# PBT205 Project Based Technology Assessment 3

Max Presley

# Introduction

My project develops on charter recognition even further by extending the functionality to include recognition of full words, not just single letters. I used a kaggle dataset with a CNN network to achieve this.

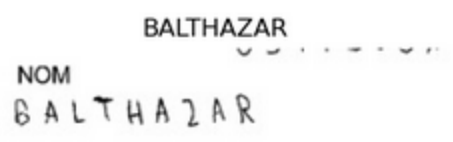
# Dataset

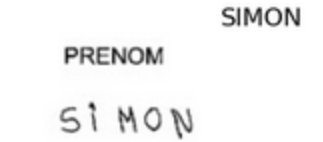
https://www.kaggle.com/datasets/landlord/handwriting-recognition

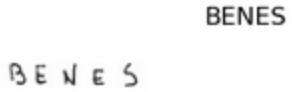
Transcriptions of 400,000 handwritten names

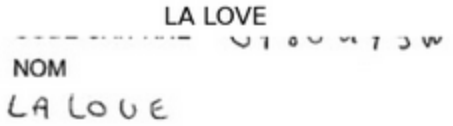
# Explorer data

- I started by exploring the dataset for better general understanding of what needed to be done.



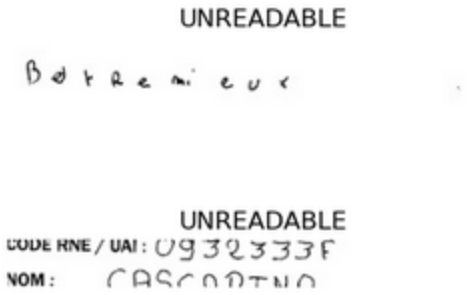



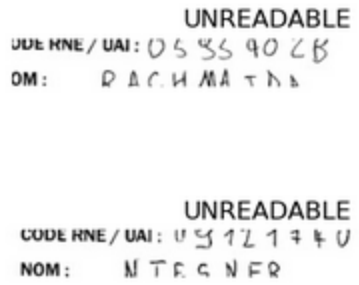


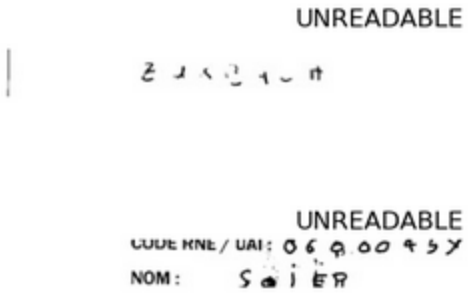
# Pre-process data

I pre-processed the data by checking for any NA data or data that may not be useful in the training process. Dropping any of the found cases

```
Number of NaNs in train set      :  565
Number of NaNs in validation set :  78
```



# Pre-process images

The images also needed some pre processing so that they were more uniform and better to work with. This included cropping and changing to black and white

- Creating a blank white image of the target size.
- Cropping the original image if it is too large.
- Placing the cropped (or original) image on the blank canvas.
- Rotating the image 90 degrees clockwise.

64x256 pixels while preserving as much of the original content as possible

```python
# Pre-process images
def preprocess(img):
    (h, w) = img.shape

    # blank white image
    final_img = np.ones([64, 256])*255

    # crop image
    if w > 256:
        img = img[:, :256]

    if h > 64:
        img = img[:64, :]


    final_img[:h, :w] = img
    return cv2.rotate(final_img, cv2.ROTATE_90_CLOCKWISE)
```

# Pre-training

train_size = 30,000

valid_size = 3,000

- Saving images to a list/array
- Labelling - abcdefg..

```
In [12]: name = 'JEBASTIN'
         print(name, '\n',label_to_num(name))

JEBASTIN
 [ 9  4  1  0 18 19  8 13]
```

# **Model** - CNN-RNN hybrid model

16 layers (including Input and Activation layers)

- **Total Convolutional Layers**: 3
- **Reshape Layer**: 1
- **Dense Layers**: 2
- **Recurrent Layers**: 2
- **Activation Layers**: 4

