Deep Learning Project in Handwritting Recognition Models

Limeng Liu
Indiana University - Bloomington
Bloomington, United State
liulim@iu.edu

***Abstract*—This document is a final report for Big Data Application and Deep Learning course project in deep learning application. Using TensorFlow and Keras to build neural networks to train text in-line image data. Compare three different neural network models and their accuracy rate.**

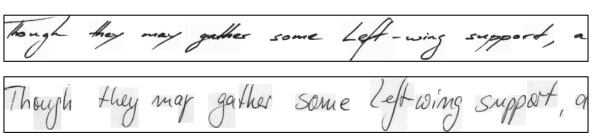
***Index Terms*—deep learning, handwritten, recognition, text**

I. Introduction

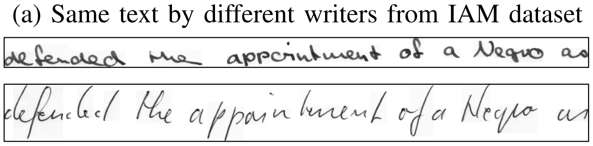
Deep learning is a branch of machine learning that works by mimicking neurons in neural networks that exist in the human brain. Deep learning models are primarily used in the field of "computer vision," which allows computers to see and visualize like human beings. The deep learning model is a kind of artificial neural network. The deep learning algorithm will gradually learn the image as it passes through each layer of the neural network. Early layers learn how to detect low-level features, such as the edges of objects, and later layers combine features from earlier layers into a more complete and complete representation.

This project aims to address the problem of handwritten text identification using off-line handwritten text recognition database images. Writer can be recognized by capturing characteristics of handwriting habits of one author, which differ from other authors. [1]

The challenge of the project is how to capture handcrafted features which are vary for different samples' writing habits, for example Figure 1 shows two examples of handwritten English sentences by different writers. As we can see, these handcrafted features includes not only the contents (objective texts) and styles (personal, subjective), which may increase the difficulty of using deep learning models to recognize texts.



(a) Same text by different writers from IAM dataset



(b) Same text by different writers from IAM dataset

Fig. 1. Different writer examples from IAM dataset

To solve the challenging problem, this paper leverages deep learning model: CNNs (Convolutional Neural Network) as primary model to learn effective representations for handwritten recognition. CNNs have demonstrated its effectiveness in various computer vision problems by improving state-of-the-art results with a large margin in image classification [2] [3] and handwriting recognition [4].

I planned to create a CNNs model which takes multiple local regions as input and trained with softmax loss on identification. I evaluate the methods on IAM dataset [5]. I trained three different CNNs models and compare the accuracy rate for the three models. The highest accuracy rate I achieved for 5 epochs is approximately 87.95% from IAM dataset on English sentence level. The following sections will introduce the background, design of the project, results, conclusion and future work of the project.

II. Background

A. Problem Statement

In offline handwriting recognition, text is analyzed after writing. The only information that can be analyzed is the binary output of the background character. The current offline deep learning model is especially necessary for large-scale digitization of historical documents, archives, or manual fill-in forms. How to improve the recognition of notes in the absence of information is a challenge [6]. All required text fields are processed during the data capture and validation phase of any form processing activity, including identifying and extracting written characters. The problem that can be caused by cursive handwriting is that letters may not be easily identified and may result in errors and erroneous information being processed.

B. Objectives

- To provide a comprehensive review of sources and characteristics of constraints typically found in existing handwriting recognition deep learning projects
- To develop a constraint classification method for easier constraint identification and modeling for handwriting recognition
- To review current industry practices and researches in regards to handwriting deep learning modeling
- To outline a conceptual framework for total constraint management

- To improve the function of the existing model, for example color contracts between handwriting and background (e.g. all current researches are using the handwriting sample of black words in white paper, can be improved by adding more attributes of different colors), absence of information (e.g. not only need to extract the handwriting but also need to automatically add information after extracting from the scanning), and etc.

C. Related Works

For the current existing research, the lead of the area in handwriting recognition is provide by MyScript. Their technology is provided to recognize normal paragraph handwriting, graphs, music, and mathematic formulas. The rst Optical Character Recognition (OCR) software developed in 1974 by Ray Kurzweil. By reducing the problem domain, the process was more accurate. This allowed for recognition in handwritten forms. [9]

Foremost, it lacked eciency and knowledge of unexpected characters. These classical techniques carried heavy limitations in two key areas: Character extraction (Individual characters are recognized by ease with OCR. Cursive handwriting, which is connected, poses more issues with evaluation. It is dicult to interpret handwriting with no distinct separation between characters) and Feature extraction (Individual properties of symbols were hard-coded, and matched to input symbols. Properties include aspect ratio, pixel distribution, number of strokes, distance from the image centre, and reection. This requires development time, as these properties are added manually). [8]

There are also some existing models in the Handwriting Recognition. Current methods includes CNN, Tensorflow, Multi-dimensional Recurrent Neural Network, RNN, and etc.

