**Report**

Assignment 2, Deep Learning

by Carmel Shablin 305812661

and Gregory Koushnir 321889479

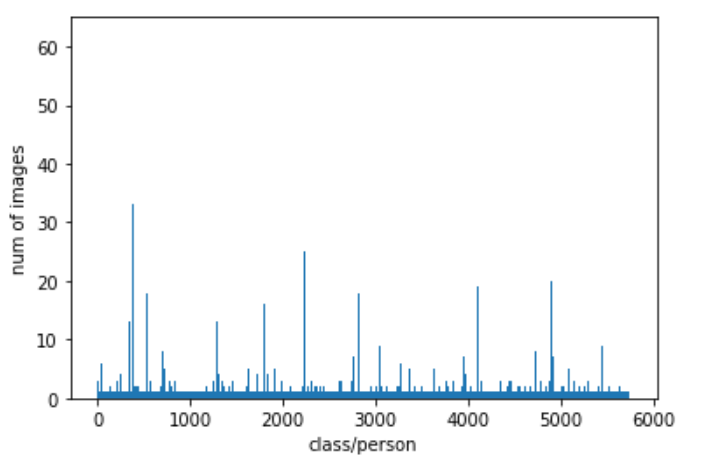
## Task

Build Siamese convolutional neural network for one-shot image recognition, given two facial images of previously unseen persons, the model has to successfully determine whether they are the same person.

## Dataset

The Labeled Faces in the Wild-a (LWF) dataset was used for the task (can be found [here](https://talhassner.github.io/home/projects/lfwa/index.html)). This dataset contains gray-scaled 250x250 images of human faces which were aligned using a commercial face alignment software. The data set was splitted by given distribution for [Train](http://vis-www.cs.umass.edu/lfw/pairsDevTrain.txt) \ [Test](http://vis-www.cs.umass.edu/lfw/pairsDevTest.txt) sets. This division is set up so that no subject from test set is included in the train set. When the images are loaded, input features (pixels) values are divided by 255 to normalize in range [0,1].

Data Set distribution:



From the above figure (X values represent different persons) can be seen that most classes has only one image, all train/test samples are taken from classes that have more than 1 image – the reason is because if we use an image from label that has only one image it will not help the model to learn from this specific image in case of similarity task.

Test Data set:

Contains 500 sample of ‘same’ pairs

Contains 500 sample of ‘different’ pairs

Train Data set:

Contains 1100 sample of ‘same’ pairs

Contains 1100 sample of ‘different’ pairs

The new training set and the validation set are generated by the split of the original training set to 80% and 20% respectively.

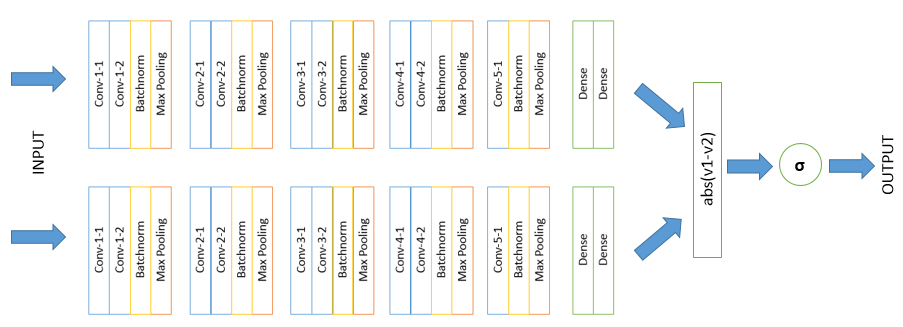
## Method

The general method is to build a Siamese Network to decide whether two given images are of the same or different persons.

The data is loaded with *load\_data\_2* function and *then prepare\_x\_y\_according\_to\_description* function is used to load, arrange and shuffle the data in a list of pairs of images attaching the label (1 for same person and 0 for different persons).

## The NN Architecture

The following architecture of the network is used for the classification task:



Every convolution layer is of size (3, 3), ReLU activation and stride of 1. Following additional parameters were used for convolution layers:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Number of filters | Padding | Initializer |
| Conv-1-1 | 32 | Same | Xavier normal |
| Conv-1-2 | 32 | Same | Xavier normal |
| Conv-2-1 | 64 | Valid | Xavier normal |
| Conv-2-2 | 64 | Valid | Xavier normal |
| Conv-3-1 | 128 | Valid | Xavier normal |
| Conv-3-2 | 128 | Valid | Xavier normal |
| Conv-4-1 | 256 | Valid | Normal distribution () |
| Conv-4-2 | 256 | Valid | Normal distribution () |
| Conv-5-1 | 512 | Valid | Normal distribution () |

Every max-pool layer has shape of (2, 2) and stride of 2.

