Application Project

Digit Recogniser using Tensorflow backend

Project by: Ignatius Tang
Supervisor: Professor Kai Warendorf
Completed on Jan 2024

ESSLINGEN UNIVERSITY

Project Overview

Objective:
Create a ML model using Tensorflow backend to accurately predict and recognise handwritten digits from input images.

Kaggle Competition link:
https://www.kaggle.com/competitions/digit-recognizer/overview

Resources used:

- Kaggle Learn
- Tensorflow website/youtube
- Stack Overflow
- OpenAI

ESSLINGEN UNIVERSITY

Table of contents

ESSLINGEN UNIVERSITY

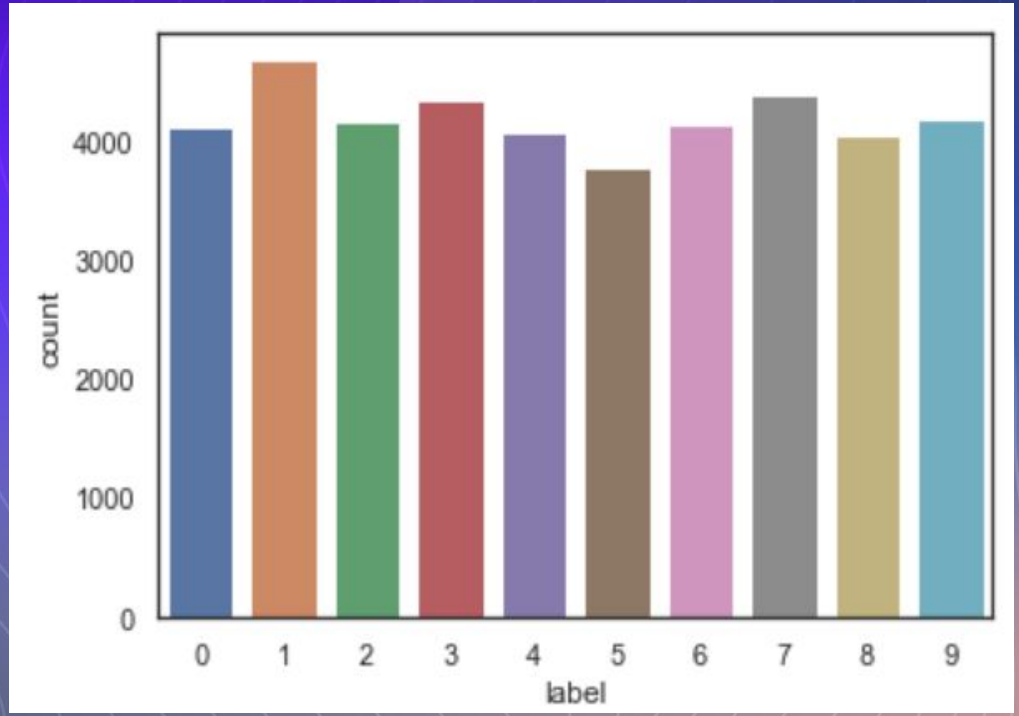
What is Tensorflow?

- Open-source ML library by Google Brain
- Used for building and training deep neural networks
- Utilises a data flow graph for efficient computational workflows
- Widely applied in tasks like image recognition and natural language processing
- Provides tools and libraries for development in multiple programming languages.

1. Data Preparation

Loading Data:

```
Out[3]: 1    4684
        7    4401
        3    4351
        9    4188
        2    4177
        6    4137
        0    4132
        4    4072
        8    4063
        5    3795
        Name: label, dtype: int64
```



^Count for all 10 digits 0-9 is similar around 4000.

ESSLINGEN UNIVERSITY

1. Data Preparation

Check for null and missing values:

```
In [4]: # Check the data
        X_train.isnull().any().describe()

Out[4]: count       784
        unique        1
        top       False
        freq        784
        dtype: object

In [5]: test.isnull().any().describe()

Out[5]: count       784
        unique        1
        top       False
        freq        784
        dtype: object
```

Both train and test data count = 784

No missing values, dataset is free of corrupted images, safe to proceed.