D. Benchmarks

I used the Google Colaboratory as the environment for training the data. Figure 2 shows the detailed inforamtion about the benchmark.

Machine Attribute	Value
BUG_REPORT_URL	"https://bugs.launchpad.net/ubuntu/"
DISTRIB_CODENAME	bionic
DISTRIB_DESCRIPTION	"Ubuntu 18.04.3 LTS"
DISTRIB_ID	Ubuntu
DISTRIB_RELEASE	18.04
HOME_URL	"https://www.ubuntu.com/"
ID	ubuntu
ID_LIKE	debian
NAME	"Ubuntu"
PRETTY_NAME	"Ubuntu 18.04.3 LTS"
PRIVACY_POLICY_URL	"https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
SUPPORT_URL	"https://help.ubuntu.com/"
UBUNTU_CODENAME	bionic
VERSION	"18.04.3 LTS (Bionic Beaver)"
VERSION_CODENAME	bionic
VERSION_ID	"18.04"
cpu_count	2
mac_version	
machine	('x86_64',)
mem_active	881.4 MiB
mem_available	11.8 GiB
mem_free	9.9 GiB
mem_inactive	1.7 GiB
mem_percent	7.3%
mem_total	12.7 GiB
mem_used	692.4 MiB
node	('bdc22e9a5382',)
platform	Linux-4.14.137+-x86_64-with-Ubuntu-18.04-bionic
processor	('x86_64',)
processors	Linux
python	3.6.9 (default, Nov 7 2019, 10:44:02)
	[GCC 8.3.0]
release	('4.14.137+',)
sys	linux
system	Linux
user	
version	#1 SMP Thu Aug 8 02:47:02 PDT 2019
win_version	

Fig. 2. Environment Benchmark

III. Project

A. Methodology

The dataset I am going to use is the IAM Handwriting Database. The methodology I plan to use for this project can be described as Figure 3.

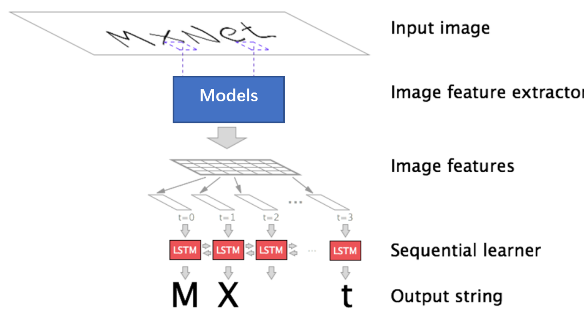
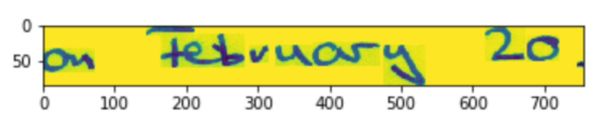


Fig. 3. Proposed flowchart

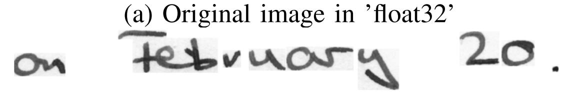
To process the images from the database, it will need to transfer the image to a binary array, while most of the images in IAM dataset are decoded by 'float32' but for the recognition use, it will be better to have the image in the type of 'uint8'. Resize and reshape all images to the same size which will be easy for computation. Using the sklearn to split the dataset into training dataset, testing dataset and validation dataset. Created several models (CNNs [7]) to extract and learn the array for times and find what is the best times of learning. Training the model and fit the dataset.Lastly, use the testing dataset to calculate the accuracy of the model and output the string.

B. Data Pre-Processing

The default datatype for the images in IAM dataset are 'float32' where the images will show in the yellow background and words will be written in dark blue. To process the images data, normalize the data by dividing by the maximum number in the data array, scale to 255 and change the datatype to 'uint8'. Figure 4 shows an example of the image processing.



(a) Original image in 'float32'



(b) After processing in 'uint8'

Fig. 4. Data processing

C. Model Choosing

I chose three CNNs model for training the IAM dataset. The final model which contains 827,698 parameters which are all trainable. The other two models both have non-trainable parameters (one has 960 non-trainable parameters and the other one has 832). Declare CNNs model as Keras sequential. The model will first resize the image again within the neural network for easy interpretation and add layers of Convolution2D, MaxPooling2D, activition layer 'relu' and 'softmax' with several dense layers. Figure 5 shos the summary of the model.

