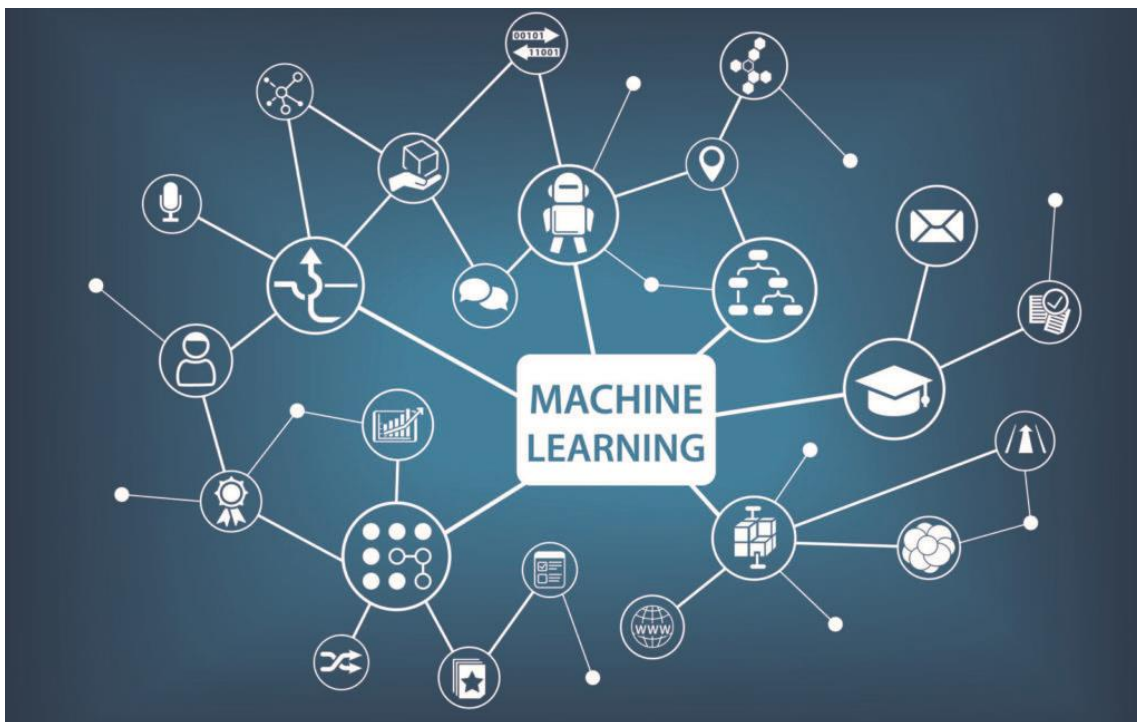


Data Representation, Reduction and Analysis

Project: OCR FOR HANDWRITING



Name: Vaibav Paliwal, Oumbe
Gabin, Laurent Nlemba

DECLARATION

This is to declare that the work done and displayed via this report is performed by **Vaibav Paliwal, Oumbe Gabin, Laurent Nlemba**. This report is a part of the course Data Representation and Analysis which comes under the program of Master of Science: Applied Computer Science at the Vrije Universiteit Brussel.

INTRODUCTION

Scanning a document allows us to obtain a numeric version of a paper document. Being able to edit this document is even better. For instance, we can think about benefit such as searchability in a text, text editing, mathematical operations, etc. This is possible with computer recognition of printed or written character called optical character recognition.

Optical character recognition (OCR) has always occupied a central place in the next generation user interface. Earlier in the years of the 1920s, the Austrian engineer Gustav Tauscher developed the first device that was able to recognize characters. This device was called Reading Machine and obtained a patent in Germany in 1929. In 1933, Paul Handel continued along the same line with the invention of the Statistical machine. Nowadays, the development of machine learning and deep learning techniques allows going a step further with more accurate optical character recognition.

Optical character recognition systems

Most of the OCR systems are based on text detection and character recognition. The recognition phase involves 5 steps:

- Data acquisition: The validation with extra data was done with images acquired through the Webcam camera with a 1440*1919 resolution. The user captures the images by presenting a sheet with the handwritten letter and pressing the keyboard letter "c".
- Pre-processing with scale conversion, normalization and noise removal. This step consists in resizing the images to the format (28*28) of the OCR database if necessary and adding transformation (rotation, equalization) when needed.
- Segmentation, if the input consists of words or sentences. The goal of this operation is to perform identification of characters.
- Feature extraction;
- Classification, where different techniques can be used such as: Convolutional Neural Network (CNN), Neural Network (NN), Support Vector Machine (SVM), K-Nearest Neighbours (KNN), etc.