ESSLINGEN UNIVERSITY

1. Data Preparation

Normalisation:
Perform greyscale normalisation to reduce the effect of illumination's differences.

```
In [6]: # Normalize the data
        X_train = X_train / 255.0
        test = test / 255.0
```

Reshape:
Train and test images (28px x 28px) has been stock into pandas. Dataframe as 1D vectors of 784 values. We reshape all data to 28x28x1 3D matrices.

```
In [7]: # Reshape image in 3 dimensions (height = 28px, width = 28px , canal = 1)
        X_train = X_train.values.reshape(-1,28,28,1)
        test = test.values.reshape(-1,28,28,1)
```

ESSLINGEN UNIVERSITY

1. Data Preparation

Label Encoding:

```
In [8]: # Encode labels to one hot vectors (ex : 2 -> [0,0,1,0,0,0,0,0,0,0])
        Y_train = to_categorical(Y_train, num_classes = 10)
```

- Labels are 10 digits numbers from 0 to 9. We need to encode these labels to one hot vectors (eg : 2 -> [0,0,1,0,0,0,0,0,0,0]).

ESSLINGEN UNIVERSITY

1. Data Preparation

Split Training and Validation Set:

```
In [9]: # Set the random seed
        random_seed = 2

In [10]: # Split the train and the validation set for the fitting
         X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size = 0.1, random_state=random_seed)
```

I chose to split the train set in two parts:

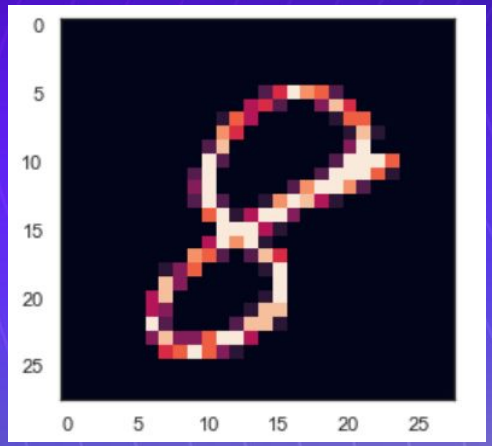
- a small fraction (10%) became the validation set which the model is evaluated and
- the rest (90%) is used to train the model.

Since we have 42,000 training images of balanced labels, a random split of the train set does not cause some labels to be over-represented in the validation set.

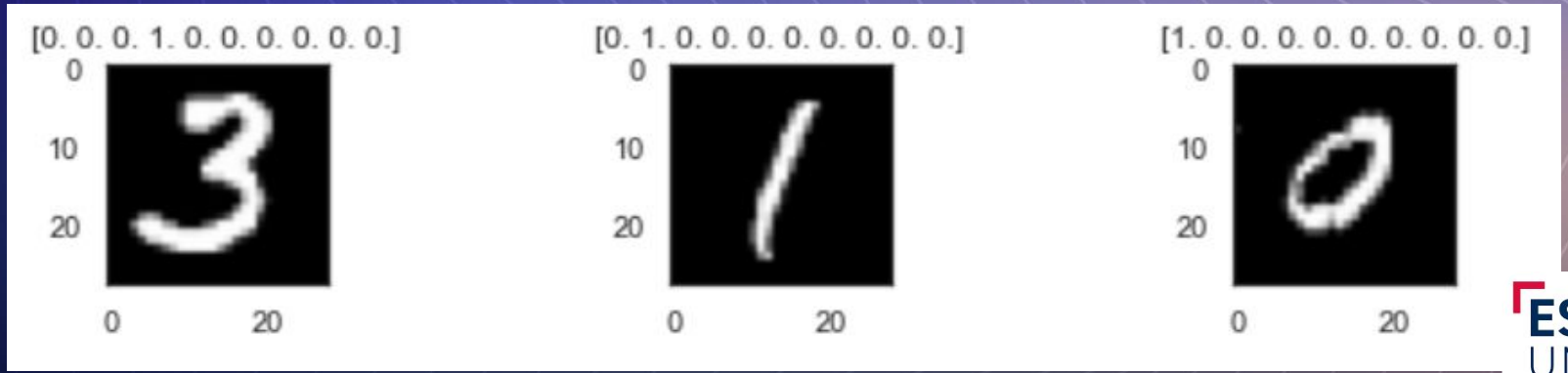
ESSLINGEN UNIVERSITY

1. Data Preparation

Example for train dataset:



Examples for raw train dataset:



ESSLINGEN UNIVERSITY

2. Train/Test Model

Define the CNN model:

- Build CNN using Keras Sequential API, adding layers incrementally.
- Conv2D layers with 32 and 64 filters transform image portions to generate feature maps.
- MaxPool2D layers downsample, reducing computational costs and overfitting.
- Combined layers enable discernment of local and global image features.
- Dropout for regularisation, randomly ignoring nodes to enhance generalization.

