OCR - Text

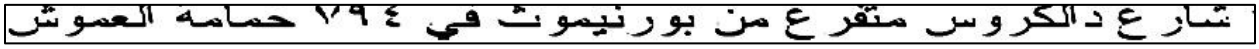
A common problem with digital transformation is to automatically digitize the documents, this is time consuming as someone manually data-entries the information, but as AI emerged, there are a lot of ways to do so without human involvement. Optical character recognition (OCR) is the process of reading the numbers, characters and words from images

Steps to build this OCR Model:

1. **Preprocessing the Data:**
   - Convert colored image to a Grayscale image
   - Convert it from gray to just black and white and remove the background by applying thresholding
   - Enhance the image quality using smoothing and sharpening
   - Perform morphological operations to clean up the image
   - Segment the text part of the image from white background
   - Apply these steps to all the images then save them in another folder for the next step

Before:

After:



2. **Preparing the date for training process:**
   - Load images and texts
   - Normalize the image to be just 0 or 1
   - vectorize the text using lookup function that create a dictionary (map) and replace each character with its index

3. **building the model architecture**
   - Built an CTC loss function in a custom layer
   - Built the model architecture using
     - Convolutional, Max pooling layers to capture image features

- Batch normalization and dropout layers to regularize and prevent overfitting
- Fully connected dense layers
- LSTM layers to capture features of sequence data
- CTC layer to be my loss function

Layer (type)	Output Shape	Param #	Connected to
image (InputLayer)	[(None, 20, 300, 1)]	0	[]
Conv1 (Conv2D)	(None, 20, 300, 16)	416	['image[0][0]']
pool1 (MaxPooling2D)	(None, 10, 150, 16)	0	['Conv1[0][0]']
Conv2 (Conv2D)	(None, 10, 150, 32)	4640	['pool1[0][0]']
pool2 (MaxPooling2D)	(None, 5, 75, 32)	0	['Conv2[0][0]']
Conv3 (Conv2D)	(None, 5, 75, 64)	18496	['pool2[0][0]']
pool3 (MaxPooling2D)	(None, 2, 37, 64)	0	['Conv3[0][0]']
batch_normalization_6 (BatchNormalization)	(None, 2, 37, 64)	256	['pool3[0][0]']
reshape (Reshape)	(None, 128, 37)	0	['batch_normalization_6[0][0]']
dense1 (Dense)	(None, 128, 1000)	38000	['reshape[0][0]']
dropout_6 (Dropout)	(None, 128, 1000)	0	['dense1[0][0]']
batch_normalization_7 (BatchNormalization)	(None, 128, 1000)	4000	['dropout_6[0][0]']
bidirectional_6 (Bidirectional)	(None, 128, 256)	1156096	['batch_normalization_7[0][0]']
bidirectional_7 (Bidirectional)	(None, 128, 128)	164352	['bidirectional_6[0][0]']
dense2 (Dense)	(None, 128, 64)	8256	['bidirectional_7[0][0]']
dropout_7 (Dropout)	(None, 128, 64)	0	['dense2[0][0]']
dropout_7 (Dropout)	(None, 128, 64)	0	['dense2[0][0]']
label (InputLayer)	[(None, None)]	0	[]
dense3 (Dense)	(None, 128, 68)	4420	['dropout_7[0][0]']
ctc_loss (CTCLayer)	(None, 128, 68)	0	['label[0][0]', 'dense3[0][0]']

Total params: 1,398,932
Trainable params: 1,396,804
Non-trainable params: 2,128

- Used For optimization Adam optimizer
- Chose For the learning rate to use scheduler exponential Decay that make the learning rate decreases every training step and that helps the model to keep getting better

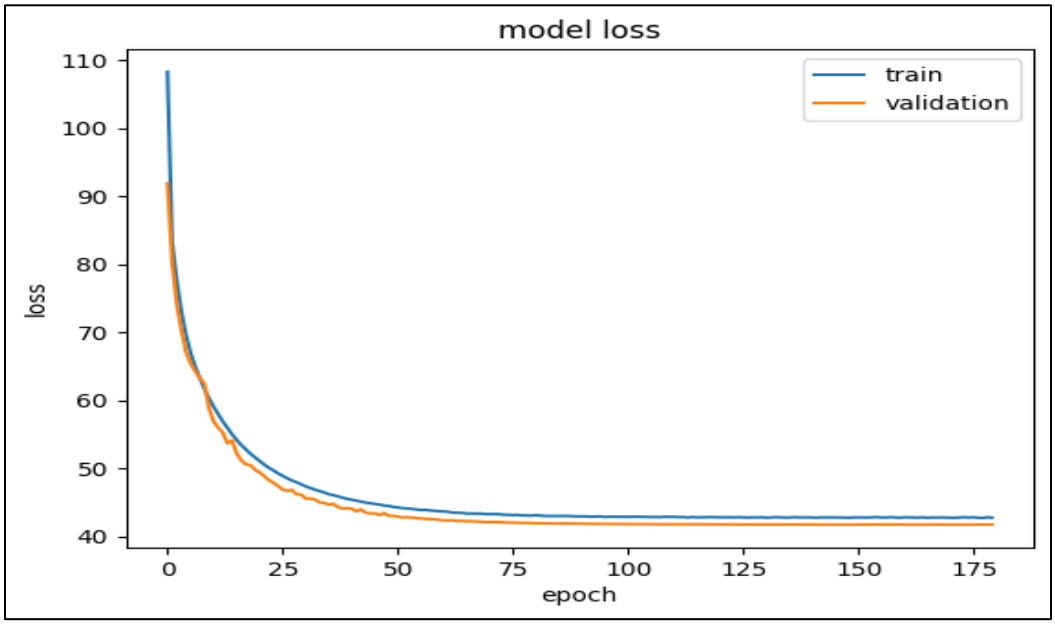
```
# Optimizer.
lr_schedule = keras.optimizers.schedules.ExponentialDecay(
                              initial_learning_rate=0.001,
                              decay_rate=0.90,
                              decay_steps=1000)

opt = keras.optimizers.Adam(learning_rate=lr_schedule)
```

- Make a validation data 10% of the data with batch size of 32

- Create a callback function that stops the training process If the model did not get better by monitoring the validation loss

```
stopping=tf.keras.callbacks.EarlyStopping(
    monitor="val_loss",
    patience=15,
    restore_best_weights=True,
    verbose=1,
    start_from_epoch=3
)
```

- Train the model on our data



4. **Saved the model as h5 and as ONNX file**
5. **Loaded the saved h5 and ONNX model and used them using the test data to make sure every thing works well**
6. **Deploying the model was [optional] I focused on the model and the preprocessing pipeline and I did not get the time to deploy it but it is not a hard thing to be done**