IDs:
ID #1: 300822954
ID #2: 307963538

Deep Learning

Assignment 2 – Siamese Neural Networks and one-shot learning

As instructed, our implementation is based on the paper "Siamese Neural Networks for One-shot Image Recognition". In this assignment, we extend and explored various architectures, methods (such as transfer learning), etc. to train a Siamese NN model based on Labeled Faces in the Wild dataset that able to perform face image verification task i.e., given two faces' images decides if those two belong to the same person.
The general idea is that the network learns to naturally rank similarity between faces images inputs, aka, similarity score, thus, later on, the network can generalize the prediction and employ it in a zero/one-shot learning fashion.
The hypothesis for Siamese NN is that if two input faces images belong to the same person, then their feature vectors must also be similar, while if the two input images belong to different persons, then their feature vectors will also be different. Hence, the element-wise absolute/euclidean-distance difference between the two feature vectors must be very similar/different and the similarity score generated by the output sigmoid layer must also be similar/different for the above cases in which this type of learning method is different than learning to classify an image directly to any of the output possible persons.

Moreover, multiple experiments have been conducted such as examine various network architectures, different learning rates, different dense layer sizes, augmented the dataset by applying transformations, applying regularization methods, etc. By doing so, we present below and explain the effect and sensitivity of the model's performance caused by varying values of the hyper-parameters, different architectures, and methods and show the effect on the training process and performance.

Experiments setup:

Pre-processing and dataset pipeline steps:

Note:

a. Our logic for the pre-processing and dataset pipeline is located under preprocessing_utils.py script.
b. We decided to use a validation set by splitting the given 2200 training pairs with validation_size=0.2 which represents the proportion of the dataset to include in the validation split. It is important to note that although the given training dataset is relatively small and our models overfitted, we decided to mitigate this issue by other methods such as data augmentation and regularization methods. Moreover, we tried to train a model with the whole given training dataset which still produces an overfitted model.

Steps:

1. LFWA dataset was downloaded, which contains directories with face images per person. Also, includes 'pairsDevTrain.txt' and 'pairsDevTest.txt' files that determine the training/test datasets. These files hold pairs of matching/non-matching images.
2. Parse each file and produce a relevant data structure that holds the matching and the non-matching records.
3. Split the given training pairs into train/validation datasets – the split was performed on the matching and non-matching pairs separately, to ensure each set contains samples of both types of pairs, and then the validation/training set of the non-matching pairs was united with the appropriate matching pairs set (validation-matching with validation-non-matching and similarily with the training sets of pairs).
4. Convert the current data structures that hold images path into tensors with the content of the images by decoding, apply image resize if configured, and normalization.
5. We decided to work with Tensorflow Dataset API, thus, converted the above pair's images data structure into training/validation/test tf.data.Dataset and configured it by applying if necessary per dataset type: cache, shuffle, batch and prefetch.

Training strategy:
The models were obtained by a training procedure which was bounded by 50 epochs and early stopping that stop training when the binary accuracy on the validation has stopped improving with patience=15 and min_delta=0.03. Also 'restore_best_weights' flag was set to True.

To monitor performance during training, we used the binary accuracy metric, namely, verification on the validation set pairs generated as depicted above. We could have chosen to apply n-way one-shot learning evaluation, however, in the paper they stated that both strategies yield similar results, thus for ease of implementation, we've decided to configure our optimizer and early stopping based on validation error for the verification task.

However, we also provide details with respect to an n-way one-shot learning evaluation on the tuned model evaluated on the validation and test dataset.
For performing n-way one-shot learning evaluation we added logic that generates those tests for both validation and test dataset into the above dataset pipeline.
Our logic creates per pair in the dataset under the matching section n-way test, namely, the same image is compared to n different images out of which only one of them matches the original person. Specifically, for each matching pair, we first search for pairs associated with the relevant person in the non-matching section, and in case that no enough n-1 cases exist, we randomly choose non-matching person images from the non-matching section.
In our configuration, we set n=3 and we didn't include it in our hyper-parameter search space. It is important to note that larger values of n will lead to relatively less correct predictions.

All training data was saved into Tensorboard as can be seen below in the empirical results section.

We wrap our experiments with Bayesian optimization to perform hyperparameter selection. We utilize the Keras-Tuner framework and configured it as follows:
Max_trials=for some experiments it was set with 150, for others, 50 was configured
Objective=max 'val_binary_accuracy'
Num_initial_points=3

With hyper-parameters as follows:
'learning_rate': [5e-5, 1e-5, 5e-4, 1e-4, 1e-3, 5e-6]
'dense_layer_size': [1028, 4096, 512]
'enable_batch_normalization': [True, False]
'bias_initializer': ["default", "zeros"]
'conv2D_kernel_initializer': ["default", "he_normal"]
'dense_kernel_initializer': ["default", "he_normal"]
'dropout_rate': [0.0, 0.2, 0.5]
'distance_metric': ["abs", "euclidean_distance"]
'l2_regularizer': [-1.0, 0.01, 0.05, 0.1, 0.5]
'optimizer': ["adam", "sgd", "RMSprop"]

Note: we fixed batch_size=64 for all experiments. We set it to 64 and not to 128 as the author chose in their paper since it worked well and provide good results and mitigate our issues with OOM during training due to the large image dimension and limited hardware.

As can be derived from the above search space, we enabled per experiment settings as demonstrated in the empirical result section below various inner model architectures and regularization methods such as if to apply batch normalization, dropout, l2 regularization, etc.

Also, regarding weight initialization, for "default" mode above, we followed the authors' suggestion in the paper, namely, for the convolutional layers the weights are initialized from a normal distribution with zero-mean and standard deviation of 10^{-2} and for biases, the weights are initialized from a normal distribution with a mean of 0.5 and standard deviation 10^{-2}.
For the full-connected layers, the weights were drawn from a much wider normal distribution with zero-mean and standard deviation of $2\ x\ 10^{-1}$ and the biases were initialized in the same way as the convolutional layers.

We stored the results including the training log, model performance in Tensorboard hparams table, and CSV file. Also, all trials and their final model are saved as well.
Remark: due to file sizes and submission limitations, we couldn't submit all this information, thus, only the log files of the best model found In our trials was submitted alongside the code and this report.
Link to full logs and information can be sent upon request.

