Face verification System

# Introduction

Face verification system is one of the most popular applications that utilizes CNN model for featuring face. There are many ways to implement face verification. This report implements CNN model, combining 2 images and using metrics learning for verifying if it’s from the same label or not. Additionally, it also does the anti-spoofing task that detects non-real faces.

# Methodology

## Data Preprocessing

### Classification Data preprocessing

For classification preprocessing, ImageDataGenerator is used to create augmentation for images:

A screen shot of a computer code

AI-generated content may be incorrect.

Figure : ImageDataGeneartor()

And flow\_from\_directory is used to get images from directories

A computer screen shot of text

AI-generated content may be incorrect.

Figure : get images from directories

### Verification Data Preprocessing

Verification Data is input for Siamese network which needs a pair of images. First, it is needed to load images from directories. Because there is no text file that is used to create pairs for images, I need to create a custom pair of images.

A screen shot of a computer code

AI-generated content may be incorrect.

Figure : Image Reading

First of all, images are read from each folder and assigned to images. After that, a pair is ready to be initiated. A pair consists of 2 images and a label 0 and 1, interpreting if it is the same or not. There are 4000 different labels in the train\_data. In each label, 36 pairs are created, half of them are positive pairs, and the remaining half are negative pairs

A screen shot of a computer code

AI-generated content may be incorrect.

Figure : Positive pairs creation

In each directory, 18 positive pairs are created and appended to pairs. Negative pairs are created using different directories to get the images, and create pairs based on that

A computer screen shot of text

AI-generated content may be incorrect.

Figure : Negative pairs creation

After completing loading pair images from the classification\_data folder, the next step is to preprocess them. In the notebook, prepare\_dataset is the function to preprocess images to work with the model

A screen shot of a computer program

AI-generated content may be incorrect.

Figure : Prepare\_dataset function (first part)

In the load\_pair function, 2 images go through load\_and\_preprocess\_images to do things:

* Convert image to float32
* Clip image on range 0,255 (optional)
* resize the image to the desired image size (112,112).

A computer screen with text

AI-generated content may be incorrect.

Figure : load and preprocess image

Then, images are formatted into tf.data.Dataset, shuffle the data, and apply paralleling function.

A screen shot of a computer program

AI-generated content may be incorrect.

Lastly, the images go through augmentation layers such as Flip, Rotation, and zoom to add noise to the images to improve generalization of performance for the model. Additionally, the images are processed in batches for faster training and evaluation.

A screen shot of a computer program

AI-generated content may be incorrect.

Figure :data augmentation and batching

### Combine Dataset

The combine dataset is used for training a multi-task model that verify images and detect non-real faces.

A computer screen shot of a code

AI-generated content may be incorrect.

Figure : combine\_generator

The combined dataset combines data from pairs for verification and single image for anti-spoofing and yields batches of data as output

A computer code with colorful text

AI-generated content may be incorrect.

Figure : train\_dataset desired format

Lastly, a train\_dataset is created with the desired output shape and the label according to the images for training. Using pre-fetch Autotune to optimize data loading by pre-fetching batches to reduce bottlenecks.

## Model Building

From the model building, the backbone model is EfficientNetV2-B0, which is suitable for processing images with average resolution, low image size, and fast training.

### Classification Model

The model is created using EfficientNetV2B0 as backbone for training. Then, the output will go through GlobalAveragePooling2D() for features selection. Next, it goes through a block of layer: Dense, Batch Normalization, Dropout. And the output is the dense layer with SoftMax activation function.

A computer screen with text on it

AI-generated content may be incorrect.

For hyperparameters tuning, the optimizer function used is Adam with the learning rate = 0.0001. Loss function is categorical\_crossentropy, which is a popular loss function used for classifying multiple classes, metrics are top-1-accuracy.

A screen shot of a computer code

AI-generated content may be incorrect.

Figure : model compiled with loss function and optimizer

For the callbacks, there are 3 supporting callbacks, EarlyStoppig to that stops the training if the val\_accuracy did not increase, ReduceRLOnplateau, to adjust the learning rate if the training phase is not stable, and ModelCheckPoint which is used to save the model with best performance among the training epoches.

A screen shot of a computer program

AI-generated content may be incorrect.

Figure : callbacks hyperparameters

### Verification Model

For verification model, the model takes a pair of images, processing them using the backbone model. This phase acts as an embedding network that process the image to select the important features of a face.

