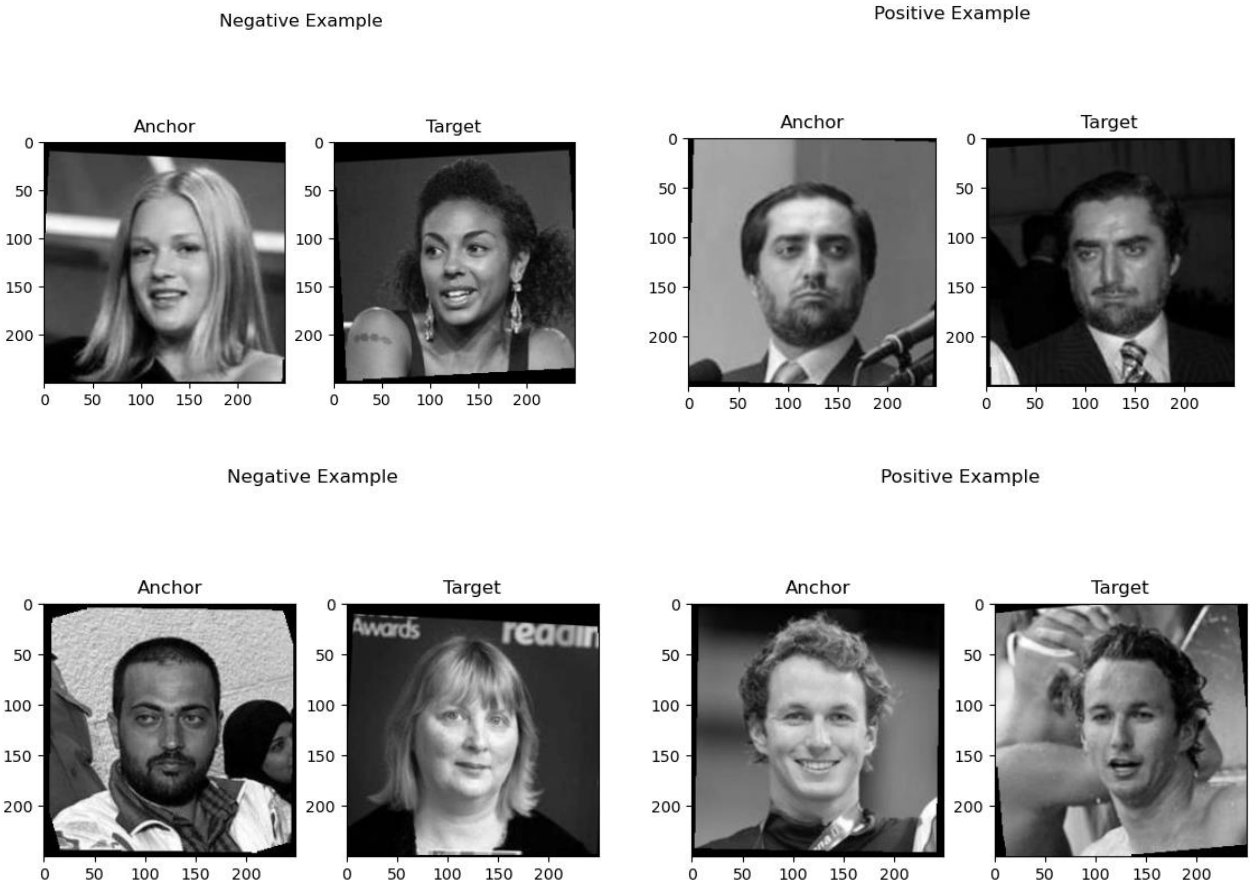
Saar Ben David – 305560591
Ido Rabia – 201181450

Summary:

In this task, we have implemented a one-shot classification solution based on a Siamese NN. Our NN's architecture is based on the one described in the paper: Siamese Neural Networks for One-shot Image Recognition.
In one-shot classification we aim to predict whether two **unseen** objects (images) are of the same class or not. That type of classification differs from the regular one because we try to make a prediction on objects that the model sees for the first time (didn't train on them).