ESSLINGEN UNIVERSITY

2. Train/Test Model

Define the CNN model:

- Relu activation adds non-linearity to the network.
- Flatten layer converts final feature maps to a 1D vector.
- Conclude with two Dense layers forming an ANN classifier.
- Last Dense layer with softmax activation outputs class probability distribution.

CNN architecture>>
[[Conv2D->relu]*2 -> MaxPool2D -> Dropout]*2 -> Flatten -> Dense -> Dropout -> Out

ESSLINGEN UNIVERSITY

2. Train/Test Model

Set the Optimizer and Annealer:

- After adding layers to the model, defining "*categorical_crossentropy*" as the loss function measures the disparity between observed and predicted labels.
- Opting for the RMSprop optimizer, which iteratively refined parameters, is crucial for efficient learning, while the "*accuracy*" metric evaluates performance without influencing training.

```
In [14]: from tensorflow.keras.optimizers import RMSprop

         # Define the optimizer
         optimizer = RMSprop(learning_rate=0.001, rho=0.9, epsilon=1e-08)
```

```
In [15]: # Compile the model
         model.compile(optimizer = optimizer , loss = "categorical_crossentropy", metrics=["accuracy"])
```

ESSLINGEN UNIVERSITY

2. Train/Test Model

Set the Optimizer and Annealer:

- Implemented LR annealing in the optimizer aims to expedite convergence toward the global minimum of the loss function.
- A dynamic LR reduction strategy, utilizing the ReduceLROnPlateau function from Keras.callbacks, adjusts the LR by half every X steps (epochs) when accuracy fails to improve after 3 epochs.
- This approach balances faster computation with the need to efficiently reach the optimal loss.

In [16]:
```
# Set a learning rate annealer
learning_rate_reduction = ReduceLROnPlateau(monitor='val_acc',
                                            patience=3,
                                            verbose=1,
                                            factor=0.5,
                                            min_lr=0.00001)
```

In [17]:
```
epochs = 30 # Turn epochs to 30 to get 0.9967 accuracy
batch_size = 86
```

ESSLINGEN UNIVERSITY

2. Train/Test Model

Set the Optimizer and Annealer:

To mitigate overfitting, I artificially expand the handwritten digit dataset by applying minor transformations, such as off-centering, varying magnitude scales, and different orientations, to replicate the variations encountered during the writing process.

Augmentation:

- Randomly rotate some training images by 10 degrees
- Randomly Zoom by 10% some training images
- Randomly shift images horizontally by 10% of the width
- Randomly shift images vertically by 10% of the height

I did not apply a vertical_flip nor horizontal_flip since it could have lead to misclassify symmetrical numbers such as 6 and 9.

ESSLINGEN UNIVERSITY

2. Train/Test Model

Fit the training dataset:

```
In [19]: # Fit the model using Model.fit
         history = model.fit(datagen.flow(X_train, Y_train, batch_size=batch_size),
                             epochs=epochs, validation_data=(X_val, Y_val),
                             verbose=2, steps_per_epoch=X_train.shape[0] // batch_size,
                             callbacks=[learning_rate_reduction])
```

1st epoch - accuracy = 0.8659

```
Epoch 1/30
439/439 - 23s - loss: 0.4164 - accuracy: 0.8659 - val_loss: 0.0750 - val_accuracy: 0.9750 - lr: 0.0010 - 23s/epoch - 53ms/ste
```

