Handwritten Signature Identification and Verification

Team: SC\_1

Omar Ashraf Darwish

20191700402

Amer Yasser Abdulmagid

20191700802

Abdullah Ahmed Metwally

20191700377

Sameh Mohamed Fekry

20191700281

Abdelrahman Magdy Abdulwahid

20191700369

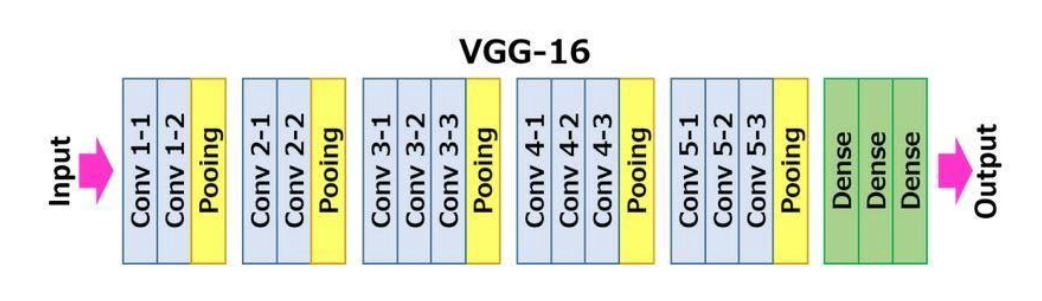
**Stage 1: First Model – Regular Deep Learning:**

**Data Preparation Process:**

* We began by extracting the dataset file, then We read the Training Images and Testing Images as colored and in size of (80,80) and inserted them in the trainingData and testingData list where every place of the list consists of a list that holds the image and its label based on different persons.
* Then We transformed the list of trainingData into a NumPy Array.
* We split the trainingData Array into Two X and Y lists where X holds the Image and Y holds The Label.
* We split the testingData Array into Two X\_test and Y\_test lists where X\_test holds the Image and Y\_test holds the Label.
* By transforming X and Y into NumPy Arrays.
* By transforming X\_test and Y\_test into NumPy Arrays
* Then We normalized X by dividing it by 255
* Then We normalized X\_test by dividing it by 255
* We split the Training Data into Training Data (80%) and Validation Data (20%) with shuffling.

**Model Process – VGG-16 Model:**

* In This Model We used Transfer Learning and VGG-16 Model.
* VGG-16 Model is a convolutional neural network that is famous for its amazing ability to classify Images, in this model the depth is increased using an architecture with very small (3 × 3) convolution filters, which showed a significant improvement on the prior-art configurations.



* It’s mainly one of the most popular algorithms for image classification and is easy to use with transfer learning.
* We utilized the VGG-16 Model by using this pretrained architecture that can detect generic visual features present in our dataset.
* We imported the Model from keras.applications.vgg16 Library, We added the ‘imagenet’ weights and we made sure that the top part is not included in our model as we want to cut off the Fully-Connected Layers.
* By applying Fine Tuning on the Model by freezing the Trainability of The Layers.
* We Flattened the Model and added a Dense Layer with The Number of Classes and Activation Function of SoftMax.
* We used the ‘sparse\_categorical\_crossentropy’ as The loss function, ‘adam’ as an optimizer and ‘accuracy’ as the metric.
* By Training Model on 30 Epochs.

**Training and Testing Time:**

Training Time = 21 Seconds.

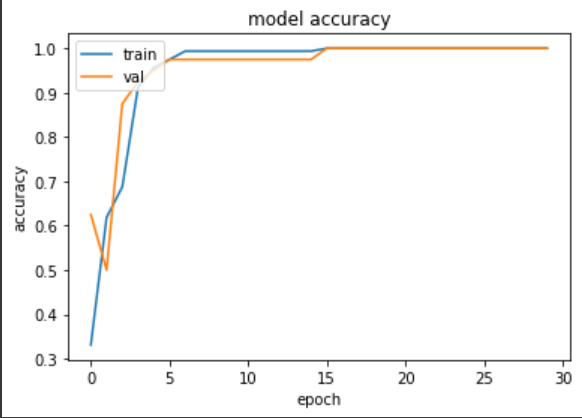
Testing Time = 0.117813 Seconds.