Models' Network Architecture:
As stated above our implementation is based on the paper "Siamese Neural Networks for One-shot Image Recognition".
Regarding the model's architecture we mainly follow their largest network, i.e.:

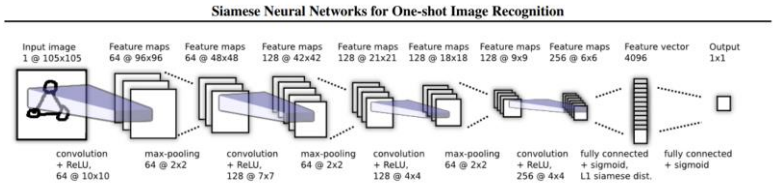


Figure 4. Best convolutional architecture selected for verification task. Siamese twin is not depicted, but joins immediately after the 4096 unit fully-connected layer where the L1 component-wise distance between vectors is computed.

Yet, we added the following:

1. Enable the feature vector size to be a hyperparameter.
2. Enable batch normalization after the convolutional layers controlled via a hyperparameter.
3. Enable dropout regularization after max pooling on each convolutional layer controlled via a hyperparameter.
4. Implemented two types of component-wise distance controlled via a hyperparameter. The first is L1 as they used in the paper and the second is the L2 distance metric.

HyperModel - To work with the Keras-Tuner framework, we created a class that inherits from HyperModel which defines a searchable space of Models and builds Models from this space. Namely, this wrapper class allows us to utilize our Siamese neural network implementation in the Keras-Tuner framework which makes it easy to perform distributed hyperparameter search with little changes to a "regular" TensorFlow model.

Data Augmentation
Although in the paper the authors augmented the training set with small affine distortions we started our experiments without performing any kind of augmentation to the given training set, since those kinds of transformations, in our opinion are less suited for the given dataset.
However, our models suffer from overfitting, namely, training binary accuracy reaches 1 while validation binary accuracy is ~0.7.
Thus, we've decided to augment the training dataset with the following transformations:

1. Salt and pepper noise.
2. Rotation by [45, -45] randomly.
3. Center crop (0.5% center) and resize with padding.
4. Flip left-right.
5. Salt and pepper noise + center crop.

To combine data augmentation as part of our training procedure, we first perform offline one time per image in the training dataset all above transformations and saves the new images into relevant directories. Then, as part of our dataset pipeline, we added all the above augmented images to be part of the training dataset only, excluding validation/test datasets.

Examples:

Adam_Sandler_0002.jpg:

Original | center crop | flip left-right | noise

Rotation-45 | noise + center crop

Note: our logic for the data augmentation is located under the data_augmentation.py script

Section 2C – EDA:

We started our EDA by verifying that no subject (which we didn't know whether it refers to pairs or individuals) is shared between the training and testing sets, as stated in the instructions. This was verified to be true for individuals and, as an outcome, for pairs as well. From visually inspecting the image files, we also noticed that aside from centering and cropping the images to contain the individuals' faces, some images were also cropped and rescaled, adding a completely black background in the edges of the frame.

To assess the scale of the provided image set, we checked how many image files each individual has in the image ('lfw2Data') directory supplied and plotted the histogram of the individual counts of image files (Fig. 1). We discovered that the majority of individuals have only a single image file, while some individuals have hundreds of image files, with the maximum amount saved under 'George_W_Bush' name, with 530 images.

The individuals appearing in the image files themselves were also of interest: after browsing through hundreds of images (through the process of performing the assignment) we noticed that most, if not all, individuals were of a narrow demographic spectrum, no images of children and very old people seem to exist, most images appear to be of white middle-aged males. The database webpage itself seems to agree and even provides a friendly warning that this database does not fit well for the 1:N recognition task.

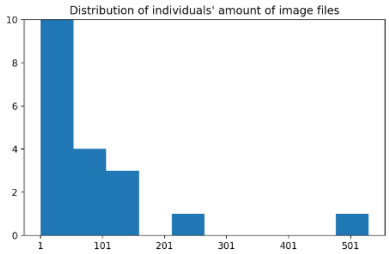


Figure 1:Distribution of Image files count - limit of y-axis is set to 10. Maximum count of image files detected is 530.

We then focused on the training and testing set only, disregarding any finding related to the image files themselves, as we are instructed to train and test our data using only the images addressed by the 'pairDevTrain.txt'& 'pairDevTest.txt'; We found that the individual appearing in the training set and the test set are 'Alec_Baldwin' 'Tang_Jiaxuan', respectively, both with 6 appearances only (both under matching pairs and non-matching pairs). We also plotted a histogram of all individuals' amount of appearances in the training and testing sets (Fig. 2), and although most individuals appeared once, it seems there is some sort of a power-law restricting the amount of

duplicated (or more) individuals appearances in both training and testing sets. In the training set, only 351 individuals appeared in both the matched pairs and non-matched pairs, whilst 138 appeared in both matching and non-matching pairs in the testing set.

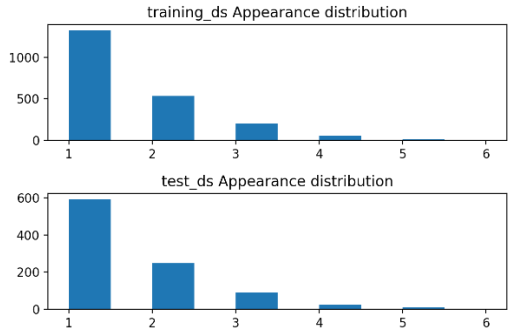


Figure 2 - Distributions of appearances in training and testing sets. The total number of unique individuals in the training set is 2132, the number of unique individuals in the matching pairs of the training set is 788, and in the non-matching pairs 1695, totaling 2132. The total number of unique individuals in the testing set is 963, the number of unique individuals in the matching pairs of the testing set is 353, and in the non-matching pairs 748, totaling 1101.

Checking for shared individuals across matching and non-matching pairs is not sufficient, as an individual might be represented by a different image (notated with the image index) in the matching samples than in the non-matching samples. Hence, we checked whether the set of images used for the matching samples is full/disjoint/contiguous with the set of images used for the non-matching sample with all shared individuals. We found that all the images of individuals appearing in both matching and non-matching pairs are the same in the matching and non-matching samples.