Dataset:

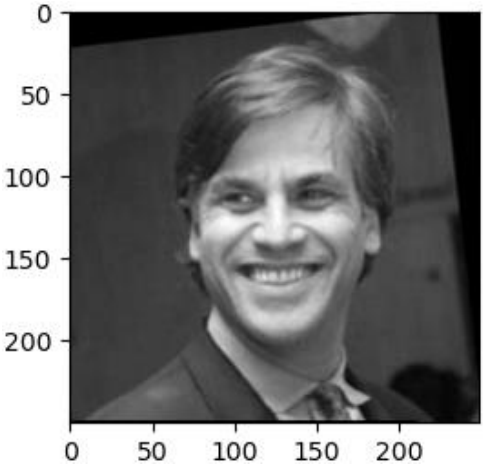
In this task, we used the dataset called LFW-a (downloaded from Labeled Faces in the Wild). This dataset contains greyscale facial images, each image is 250X250 pixels.
Our training-set consists of 2200 examples - 1100 positive examples (pair of images of the same person) and 1100 negative ones (pairs of images of two different persons).
Our test-set consists of 1000 examples, 500 positive examples and 500 negative examples.



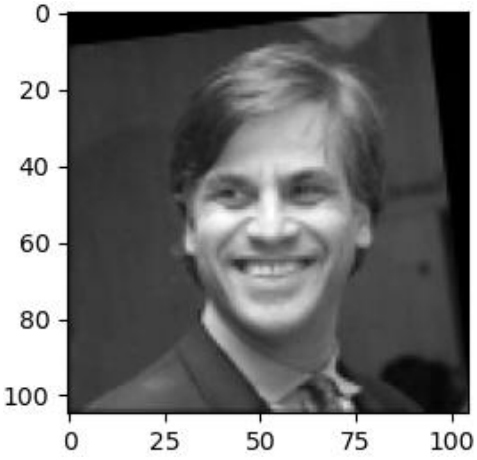
Preprocessing:

First, we resized the images in our dataset to the size of 105X105 pixels (same input size as described in the paper). We did it to reduce the number of parameters in our network without harming the learning process.
In addition, we have normalized our pictures by dividing each image-matrix by 255. We did it to make each of our inputs to be on the scale of [0,1].

Regular Image



Under-Sampled Image



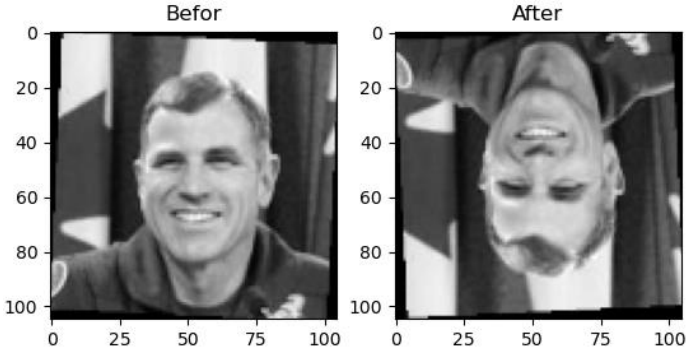
Lastly, we performed two kinds of preprocessing: Normalization and Augmentation.
The normalization phase includes:

1. Centering – subtracting the featurewise mean from the dataset.
2. Standardizing – dividing the dataset by the featurewise standard deviation.

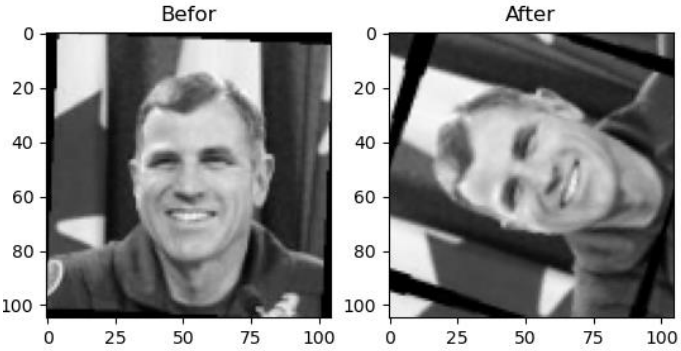
Normalization has been applied on all the data set (train, validation, and test). The final value of a feature after the normalization is: $x^i = \frac{\frac{x^i}{255} - \mu}{\sigma^2}$