**Training and Testing Accuracy:**

Training Accuracy = 100%

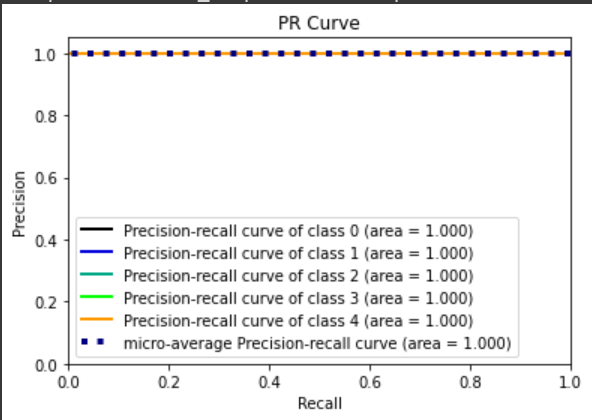
Validation Accuracy= 100%

Testing Accuracy = 100%



**Visualization of Test Set Classification:**

**Percision-Recall Curve:**

****

**Screenshots of Classification of Test Set**

**Text, letter

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

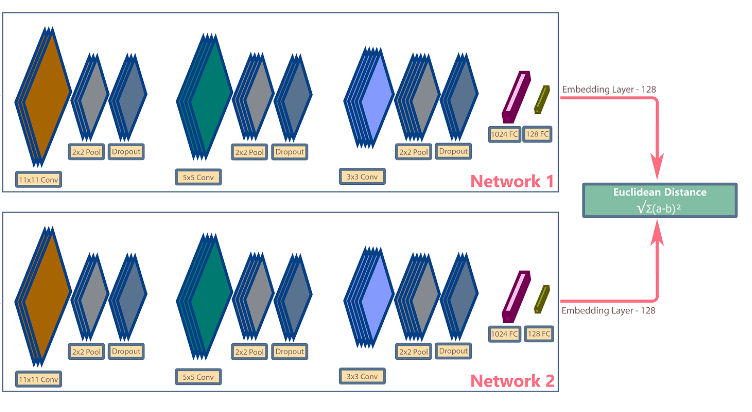
**Stage 2: Single Siamese model**

**Data Preparation Process:**

* We began by extracting the dataset file, then We read the Training Images and Testing Images as colored and in size of (128,128,3) and converted them from BGR to RGP.
* We created a triplet list for Training (train\_triplet) and another for Testing (test\_triplet), each triplet consists of anchor (real image) and positive (another real image) negative (forgery image), all images for the same person
* We made shuffle for the data (training and testing)
* We had converted each image to np array.
* Each image has been preprocessed by the function preprocess\_input.

**Model Process –Sequential Model:**

the Siamese Network only finds out the distance between any two given images. If the images have the same label, then the network should learn the parameters, the weights and the biases in such a way that it should produce a smaller distance between the two images, and if they belong to different labels, then the distance should be larger.



We're using a pretrained model, Xception model which is based on Inception\_V3 model. By using transfer learning, we can significantly reduce the training time and size of the dataset.

The Model is connected to Fully Connected (Dense) layers and the last layer normalises the data using L2 Normalisation. (L2 Normalisation is a technique that modifies the dataset values in a way that in each row the sum of the squares will always be up to 1)