The model compiles with a loss function of 'categorical crossentropy' and optimizer of 'adam'. It is important to choose the epochs and number f layers. Several attempts has made to choose the optimizer and number of layers from related works by Jin, L., Qian, X., Wang, Y. [10] As shown in Figure 6, the increasing of epochs will return positive effect

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
zero_padding2d_1 (ZeroPaddin (None, 115, 115, 1)       0
_________________________________________________________________
lambda_1 (Lambda)            (None, 56, 56, 1)         0
_________________________________________________________________
conv1 (Conv2D)               (None, 28, 28, 32)        832
_________________________________________________________________
activation_1 (Activation)    (None, 28, 28, 32)        0
_________________________________________________________________
pool1 (MaxPooling2D)         (None, 14, 14, 32)        0
_________________________________________________________________
conv2 (Conv2D)               (None, 14, 14, 64)        18496
_________________________________________________________________
activation_2 (Activation)    (None, 14, 14, 64)        0
_________________________________________________________________
pool2 (MaxPooling2D)         (None, 7, 7, 64)          0
_________________________________________________________________
conv3 (Conv2D)               (None, 7, 7, 128)         73856
_________________________________________________________________
activation_3 (Activation)    (None, 7, 7, 128)         0
_________________________________________________________________
pool3 (MaxPooling2D)         (None, 3, 3, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1152)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 1152)              0
_________________________________________________________________
dense1 (Dense)               (None, 512)               590336
_________________________________________________________________
activation_4 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense2 (Dense)               (None, 256)               131328
_________________________________________________________________
activation_5 (Activation)    (None, 256)               0
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
output (Dense)               (None, 50)                12850
_________________________________________________________________
activation_6 (Activation)    (None, 50)                0
=================================================================
Total params: 827,698
Trainable params: 827,698
Non-trainable params: 0
```

Fig. 5. CNNs model summary

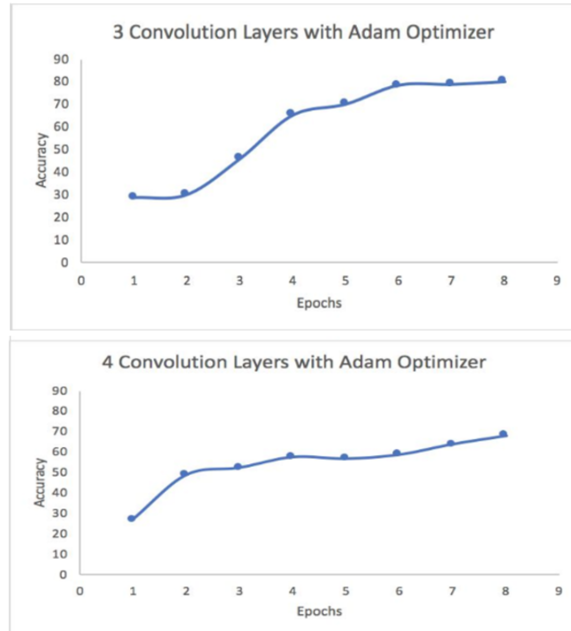


Fig. 6. Relationship between accuracy rate and number of layers

to accuracy rate, however; the increasing in the number of layers will have negative relationship with accuracy rate.

D. Train and Test Data

Using the sklearn to split the data into training, testing and validation in the rate of 4:1:1. By using the test data to get the accuracy score of the model, we can find that the accuracy rate for the model has 87.95% for 5 epochs and the accuracy rate will increase if training more epochs.

E. Other Models

Figure 7 displays other two CNNs models I made but did not get expected results for accuracy rate.

IV. Result

The CNNs model of Keras sequential with three layes of Convolution2D, MaxPooling2D, activition layer 'relu' and 'softmax' with several dense layers. As well as compiling loss function of 'categorical crossentropy' and optimizer of 'adam'. The model returns greater rate of accuracy than other 2 models. For 5 epochs, the final accuracy rate of testing data is approximately 87.95%. From learning from related works, I concluded that number of epochs has positive influence to the accuracy rate, but number of layers may have negative effect to accuracy rate.

V. Conclusion

In this project, I created a data-driven text independent model to identify writer for off-line handwritten scanned image from IAM dataset. To design a deep Convolutional Neural Network to extract sentences features. I investigated how the network structure affects identification accuracy by changing