The augmentation includes four types of changes:

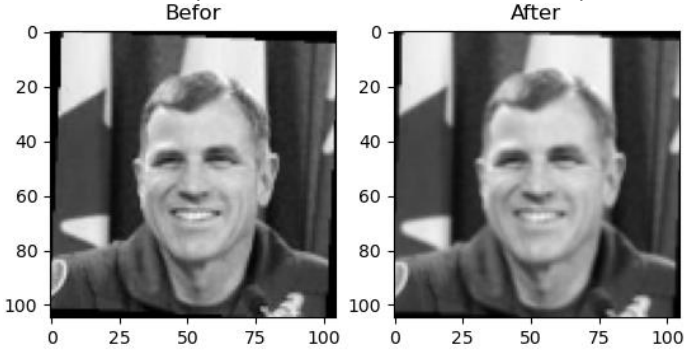
1. *RandomFlip* ("horizontal_and_vertical")



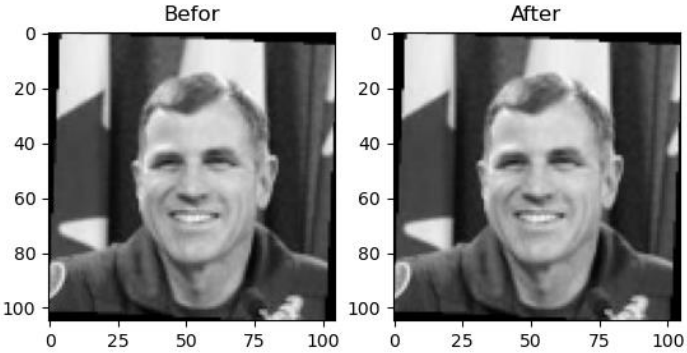
2. *RandomRotation* (factor =between 0.2 and 0.4)



3. *RandomZoom* (factor =between -0.3 and 0.3)



4. *RandomContrast* (factor =between 0.1 and 0.5)



Augmentation was only applied on the training data set. After augmentation, we ended up increasing the amount of the training data set from 1760 examples to 8800.

Experimental Setup:

We used the network architecture of the paper as a baseline for our experiments. We configured following hyperparameters:

- batch_size = 128
- epoch = 25 (Original paper used 200 epochs. We decided to decrease to 25 because we have less data)
- optimizer was set to Adam
- learning_rate = 0.0001 (We followed the paper instructions here: $\eta_j \in (10^{-4}, 10^{-1})$) In addition, each epoch we decreased the learning rate by 1%.
- momentum = 0.5 (Same as in the paper)

- loss function was set to binary crossentropy

We allocated 20% of the training examples to the validation-set.
We also set a stopping criterion for the learning process once the validation-loss stop decreasing for 5 consecutive epochs. In case of early stopping, the model with the lowest validation-loss is saved (using tensorflow checkpoint-callback), then reloaded for the evaluation part.

We also used L2 regularization factors both in our convolutional and fully connected layers. For our convolutional layers we used regularization factor of 0.001, and for our fully connected layers we used factor of 0.0001. It is worth mentioning that we also tried factors of 0.01 and 0.001, but we have got the same results, but the convergence time was higher (took more epochs).

In addition, we also tried change the regularization technique for the fully connected layers, and instead using L2 we used dropout with 15% drop chance.

Evaluation:

We evaluated the learning process by looking at the change in the loss and accuracy along the epochs. These graphs were generated by Tensorboard-API for both the training and the validation sets.

Then, we evaluated our trained models in two ways:

1. For each of our models we evaluated its binary-accuracy, recall, precision, and AUC metrics on standard binary-classification tasks.
2. In addition, we evaluated the accuracy of each of our models on One-Shot classification tasks.