EDA conclusions:

As most individuals appear in either the matching pairs or the non-matching pairs, the task in hand (to classify if two images are of the same individual) is more difficult than we initially thought. During most of the training, our model will learn from each individual only once, either from matching samples or non-matching samples, and will have very few opportunities to learn matching and non-matching samples of a single person. This makes the task of training closer to zero-shot learning. This is emphasized by the fully disjoint train and test sets where no individuals are contiguous about the two sets.

All of the above and the disclaimers mentioned at the beginning of this section, suggest that in “real world” research, where the performance of the model actually matter, one should not use this dataset on its own as it might (among other issues) cause a bias towards specific demographics (e.g. it might always classify two images of children as a matching pair).

Empirical results:

Remarks:

1. We present the 10 best models selected per experiment settings based on ‘val_binary_accuracy’. Then, presents in more detail the 2 best models from across the below experiments.
2. The experiments have been conducted with the configuration and training strategy depicted above.
3. It is important to note that some models converge more slowly compared to others, thus in a different training strategy with different stopping criteria, they might yield better test/validation accuracy scores since might not yet reach a plateau. Hence, it may worth running those configurations again with a different strategy that allows them more iterations for training, however, will require more computation time until convergence.
4. For each model, we’ve conducted a one-shot accuracy test on the final trained model using 3-way tests that we’ve generated based on the validation set and test set. The results are presented in the below tables.

Top 2 – based on test binary accuracy score:

1. From experiment #2:

Trial ID	test loss	test accuracy	test one shot accuracy	val loss	val accuracy	val one shot accuracy	train loss	train accuracy	learning_rate	dense_layer_size	enable_batch_normalization	bias_initializer	conv2D_kernel_initializer	dense_kernel_initializer	dropout_rate	distance_metric	l2_regularizer	optimizer
6a20f23d6202f7e56a2d7856f71ecf97	0.880382	**0.759**	0.782	0.744618	0.786364	0.836364	3.17E-06	1	5.00E-05	1028	1	default	he_normal	he_normal	0	abs	-1	RMSprop

Epochs: 26
Time per epoch: 14s
Trial Complete [00h 06m 22s]

2. From experiment #6:

Trial ID	test loss	test accuracy	test one shot accuracy	val loss	val accuracy	val one shot accuracy	train loss	train accuracy	learning_rate	dense_layer_size	enable_batch_normalization	bias_initializer	conv2D_kernel_initializer	dense_kernel_initializer	dropout_rate	distance_metric	l2_regularizer	optimizer
b33f90162bcacd6036f4ba7afce2a244	0.533389	**0.757**	0.658	0.529366	0.763636	0.681818	0.181441	1	5.00E-05	1028	0	default	default	he_normal	0.5	abs	-1	RMSprop

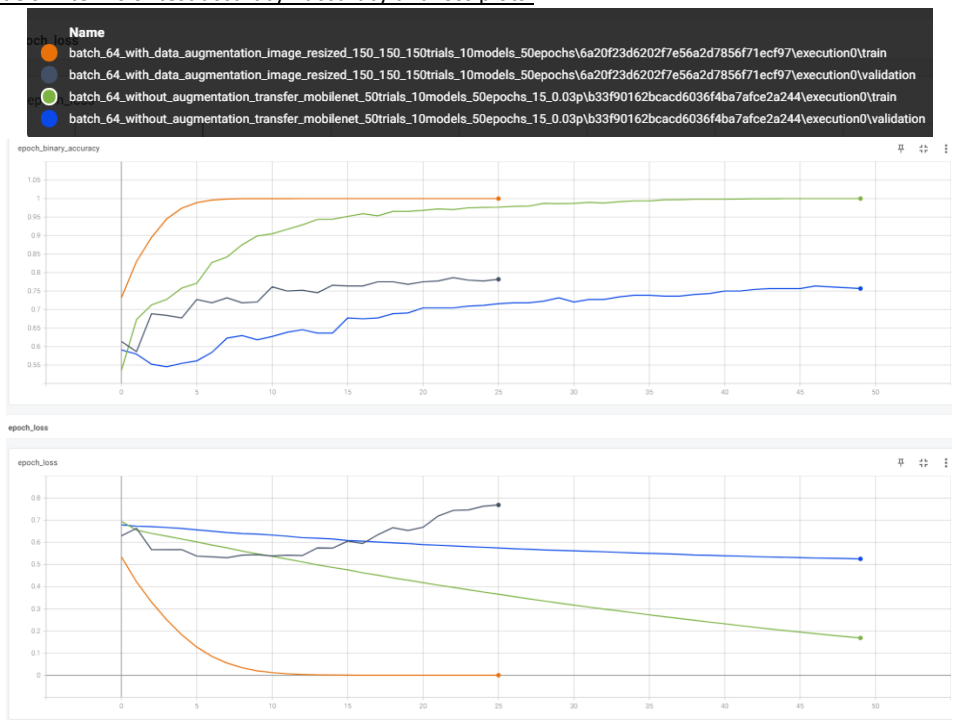
Epochs: 50
Time per epoch: 3sec
Trial Complete [00h 02m 45s]

Top 1 – based on test one-shot accuracy score:

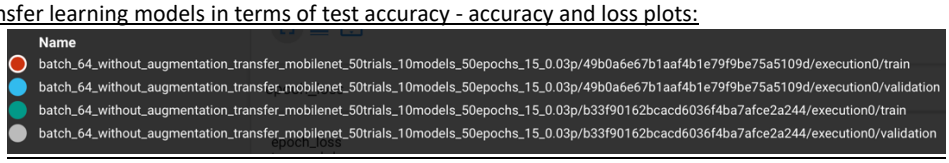
From experiment #2:

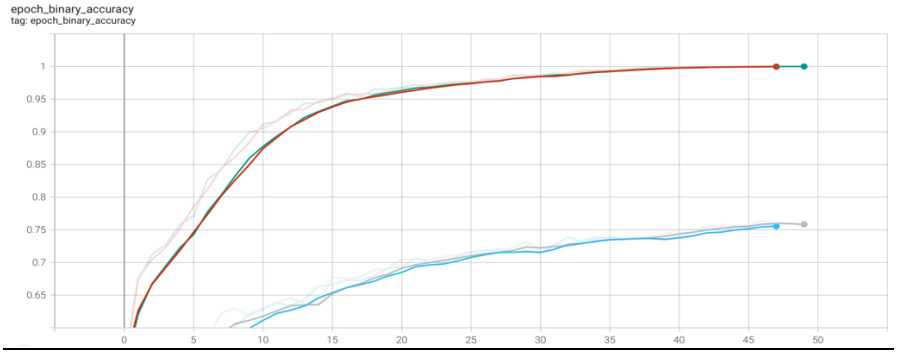
Trial ID	test loss	test accuracy	test one shot accuracy	val loss	val accuracy	val one shot accuracy	train loss	train accuracy	learning_rate	dense_layer_size	enable_batch_normalization	bias_initializer	conv2D_kernel_initializer	dense_kernel_initializer	dropout_rate	distance_metric	l2_regularizer	optimizer
6a20f23d6202f7e56a2d7856f71ecf97	0.880382	0.759	**0.782**	0.744618	0.786364	0.836364	3.17E-06	1	5.00E-05	1028	1	default	he_normal	he_normal	0	abs	-1	RMSprop

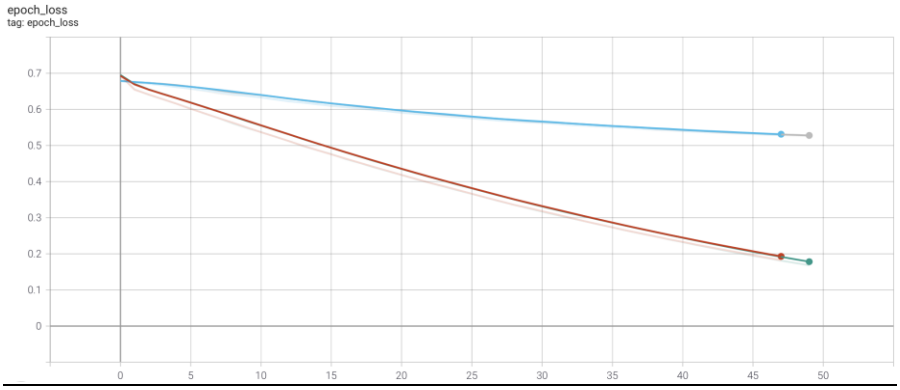
<u>Best 2 models in terms of test accuracy - accuracy and loss plots:</u>



<u>Best 2 transfer learning models in terms of test accuracy - accuracy and loss plots:</u>




epoch_binary_accuracy
tag: epoch_binary_accuracy
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0
5
10
15
20
25
30
35
40
45
50


epoch_loss
tag: epoch_loss
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
5
10
15
20
25
30
35
40
45
50

Examples based on model from trial - 6a20f23d6202f7e56a2d7856f71ecf97:

Incorrect classification:

Incorrect test-case #2 (matching use-case):

Abdullah_Gul 13 16

Prediction:
1.41772225e-01

We think that the model was not able to predict correctly this use-case because of the following:

1. Image 16 is a profile face image while our model mostly learns from images with direct center-oriented faces.
2. 2 other faces appear in image 16 which again, most of the images in the training set mostly contain one face per image.

Incorrect test-case #996 (non-matching use-case):

Susan_Whelan 1 Wolfgang_Schneiderhan 1

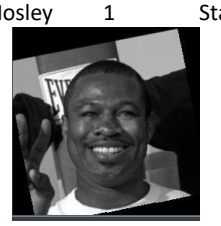
Prediction:
5.73490262e-01

We think that the model was not able to predict correctly this use-case because of the following:

1. The model is not able to distinguish between the short black hair that covers the lady's forehead and the black military barrette that covers the man's forehead and might mistake it for human hair.
2. They look kind of similar to us in terms of their neck is fully covered by cloths, the same skin color, and similar facial features, both individuals' eyes are aimed directly at the camera with their face slightly tilting.

Incorrect test-case #991 (non-matching use-case):

Shane_Mosley 1 Stacey_Dales-Schuman 1



Prediction:
6.91941202e-01

We suspect that the model was not able to predict correctly this use-case because as stated in the EDA section, the training dataset contains a narrow demographic spectrum, namely, not enough images of individuals with dark skin tones. We can still try to speculate that perhaps the identical white color of the top cover (shirt and tanktop) played a role as well, and maybe the tightly packed hair of Stacey Dales in this image looks as short trimmed hair to the model, making her similar to Shane Mosley.

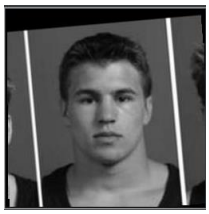
Incorrect test-case #992 (non-matching use-case):
Sheila_Taormina 1 Stephan_Eberharter 1


Prediction:
6.52905405e-01

We think that the model was not able to predict correctly this use-case mainly because of the medal Olympics necklace appearing in both images. The two cap hats in the images, on Stephan Eberharter and the person next to Sheila Taormina although mostly out of frame, might also skew the model towards the wrong classification.

Correct classification:

Correct test-case #998 (non-matching use-case):
Tamara_Mowry 1 Zach_Parise 1

Prediction:
7.70333260e-02

Correct test-case #6 (matching use-case):
Alan_Greenspan 1 5


Prediction:
9.97652113e-01

Experiment #1 – without data augmentation and with image resize to (150, 150, 1)

Image resize:
One of the hyper-parameters we tested for was whether to perform image resizing, where images are scaled to a 150x150 pixels matrix instead of the original 250x250 pixels matrix. The trials in regards to this hyperparameter are aimed to answer two concerns: (1) is the amount of information retained in a reduced size image sufficient for the task at hand, i.e. will the model be able to learn the important facial features that discriminate between individuals or is the information encoding said facial features lost when images are scaled-down in size. (2) Out of memory (OOM) issues; the domain of image processing is very resource-demanding by its nature, so reducing the physical amount of memory each image occupies might accelerate training and testing processes.

In this experiment, we look at the effect of performing image resize as part of the dataset pipeline on model performance/convergence.