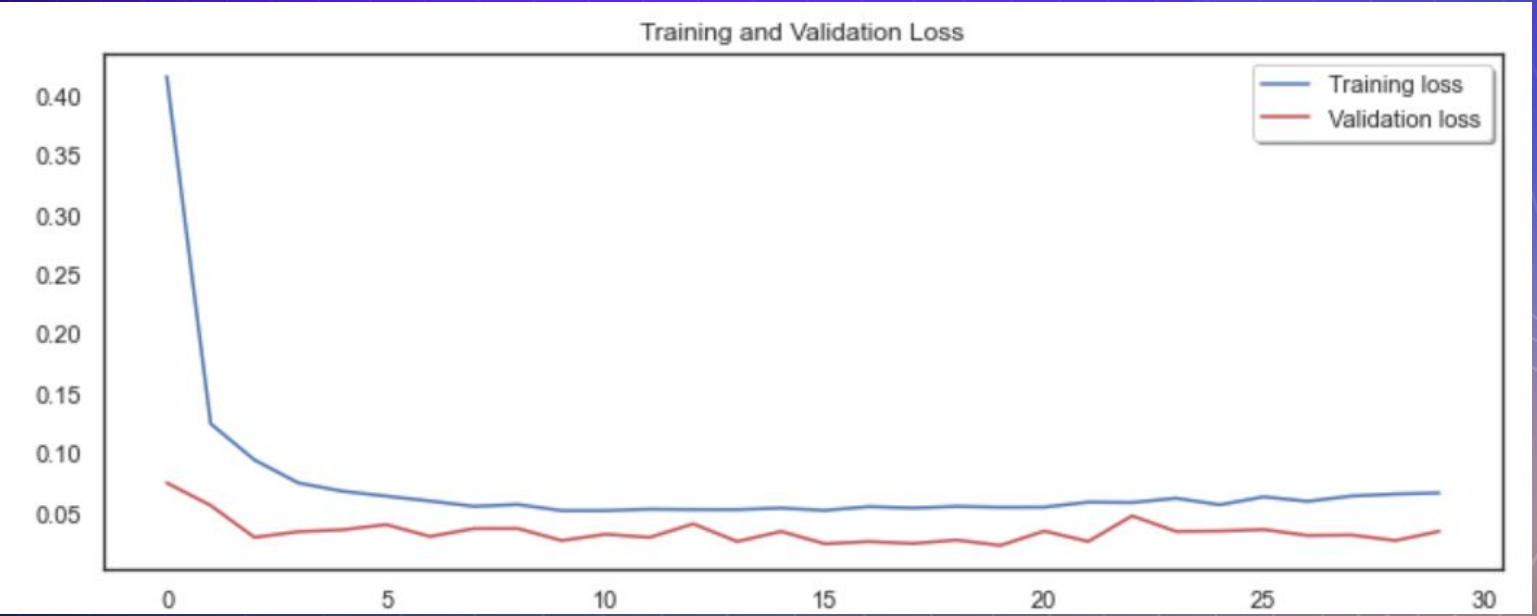
```
Epoch 30/30
439/439 - 31s - loss: 0.0664 - accuracy: 0.9839 - val_loss: 0.0342 - val_accuracy: 0.9936 - lr: 0.0010 - 31s/epoch - 71ms/ste
```

30th epoch - accuracy = **0.9839**

ESSLINGEN UNIVERSITY

3. Model Evaluation

Training and Validation Loss:

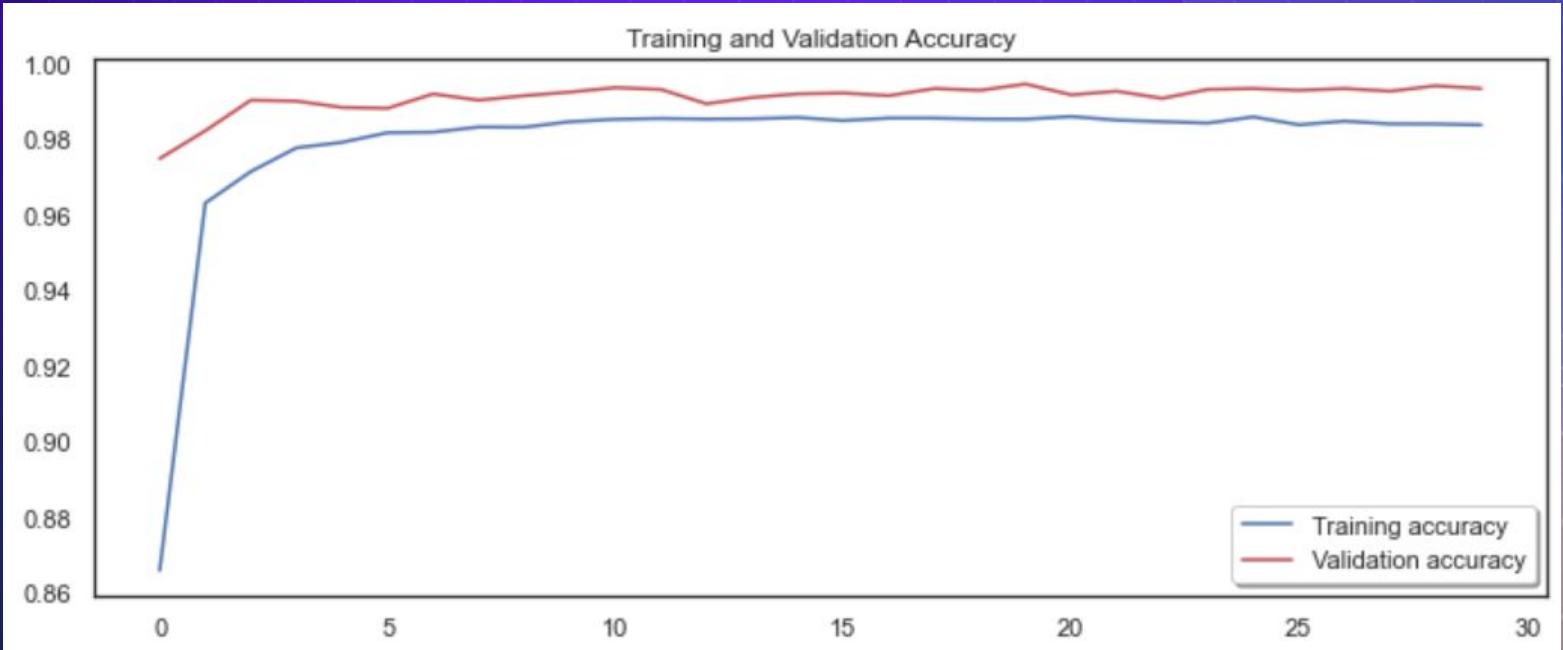


^Training loss generally plateaus after 10 epochs.

ESSLINGEN UNIVERSITY

3. Model Evaluation

Training and Validation Accuracy:

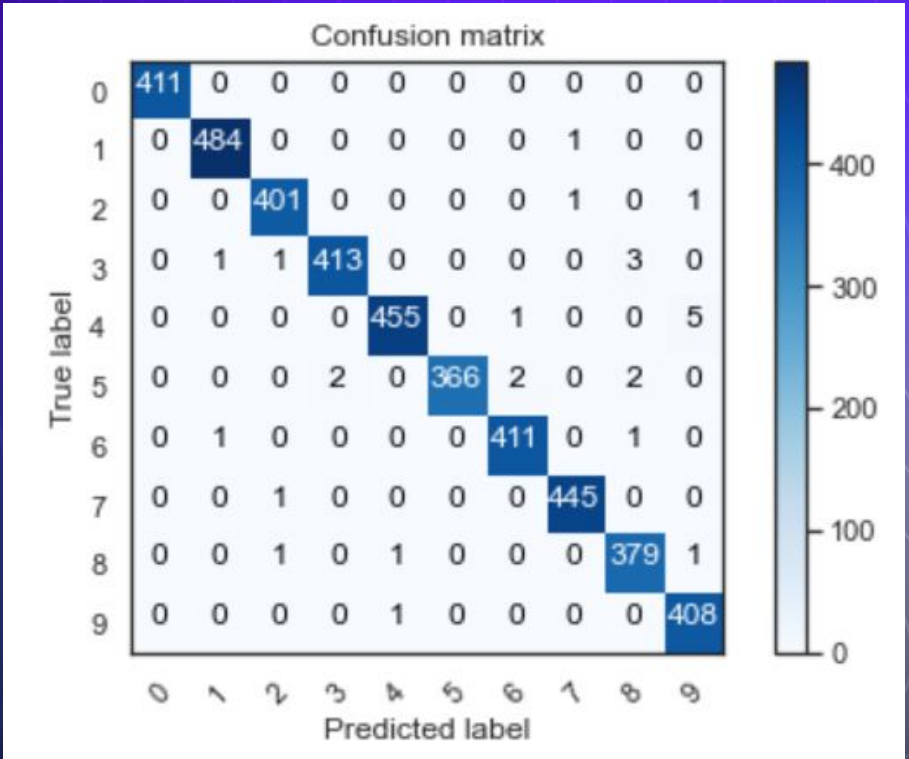


^Training accuracy generally plateaus after 10 epochs.