We generated the One-Shot classification tasks from each of our **positive** pairs in the test-set.
For each positive pair, we generated 9 other negative pairs for the same anchor. Each 10-pairs-combo (1 positive – 9 negatives) was counted as a single One-Shot classification task. For each task we tested whether the model succeeded in predicting the positive pair (a simple argmax over 10 predictions). The One-Shot classification accuracy was calculated as $\frac{\#success}{\#tasks}$.
Each model was tested on 500 One-Shot classification tasks.

Experiments:

Experiment 1 – baseline (no normalization, no dropout, no augmentation)

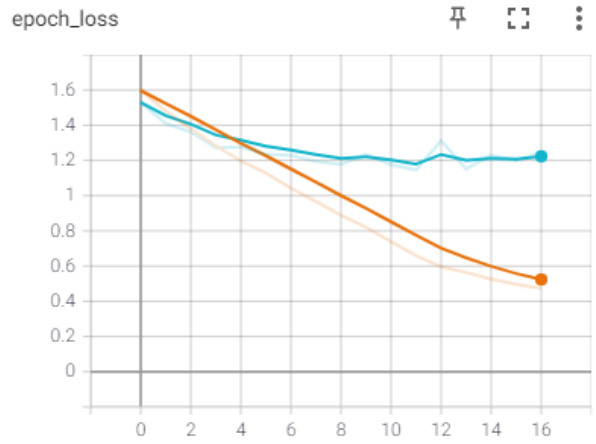
For our baseline network, we used the Siamese Network architecture as described in the paper:

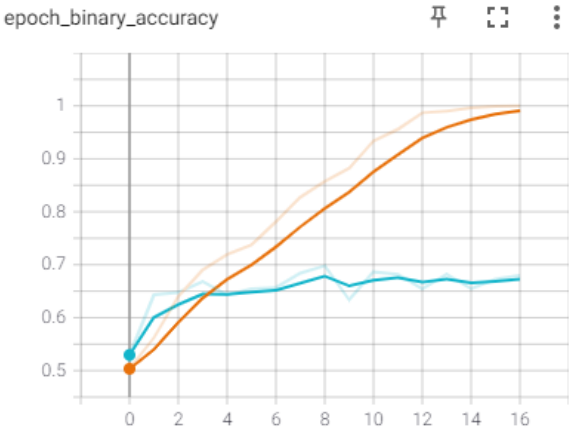
1. Input – 2 images of dimensions (105X105X1)
2. Conventional + relu 64 @ (10,10)
3. MaxPooling2D
4. Conventional + relu 128 @ (7,7)
5. MaxPooling2D
6. Conventional + relu 128 @ (4,4)
7. MaxPooling2D
8. Conventional + relu 256 @ (4,4)
9. Flatten - fully connected
10. Dense - fully connected (4096 neurons) + sigmoid
11. Dense – fully connected (1 neuron) + sigmoid

In this experiment, there were 38,951,745 parameters and the training stopped after 17 epochs (out of 25 available).

Results on test-set	
Binary-Accuracy	0.668
One-Shot Accuracy	0.242
Recall	0.692
Precision	0.660
AUC	0.725

In the following 2 graphs we can see the changes in loss and in binary_accuracy between epochs. The **orange** line stands for the **training**, and the **cyan** line stands for the **validation**.