Results:

Trial ID	test loss	test accuracy	test one shot accuracy	val loss	val accuracy	val one shot accuracy	train loss	train accuracy	learning_rate	dense_layer_size	enable_batch_normalization	bias_initializer	conv2D_kernel_initializer	dense_kernel_initializer	dropout_rate	distance_metric	l2_regularizer	optimizer
64b4b426981adeb1c606ee1eb8231507	0.565809071	0.713	0.688	0.547670007	0.7409091	0.75	0.002298895	1	5.00E-05	512	1	default	he_normal	he_normal	0	abs	-1	RMSprop
0b7f3631d184aad56a63617d924a0e69	0.577563465	0.712000012	0.712	0.555291951	0.743181825	0.790909091	0.003661512	1	5.00E-05	1028	1	default	he_normal	he_normal	0	abs	-1	RMSprop
2704e613b1f0e4fc0b6b8c9a58782646	0.565164804	0.709999979	0.722	0.558400035	0.743181825	0.727272727	0.039542217	1	5.00E-05	1028	1	default	he_normal	he_normal	0	abs	-1	RMSprop
3e9fe886db9bd378b221ed0f4c16ac0c	0.559289455	0.708000004	0.768	0.542463183	0.752272725	0.727272727	0.026102604	1	5.00E-05	1028	1	default	he_normal	he_normal	0	abs	-1	RMSprop
354ddc0dd3102109955663a6dcc5c71a	0.573514283	0.708000004	0.696	0.557361126	0.7409091	0.754545455	0.012923067	1	5.00E-05	1028	1	default	he_normal	he_normal	0	abs	-1	adam
e5b6ad9c0a39fabd6a2c0ccf69720515	0.582733393	0.708000004	0.72	0.573180497	0.7409091	0.768181818	0.001056077	1	5.00E-05	1028	1	default	he_normal	he_normal	0	abs	-1	RMSprop
b782447081c22ce5ec73d21dcda8818d	0.559720039	0.70599997	0.662	0.548635542	0.738636374	0.709090909	0.028006701	1	5.00E-05	1028	1	default	he_normal	he_normal	0	abs	-1	RMSprop
35ea151c449fffa170d2f1920aeb37ac	0.568757236	0.704999983	0.716	0.560004294	0.7409091	0.731818182	0.021893505	1	5.00E-05	1028	1	default	he_normal	he_normal	0	abs	-1	adam
b1b1583fda89102aac9bbb507909e5e3	0.622828722	0.703000009	0.708	0.620171309	0.74545455	0.763636364	0.000394222	1	5.00E-05	1028	1	default	he_normal	he_normal	0	abs	-1	RMSprop
7b1f650dd2df1fe02fb4b49a49ebbfa5	0.581611931	0.699000001	0.692	0.554746747	0.743181825	0.759090909	0.000855004	1	5.00E-05	512	1	default	he_normal	he_normal	0	abs	-1	RMSprop

Experiment #2 – with data augmentation and with image resize to (150, 150, 1)

In this experiment, we look at the effect of performing image resize and as part of the dataset pipeline and enriching the training set with augmentation on model performance/convergence.

Results:

Trial ID	test loss	test accuracy	test one shot accuracy	val loss	val accuracy	val one shot accuracy	train loss	train accuracy	learning_rate	dense_layer_size	enable_batch_normalization	bias_initializer	conv2D_kernel_initializer	dense_kernel_initializer	dropout_rate	distance_metric	l2_regularizer	optimizer
6a20f23d6202f7e56a2d7856f71ecf97	0.880381584	0.759	0.782	0.744618	0.786364	0.83636364	3.17E-06	1	5.00E-05	1028	1	default	he_normal	he_normal	0	abs	-1	RMSprop
1f79ff5cbb9a9c646adfed705e106641	0.717002451	0.751	0.772	0.667958	0.779545	0.80909091	6.90E-05	1	5.00E-05	1028	1	default	he_normal	he_normal	0	abs	-1	RMSprop
012f2baf0c4dc21d396b8a7c46064be7	0.555982351	0.743	0.728	0.530018	0.775	0.81363636	0.010552	1	5.00E-05	1028	1	default	he_normal	he_normal	0	abs	-1	adam
5989e37215ec56a59e8e8b168dd6216e	0.857138276	0.742	0.772	0.711063	0.775	0.84545455	5.62E-06	1	5.00E-05	1028	1	default	he_normal	he_normal	0	abs	-1	RMSprop
3770b7258945b190d280a7721ba997a2	1.042405725	0.742	0.738	0.864917	0.775	0.81363636	3.08E-07	1	5.00E-05	1028	1	default	he_normal	he_normal	0	abs	-1	RMSprop
309a466ab02ae6082fe8fd9576d6f616	0.561938882	0.736	0.774	0.528978	0.788636	0.81363636	0.010498	1	5.00E-05	1028	1	default	he_normal	he_normal	0	abs	-1	adam
404337e93a96b25be2c125152ae6c169	0.54262048	0.734	0.766	0.518614	0.768182	0.82272727	0.030472	1	5.00E-05	1028	1	default	he_normal	he_normal	0	abs	-1	adam
882f2f4678690d51879f5128a5de7677	0.534906745	0.733	0.766	0.50628	0.775	0.77272727	0.059437	0.999905	5.00E-05	1028	1	default	he_normal	he_normal	0	abs	-1	adam
b919c79e718db8a3b7cde819e87bc725	0.557698727	0.731	0.752	0.50048	0.781818	0.85	0.019645	1	5.00E-05	1028	1	default	he_normal	he_normal	0	abs	-1	RMSprop
cd676cf33ca6b69349486b6066deb973	0.56139344	0.728	0.736	0.525398	0.754545	0.8	0.021727	1	5.00E-05	1028	1	default	he_normal	he_normal	0	abs	-1	RMSprop

Experiment #3 – without data augmentation and without image resize

In this experiment, we look at the effect of not performing image resize and as part of the dataset pipeline and not enriching the training set with augmentation on model performance/convergence.