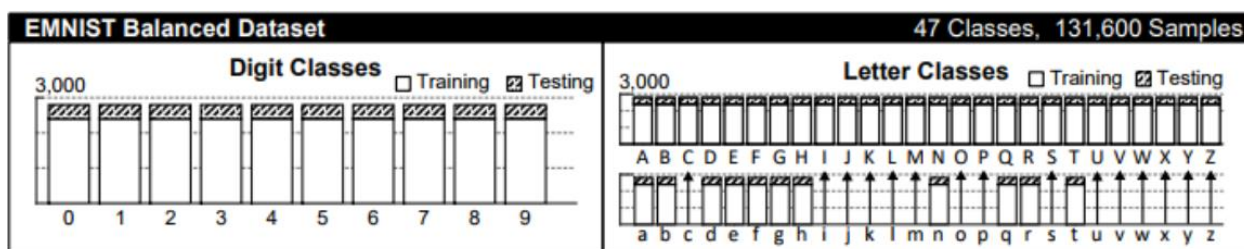
Objectives of the project

In this project, we will build an OCR system, employing CNN models. We will use the EMNIST balanced dataset that intended to be the most widely applicable dataset as it seeks to further reduce the errors occurring from case confusion by merging all uppercase and lowercase classes to form a balanced 26-class classification task.

Our OCR system will be able to recognize in first instance printed letters, captured by a laptop camera (image acquisition) and then to proceed to some pre-processing to make it similar to images of the EMNIST dataset.

Dataset

For our study, we use the EMNIST balanced dataset. The EMNIST Balanced dataset is meant to address the balance issues in the ByClass and ByMerge datasets. It is derived from the ByMerge dataset to reduce mis-classification errors due to capital and lower case letters and also has an equal number of samples per class. This dataset is meant to be the most applicable. It contains 112 800 train data, 18 800 test data, and 47 balanced classes.



Results and discussion

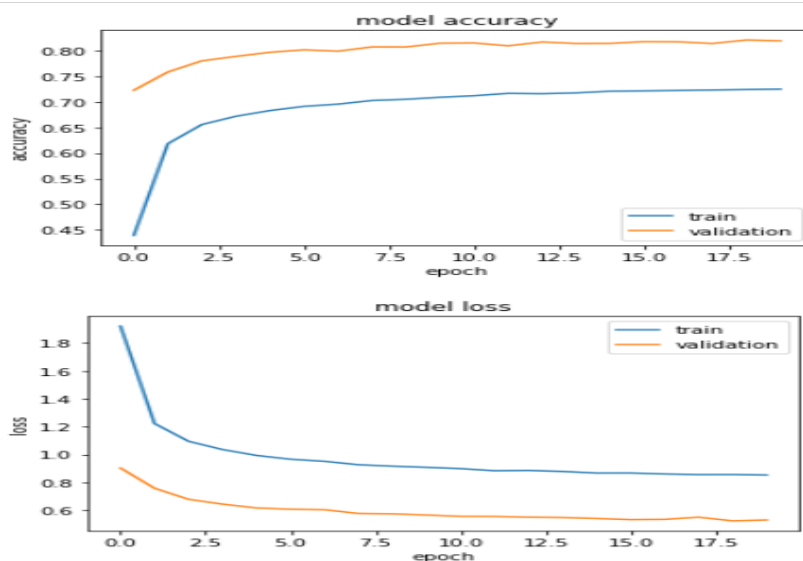
RESULT ANALYSIS: The models used to train the EMNIST balanced dataset are presented in the table below. They are all based on the CNN model.

Model Name	No. of layers	Conv Layer	Dropout	Flatten	Maxpooling	Dense	BatchNormalization	Test_Result
Model2	11	3	3	1	3	1	0	81.08%
Model3	5	3	0	1	0	1	0	83.47%
Model4	8	3	3	1	0	1	0	86.19%
Model5	19	7	3	1	0	1	7	89.48%
Model6	11	3	3	1	0	1	3	86.33%
all_cnn	21	8	3	1	0	1	8	89.31%
create_complex_model	25	10	3	1	0	1	10	89.88%

Model Analysis:

Let's have a look on the loss and accuracy graph of all the models present in the table. We have also calculated the accuracy for test data and visualize some of them.