ESSLINGEN UNIVERSITY

3. Model Evaluation

Confusion Matrix:



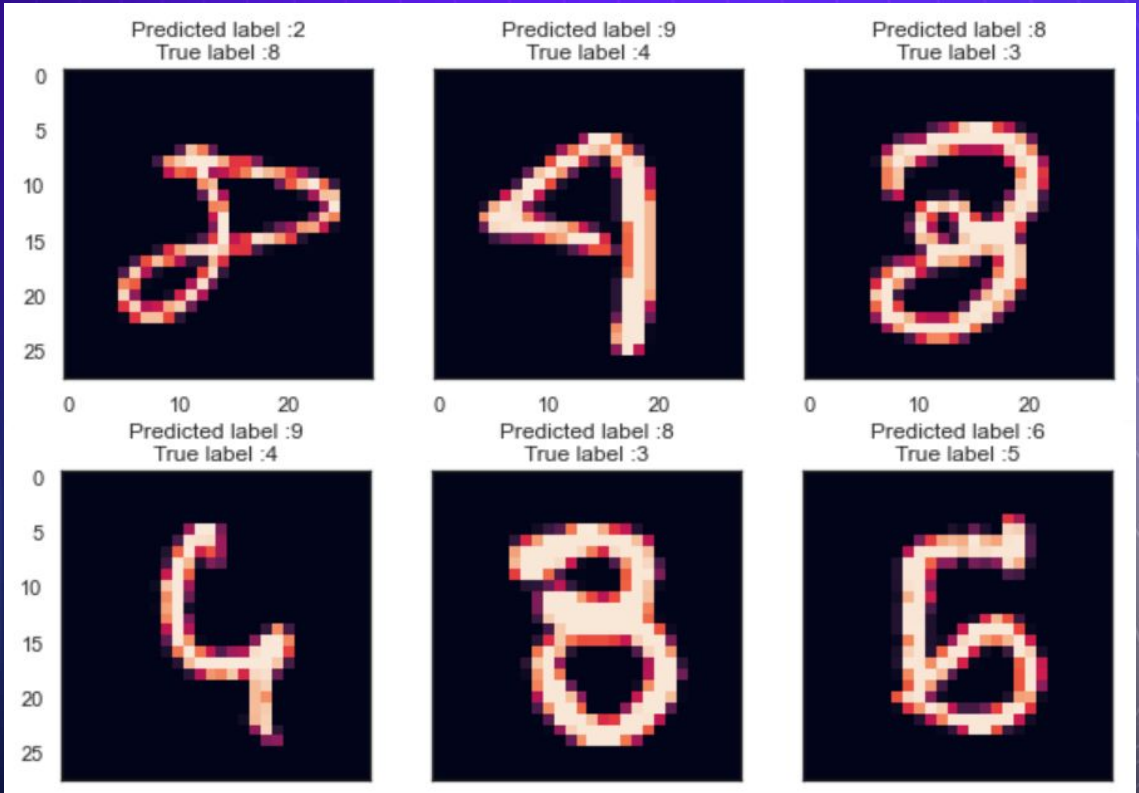
Here we can see that our CNN performs very well on all digits with few errors considering the size of the validation set (4 200 images).

However, it seems that our CNN has some little troubles with the 4 digits, hey are misclassified as 9. Sometimes it is very difficult to catch the difference between 4 and 9 when curves are smooth.

ESSLINGEN UNIVERSITY

3. Model Evaluation

Error Investigation: I need to get the difference between the probabilities of real value and the predicted ones in the results.