Results:

Trial ID	test loss	test accuracy	test one shot accuracy	val loss	val accuracy	val one shot accuracy	train loss	train accuracy	learning_rate	dense_layer_size	enable_batch_normalization	bias_initializer	conv2D_kernel_initializer	dense_kernel_initializer	dropout_rate	distance_metric	l2_regularizer	optimizer
3616ecf6ed2be0d685a34268eebb27d0	0.591839	0.686	0.672	0.554336	0.731818	0.690909	0.00413	1	5.00E-05	1028	1	default	default	he_normal	0	abs	-1	RMSprop
1660ee5ea70fa47f8fbbc6513535b1a5	0.614451	0.668	0.616	0.553238	0.736364	0.763636	0.00011	1	5.00E-05	1028	1	zeros	default	he_normal	0	abs	-1	RMSprop
d0a82d6ce081284e286f0f3fed28af06	0.608062	0.664	0.572	0.557606	0.734091	0.759091	0.000104	1	5.00E-05	512	1	zeros	default	he_normal	0	abs	-1	RMSprop
5dc98608a72eb651d1c0e41696b6293f	0.604309	0.662	0.6	0.554757	0.736364	0.722727	0.000853	1	5.00E-05	512	1	zeros	default	he_normal	0	abs	-1	RMSprop
de4a18191c7ee8b54939e6bbcb47890c	0.631655	0.661	0.562	0.56399	0.734091	0.763636	7.63E-06	1	5.00E-05	1028	1	zeros	default	he_normal	0	abs	-1	RMSprop
f27c4fb0258fe1d2c4fb66a3d9e4dee0	0.607397	0.66	0.65	0.561871	0.734091	0.727273	0.000882	1	5.00E-05	512	1	zeros	default	he_normal	0	abs	-1	RMSprop
db27de45e802a2c46569229e18b5136e	0.621617	0.656	0.568	0.557005	0.738636	0.759091	9.63E-05	1	5.00E-05	1028	1	zeros	default	he_normal	0	abs	-1	RMSprop
606c37d03057a631ff3546bbc5f21fcb	0.608034	0.654	0.558	0.55522	0.729545	0.754545	0.001068	1	5.00E-05	1028	1	zeros	default	he_normal	0	abs	-1	RMSprop
5170bd43f4ecc297547eb98a1375ce38	0.633268	0.647	0.594	0.566091	0.736364	0.731818	0.000168	1	5.00E-05	512	1	zeros	default	he_normal	0	abs	-1	RMSprop
7665c90d9d58f6d71da2a3fa7fbb17b3	0.616196	0.642	0.55	0.557071	0.745455	0.75	0.000114	1	5.00E-05	512	1	zeros	default	he_normal	0	abs	-1	RMSprop

Experiment #4 – with data augmentation and without image resize

In this experiment, we look at the effect of not performing image resize and as part of the dataset pipeline and enriching the training set with augmentation on model performance/convergence.

Results:

Trial ID	test loss	test accuracy	test one shot accuracy	val loss	val accuracy	val one shot accuracy	train loss	train accuracy	learning_rate	dense_layer_size	enable_batch_normalization	bias_initializer	conv2D_kernel_initializer	dense_kernel_initializer	dropout_rate	distance_metric	l2_regularizer	optimizer
dbaab2f450d0aec751d9a6da0a93932d	0.63159	0.726	0.728	0.585848	0.752273	0.786364	0.000143	1	5.00E-05	512	1	zeros	he_normal	he_normal	0	abs	-1	RMSprop
32fa4f466454070782 4e89ff3d281438	0.688504	0.717	0.706	0.639174	0.75	0.786364	0.000333	1	5.00E-05	1028	0	zeros	he_normal	he_normal	0	abs	-1	RMSprop
f00b3c2d2416cb59592f41c171eff9a8	0.688609	0.716	0.708	0.632288	0.752273	0.804545	0.000684	1	5.00E-05	1028	0	zeros	he_normal	he_normal	0	abs	-1	RMSprop
fa787b133f7fe34e95e0299bcbe226f3	0.675786	0.715	0.742	0.599807	0.752273	0.768182	0.000769	1	5.00E-05	512	0	zeros	he_normal	he_normal	0	abs	-1	RMSprop
3fb11a151540d49643345b93d2b41663	0.831978	0.714	0.718	0.722848	0.768182	0.840909	2.49E-06	1	5.00E-05	1028	0	zeros	he_normal	he_normal	0	abs	-1	RMSprop
e871013e84b03ca111ad0542503e5a80	0.679345	0.713	0.698	0.622057	0.756818	0.804545	3.88E-05	1	5.00E-05	512	1	zeros	he_normal	he_normal	0	abs	-1	RMSprop
16b9b46bb4671e5ddeb6314398bc1197	0.794483	0.71	0.708	0.657504	0.759091	0.804545	1.75E-05	1	5.00E-05	1028	0	zeros	he_normal	he_normal	0	abs	-1	RMSprop
592ef4cb40e927bf39305db52c827cd3	0.74177	0.708	0.676	0.688438	0.754545	0.813636	0.000155	1	5.00E-05	1028	0	zeros	he_normal	he_normal	0	abs	-1	RMSprop
bf2f9f5dd9f56f3d5f616182ebca588b	0.563125	0.701	0.676	0.546771	0.729545	0.7	0.148099	0.973201	5.00E-05	1028	0	zeros	he_normal	he_normal	0	abs	-1	RMSprop
e9a54d5bbfc592d4aad0117969eafeff	0.918947	0.694	0.714	0.760546	0.761364	0.795455	1.02E-06	1	5.00E-05	1028	0	zeros	he_normal	he_normal	0	abs	-1	RMSprop

Experiment #5 – without data augmentation with image resize to (150, 150, 1) and exponential learning rate decay mechanism with learning_rate_decay=0.97

In this experiment, we look at the effect of performing image resize and as part of the dataset pipeline, not enriching the training set with augmentation and adding exponential learning rate decay with on model performance/convergence.

