Liat Cohen- 205595283
Amit Shakarchy- 313278889

DEEP LEARNING

ASSIGNMENT 2

DATA EXPLORATION

In this section, we will perform an analysis of the given dataset.

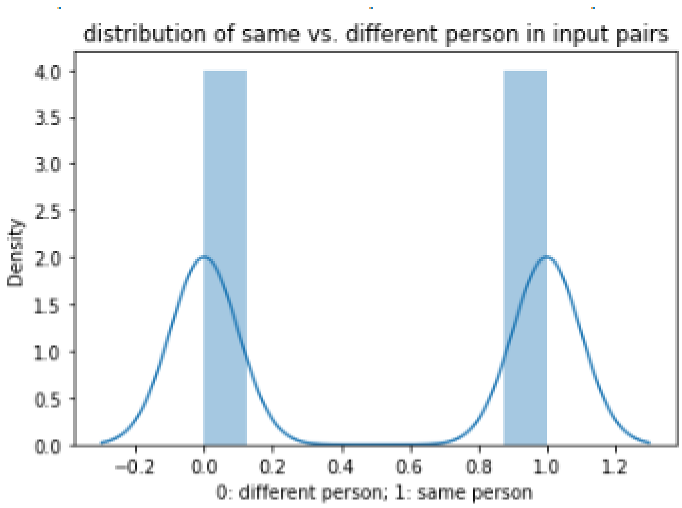
General information:

number of classes (number of different people)	5749
number of total examples	13233
average number of images per a person	2.302 +- 9.016
Number of examples (pairs)- training set	2200
Number of examples (pairs)- test set	1000

Distribution of number of images per a person:

As we can see, the majority of the individuals in the dataset have only one photograph, and only a handful have several photos.

Looking at the distribution on the input image pairs, we can see that our dataset is balanced - the number of image pairs of the same person is equal to the number of pairs with different people. The graph below shows that distribution (0 for different people, 1 for the same person):



Let’s sample a few pairs, of the different and same person:

EXPERIMENTAL SETUP

INITIAL SETUP:

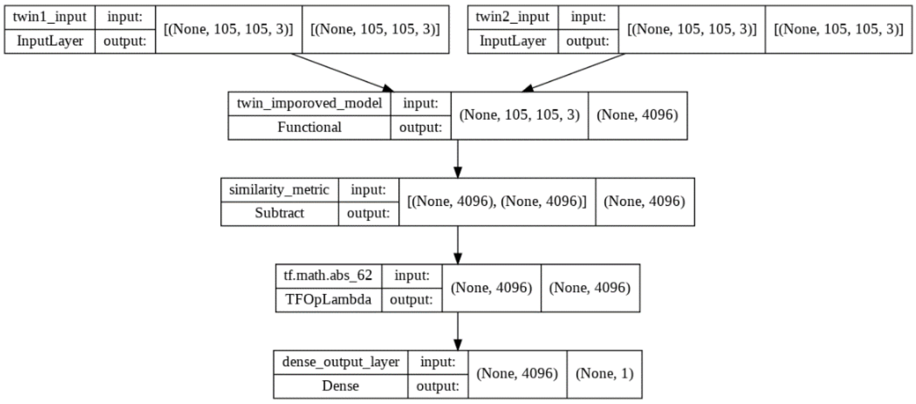
Data:

- We divided the training subset into train & validation sets, where the validation set’s size is 20% of the original training subset. We used a stratified split, to reserve the balance of image pairs with the same/different people.
- We resized the input images to a size of 105X105X3, to match the paper’s architecture.

Network architecture:

We stuck to the architecture that was described in the paper, creating two identical convolutional networks, connected by a similarity metric layer that subtracts one twin output vector from the other to determine whether the inputs belong to the same person or not. On the left image is a description of a “twin” network; on the right is the Siamese network.

```
Model: "twin_model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_6 (InputLayer)        [(None, 105, 105, 3)]     0

 twin_conv1 (Conv2D)         (None, 96, 96, 64)        19264

 twin_max_pooling1 (MaxPooli  (None, 48, 48, 64)       0
 ng2D)

 twin_conv2 (Conv2D)         (None, 42, 42, 128)       401536

 twin_max_pooling2 (MaxPooli  (None, 21, 21, 128)      0
 ng2D)

 twin_conv3 (Conv2D)         (None, 18, 18, 128)       262272

 twin_max_pooling3 (MaxPooli  (None, 9, 9, 128)        0
 ng2D)

 twin_conv4 (Conv2D)         (None, 6, 6, 256)         524544

 twin_flatten (Flatten)      (None, 9216)              0

 twin_dense (Dense)          (None, 4096)              37752832

=================================================================
Total params: 38,960,448
Trainable params: 38,960,448
Non-trainable params: 0
_________________________________________________________________
```