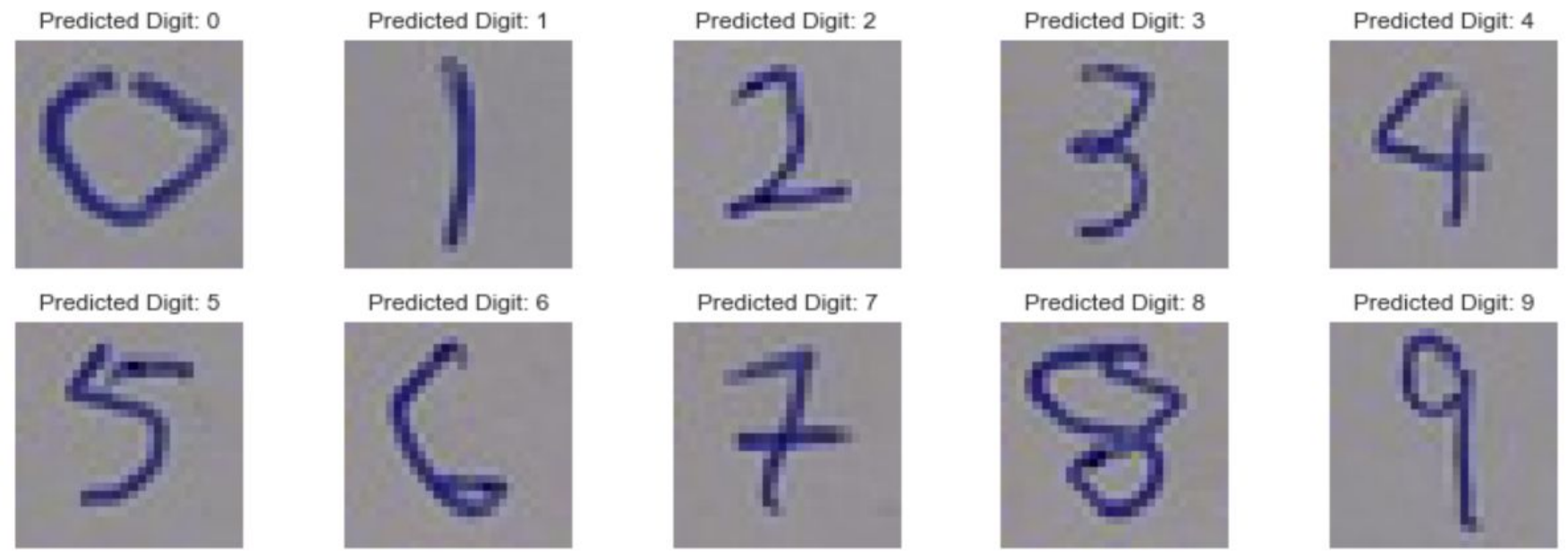


The most important errors are also the most intricate. For these six cases, the model is not absurd. Some of these errors can also be made by humans.

ESSLINGEN UNIVERSITY

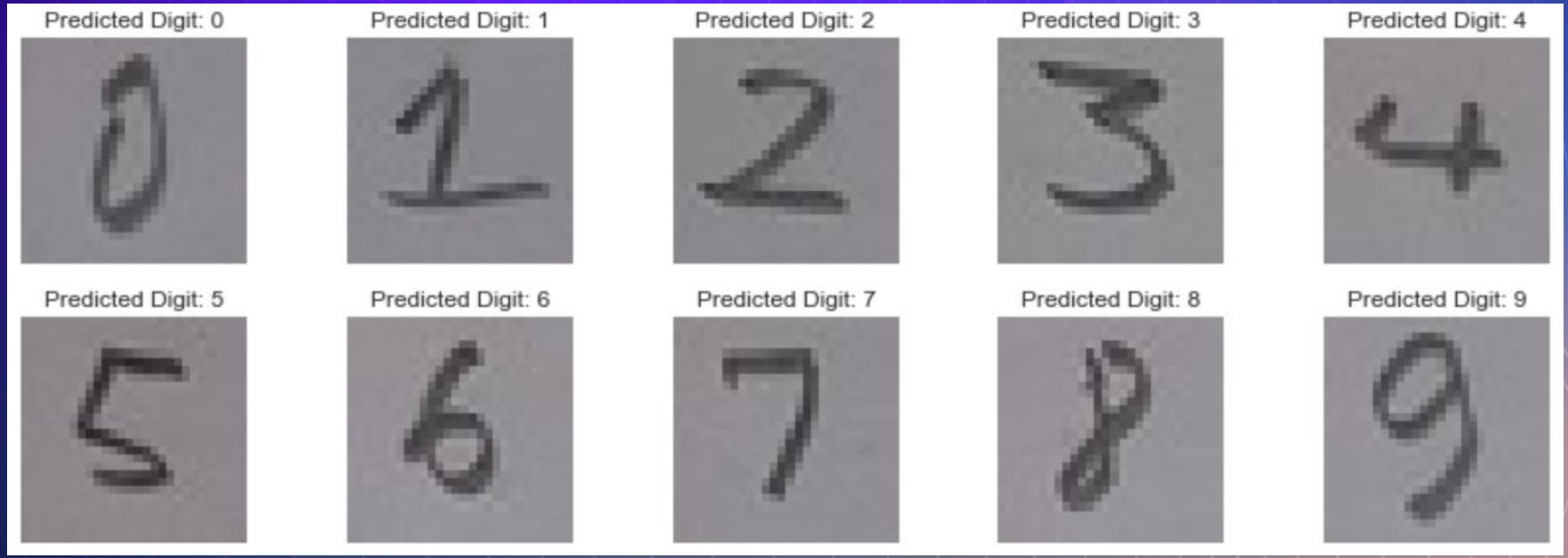
4. Real-Time Prediction

I compiled 3 unique sets of handwritten digits from 0-9 and ran the model against this new loaded image.