Results:

Trial ID	test loss	test accuracy	test one shot accuracy	val loss	val accuracy	val one shot accuracy	train loss	train accuracy	learning_ rate	dense_layer _size	enable_batch _normalizatio n	bias_initi alizer	conv2D_kernel _initializer	dense_kernel _initializer	dropout_ rate	distance_ metric	l2_regularizer	optimizer
d3d71554ca0326ef045b59d24b69e6a5	0.579447	0.694	0.644	0.576827	0.727273	0.695455	0.097003	1	5.00E-05	1028	1	zeros	he_normal	he_normal	0	abs	-1	RMSprop
d5c3711c9186cfd11ebe22cfdfc49ad0	0.600033	0.694	0.662	0.569552	0.725	0.8	0.001253	1	5.00E-05	1028	1	zeros	he_normal	he_normal	0	abs	-1	RMSprop
17c45232c5e515d3edbf8c7d1c8023ec	0.593454	0.694	0.716	0.565989	0.720455	0.786364	0.005224	1	5.00E-05	1028	1	zeros	he_normal	he_normal	0	abs	-1	RMSprop
2b8d2e9933b0c6576503436165fc4edd	0.597642	0.692	0.634	0.567219	0.740909	0.763636	0.003106	1	5.00E-05	1028	1	zeros	he_normal	he_normal	0	abs	-1	RMSprop
52b49aeae22ab1b4e3a0ff8925667837	0.585532	0.688	0.672	0.562685	0.718182	0.781818	0.005056	1	5.00E-05	1028	1	zeros	he_normal	he_normal	0	abs	-1	RMSprop
1f86794579ca6339da3090a615f16c18	0.59966	0.686	0.638	0.568046	0.722727	0.786364	0.005025	1	5.00E-05	1028	1	zeros	he_normal	he_normal	0	abs	-1	RMSprop
c0e43614b3f4256490b4071503b53d14	0.591568	0.686	0.586	0.567213	0.720455	0.731818	0.033704	1	5.00E-05	1028	1	zeros	he_normal	he_normal	0	abs	-1	RMSprop
11f5251b39a1fb3feac7cd54568cf3ff	0.603307	0.684	0.636	0.582548	0.729545	0.786364	0.001595	1	5.00E-05	1028	1	zeros	he_normal	he_normal	0	abs	-1	RMSprop
268f704869a469c55645780d0990baf5	0.582457	0.683	0.654	0.570474	0.725	0.763636	0.054471	1	5.00E-05	1028	1	zeros	he_normal	he_normal	0	abs	-1	RMSprop
03d2f511d6a5290ec71bc060887e61e7	0.5906	0.682	0.642	0.561913	0.725	0.763636	0.015453	1	5.00E-05	1028	1	zeros	he_normal	he_normal	0	abs	-1	RMSprop

Experiment #6 – transfer learning - without data augmentation and use pre-trained Mobilenet model

Note: the motivation for trying transfer learning here is although our task is of learning representation vector of person's faces and Mobilenet is used for classifying pictures, the training set is relatively small, thus if a model is already trained on a much larger and general enough dataset including persons, then we might effectively utilizing it to feature extraction, namely use the representation learned by the model to our use and retrain just the final layers for producing the input encoding.

In this experiment, we look at the effect of using a pre-trained model by applying transfer learning, namely, feature extraction, freezing all layers except the top added dense layer, and not enriching the training set with augmentation on model performance/convergence.

Results:

Trial ID	test loss	test accuracy	test one shot accuracy	val loss	val accuracy	val one shot accuracy	train loss	train accuracy	learning_ rate	dense_layer_ size	enable_batch_ normalization	bias_initializer	conv2D_kernel _initializer	dense_kernel _initializer	dropout_ rate	distance_ metric	l2_regularizer	optimizer
b33f90162bcacd6036f4ba7afce2a244	0.533389	0.757	0.658	0.529366	0.763636	0.681818	0.181441	1	5.00E-05	1028	0	default	default	he_normal	0.5	abs	-1	RMSprop
49b0a6e67b1aaf4b1e79f9be75a5109d	0.533618	0.757	0.652	0.529955	0.759091	0.677273	0.182746	1	5.00E-05	1028	0	default	he_normal	he_normal	0.5	abs	-1	RMSprop
bbcf2ea7c8891bfbbd7302bdf15da276	0.535672	0.757	0.652	0.532684	0.759091	0.677273	0.194615	0.999432	5.00E-05	1028	0	default	he_normal	he_normal	0.5	abs	-1	RMSprop
b43595d86dab64793575d70957fe4b2a	0.532567	0.756	0.654	0.528827	0.763636	0.686364	0.17492	1	5.00E-05	1028	0	default	he_normal	he_normal	0	abs	-1	RMSprop
6e35e213fac1f40c1c86baabb7e7a509	0.533592	0.755	0.648	0.529436	0.763636	0.690909	0.181868	1	5.00E-05	1028	0	default	he_normal	he_normal	0.5	abs	-1	RMSprop
b048c53e2a1de7cef4de869cbc313bd5	0.543144	0.752	0.648	0.539041	0.756818	0.672727	0.231344	0.998864	5.00E-05	1028	1	default	he_normal	he_normal	0	abs	-1	adam
0e9d9e7e6d1d76831f0080650 2f894cb	0.559759	0.747	0.648	0.560066	0.734091	0.618182	0.312564	0.989205	5.00E-05	1028	1	default	he_normal	he_normal	0.5	abs	-1	adam
63a9c6821d453afa190f1d432ba31657	0.559757	0.74	0.654	0.562244	0.729545	0.631818	0.310543	0.986932	5.00E-05	1028	0	default	he_normal	he_normal	0.5	abs	-1	RMSprop
a81da4f5468bfcd2e869c1b7db335aa8	0.554381	0.718	0.604	0.5356	0.759091	0.631818	0.192709	1	5.00E-05	1028	1	zeros	default	he_normal	0	abs	-1	adam
d50b3cd63906bc009033fc199e90dca3	0.554694	0.718	0.602	0.538098	0.756818	0.631818	0.199905	1	5.00E-05	1028	1	zeros	default	he_normal	0.5	abs	-1	RMSprop

Experiment #7 – transfer learning - with data augmentation and use pre-trained Mobilenet model

In this experiment, we look at the effect of using a pre-trained model by applying transfer learning, namely, freezing all layers except the top added dense layer and enriching the training set with augmentation on model performance/convergence.