Two dense layers have 1024 units each and their weights are initialized with the Normal distribution (). The first Dense layer is activated with ReLU and the last is activated with Sigmoid.

After the completion of the pass in twin-network by both images simultaneously, the resulting embeddings of each image are subtracted (neglecting the sign) and the resulting layer (distance vector) fully connected to the final sigmoid neuron layer, which classifies the similarity of the images.

In total, there are 16,512,673 trainable weights.

## Incremental steps behind the final architecture

After implementing the architecture similar to the [given paper](https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf), the accuracy results were not stunning, reaching about 0.6. It was decided to change the CNN architecture. The final architecture inspired by VGG Net since ImageNet images are with similar size to ours (224^2) and our images are also in the same domain (faces). Also, the VGG is known for its ability to decrease the dimensionality which leads to weights size reduction, and thus a quicker training. Main idea taken form the VGG is to decrease the size of every feature map and to increase the depth (# maps) on ever layer. Progressing this concept showed good results.

Multiple changes were made to the original VGG architecture, including the addition of batch normalization, changes in padding and initializers, addition of another convolution layer, change of last dense activation function.

Foreach twin’s neural net used final activations as Sigmoid as described in Paper, all other activations are RELU similar to VGG. By adding Batch Norm in every layer, the model started to learn before that the Validation loss almost did not improved it was zero.

Initially started with normal distribution of weights initialization N(0,0.01) similar to described in the paper. That lead to accuracy on Validation set of about 0.625. Then, change the distribution to Xavier for kernel weight initialization for the first few layers and improved marginally on first epochs, also the convergence was faster (fewer epochs). Bias weight were nullified on initialization.

After some trials, found that adding learning regularization of L = 0.95\*L after every epoch boosted a bit the accuracy and lowered the loss.

Adding last CONV layer (depth of 512) boosted by another 2% accuracy. Additional convolution layers did not help.

Firstly, we declared all CONV layers padding as ‘same’ inspired by VGG. Then tried doing without padding. Experimentally, keeping “same” padding only on the first two CONV layers showed the greatest improvement.

Nevertheless, many techniques were tried as described in the given paper, including but not limited by dropout and L2 layer-wise regularization. That did not bring a better result.

Experimental setup

Since none of the students has or has an access to GPU physically, the assignment was done in the Google Colab environment in order to speed up the training process. All of the shown results are taken from the Jupyter notebook.

The training performed with the following parameters: batch size = 8, validation split = 0.2, number of epochs = 20. Three callback functions were used:

* Early stopping for validation accuracy to interrupt training if there are no improvement for 3 consecutive epochs
* Learning rate scheduler, which decreases the learning rate each epoch by 5%
* Model checkpoint, to save the weights of model with highest validation accuracy

The Cross-entropy loss will calculate a score that summarizes the average difference between the actual and predicted probability for class 1, that is what we are interested in ‘binary-crossentropy’. The optimizer that was used is Adam - a good default optimizer with starting learning rate of 1e-4 . The metric we are interested in is accuracy for the verification task as described in the paper (not the one-shot task).

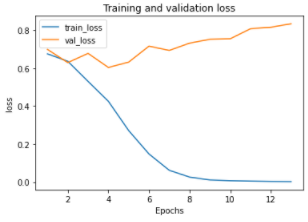
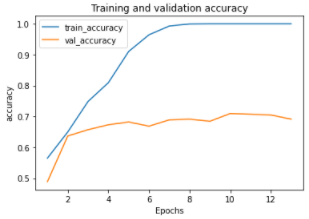
After fine-tune of the model for the best accuracy on validation set, it is loaded and retrained on the full dataset (including the validation set) for three epochs with the batch size of 8. We are retraining the model on full train and validation sets to increase the training size on all the images we have (1100 in total).

After the training, the model is evaluated on a test set.

## Results

With the described architecture and method, the best epoch accuracy on a validation set is 0.71

Graphs that describe the accuracy and loss through the training process.

The training takes ~24 seconds per epoch. In total, we can see that the early stop worked after 13 epochs (313 seconds). Retraining for 3 more epochs takes another 87 seconds. After that the test set accuracy is also **0.71**.

Example of correctly classified “same” person:

TODO print Loss giving indication on the goodness of fit better use low loss

loss: 0.0203



Example of incorrectly classified “same” person:

TODO print Loss giving indication on the badness of fit better use high loss from the options

loss: 0.0203



The haircut, face exposure and face details quality are very different.

Example of correctly classified “different” person:

TODO print Loss giving indication on the goodness of fit better use low loss

loss: 0.0203



Example of incorrectly classified “different” person:

TODO print Loss giving indication on the badness of fit better use high loss from the options

loss: 0.0203