Total params: 2,406,878

Trainable params: 2,406,430

Non-trainable params: 448

```
input_data = Input(shape=(256, 64, 1), name='input')

# Convolutional Layers
inner = Conv2D(32, (3, 3), padding='same', name='conv1', kernel_initializer='he_normal')
(input_data)
inner = BatchNormalization()(inner)
inner = Activation('relu')(inner)
inner = MaxPooling2D(pool_size=(2, 2), name='max1')(inner)

inner = Conv2D(64, (3, 3), padding='same', name='conv2', kernel_initializer='he_normal')(inner)
inner = BatchNormalization()(inner)
inner = Activation('relu')(inner)
inner = MaxPooling2D(pool_size=(2, 2), name='max2')(inner)
inner = Dropout(0.3)(inner)

inner = Conv2D(128, (3, 3), padding='same', name='conv3', kernel_initializer='he_normal')(inner)
inner = BatchNormalization()(inner)
inner = Activation('relu')(inner)
inner = MaxPooling2D(pool_size=(1, 2), name='max3')(inner)
inner = Dropout(0.3)(inner)

# Reshape Layer
inner = Reshape(target_shape=((64, 1024)), name='reshape')(inner)

# Dense Layer
inner = Dense(64, activation='relu', kernel_initializer='he_normal', name='dense1')(inner)

# Recurrent Layers
inner = Bidirectional(LSTM(256, return_sequences=True), name = 'lstm1')(inner)
inner = Bidirectional(LSTM(256, return_sequences=True), name = 'lstm2')(inner)

# Output Layer
inner = Dense(num_of_characters, kernel_initializer='he_normal',name='dense2')(inner)
y_pred = Activation('softmax', name='softmax')(inner)

# Model Compilation
model = Model(inputs=input_data, outputs=y_pred)
model.summary()
```

# Training

- epochs=10  batch_size=128
- Training of the model took a long time...
- 2.5 Hours for 10 epochs
- Results of Epoch 1:
  `842s 4s/step - loss: 24.8908 - val_loss: 20.7982`
- Ideally would prefer around 50 Epochs

# Evalutation

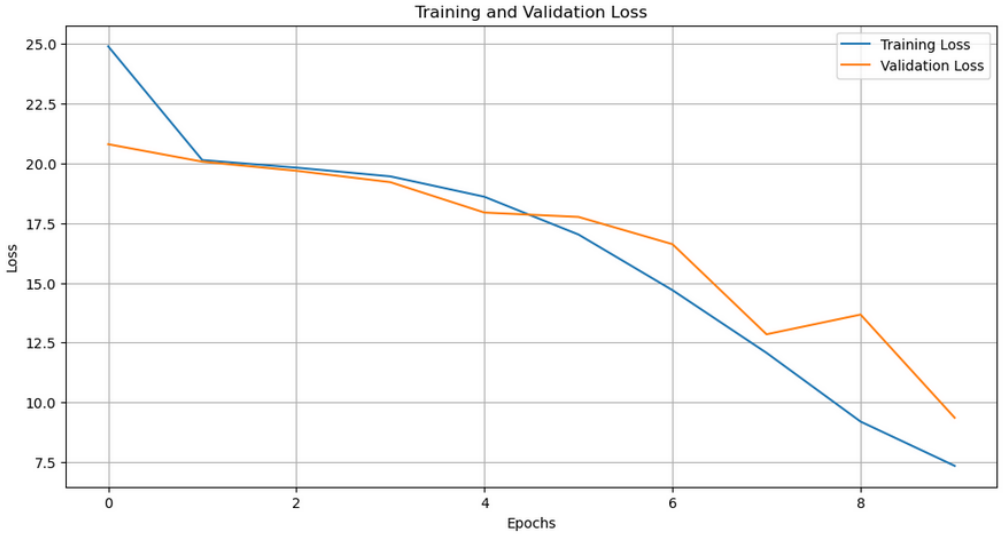
**First Train**

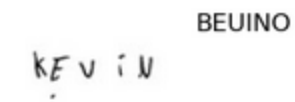
Correct characters predicted : 65.82%
Correct words predicted : 30.21%

**Second Train**

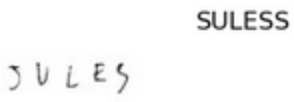
Correct characters predicted : 33.65%

Correct words predicted : 2.10%







# Conclusion

Overall, I processed and managed the data well. However, I was limited by time and hardware which resulted in a long wait times for training and low accuracy of making the predictions. I think I could have achieved a better accuracy if these issues weren't as prevalent. I was also unable to make a GUI for this. Regardless, I found I learned a lot about CNN networks during this time.

# Thank you

Q&A