Results:

Trial ID	test loss	test accuracy	test one shot accuracy	val loss	val accuracy	val one shot accuracy	train loss	train accuracy	learning_ rate	dense_layer_ size	enable_batch _normalizatio n	bias_initi alizer	conv2D_kernel _initializer	dense_kernel_ initializer	dropout_ rate	distance_ metric	l2_regularizer	optimizer
9c7a215d82a507d1aca55af87d207ab7	0.523437	0.749	0.644	0.49988	0.784091	0.654545	0.190729	0.985511	5.00E-05	512	1	zeros	default	he_normal	0.5	abs	-1	adam
d605f2ccd4241832863baea809cff7d0	0.523042	0.748	0.64	0.501768	0.784091	0.654545	0.219394	0.980776	5.00E-05	512	0	zeros	default	he_normal	0.5	abs	-1	adam
2dcae7a20a4c3a4c78bb0a87bbff379a	0.522996	0.747	0.644	0.499931	0.786364	0.654545	0.204595	0.983333	5.00E-05	512	0	zeros	he_normal	he_normal	0.5	abs	-1	adam
50bd785d0f79588cb9eb661ce75faae7	0.522826	0.747	0.642	0.502106	0.786364	0.645455	0.227024	0.979451	5.00E-05	512	1	zeros	default	he_normal	0.5	abs	-1	adam
8c4963158e286271f3a507bb6331a522	0.522585	0.747	0.634	0.501941	0.786364	0.645455	0.227168	0.978504	5.00E-05	512	1	zeros	default	he_normal	0.5	abs	-1	adam
d64b98fd8b4e95ce6e68ecc2c0057150	0.522904	0.747	0.636	0.503268	0.784091	0.65	0.24344	0.974337	5.00E-05	512	1	zeros	default	he_normal	0.5	abs	-1	adam
b4b6ae66b5a4a7347fdf4b802efa273a	0.522976	0.746	0.636	0.502656	0.786364	0.645455	0.227173	0.978977	5.00E-05	512	1	zeros	default	he_normal	0.5	abs	-1	adam
af089a47c80f60419fbca0697ffb4d27	0.522979	0.746	0.642	0.502631	0.786364	0.65	0.227178	0.978977	5.00E-05	512	0	zeros	default	he_normal	0.5	abs	-1	adam
708b57a0df6412663826a47835522156	0.522837	0.745	0.64	0.50224	0.786364	0.65	0.2274	0.979356	5.00E-05	512	1	zeros	default	he_normal	0.5	abs	-1	adam
ca90e545a505bf70ac2d2d07f271773e	0.522705	0.745	0.64	0.501429	0.784091	0.659091	0.218278	0.980966	5.00E-05	512	0	zeros	default	he_normal	0	abs	-1	adam

Remarks, Observation, and Comparison:

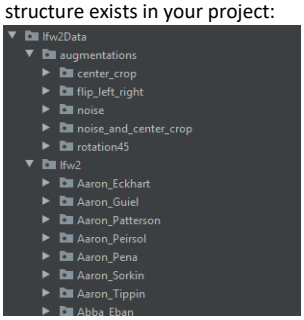
1. From our experiments above it appears that all models overfitted, the training binary accuracy reaches ~1 while the validation score is between 0.7-0.8.
2. As can be seen in the above tables, experiments that perform image-resize was able to yield better results compared to models that trained with the original data. Hence we can conclude that the amount of information retained in the reduced size image is sufficient for the task at hand. Moreover, this might assist and focus the training procedure since the model doesn't try to learn feature vectors including the noisy background that can contain other faces and elements that can damage the model accuracy. Also, the computation overhead and training time is considerably smaller and with a faster convergence rate.
3. For all models results above, L2 regularization method wasn't configured. Also, dropout wasn't configured except for models from the transfer learning experiments. Thus integrating a regularization method to control the convergence and maintain a smaller accuracy gap between training and validation to avoid overfitting doesn't produce better models.
4. For all models above, the absolute distance metric yields better models compared to the euclidean distance method.
5. It appears that enabling batch normalization helps models to achieve better performance.
6. It appears that setting a smaller dense layer size of 512/1028 was able to yield better models compared to 4098.
7. It seems that a relatively small learning rate is required to achieve good results. We can assume that this is due to the small training dataset size given and the nature of the task itself where the model should be as generalizing as possible and avoid learning significantly from a single sample.
8. Applying learning rate decay in experiment #5 did not help and we didn't get better results.
9. It seems that SGD optimizer is less suitable and produces less good results compared to RMSprop and Adam. Also, for our network architecture, it seems that RMSprop yields better models compared to Adam except for experiment #7 when applying transfer learning with data augmentation.
10. The one-shot learning accuracy scores of models based on the paper's architectures are superior compared to models based on the transfer learning paradigm (that we tested).
11. As can be seen in the above tables, there isn't a direct correlation between loss and accuracy.
12. Another aspect to take into consideration is that each configuration has a different computation overhead and training time is considerably larger when training without image-resize and data augmentation compared to other configurations which have a faster convergence rate.
13. When using the augmented data, as part of the training set, the validation loss has more variance across experiments than when not using augmented data during training. This could be explained by the fact that the models that were trained with the augmented data were more varied as they attempted to generalize over a larger set of samples.

How To Run:

1. open a terminal and cd to 'HW2_300822954_307963538' directory
2. conda env create --file HW2_300822954_307963538.yml
3. WINDOWS: activate HW2_300822954_307963538
   LINUX, macOS: source activate HW2_300822954_307963538
4. pip install keras-tuner

For running our code, perform the following:

1. Unzip lfwa.zip into 'lfw2Data/lfw2' and make sure lf2Data contains pairsDevTrain.txt and pairsDevTest.txt files
2. Run data_augmentation.py script
3. Validate that the following directories structure exists in your project:



4. Create tuner_results directory
5. Create tf_logs/tuner directory
6. For training, the best model with the above hyperparameters run main_bayes.py script
7. For running Bayesian optimization set the following parameters as desired (we present here the default configuration):
   # configuration
   seed = 0
   augment_dataset = True
   run_bayes_search = False
   use_transfer_learning_architecture = False
   image_resize = True
   batch_size = 64
   max_trials = 150
   num_models = 10
   epochs = 50
   patience = 15
   min_delta = 0.03