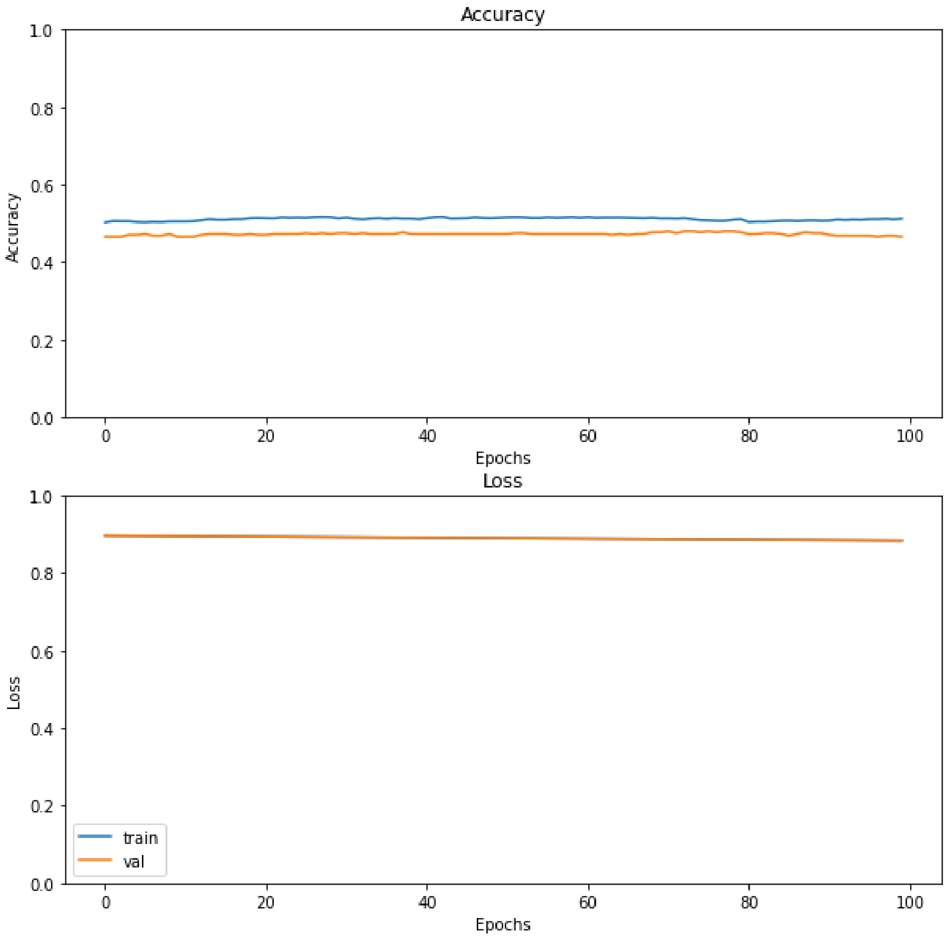


Additional parameters:

- Batch size - 128 as mentioned in the paper.
- We performed L1 distance as a similarity metric, as presented in the paper.
- We used binary cross-entropy loss to comply with the paper.
- We used L2 regularization. We used a regularization factor of 0.1, to comply with the paper.
- Epochs- for the initial setup, we ran our network through 100 epochs.
- Weights initialization- We initialized the network weights and biases in the convolutional layers from a normal distribution with zero mean and a standard deviation of 10^{-2} for the weights and mean 0.5 and standard deviation of 10^{-2} – as described in the paper.
- Optimizer- We used SGD optimizer with a 0.01 learning rate as described in the paper.

