Deep Learning – Assignment 2

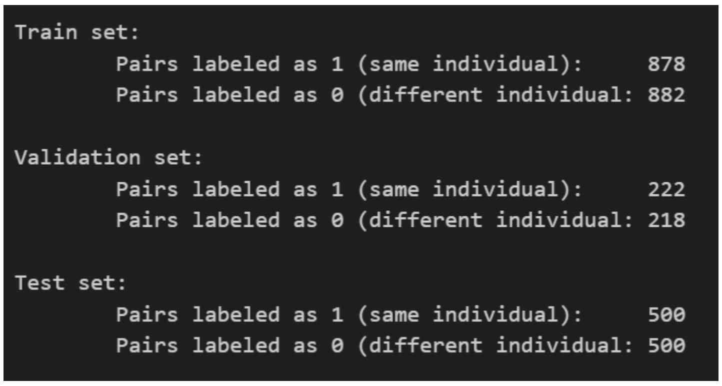
1. Overview

1.1. This assignment addresses the task of facial recognition using Convolutional Neural Networks (CNNs), specifically employing a one-shot learning approach. One-shot learning enables a model to accurately recognize individuals even when presented with only a single reference image per person. This capability is particularly valuable when extensive training data is scarce and mirrors the human ability to recognize faces after just one viewing.

1.2. Method: Our implementation is based on the paper "Siamese Neural Networks for One-shot Image Recognition" by Koch et al., which proposes the use of Siamese Networks—a specialized neural network architecture designed to determine if two previously unseen images depict the same individual or object. In this report, we describe our approach to building a Siamese Network tailored for the one-shot facial recognition task, detail our methodology for data preparation and model evaluation, and discuss the experimental results we obtained.

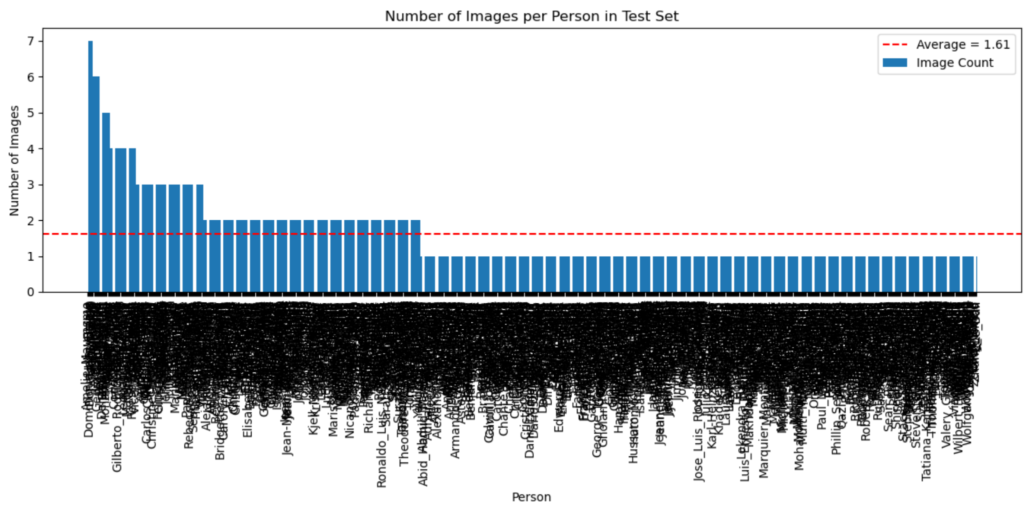
1.3. Dataset: For this assignment, we utilized the LFW-a dataset, which was originally presented by Taigman et al. in their paper "DeepFace: Closing the Gap to Human-Level Performance in Face Verification." This dataset contains a total of 13,233 facial images featuring numerous distinct individuals—some individuals appear in multiple images, while others are represented by just a single image. In addition to the image collection, the dataset included two CSV files defining the division of images into training and testing subsets. Each row in these files represents a specific pair of images. These pairs are labeled according to whether both images depict the same individual (label = 1) or two different individuals (label = 0).
To improve model training and parameter tuning, we further divided the provided training data by setting aside 20% as a validation subset, following standard machine learning practices.