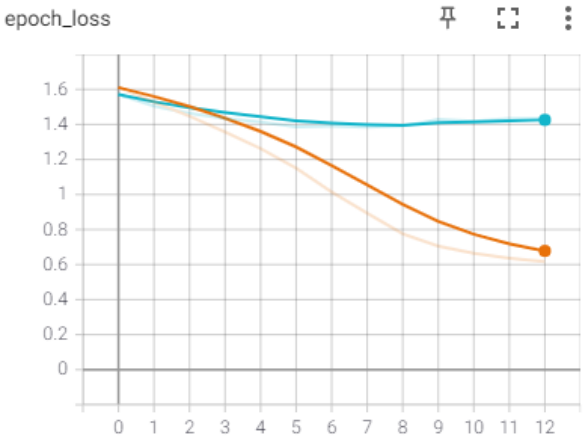


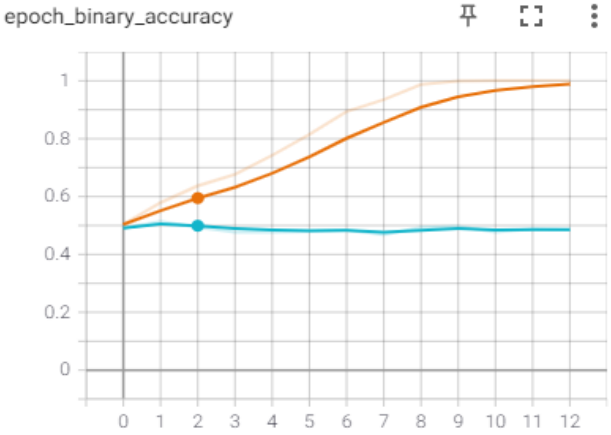
Experiment 2 – baseline with featurewise-normalization

In this experiment we used the same baseline architecture, but we have added a featurewise normalization preprocessing to it. There were 38,951,745 parameters and the training stopped after 13 epochs (out of 25 available).

Results on test-set	
Binary-Accuracy	0.47
One-Shot Accuracy	0.108
Recall	0.516
Precision	0.472
AUC	0.450

In the following 2 graphs we can see the changes in loss and in binary_accuracy between epochs. The **orange** line stands for the **training**, and the **cyan** line stands for the **validation**.





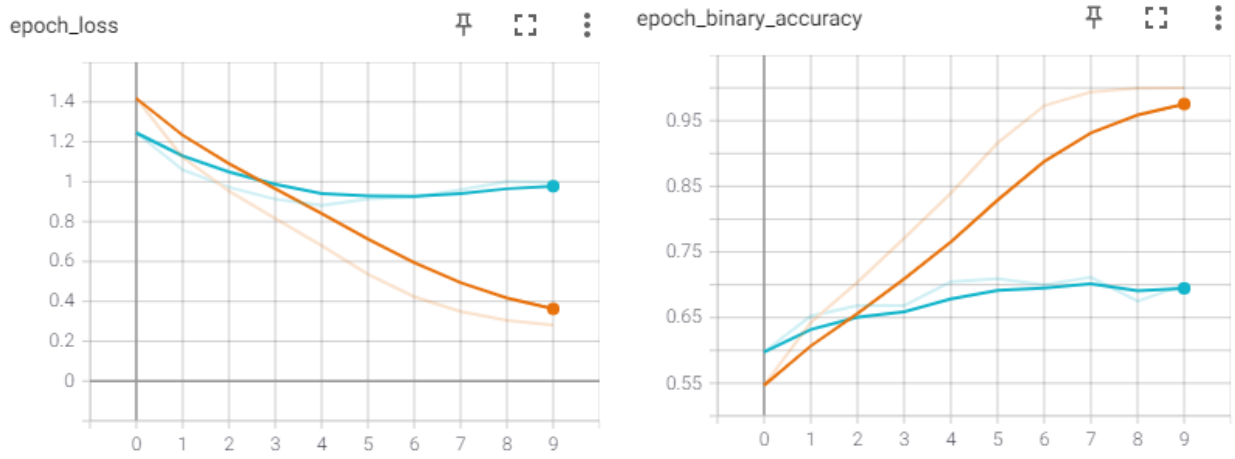
We can easily see that our model hits a plateau really fast on the validation set, both in loss and in accuracy. We deduce that featurewise-normalization might not suit the task we are trying to learn.

Experiment 3 – baseline with augmentation

In this experiment we used the same baseline architecture, but we have added augmented images to the training set. There were 38,951,745 parameters and the training stopped after 10 epochs (out of 25 available).

Results on test-set	
Binary-Accuracy	0.666
One-Shot Accuracy	0.222
Recall	0.62
Precision	0.679
AUC	0.739

In the following 2 graphs we can see the changes in loss and in binary_accuracy between epochs. The **orange** line stands for the **training**, and the **cyan** line stands for the **validation**.