A computer screen shot of text

AI-generated content may be incorrect.

Figure : Model's embedding network

The backbone layer’s output goes through the GlobalAveragePooling2D() and another block of fully connected layer like in classification task. This neural network is applied to 2 images for features selection and processing.

A screen shot of a computer program

AI-generated content may be incorrect.

Figure : Verification task using Concatenation layer

At this stage, all 2 input images go through the embedding network and process by a Concatenation layer. Lastly, the output will be created based on the sigmoid activation function.

On the other hand, 2 input images can be processed through a lambda layer to track if they are the same or not.

A screen shot of a computer program

AI-generated content may be incorrect.

Figure : Verification task using metrics learning

For hyperparameters, the loss function used is ‘binary\_crossentropy’ for binary classification, since the output label is either 0 or 1 (same or different). All other hyperparameters remain the same.

A computer screen shot of text

AI-generated content may be incorrect.

Figure : model hyperparameters

### Anti spoofing Model

An anti-spoofing model utilizes face verification model and anti-spoofing model to create a model with multi-task. First of all, the embedding layer remains the same.

For the verification model, while all layers for verification task remain the same, new layer is added to do the anti-spoofing task.

## Model Training

Using Early stopping, the models will train with epoch=100,

# Results

## Classification model results

A screenshot of a computer program

AI-generated content may be incorrect.

Figure : classification model's evaluation

## Siamese model using metric learning

A screenshot of a computer program

AI-generated content may be incorrect.

Figure : cosine-model evaluation

## Siamese model using concatenation layer

A screenshot of a computer program

AI-generated content may be incorrect.

Figure : concatenation layer model's evaluation

# Discussion

## Model comparison

For face verification task, classification approach is not suitable due to the very large number of labels (4000 labels). Furthermore, training a classification model takes a large amount of time compared to another approach. It took 1 week to finish training a classification model. However, the highest validation accuracy was about 75%, which is a fair performance.

Using Siamese Network, the model outstands classification. By using 2 images as input and verifying if it’s the same or not, this approach takes a shorter time to train due to the binary output (0 or 1), and the performance of the model is far better than the classification model (about 83% validation accuracy).

## Challenges

In the testing, I make sure to use images from the test folder, which is not included in the train\_folder, and the accuracy of all models seem to be even higher than the accuracy of the validation. However, when testing on a real-time face verification system, the model does not perform well, it often detects faces as the same while they are not.

## GUI

For GUI applications, I have created 2 applications, one for testing the performance of Siamese network, and one for a verification system.

The GUI system includes login and signs up page, the sign-up page register a face, and the login page verifies the image to get through.

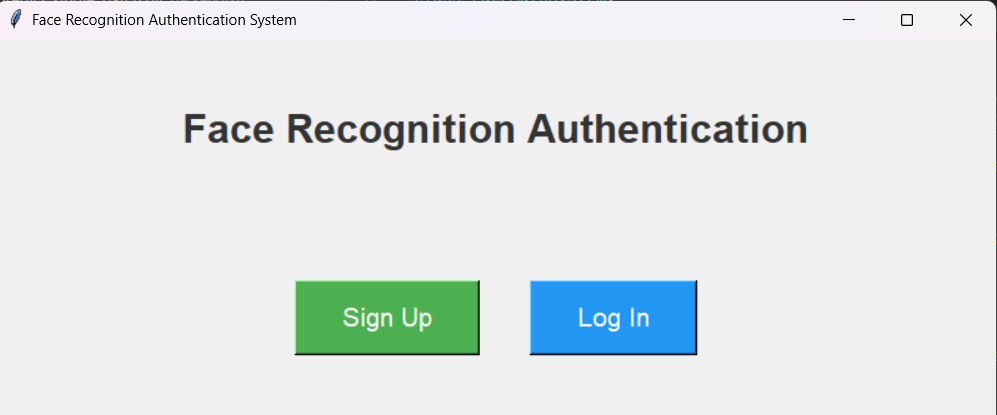


Figure : verification interface

# Conclusion

The face verification project involves developing a face recognition attendance system using CNN-based face verification, comparing metric learning and classification approaches, and evaluating performance with ROC and AUC metrics. An anti-spoofing module with liveness detection counters spoofing attempts, tested via a live demo. The system requires a user interface for registering and identifying employees, with submissions including detailed reports, source code, and optional innovations.