English Handwritten Text Recognition and OCR

Shaimaa Mohamed
Smoha295@uOttawa.ca
300327276

Hassan Elhefny
helhe043@uOttawa.ca
300327238

Yomna Ahmed
yahme022@uottawa.ca
300327217

Nada Abd-Elmageed
nabde013@uOttawa.ca
300327207

***Abstract*— Optical Character Recognition that became more famous in the last few years. It is difficult to translate handwritten text into machine-readable text forms, and handwritten material is of inferior quality to machine-printed text. This paper aims to enhance the prediction of handwritten text using the Neural network on the 'OCR' dataset and using the EasyOCR library to extract the complete text from the handwritten text image, then compare it to our model to see if we can perform better than the pertained "EasyOCR" model. Several businesses, including banking, insurance, and healthcare, need to address this issue.**

Index Terms - text recognition, prediction, Neural network, pertained model

I. INTRODUCTION

This is the era of online banking and healthcare text recognition. The field of text recognition has seen a lot of scientific investigation. Attention mechanisms are at the core of text recognition systems. This research will employ deep learning to develop a model that can recognize English handwriting despite image noise, while standard attention techniques face major alignment issues due to their recurring alignment procedure. This project aims to help predict the text by making good use of Neural Networks over the OCR images dataset then images fed to Neural Networks to predict the character and text. Optical character recognition [1] is one of the automatic identification techniques that can be used in a variety of applications (OCR). OCR enables a machine to read data in any format from natural scenery or other resources. Character recognition during typing and printing is simply because of the character's well-defined size and shape. The aspects of handwriting differ among individuals. To recognize a character, the handwritten OCR algorithm must consequently work hard to comprehend this distinction. Every character in our dataset has an image, and we utilize these images to train the model and enable it to identify these characters in the image. Images of English names written by hand are included in the second dataset, which is divided into three sections for training, testing, and validation. We have used Handwritten Character Recognition Model and Easy OCR-pertained model. The handwritten OCR systems and the methods involved in OCR one of the automatic identification techniques were covered in this paper.

II. DATASET DESCRIPTION

The data set includes all English letters and numbers (0–9) as well as some special characters (@, #, $, and &). The pictures are in black and white and are 32 by 32 pixels RGB 'Three channels'. There are 39 categories in total. 9 for Digits, 26 for Alphabets (which mix tiny and capital letters to form a single class of each character) (i.e., 1 to 9). The digit 0 is coupled with the character O category to prevent misclassification. and some unique ones (@, #, $, &). The dataset contains photos for every character, and we use these images to train the model and let it recognize these characters in the image. The second dataset, which is split into three parts for training, testing, and validation, includes pictures of English names written by hand. The text from each image is included in the ".csv" file that corresponds to each section. The Train and Validation files contain 0.8+ million training records and 20,000+ validation records. It was collected from Kaggle. Applying feature engineering and label binarization on the dataset.
We convert all images to be in Grayscale 'only one channel' and resize all images to be '32 * 32 * 1'.

III. RELATED WORK

In the field of computer vision, there is a lot of research interest in text recognition. This architecture is shown in the figure. 1 is used to get a text from images using the concept of Image to Sequence to get the sequence of characters that

create words and words that create sentences [2]. Then use the concept of Formatting and Auxiliary Markup to only get a text from text regions and discard any un-transcribed regions. This architecture did not use any image segmentation but used image sequence to get the sequence of characters. This architecture gave them an average error of 4.4% in a single line and 5.5% in a paragraph level [2]. Compared with the best cloud API 'Wiki Text,' the mean character error for cloud API was 14.4% but the model gave them 7.6% as a mean character error. This architecture is used in multiline English handwritten text recognition because it uses a different structure of image segmentation. This architecture isn't good enough in single-word handwritten recognition because image segmentation gave us less Character Error Rate (CER) so in our project, we will use another architecture that uses image segmentation [2]. The authors proposed a new method called decoupled attention network (DAN) for text recognition by decoupling the decoder of the traditional attention mechanism into an alignment module and a decoupled text decoder. DAN is effective and has high performance in recognition tasks, including handwritten text recognition and regular/irregular scene text recognition. Which consists of three components as shown in the figure. 2: feature encoder depends on the convolutional neural network (CNN) to extract visual features from the input image. Convolutional alignment module (CAM): it uses multi-scale visual features from the feature encoder as input instead of the traditional score-based recurrency alignment module, this method constructs attention mappings using a fully convolutional network in a channel-wise way. Decoupled text decoder: It uses a gated recurrent unit with the feature map, attention maps, and to make the final prediction (GRU). Decoupled Attention Network (DAN) decoupling the decoder of the traditional attention mechanism into an alignment module and a decoupled text decoder. DAN is effective and has high performance in recognition tasks, including handwritten text recognition and regular/irregular scene text recognition [3].