1.3.1. Split Analysis: The final dataset partitioning resulting from this procedure is with balanced stratification, as follows:

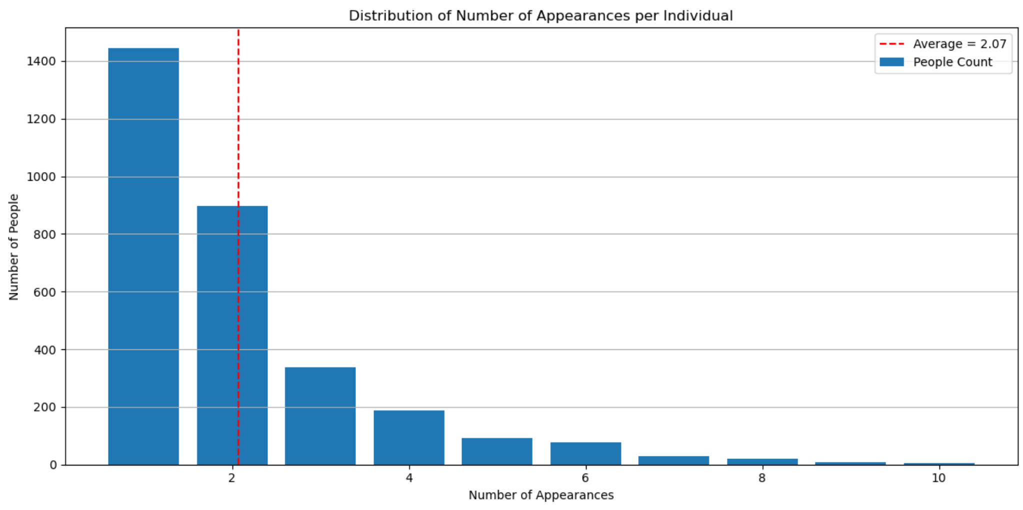


```
Train set:
        Pairs labeled as 1 (same individual):     878
        Pairs labeled as 0 (different individual: 882

Validation set:
        Pairs labeled as 1 (same individual):     222
        Pairs labeled as 0 (different individual: 218

Test set:
        Pairs labeled as 1 (same individual):     500
        Pairs labeled as 0 (different individual: 500
```

1.3.2. Distribution Analysis of images per individual in the dataset:



1.3.3. Frequency Distribution of Individual Appearances in LFW Train & Test Sets



Most individuals have only 1–2 images, with an average of ~1.6 images in the test set and ~2.1 appearances overall. The dataset is highly imbalanced, with a small number of identities appearing frequently and the majority appearing very few times.

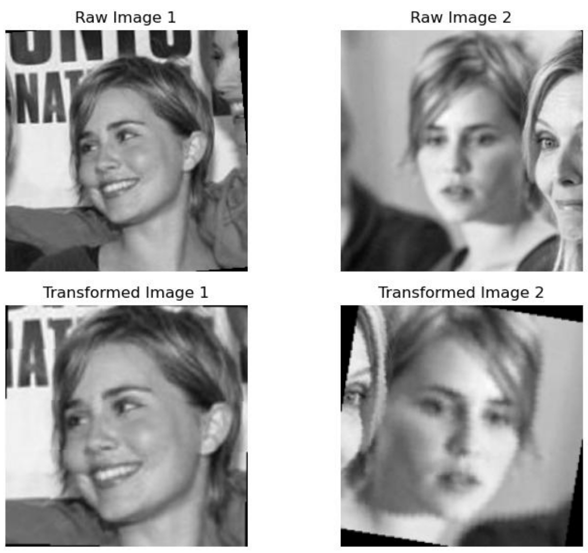
This reflects a few-shot learning scenario, making it challenging for models to generalize. Consequentially, we chose to use data augmentation techniques to mitigate the impact of limited samples per identity.