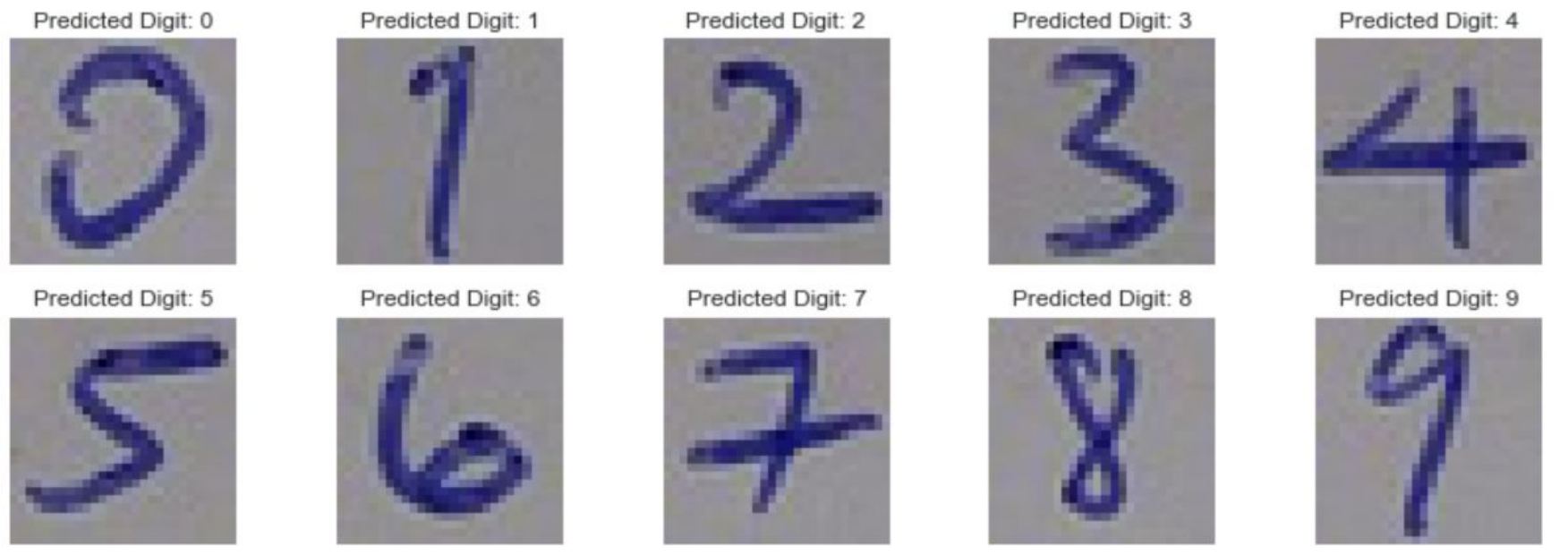
ESSLINGEN UNIVERSITY

4. Real-Time Prediction



ESSLINGEN UNIVERSITY

4. Real-Time Prediction



ESSLINGEN UNIVERSITY

Project Findings

Accuracy

-Accuracy of prediction increases as the number of epochs increases.
-Significance of each consecutive epoch training becomes diminishing.

Reproducibility

-For the model to be applied to new dataset, the dataset must first be resized and grey-scaled to fit the compatibility of the trained model.
-Can be done both automatically or manually.

Error Detection

-Model is not well-trained in recognising unconventional writings.
(e.g. ambiguous handwritings)

ESSLINGEN UNIVERSITY

Implications

- Similar training methods can be used to train other CNN models to recognise and classify images accurately.
- Applies to alphabet recognition or character recognition for other languages.
- Current model can be further built to identify numbers/letters and hence texts from a complex images(e.g. hand-written physical forms).

ESSLINGEN UNIVERSITY

Conclusion

Convoluted Neural Networks(CNN) are fundamental in computer vision and machine learning as the model trains itself from correctly executing its designated task.

With more useful data available for teaching and training the model, its performance improves to a point where it is more accurate and efficient than humans.

ESSLINGEN UNIVERSITY