Model2: The loss and accuracy graph look like.



Testing data result:

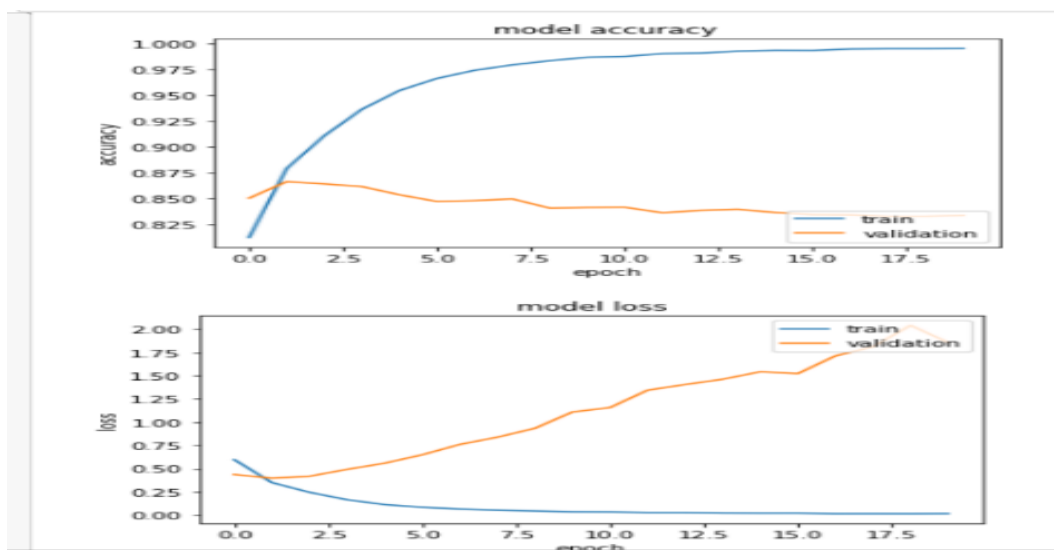
```
In [31]: eval_model(model2,test_x,test_y)
```

```
18799/18799 [=====] - 5s 289us/sample - loss: 0.5635 - acc: 0.8109  
The accuracy of the model is: 0.8108942
```

```
Out[31]: [0.5634857957791417, 0.8108942]
```



Model3: The loss and accuracy graph look like.



Testing data result:

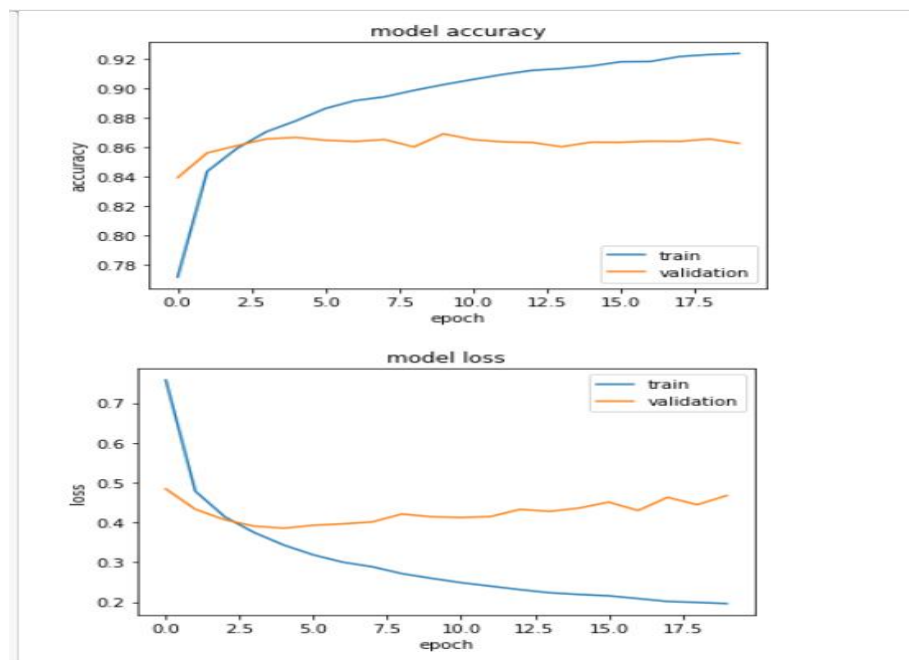
```
In [21]: eval_model(model3,test_x,test_y)
```

```
18799/18799 [=====] - 23s 1ms/sample - loss: 1.8857 - acc: 0.8348
The accuracy of the model is: 0.8347784
```

```
Out[21]: [1.8857239498103626, 0.8347784]
```



Model4: The loss and accuracy graph look like.