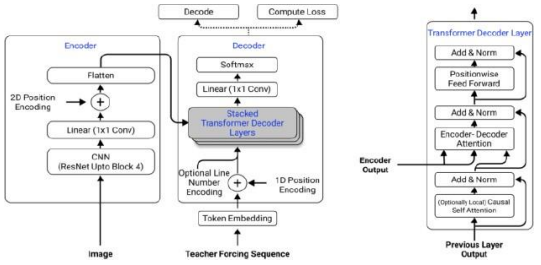

Fig. 1 CNN Encoder and Transformer Decoder

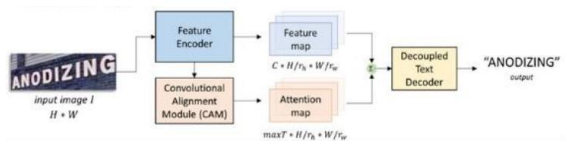

Fig. 2 Architecture of DAN

IV. METHODS

We use two Convolutional Neural Network models, the first one consists of:

1. Three Conv2d Layers
2. Three Max Pooling layers
3. Two Drop Out layers
4. Flatten Layer
5. Two Dense Layers contain the Output Layer

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 32)        64

 max_pooling2d (MaxPooling2D  (None, 16, 16, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 14, 14, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 7, 7, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 5, 5, 128)         73856

 max_pooling2d_2 (MaxPooling  (None, 2, 2, 128)        0
 2D)

 dropout (Dropout)           (None, 2, 2, 128)         0

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 128)               65664

 dropout_1 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 35)                4515

=================================================================
Total params: 162,595
Trainable params: 162,595
Non-trainable params: 0
_________________________________________________________________
```

Fig 3. the structure of model 1

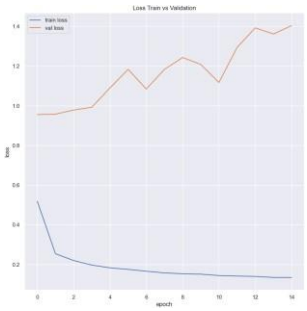

Fig 4. Train vs Validation Loss

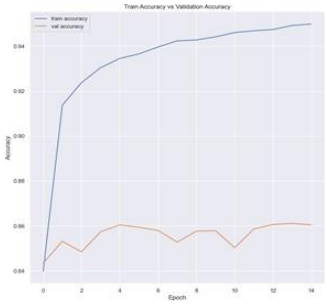

Fig 5. Train vs Validation Accuracy

The second model consists of the same layers in model one and some additional layers like:

1. Layer Normalization
2. Batch Normalization

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 32, 32, 64)        128

 max_pooling2d_6 (MaxPooling  (None, 16, 16, 64)       0
 2D)

 conv2d_7 (Conv2D)           (None, 14, 14, 128)       73856

 max_pooling2d_7 (MaxPooling  (None, 7, 7, 128)        0
 2D)

 conv2d_8 (Conv2D)           (None, 5, 5, 256)         295168

 max_pooling2d_8 (MaxPooling  (None, 2, 2, 256)        0
 2D)

 batch_normalization_1 (Batc  (None, 2, 2, 256)        1024
 hNormalization)

 dropout_4 (Dropout)         (None, 2, 2, 256)         0

 flatten_2 (Flatten)         (None, 1024)              0

 dense_4 (Dense)             (None, 128)               131200

 layer_normalization_1 (Laye  (None, 128)              256
 rNormalization)

 dropout_5 (Dropout)         (None, 128)               0

 dense_5 (Dense)             (None, 35)                4515

=================================================================
Total params: 506,147
Trainable params: 505,635
Non-trainable params: 512
_________________________________________________________________
```

Fig 6. the structure of model 2

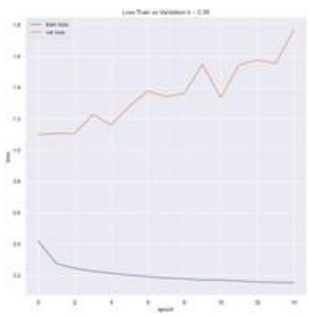

Fig 7 Train vs Validation Loss

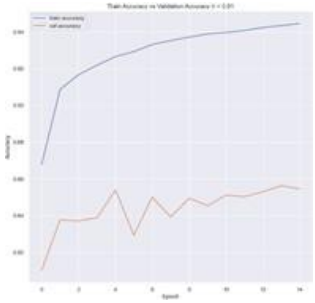

Fig 8. Train vs Validation Accuracy

V. EXPERIMENTS

1. Change Optimizer
   In two models we use the 'Adam' optimizer, so we try using the 'SGD' optimizer and make 'Nesterov ' True. The result changed and the performance of the model was not good.

