Deep Learning – CNN Assignment – Facial Recognition

Analysis of the dataset

The LFW-a dataset contains 13,233 different images of 5,749 different persons. In addition, the given train set contains 2200 samples (pairs), in which, 1100 samples are positive samples (same person = positive class = 1), and the other 1100 samples are negative samples (different persons = negative class = 0). The test set contains 1000 samples – 500 of the positive class and 500 of the negative class.

Moreover, we divided the train set into train and validation sets while the validation set portion was 15% of the original train set, i.e., 330 samples were selected randomly. Both sets were balanced exactly as the original train set – 50% of each label.

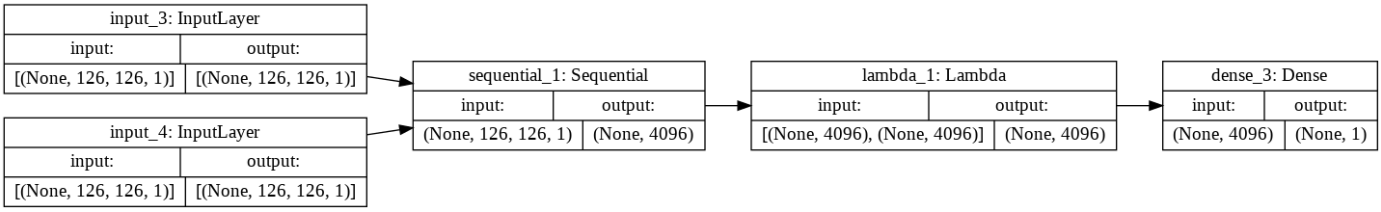
Full experimental setup

We used cross entropy as our loss function with Adam optimizer and learning rate of 0.0001. We tried few learning rates between [0.1,0.0001], as stated in the paper and 0.0001 got the best performances. The model's training was conducted with batch size of 16.

In addition, our stopping criteria was 40 epochs or early stopping when reached 4 epochs without improvement of the validation loss.

The experiment was conducted on google colab environment using GPU.

Description of our architecture:



Our model contains two input layers of shape (126,126,1). Then, each of the inputs is connected to a sequential model with the following architecture:

Conv2D(filters=64, kernel size=(10,10), activation = relu)

Batch Normalization

Max Pool2D(pool size=(2,2)

Conv2D(filters=128, kernel size=(7,7), activation = relu)

Batch Normalization

Max Pool2D(pool size=(2,2)

Conv2D(filters=128 kernel size=(4,4), activation = relu)

Batch Normalization

Max Pool2D(pool size=(2,2)

Conv2D(filters=256 kernel size=(4,4), activation = relu)

Batch Normalization

Flatten

Fully Connected (units=4096, activation = sigmoid)

After that, both inputs were inserted into the sequential model. The output was two vectors which was inserted into a distance layer and calculated the absolute difference between them. Finally, we added a dense layer with sigmoid activation with 1 unit which gives us the probability of the pair being 1 (positive=same) or 0 (negative=different).

Explanation of our architecture:

We started with the paper's architecture. We found out that the initialization of the bias, kernel weights and the regularization that were implemented in the paper were unhelpful, so we decided not to use them. Then, we tried to add batch normalization layers after each conv2d layer and saw an improvement of the performances. As we saw on other paper, they recommended to add dropout layer after dense layers, but it gave no improvement.

Another improvement came from changing the optimizer. We used an Adam optimizer instead of an SGD optimizer.

Analysis of architecture's performance

- Final Details:

Convergence time: 156.887 seconds

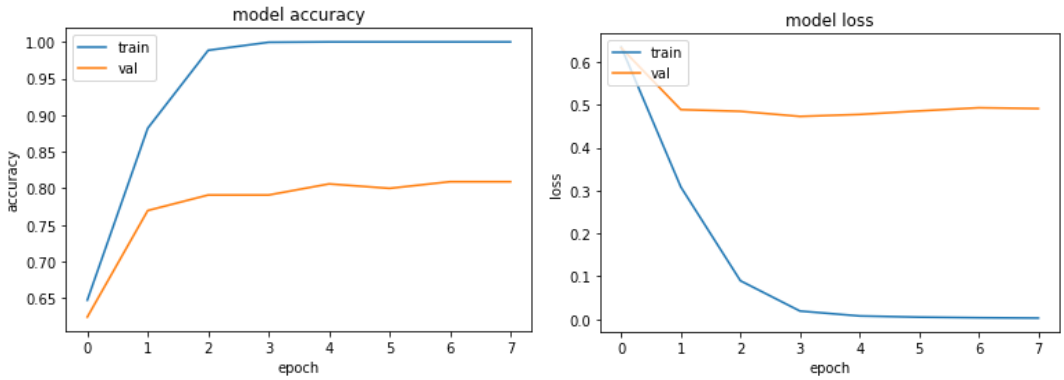
Final loss test set: 0.57

Accuracy test set: 78%

Final loss validation set: 0.49

Accuracy validation set: 80.91%

- Graphs describing the loss on the training set throughout the training process:



As we can see, the model reached plateau after 3 iterations, so the learning stopped after 7 epochs because of our early stopping criteria. Moreover, the gap between the graphs is too wide so we can understand that the model was overfitted. We tried to eliminate/reduce this problem by finding better hyperparameter configurations and data augmentation.

- Performance when experimenting with the various parameters:

Learning rate - we tried many learning rates between [0.1,0.0001] as mentioned in the paper and found out that when the learning rate is 0.0001 the model reached the best performance.

Batch size – we tried different batch sizes such as [8,16,32,64], when trying to reach fast convergence and fasts running time. We experienced that batch size of 16 gets the best performances in this matter.

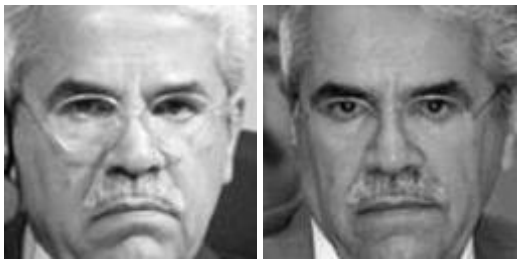
Early stopping – we ran the model and found out that after 4 epochs if there is no improvement it is better to stop because the model is diverging.

- examples of misclassifications

As we can see, the model struggles to correctly classify those images, and our intuition is because there is more than one face in the picture. Because of this insight we decided to

crop all the data and take only the center of the image with ratio of 50%. The new misclassifications can be seen below.





As we can see, different angles of the faces can lead to misclassifications. Moreover, different facial expressions might lead to a difficulty of identifying the right person. So, we decided to add horizontally flipped images to the train data (those images taken only from the train set). The added data is now all the pairs of the training data with only the first picture horizontally flipped. In this way we multiplied the amount of the training data samples.

- examples of accurate classification:



We can see that accurate classifications are mainly in pairs that have the same face angles, colors, and face expressions.