We have used the triplet loss function

Shape

Description automatically generated with medium confidence

We used the adam optimizer with learning rate=1e-3,

epsilon=1e-01

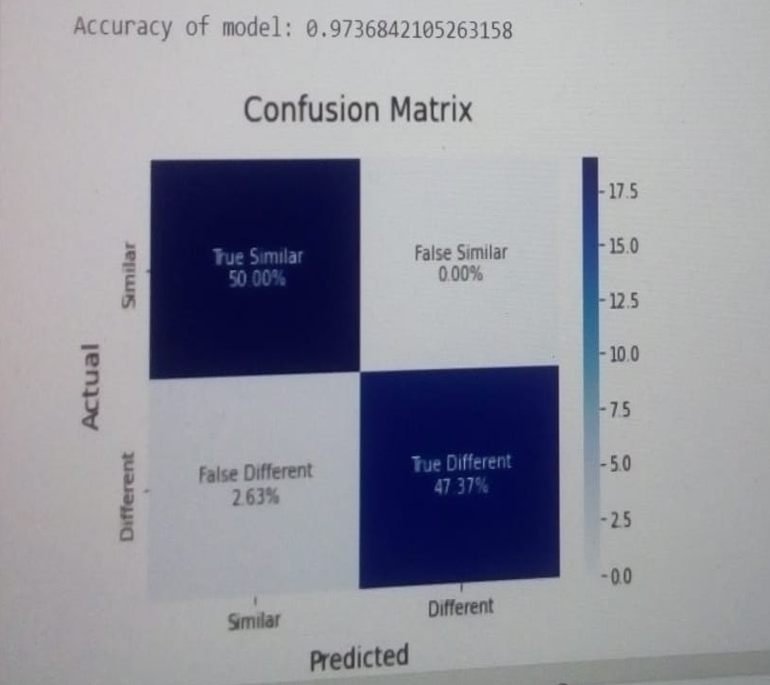
**Training and Testing Time:**

5 m and 37 s for training with epochs=25 and batch size =128

1 s for testing

**Training accuracy**: 100%

**General Accuracy:**



**Stage 3: Signature object detection:**

**Data Preparation Process:**

* We began by extracting the dataset file, then we split train images into train (80%) and validation images (20%) .
* We created dataset.yaml file that describe to YOLO which object we need to detect “Signature“.
* We changed the format of current labels to YOLO format.
* We put train, validation data and dataset.yaml file in the same path so that YOLO method can train our images.