2. Pre-processing & Augmentation Pipeline

2.1. Pre-process: Some of the input images exhibited tilt or misalignment, resulting in visible black borders around the facial region. To address this and enhance model performance, we cropped 30 pixels from each side of the image before resizing. Through an Optuna-based hyperparameter search—where different image compression levels were tested with validation loss as the optimization objective—we aimed to receive the best trade-off between validation loss, accuracy, and retention of meaningful visual features, while also reducing runtime overhead. Overall, the pre-process steps applied uniformly to all datasets, included: cropping, compression (resize), normalization ([0, 1] → [-1, 1]) and conversion to tensors shape.

2.2. Trainset Augmentation: After splitting train set and validation set, to further improve generalization and expand the diversity of the training data, we applied a range of standard data augmentation techniques **exclusively to the training set images** (not test set nor validation set), as detailed below:

Split	Transformations (by order)	Purpose	Overall Goal
Train	RandomHorizontalFlip RandomRotation(±10°)	Faces in the LFW dataset are approximately aligned but still exhibit slight variations in yaw and pitch angles. Applying a ±10° rotation and horizontal mirroring expands the range of facial poses without significantly distorting critical identity features such as the eyes, nose, and mouth.	Increase pose / illumination diversity before down-sampling.



3. Model Architecture

3.1. Input Layer: Receives grayscale images of size 200X200X1, 160X160X1, or 105X105X1 (run-dependent), preprocessed to remove border noise and centered on facial features.

3.2. Convolution Layers: A sequence of four convolutional blocks:

3.2.1. Conv2d (1 → 64, kernel = 10)

3.2.2. Conv2d (64 → 128, kernel = 7)

3.2.3. Conv2d (128 → 128, kernel = 4)

3.2.4. Conv2d (128 → 256, kernel = 4)

Each layer is followed by ReLU activation.

3.3. Max Pooling Layers: Applied after the first three convolution layers, each using a 2 × 2 window to reduce spatial dimensions and enforce translational invariance.

3.4. Fully Connected Layers: After flattening the final feature map, the network passes through:

3.4.1. Linear layer (≈70k → 4096) with ReLU, and dropout for regularization

3.4.2. Linear layer (4096 → 1) producing a scalar embedding per input image

3.5. Output Layer: For a pair of images, the absolute difference between their final scalar embeddings is computed. This value is passed through a sigmoid function to yield a similarity score between 0 and 1, indicating the likelihood that the images belong to the same person.

4. Deviations from Koch et al. 2015:

Aspect	Koch et al. (2015)	In Our implementation	Clarifications
Input size	105×105	200X200, 160X160, 105X105	The original 105 × 105 crop was chosen to fit 2015 GPUs (3 GB). On modern cards 160 × 160 or 200 x 200 is a more natural divisor for the 2× max-pool chain, avoiding fractional features after the third pool.
Color	RGB	Grayscale	Using one channel cuts the first convolutional weight tensor by 3×, yielding fewer parameters and faster forward pass.
Dropout	No	Optional	We place dropout after the 4096-dim dense layer. Earlier dropout hurt because random feature masking before the fc1 flatten disturbed spatial correspondence.
Optimizer	SGD	Adam/AdamW/RMSprop/SGD	

Early stopping	–	Varying Patience
Hyper-param search	Manual	Optuna
Loss and Regularization	BCE, L1	BCE, L1

5. Experimental Setup

5.1. Our experimental setup emphasizes thorough hyperparameter optimization to identify the best-performing model configuration. We conducted a total of 300 distinct training runs, each with a unique set of hyperparameters, using the Optuna package.

5.2. The combinations were informed by both the paper's suggestions and widely accepted practices in the deep learning community. The following ranges were explored:

Key	Search Space	Justification
Resize Spatial Dimension	[105X105X1, 160X160X1, 200X200X1]	allows us to systematically evaluate the trade-off between computational efficiency and representational richness. The size 105×105×1 replicates the original configuration used in the reference paper, serving as a validated baseline. Increasing the resolution to 160×160×1 and 200×200×1 enables the model to capture finer-grained spatial features that may improve discriminative performance, particularly in tasks involving subtle visual differences.
Learning rate	Range of [1e-6, 1e-3], log-uniform	Covers five orders of magnitude, wide enough for Adam **and** RMSprop to find their natural sweet spot.
Weight decay	Range of [1e-6,1e-2], log-uniform	Below 1e-6 effect is negligible; above 1e-2 network under-fits in pilot runs.
Dropout rate	Range of [0.0, 0.5]	0 disables dropout entirely; Optuna often converged to 0.25.
Optimizer	Adam, AdamW, RMSprop, SGD	AdamW sometimes wins when weight-decay is large; RMSprop occasionally stabilises noisy early gradients. SDG was used in the paper.
Batch Size	32 across all experiments	
Loss Function	Binary Cross Entropy across all experiments	
Regularization	L1 across all experiments as in the paper	