Optimizer	Mean Accuracy
SGD Optimizer	92.6 %
Adam Optimizer	93.25 %

Table 1. Optimizer vs Mean Accuracy

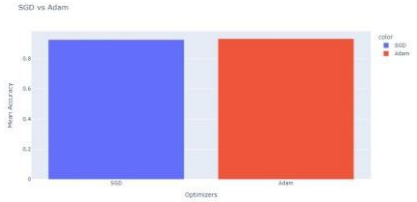

Fig 9. Optimizer vs Mean Accuracy

2. Tune Learning Rate using Adam Optimizer
   We use the 'Adam' optimizer because it gave us good results than the 'SGD' optimizer and tune the learning rate. We use the learning rate from a list consisting of [0.1, 0.01, 0.3, 0.03, 0.9, 0.09].

LR	Mean Accuracy
0.1	87.4%
0.01	92.8%
0.3	86.2%
0.03	90.97%
0.9	86.4%
0.09	87.7%

Table 2. LR vs Mean Accuracy

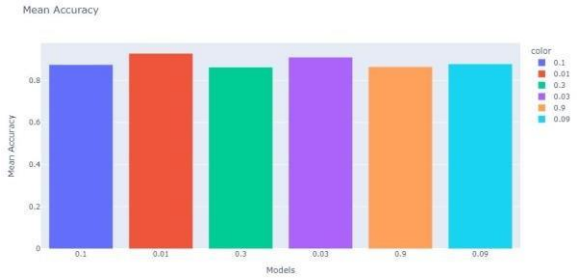

Fig 10. LR vs Mean Accuracy

From the figure above, we find that the best value for the learning rate is 0.01.

Final Model Hyperparameters

optimizer	ADAM
learning rate	0.01

Table 3. Best Model Hyperparameters

Final Model

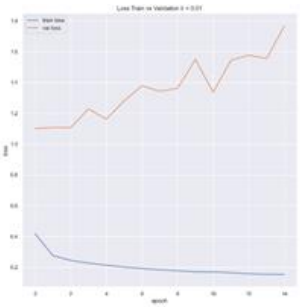

Fig 10 Train vs Validation Loss

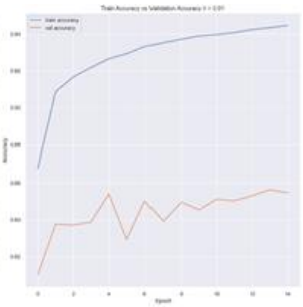

Fig 11. Train vs Validation Accuracy

Result of Final Model

After using the model and using Region of Interest (ROI) in the image, simulating the decoder, and making our model able to deal with English words based on the direction of the English language 'left to right, our model predicts handwritten names like the figure below.

Fig 12 result from model

Easy-OCR

Easy-OCR is a pre-trained model and we try to improve the results of the Easy-OCR model using a simple CNN model depending on simulating the decoder structure used in Easy-OCR.