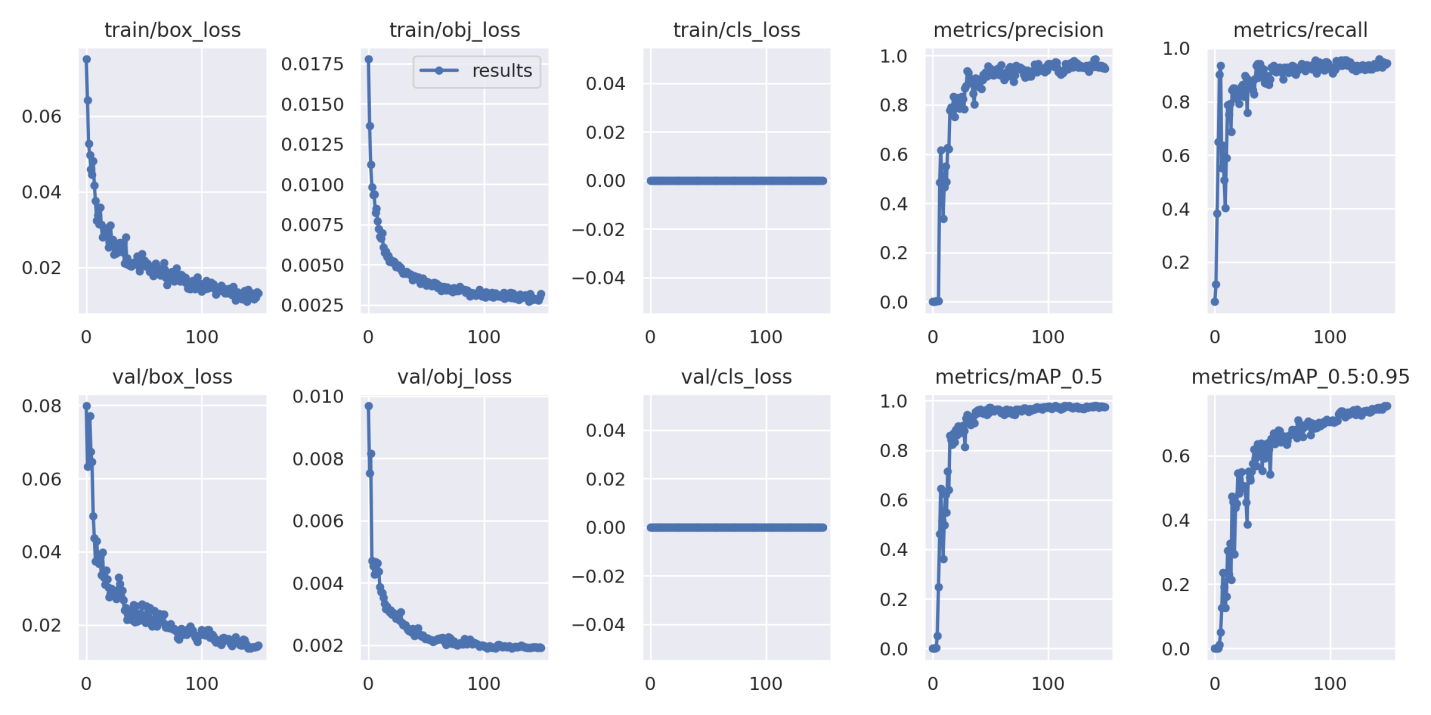
**Model Process – YOLO-v5 Method**

* YOLO-v5 is one of the most high-performing object detectors out there. It is fast, has high accuracy and is incredibly easy to train.
* YOLOv5 is shipped with a set of YOLOv5s, YOLOv5m, YOLOv5n and Others.
* They’re pre-trained using MS COCO dataset
* COCO dataset is a large-scale object detection , segmentation dataset
* We used YOLOv5s6 in our Project.