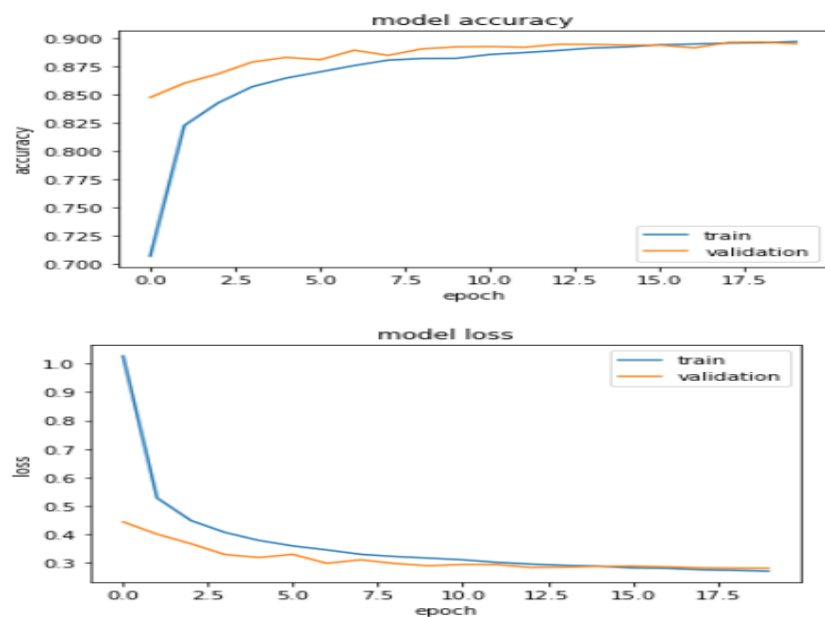
Testing data result:

```
In [26]: eval_model(model4,test_x,test_y)

18799/18799 [=====] - 26s 1ms/sample - loss: 0.4924 - acc: 0.8620
The accuracy of the model is: 0.86196077

Out[26]: [0.49239890185199686, 0.86196077]
```

Model5: The loss and accuracy graph look like.



Testing data result:

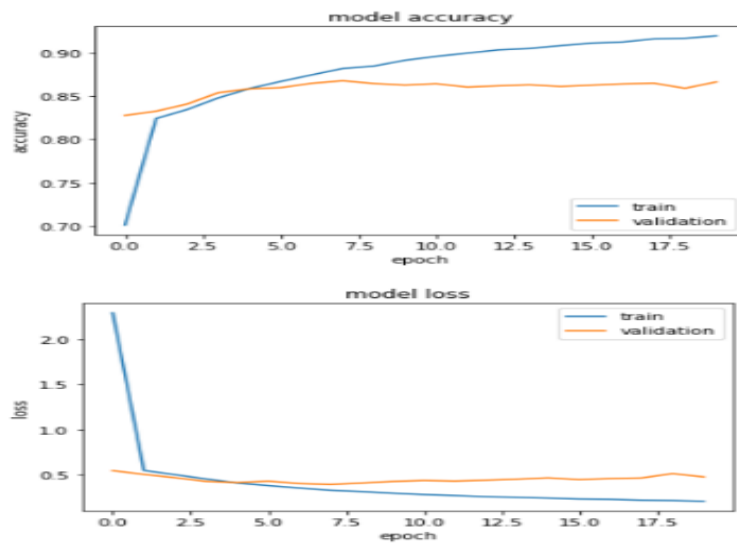
```
In [32]: eval_model(model5, test_x, test_y)
```

18799/18799 [=====] - 56s 3ms/sample - loss: 0.2953 - acc: 0.8948
The accuracy of the model is: 0.8948348

```
Out[32]: [0.295271482146435, 0.8948348]
```



Model6: The loss and accuracy graph look like.