Performance:

run time in seconds	503.148
train accuracy	0.512
train loss	0.883
validation accuracy	0.467
validation loss	0.883
test accuracy	0.5
test loss	0.883



The initial performance of the described architecture is quite poor. The model does not converge at all and predicts randomly whether the input images are of the same person or not. We thought that the initialization of the weights and biases, in addition to the low learning rate, lead to those bad results. Now we will explore some of the model's classifications:

We can see that the model isn't confident with its predictions, and all predictions are made randomly.

IMPROVED SETUP:

We modified the weights initialization of the model, and then we had to cope with heavy overfitting of the model. We made a few modifications to reduce overfitting:

- Dropout layers- we added dropout layers after every convolutional layer in the network's architecture.
- Batch normalization- we added Batch normalization layers after every max-pooling layer for regularization.
- Data augmentation- we increased the size of our model by 10%. For each pair of the randomly selected pairs, we flipped one of the images.

The modified "twin" network:

```
Model: "twin_imporoved_model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_10 (InputLayer)       [(None, 105, 105, 3)]     0

 imporoved_conv1 (Conv2D)    (None, 96, 96, 64)        19264

 batch_normalization_15 (Bat  (None, 96, 96, 64)       256
 chNormalization)

 imporoved_max_pooling1 (Max  (None, 48, 48, 64)       0
 Pooling2D)

 dropout_18 (Dropout)        (None, 48, 48, 64)        0

 imporoved_conv2 (Conv2D)    (None, 42, 42, 128)       401536

 batch_normalization_16 (Bat  (None, 42, 42, 128)      512
 chNormalization)

 imporoved_max_pooling2 (Max  (None, 21, 21, 128)      0
 Pooling2D)

 dropout_19 (Dropout)        (None, 21, 21, 128)       0

 imporoved_conv3 (Conv2D)    (None, 18, 18, 128)       262272

 batch_normalization_17 (Bat  (None, 18, 18, 128)      512
 chNormalization)

 imporoved_max_pooling3 (Max  (None, 9, 9, 128)        0
 Pooling2D)

 dropout_20 (Dropout)        (None, 9, 9, 128)         0

 imporoved_conv4 (Conv2D)    (None, 6, 6, 256)         524544

 imporoved_flatten (Flatten)  (None, 9216)             0

 imporoved_dense (Dense)     (None, 4096)              37752832

=================================================================
Total params: 38,961,728
Trainable params: 38,961,088
Non-trainable params: 640
_________________________________________________________________
```

In the next section, we will show hyperparameter optimization to select the best-performing model.

ARCHITECTURE'S PERFORMANCE – EXPERIMENTS

We have trained our model using combinations of several hyperparameters, to find the parameters that yield the best performance on the validation set.
We have evaluated the following parameters:

Parameter	Evaluated values
Learning rate	[0.0001, 0.001, 0.01]
dropout	[0.2, 0.5, 0.8]
Batch size	[64, 128, 256]
L2 regularization factor	[0.1, 0.01]

Note: We used 40 epochs for each training combination, to compare results more easily and due to long-running times.

	Parameters				Results				
run #	**learning rate**	**dropout rate**	**batch size**	**l2**	**test loss**	**test accuracy**	**val loss**	**val accuracy**	**run time**
1	0.0001	0.2	64	0.1	0.877	0.541	0.879	0.534	217.743
2	0.0001	0.2	64	0.01	0.710	0.515	0.708	0.532	202.648
3	0.0001	0.2	128	0.1	0.893	0.488	0.886	0.523	203.732
4	0.0001	0.2	128	0.01	0.716	0.499	0.714	0.482	192.236
5	0.0001	0.2	256	0.1	0.893	0.505	0.892	0.509	263.488
6	0.0001	0.2	256	0.01	0.717	0.498	0.714	0.482	203.455
7	0.0001	0.5	64	0.1	0.882	0.502	0.883	0.534	203.599
8	0.0001	0.5	64	0.01	0.712	0.516	0.713	0.486	204.458
9	0.0001	0.5	128	0.1	0.886	0.532	0.889	0.502	192.885
10	0.0001	0.5	128	0.01	0.713	0.506	0.713	0.514	204.296
11	0.0001	0.5	256	0.1	0.897	0.483	0.896	0.468	203.486
12	0.0001	0.5	256	0.01	0.711	0.510	0.709	0.536	188.679
13	0.0001	0.8	64	0.1	0.881	0.516	0.881	0.464	202.656
14	0.0001	0.8	64	0.01	0.713	0.515	0.714	0.475	203.293
15	0.0001	0.8	128	0.1	0.886	0.526	0.886	0.516	191.720
16	0.0001	0.8	128	0.01	0.713	0.503	0.714	0.459	192.286
17	0.0001	0.8	256	0.1	0.889	0.512	0.889	0.470	188.080