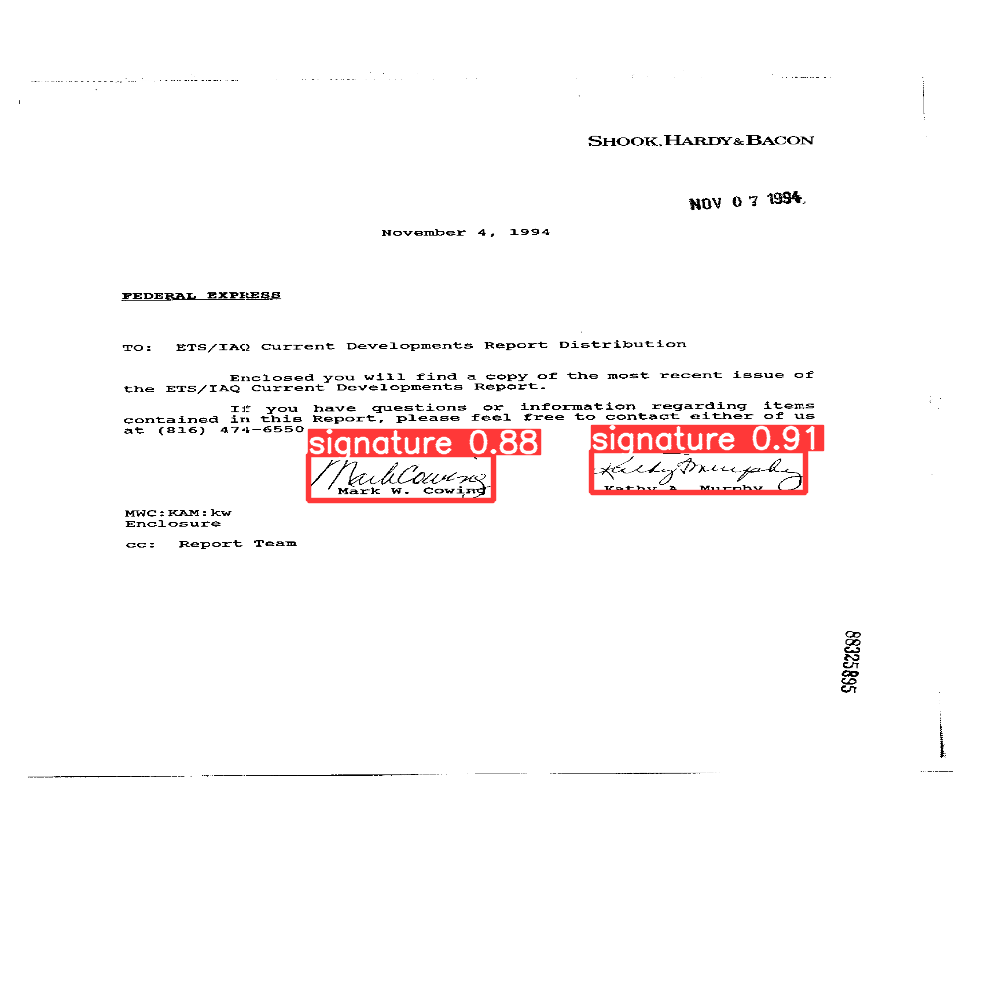
**Training and Testing Time:**

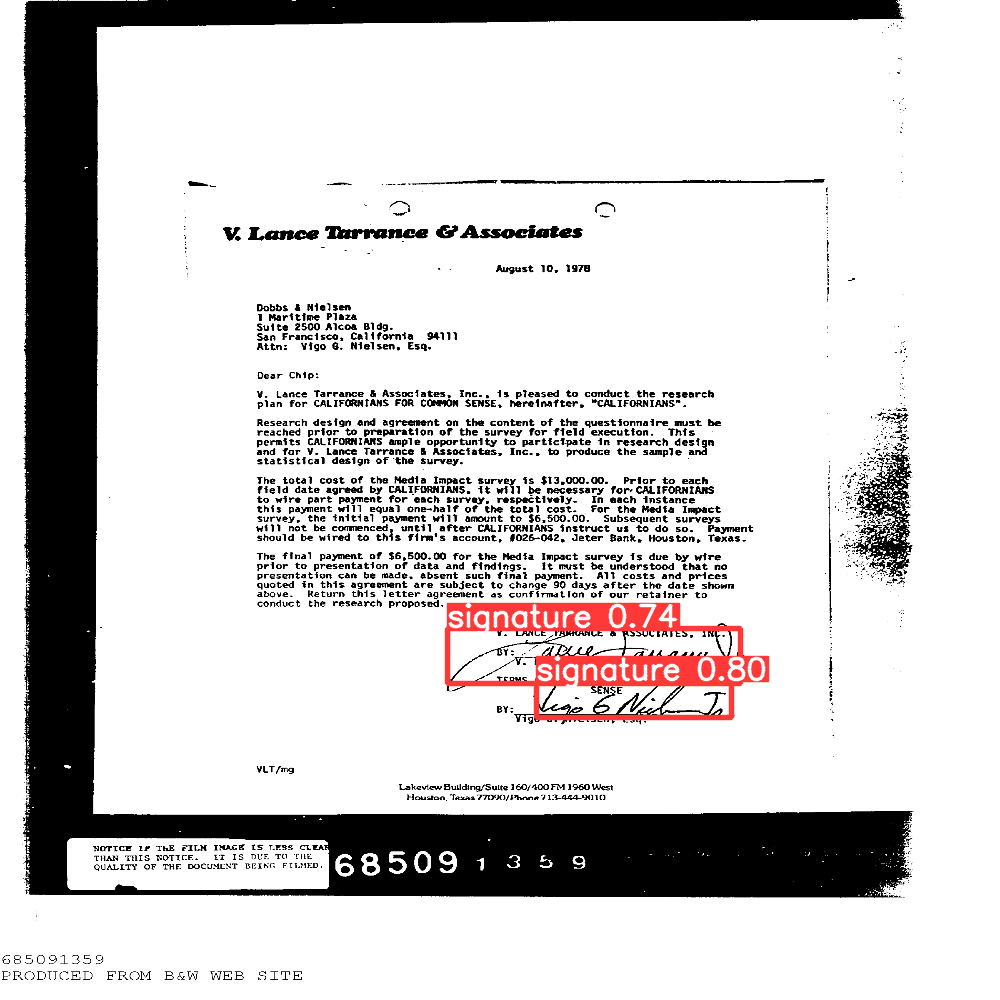
* Training Time = 27 minutes.
* Detecting Time = 16 Seconds.

**Object detection performance**



**Screenshots of the test sets detection**

****

****