5.3. Early Stopping Mechanism: For the hyperparameter tuning phase using Optuna, we limited each run to 200 epochs to keep the experiments efficient, while employing an early stopping patience of 20 epochs without improvement in validation loss. After identifying the top-performing hyperparameter configuration for each input resize option (105×105, 160×160, and 200×200), we retrained these selected models with an extended training setup. In this retraining stage, the maximum epoch count remained at 200, but we applied a stricter early stopping condition.

5.4. Data augmentation: We chose to augment each training sample deterministically to ensure consistent exposure, balanced feature learning, and reproducibility across runs. This approach improves generalization, and ensures reproducibility—crucial in low-data tasks like one-shot learning where every example counts.

5.5. Class-balance check**:** After the CSV-level 80 / 20 split we verified pair parity: 49.9 % positive vs 50.1 % negative in both train and validation.

5.6. Parameter Initialization: Kaiming-uniform for Conv/Linear.

6. Results

After identifying the top-performing hyperparameter configuration for each input resize option (105×105, 160×160, and 200×200), we retrained the network, each time with the best hyperparameter combination.

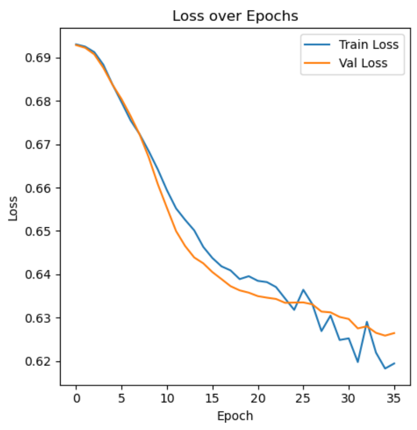
6.1. First Run |Resize to 105X105X1

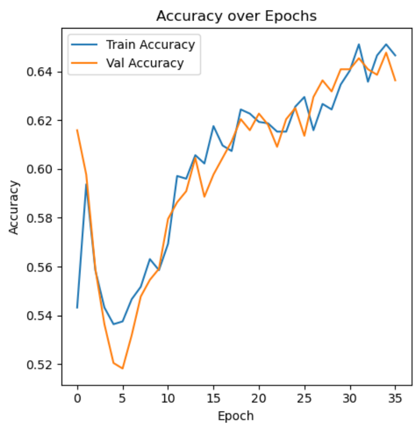
6.1.1. Hyperparameters:

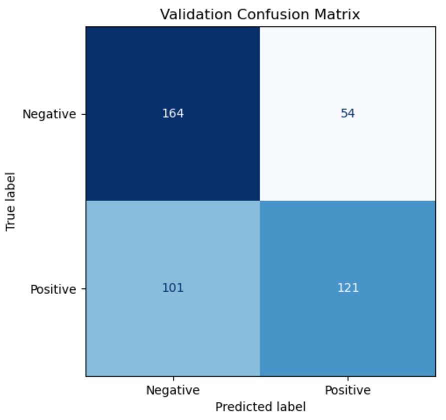
- Learning rate: 2.100918174668847e-06
- Weight decay: 2.340640334247835e-05
- Dropout Rate: 0.07611809725042018
- Batch size: 32

6.1.2. Statistics

Category	Value
Early Stopping triggered at epoch	36
Train Loss	0.6194
Train Accuracy	0.6466
Validation Loss	0.6264
Validation Accuracy	0.6364
Test Accuracy	0.6580
Confusion Matrix	TP: 121, FP: 54, TN: 164, FN: 101