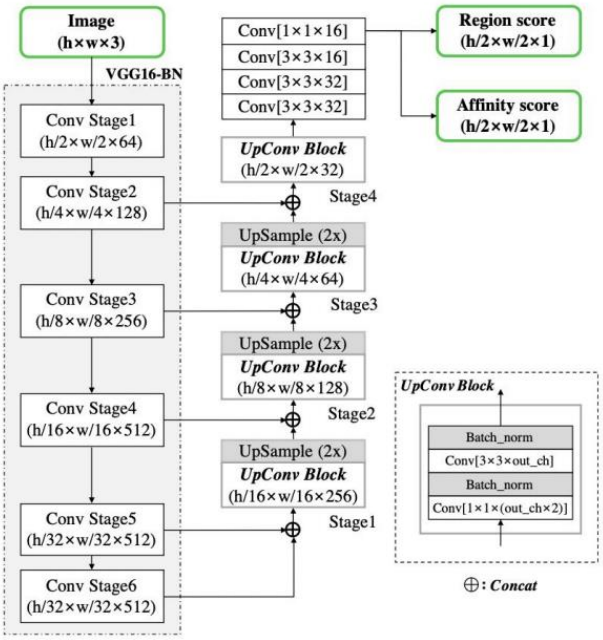

Fig 13 Easy OCR architecture

VI. EVALUATION

OCR (Optical Character Recognition) converts scanned graphical text into editable computer text. This can significantly improve the usability of digitized documents, allowing for more efficient searching and other NLP applications. Punctuation errors, case sensitivity, character format, word meaning, and segmentation errors occur when spacings in different lines, words, or characters cause the misrecognition of white spaces. Other causes of OCR errors include font variation across different materials, historical spelling variations, material quality, and language specific to different media texts. The accuracy metric is commonly used to evaluate prediction output, where we indicate a match (1) ora no match (0). However, this does not provide enough granularity to effectively assess OCR performance. We used error rates to determine how much the OCR transcribed text and ground truth text (i.e., manually labeled reference text) differed. A common instinct is to count the number of misspelled characters. Although this is correct, the actual error rate calculation is more complicated. This is because the OCR output may differ in length from the original text. When it comes to Handwritten OCR accuracy, two objective metrics are used to evaluate how trustworthy OCR is: Character Error Rate (CER) and Word Error Rate (WER). CER counts the number of character-level operations such as substitutions, insertions, and deletions of characters required to transform the ground truth text into the predicted text from the model.

$$CER = \frac{S + D + I}{N}$$

Character Error Rate (CER) formula

where:

S = Number of Substitutions
D = Number of Deletions
I = Number of Insertions
N = Number of characters in reference text (aka ground truth)

And WER also counts the number of word-level operations required to transform the ground truth text into the OCR model prediction.

$$WER = \frac{S_w + D_w + I_w}{N_w}$$

Word Error Rate (WER) formula

WER uses the same formula as CER, but it operates at the word level instead. It is the number of word substitutions, deletions, or insertions required to convert one sentence to another.

	OUR MODEL	**EASY OCR**
Character Error Rate:	0.56	0.48
Word Error Rate:	0.97	0.90

Table 4. CER and WER

VII. CONCLUSION & FUTURE WORK

In this paper, a flexible convolutional neural network is proposed for text recognition. seeks to improve handwritten text prediction using the Neural network on the OCR dataset and the EasyOCR library to extract the whole text from the handwritten text image, then compareit to our model to see if we can outperform the existing EasyOCR model.
Our simple model simulates the GRU decoder in EasyOCR pre-trained model depending on two importantthings:

- Direction of English language ‘left to right’
- Region Of Interest ‘ROI’ by getting the effective region of the image by calculating the change in color between white and black and focusing on black regions because it’s our target in the images.

After getting the important regions of the image we use our model to predict each character and combine them tocreate an English word.
Tune hyperparameters of our model based on two important things:

- Optimizer
- Learning Rate

And found that the best value for the learning rate was 0.01 and the best optimizer was ‘Adam’ so we build a simple model with these hyperparameters.
We also add some regularization terms in our model second model by adding:

- Batch Normalization Layer
- Layer Normalization Layer

To improve the performance of our model.
Our model needs to train on a large amount of handwritten characters images to be able to outperform Keras-OCR pre-trained model and to do that there are twoplans:

- Use data augmentation to create a lot of imagesfor a handwritten character’s images.
- Use federated learning in handwritten character images for more than one person and combine all models to make a final model able to get all characters in the image and train in all types of handwritten images thanks to federated learning that make us able to learn a model for each person and combine them after all models train.

VIII. REFERENCES

[1] K. &. Kanagarathinam, K. &. Ravindrakumar, and S. Ilankannan, “Steps Involved in Text Recognition and Recent Research in OCR,” *A Study*, vol. 8, pp. 3095–3100, 2019.

[2] “Full-page handwriting recognition via an image to sequence extraction,” *Papers With Code*, 2022.

[3] “[4] K. Leung, ‘Evaluate OCR Output Quality with Character Error Rate (CER) and Word Error Rate (WER),’ Towards Data Science, 24-Jun-2021. [Online]. Available: https://towardsdatascience.com/evaluating-ocr-output- quality-with-character-error-rate-cer-and-word-error-rate-wer-853175297510. [Accessed: 01-Dec-2022].” .

[4] S. Mutuvi, A. Doucet, M. Odeo, and A. Jatowt, “Evaluating the impact of OCR errors on topic modeling,” in *Lecture Notes in Computer Science*, Cham: Springer International Publishing, 2018, pp. 3–14.

[5] R. F. Wheeling, “Optimizers: Their structure,” *Commun. ACM*, vol. 3, no. 12, pp. 632–638, 1960.

[6] “Plotly,” *Plotly.com*. [Online]. Available: https://plotly.com/python/plotly-express/. [Accessed: 01-Dec-2022].

[7] G. Blokdyk, *TensorFlow Containers A Complete Guide - 2019 Edition*. 5starcooks, 2019.

[8] T. Wang *et al.*, *Papers with code - decoupled attention network for text recognition. Decoupled Attention Network for Text Recognition | Papers With Code*. 2022.