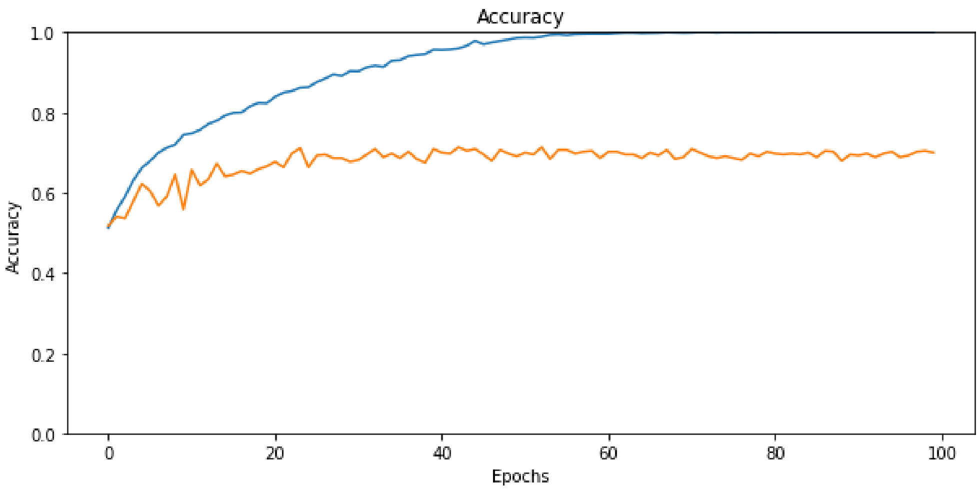
18	0.0001	0.8	256	0.01	0.712	0.494	0.713	0.473	188.889
19	0.001	0.2	64	0.1	0.795	0.588	0.798	0.564	263.497
20	0.001	0.2	64	0.01	0.686	0.584	0.682	0.600	263.495
21	0.001	0.2	128	0.1	0.844	0.568	0.848	0.525	193.039
22	0.001	0.2	128	0.01	0.699	0.549	0.700	0.543	193.651
23	0.001	0.2	256	0.1	0.867	0.531	0.867	0.545	203.510
24	0.001	0.2	256	0.01	0.707	0.537	0.707	0.523	203.912
25	0.001	0.5	64	0.1	0.814	0.567	0.816	0.532	263.459
26	0.001	0.5	64	0.01	0.709	0.528	0.713	0.516	263.568
27	0.001	0.5	128	0.1	0.855	0.497	0.851	0.557	203.807
28	0.001	0.5	128	0.01	0.710	0.528	0.712	0.523	204.600
29	0.001	0.5	256	0.1	0.872	0.494	0.872	0.489	190.047
30	0.001	0.5	256	0.01	0.712	0.499	0.712	0.477	189.369
31	0.001	0.8	64	0.1	0.818	0.500	0.818	0.500	263.541
32	0.001	0.8	64	0.01	0.712	0.483	0.712	0.475	204.746
33	0.001	0.8	128	0.1	0.850	0.523	0.850	0.498	203.516
34	0.001	0.8	128	0.01	0.713	0.490	0.713	0.470	204.345
35	0.001	0.8	256	0.1	0.870	0.513	0.872	0.459	189.223
36	0.001	0.8	256	0.01	0.713	0.485	0.714	0.477	189.179
37	0.01	0.2	64	0.1	0.633	0.677	0.640	0.657	204.260
38	0.01	0.2	64	0.01	0.628	0.687	0.643	0.693	204.582
39	0.01	0.2	128	0.1	0.651	0.674	0.659	0.652	203.492
40	0.01	0.2	128	0.01	0.617	0.682	0.637	0.666	204.401
41	0.01	0.2	256	0.1	0.720	0.633	0.725	0.598	189.972
42	0.01	0.2	256	0.01	0.649	0.625	0.664	0.611	203.526
43	0.01	0.5	64	0.1	0.665	0.624	0.680	0.584	263.552
44	0.01	0.5	64	0.01	0.659	0.612	0.701	0.564	205.201