** We deliberately tailored the early stopping criteria to ensure the divergence between training and validation performance is not temporary, while also keeping the model from heavily overfitting. In this case, overfitting begins after epoch 36.*

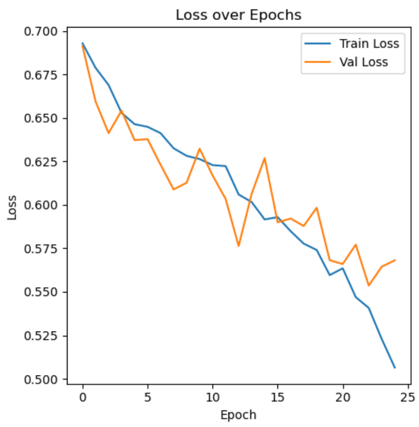
6.2. <u>Second Run |Resize to 160X160X1</u>

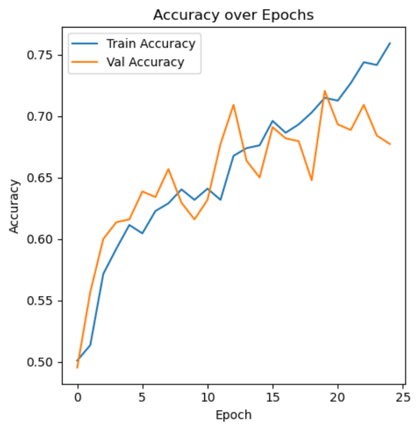
6.2.1. Hyperparameters:

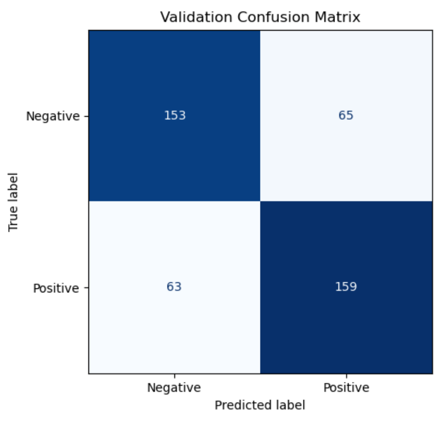
- Learning rate: 4.08367006057259e-05
- Weight decay: 0.008757687339341956
- Dropout Rate: 0.3357751894656019
- Batch size: 32

6.2.2. Statistics

Category	Value
Early Stopping triggered at epoch	25
Train Loss	0.5065
Train Accuracy	0.7591
Validation Loss	0.5681
Validation Accuracy	0. 6773
Test Accuracy	0.6800
Confusion Matrix	TP: 159, FP: 65, TN: 153, FN: 63







** We deliberately tailored the early stopping criteria to ensure the divergence between training and validation performance is not temporary, while also keeping the model from heavily overfitting. . In this case, overfitting begins after epoch 24.*

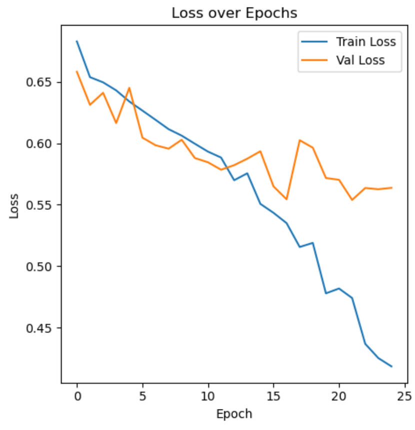
6.3. Third Run |Resize to 200X200X1

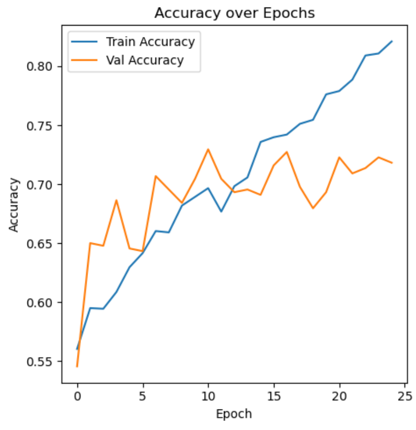
6.3.1. Hyperparameters:

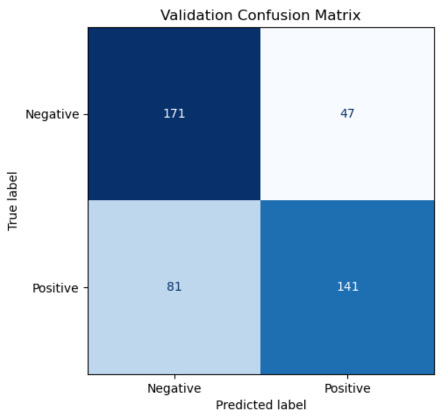
- Learning rate: 0.00022464196855043374
- Weight decay: 0.0062398539519617465
- Dropout Rate: 0.3339944605059354
- Batch size: 32

6.3.2. Statistics

Category	Value
Early Stopping triggered at epoch	25
Train Loss	0.4185
Train Accuracy	0.8210
Validation Loss	0.5637
Validation Accuracy	0.7182
Test Accuracy	0.6850
Confusion Matrix	TP: 141, FP: 47, TN: 171, FN: 81







** We deliberately tailored the early stopping criteria to ensure the divergence between training and validation performance is not temporary, while also keeping the model from heavily overfitting. . In this case, overfitting begins after epoch 17.*

7. Analysis and Conclusions

The following table summarizes the performance of the Siamese network across three different input resolutions:

Input Resolution	Train Accuracy	Validation Accuracy	Test Accuracy	Train Loss	Validation Loss	Epochs to Early Stopping
105 × 105 × 1	64.66 %	63.64 %	65.80 %	0.6194	0.6242	36
160 × 160 × 1	75.91 %	67.73 %	68.00 %	0.5065	0.5681	25
200 × 200 × 1	82.10 %	71.82 %	68.50 %	0.4185	0.5637	25

7.1.1. Effect of Increased Input Resolution

- Larger input resolutions provide more detailed visual information, which enables the network to learn more discriminative features. As a result, both training accuracy and loss improve consistently as resolution increases. Specifically, moving from 105×105 to 200×200 results in an increase of over 17.44% in training accuracy and a reduction in training loss from 0.6194 to 0.4185. This improvement is attributed in part to the increased model capacity: the feature map passed to the fully connected layer grows with the input size (e.g., $256 \times 12^2 \rightarrow 256 \times 17^2$), enabling the network to capture more complex patterns during training.

7.1.2. Generalization and Overfitting

- While all configurations show higher training accuracy compared to validation accuracy—indicating a degree of overfitting—the gap remains relatively consistent across resolutions.

7.1.3. Training Efficiency

- Higher resolutions lead to faster convergence. Both the 160×160 and 200×200 configurations required only half the number of epochs (25) compared to the 105×105 setup (38 epochs). This indicates that the richer gradients derived from higher-resolution images facilitate more efficient learning.

8. Comparison to Original Paper

8.1.1. While the original paper achieved 92% accuracy on the Omniglot dataset, our best result on LFW reached 68.7%. This gap is due to key differences: Omniglot features simple, centered characters with many samples per class, while LFW contains real-world face images with high variability, few samples per identity, and weak alignment.

Additionally, unlike the original work, our model was trained from scratch without pretrained facial features, making the task significantly more challenging.

9. Recommendations for Future Work

9.1.1. Pretrained Feature Extractors: Since LFW is a real-world face dataset, initializing the Siamese towers with weights from pretrained facial recognition models (e.g., FaceNet or VGGFace) could improve generalization and reduce training time.

9.1.2. Enhanced Regularization for High-Resolution Inputs: Although the 200×200 configuration performed best overall, it also exhibited a slightly larger training-validation gap, which could be mitigated with stronger regularization (e.g., increased dropout or weight decay).

9.1.3. Batch Normalization: use of BatchNorm between convolutional layers might Improve training stability and accelerate convergence.

9.1.4. Stochastic Data Augmentation: Instead of applying augmentation to all training samples, applying it to a random subset can improve generalization by preserving some clean data. We suggest treating the augmentation probability as a tunable hyperparameter in future work.