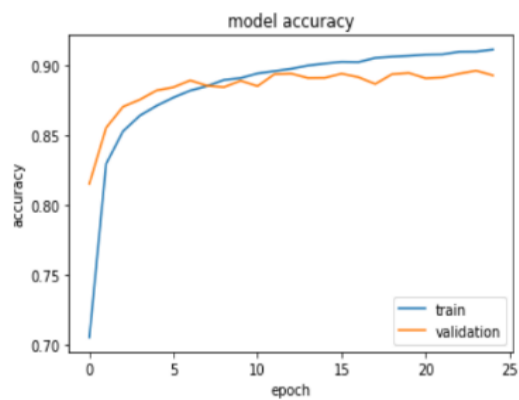
Testing data result:

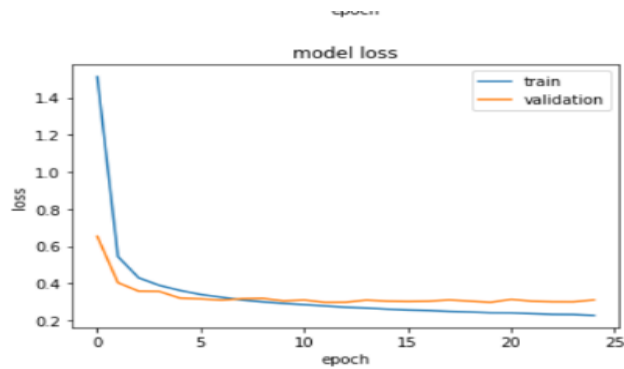
```
In [38]: eval_model(model6, test_x, test_y)
```

```
18799/18799 [=====] - 48s 3ms/sample - loss: 0.5050 - acc: 0.8634  
The accuracy of the model is: 0.863397
```

```
Out[38]: [0.5050101600461732, 0.863397]
```

All_CNN: The loss and accuracy graph look like.



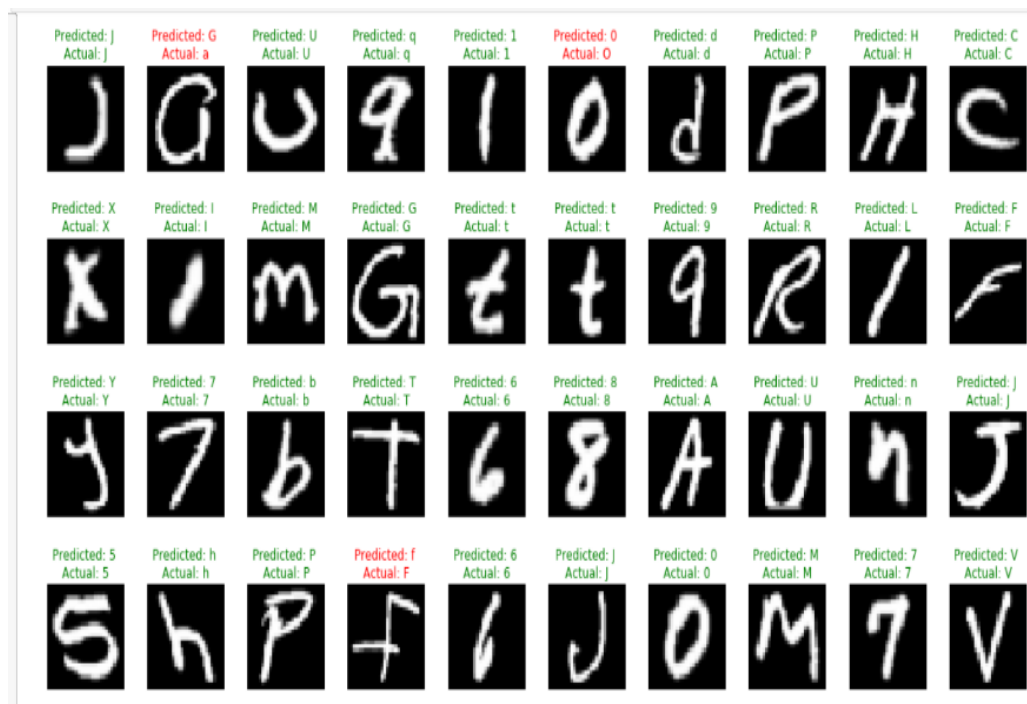


Testing data result:

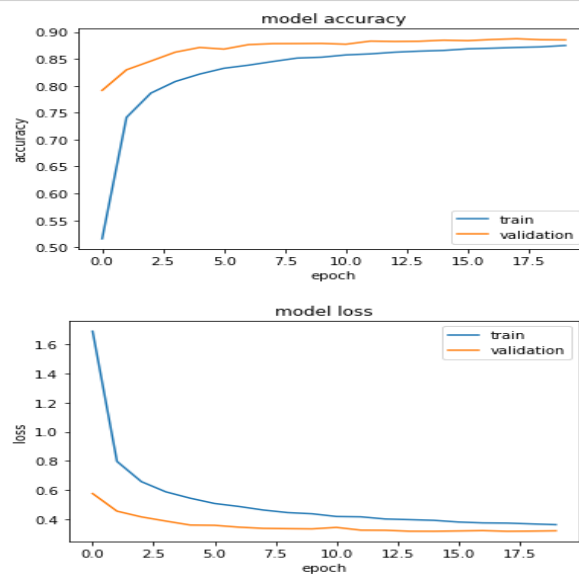
```
In [23]: eval_model(loader_model4, test_x, test_y)
```

18799/18799 [=====] - 54s 3ms/sample - loss: 0.3277 - acc: 0.8932
The accuracy of the model is: 0.8931858

```
Out[23]: [0.32768681388730997, 0.8931858]
```



Create_complex_model: The loss and accuracy graph look like.



Testing data result:

```
In [44]: eval_model(loader_model3, test_x, test_y)
```

18799/18799 [=====] - 30s 2ms/sample - loss: 0.3062 - acc: 0.8989
The accuracy of the model is: 0.8988776

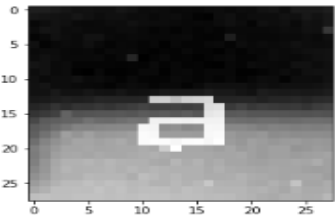
```
Out[44]: [0.30622104222175, 0.8988776]
```

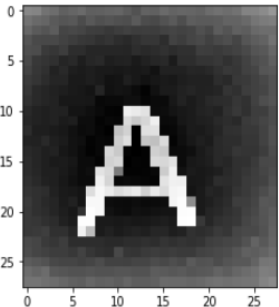


Final Work: Predicting new images.

In this section we saved some of our models and used them to predict some new images that we took from camera.

Result: For the below images we used ("complex_emnist.h5") model which is basically the (create_complex_model).

```
In [97]: image = ("sample.jpg")
In [98]: img = reshape(image)
img.shape
(4632, 3474, 3)
(28, 28)

Out[98]: (1, 28, 28, 1)
In [99]: predictions = loaded_model.predict(img)
print(np.argmax(predictions))
36
In [100]: str((classes[np.argmax(predictions)]))
Out[100]: 'a'
```

```
In [41]: image1 = ("7.jpg")
img1 = reshape(image1)
img1.shape
(1919, 1440, 3)
(28, 28)

```

Out[41]: (1, 28, 28, 1)

```
In [46]: predictions1 = loaded_model.predict(img1)
str((classes[np.argmax(predictions1)]))
```

Out[46]: 'A'

As you can see our model has successfully predicted the images and the result is also true.

CONCLUSION

In this project, we built an OCR system based on a CNN architecture that recognize handwritten characters captured through a webcam. A comparative process between the models evaluated showed that an increase in the number of convolution layers improved the performance of the system in contrast to dropout that didn't bring significant improvements. Moreover the quality and precision of the preprocessing appear to have a great incidence in the ability of our system to make accurate precision since luminosity, distortion and resolution of captured images.

The addition to our system of an identification component that would identify characters present in an image will make it more interesting in a real life context. Moreover, considering captured images in addition to EMNIST images in the training step might improve the robustness of our system.

REFERENCES

<https://www.kaggle.com/crawford/emnist>

<https://www.tensorflow.org/>

<https://keras.io/>

<https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148>

<https://history-computer.com/ModernComputer/Basis/OCR.html>

BHATT, Preeti P. et PATEL, Isha. Optical Character Recognition Using Deep Learning-A Technical Review. National Journal of System and Information Technology, 2018, vol. 11, no 1, p. 55.

Driss, S. Ben, et al. "A comparison study between MLP and Convolutional Neural Network models for character recognition." Real-Time Image and Video Processing 2017. Vol. 10223. International Society for Optics and Photonics, 2017.