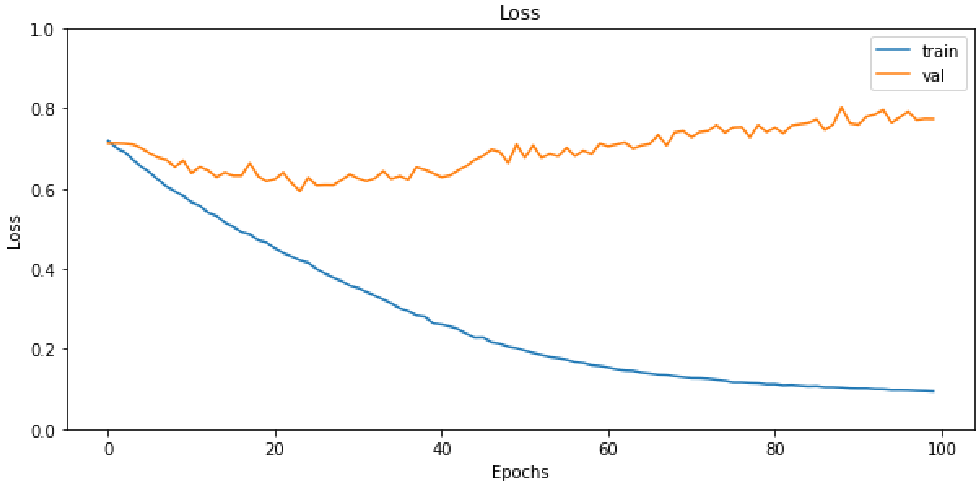
45	0.01	0.5	128	0.1	0.703	0.539	0.706	0.532	203.768
46	0.01	0.5	128	0.01	0.680	0.567	0.694	0.527	204.316
47	0.01	0.5	256	0.1	0.752	0.542	0.754	0.509	190.387
48	0.01	0.5	256	0.01	0.699	0.526	0.702	0.514	189.271
49	0.01	0.8	64	0.1	0.701	0.500	0.700	0.500	263.523
50	0.01	0.8	64	0.01	0.703	0.528	0.700	0.566	263.547
51	0.01	0.8	128	0.1	0.716	0.510	0.715	0.511	203.801
52	0.01	0.8	128	0.01	0.707	0.551	0.707	0.518	204.814
53	0.01	0.8	256	0.1	0.755	0.504	0.755	0.491	203.498
54	0.01	0.8	256	0.01	0.709	0.510	0.709	0.518	208.100

Despite all our efforts to avoid overfitting, even the best-performing model was heavily overfitted.

The best performing model's hyperparameters (also marked in green in the table above):

	Parameters				Results				
run #	**learning rate**	**dropout rate**	**batch size**	**l2**	**test loss**	**test accuracy**	**val loss**	**val accuracy**	**run time**
38	0.01	0.2	64	0.01	0.628	0.687	0.643	0.693	204.582





As you can see, even the best model is suffering from overfitting.

Let’s explore some examples of accurate/ misclassifications of the model:

If we look closely at the incorrect classifications, it appears that the model may use glasses as a separating feature. On the left misclassified pair of images, we can see images of the same person with/without glasses. On the right misclassified pair of images, we can see images of different Caucasian people wearing glasses- and the model thinks they are the same person. We assume that adding more examples of people with glasses, should help.

SUMMARY

In this assignment, we performed a one-shot learning task for previously unseen objects. We used the architecture described in the given paper, and then tried to improve it and its performance. We coped with overfitting and convergence disability using several regularization techniques and learned to analyse the model's misclassifications. If we had more time, we would:

- Try different architectures to improve the performance.
- Augmenting the data using rotations.
- Try different techniques for reducing overfitting (less complicated model, L2 regularization in more layers, use different optimizer, etc.).