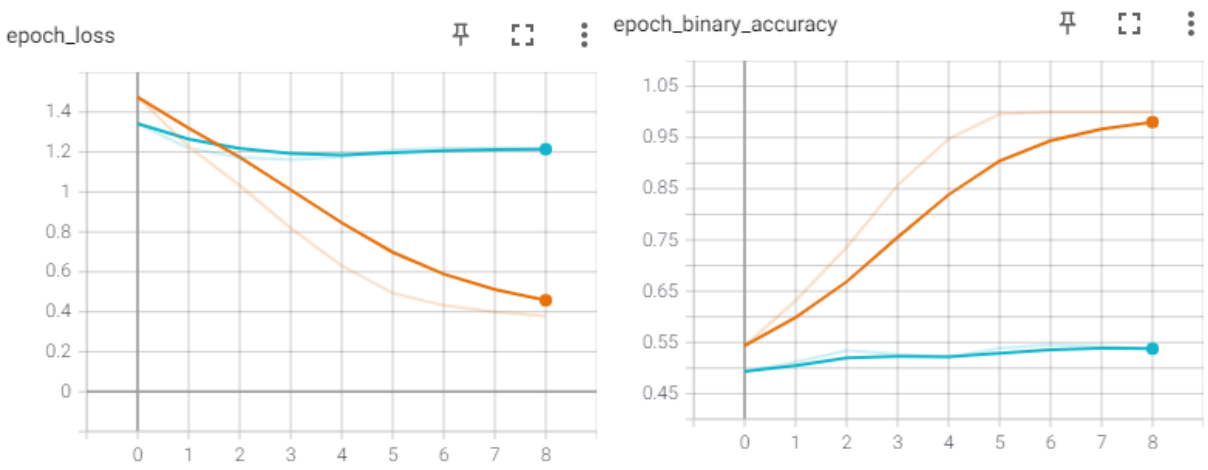
We can see that though using augmented data doesn't improve the accuracy of the model, there's a ~20% reduction in the validation loss, and the model hits the same accuracy in less epochs (10 instead of 17).

Experiment 4 – baseline with both augmentation and normalization

In this experiment we used the same baseline architecture with augmentations and normalization. There were 38,951,745 parameters and the training stopped after 9 epochs (out of 25 available).

Results on test-set	
Binary-Accuracy	0.532
One-Shot Accuracy	0.116
Recall	0.340
Precision	0.552
AUC	0.534

In the following 2 graphs we can see the changes in loss and in binary_accuracy between epochs. The **orange** line stands for the **training**, and the **cyan** line stands for the **validation**.



Experiment 5 – baseline with Dropout

In this experiment we used the Dropout mechanism on both our fully-connected layer:

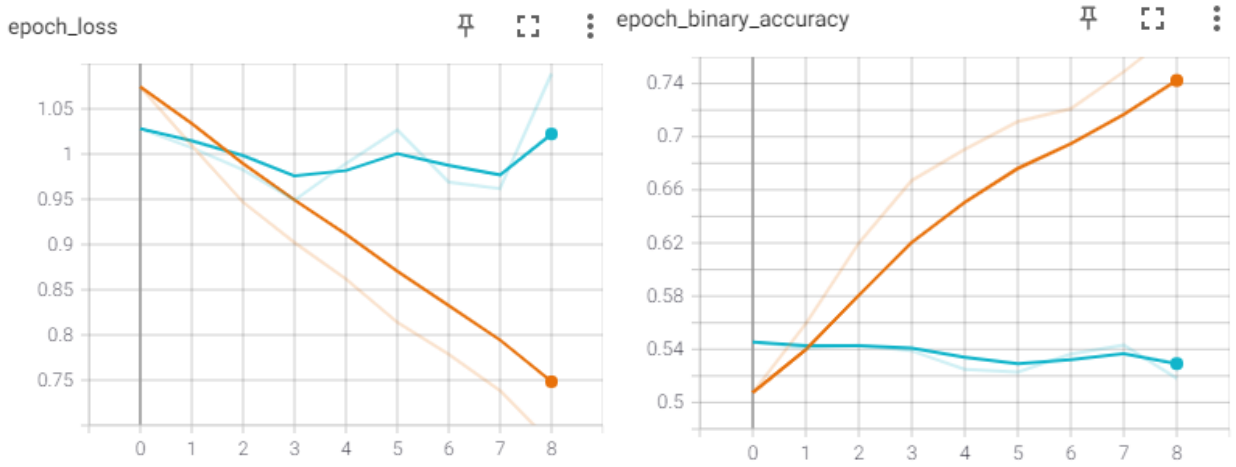
1. Input – 2 images of dimensions (105X105X1)
2. Conventional + relu 64 @ (10,10)
3. MaxPooling2D
4. Conventional + relu 128 @ (7,7)
5. MaxPooling2D
6. Conventional + relu 128 @ (4,4)
7. MaxPooling2D
8. Conventional + relu 256 @ (4,4)
9. Flatten - fully connected
10. Dense - fully connected (4096 neurons) **+ Dropout (10% drop chance)** + sigmoid
11. Dense – fully connected (1 neuron) + sigmoid

There were 38,951,745 parameters and the training stopped after 9 epochs (out of 25 available).

Results on test-set	
Binary-Accuracy	0.555
One-Shot Accuracy	0.222
Recall	0.176
Precision	0.710
AUC	0.721

While barely affects the One-Shot accuracy comparing to the baseline (a reduction of 2%), using dropout did affects the Recall and the binary-accuracy of the network.

In the following 2 graphs we can see the changes in loss and in binary_accuracy between epochs. The **orange** line stands for the **training**, and the **cyan** line stands for the **validation**.



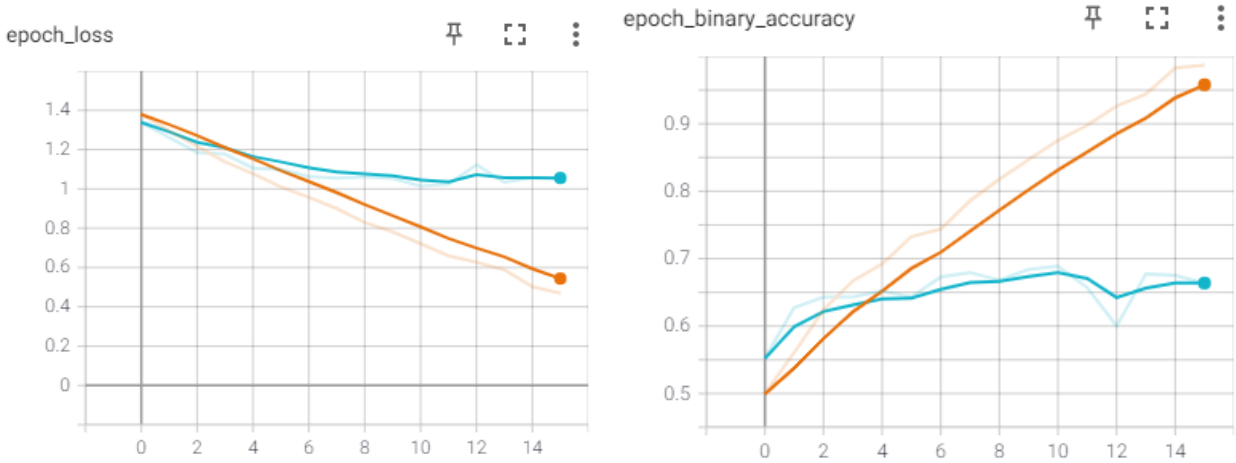
Experiment 6 – altered baseline (narrow dense layer)

In this experiment we narrowed the fully-connected layer to contain 2048 neurons (instead of 4096):

There were 20,073,281 parameters and the training stopped after 16 epochs (out of 25 available).

Results on test-set	
Binary-Accuracy	0.670
One-Shot Accuracy	0.252
Recall	0.689
Precision	0.663
AUC	0.735

In the following 2 graphs we can see the changes in loss and in binary_accuracy between epochs. The **orange** line stands for the **training**, and the **cyan** line stands for the **validation**.

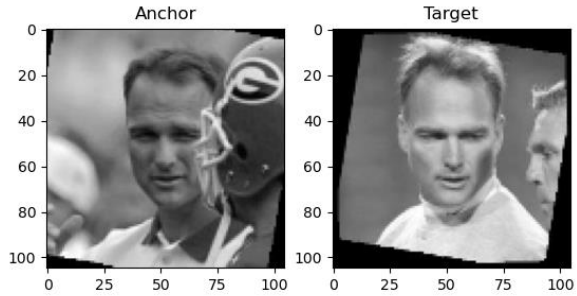


Results:

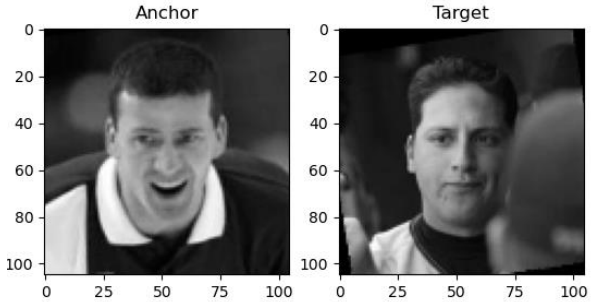
As described in the table below, we can see that the altered-baseline model slightly improves the binary accuracy and the one-shot accuracy of the model. In addition, it is worth mentioning that due to a narrowed fully-connected layer, the number of parameters is cut by almost a half, which translates to higher convergence times, and a shorter training in general.

Model	Binary-Accuracy	One Shot Accuracy	Recall	Precision	AUC
Baseline	0.668	0.242	**0.692**	0.660	0.725
Baseline + Featurewise Norm.	0.47	0.108	0.516	0.472	0.450
Baseline + Augemntations	0.666	0.222	0.62	0.679	**0.739**
Baseline + Augemntations + Featurewise Norm.	0.532	0.116	0.340	0.552	0.534
Baseline + Dropout (10%)	0.555	0.222	0.176	**0.710**	0.721
Altered Baseline (Narrowed)	**0.670**	**0.252**	0.689	0.663	**0.735**

Here we will demonstrate some of the altered-baseline model's best and worst binary-classification tasks:

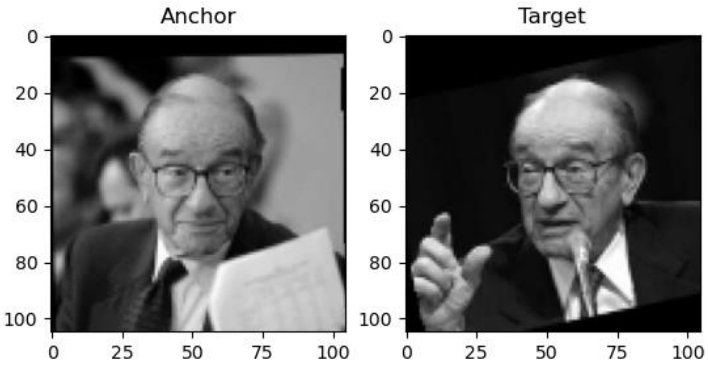


The model gave for the above POSITIVE anchor-target pair only a 0.039 probability for being the same person.



For the above NEGATIVE pair the model gave a 0.978 probability for being the same person.

These two examples were 2 of the worst predictions given by our model.



The above POSITIVE pair is received the model's most accurate prediction. It predicted the pair to be of the same person with a probability of 0.993.

Now we will demonstrate it's BEST One-Shot classification tasks:

In addition to succeeding in this task, we can see that most of the negative targets received a much lower probability, except from the negative target in the red circle.

And it's WORST One-Shot classification tasks:

We can see that both the anchor image and the image the model has picked (marked in red) have large portion of dark background, in contrast to the positive image that has a much brighter background.