Model: "sequential_4"

Layer (type)	Output Shape	Param #
zero_padding2d_4 (ZeroPaddin	(None, 115, 115, 1)	0
lambda_4 (Lambda)	(None, 56, 56, 1)	0
conv2d_5 (Conv2D)	(None, 56, 56, 32)	320
batch_normalization_5 (Batch	(None, 56, 56, 32)	128
max_pooling2d_5 (MaxPooling2	(None, 28, 28, 32)	0
dropout_8 (Dropout)	(None, 28, 28, 32)	0
conv2d_6 (Conv2D)	(None, 28, 28, 64)	18496
batch_normalization_6 (Batch	(None, 28, 28, 64)	256
max_pooling2d_6 (MaxPooling2	(None, 14, 14, 64)	0
dropout_9 (Dropout)	(None, 14, 14, 64)	0
conv2d_7 (Conv2D)	(None, 14, 14, 128)	73856
batch_normalization_7 (Batch	(None, 14, 14, 128)	512
max_pooling2d_7 (MaxPooling2	(None, 7, 7, 128)	0
dropout_10 (Dropout)	(None, 7, 7, 128)	0
conv2d_8 (Conv2D)	(None, 7, 7, 256)	295168
batch_normalization_8 (Batch	(None, 7, 7, 256)	1024
max_pooling2d_8 (MaxPooling2	(None, 7, 3, 128)	0
flatten_3 (Flatten)	(None, 2688)	0
dropout_11 (Dropout)	(None, 2688)	0
digit1 (Dense)	(None, 36)	96804
digit2 (Dense)	(None, 36)	1332
digit3 (Dense)	(None, 36)	1332
digit4 (Dense)	(None, 36)	1332

Total params: 490,560
Trainable params: 489,600
Non-trainable params: 960

Model: "sequential_6"

Layer (type)	Output Shape	Param #
zero_padding2d_6 (ZeroPaddin	(None, 115, 115, 1)	0
lambda_6 (Lambda)	(None, 56, 56, 1)	0
conv2d_15 (Conv2D)	(None, 56, 56, 32)	320
batch_normalization_9 (Batch	(None, 56, 56, 32)	128
max_pooling2d_9 (MaxPooling2	(None, 28, 28, 32)	0
dropout_12 (Dropout)	(None, 28, 28, 32)	0
conv2d_16 (Conv2D)	(None, 28, 28, 64)	18496
batch_normalization_10 (Batc	(None, 28, 28, 64)	256
max_pooling2d_10 (MaxPooling	(None, 14, 14, 64)	0
dropout_13 (Dropout)	(None, 14, 14, 64)	0
batch_normalization_11 (Batc	(None, 14, 14, 64)	256
max_pooling2d_11 (MaxPooling	(None, 7, 7, 64)	0
dropout_14 (Dropout)	(None, 7, 7, 64)	0
conv2d_17 (Conv2D)	(None, 7, 7, 256)	147712
batch_normalization_12 (Batc	(None, 7, 7, 256)	1024
max_pooling2d_12 (MaxPooling	(None, 7, 3, 128)	0
flatten_4 (Flatten)	(None, 2688)	0
dropout_15 (Dropout)	(None, 2688)	0
dense_1 (Dense)	(None, 1024)	2753536
dropout_16 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 512)	524800
dropout_17 (Dropout)	(None, 512)	0
digit1 (Dense)	(None, 36)	18468
digit2 (Dense)	(None, 36)	1332
digit3 (Dense)	(None, 36)	1332
digit4 (Dense)	(None, 36)	1332

Total params: 3,468,992
Trainable params: 3,468,160
Non-trainable params: 832

Fig. 7. Other models performs CNNs

the epochs and number of layers. For final, the model achieved high identification accuracy.

VI. Future Work

In the future, I planned to learn more about different layers and the functions of different layers. For the neural network, I will need to study more about the background knowledge of how data processing and learned by the model.

Another aspects I am interested in is how the dataset split to sentences. It is worth to learn how computer and neural network learn the concept of forms, paragraphs, sentences and words. As in IAM dataset, large amount of images are separated, even a punctuation.

Having the chance of working on the computer vision related projects, I will get more chance to improve the recognition model to not only English sentences but also other languages.

References

[1] Bulacu M, Schomaker L. Text-independent writer identification and verification using textural and allographic features[J]. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2007, 29(4): 701-717.

[2] Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 10971105).

[3] He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition[J]. arXiv preprint arXiv:1512.03385, 2015.

[4] Ciresan D, Meier U, Schmidhuber J. Multi-column deep neural networks for image classification[C]. Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. IEEE, 2012: 3642-3649.

[5] U. Marti and H. Bunke. The IAM-database: An English Sentence Database for Off-line Handwriting Recognition. Int. Journal on Document Analysis and Recognition, Volume 5, pages 39 - 46, 2002.

[6] Chescoe, D., (2017), What problems can Handwriting cause in the Digital Age?, AAC Systems – Specialists in Data Capture and Information Management- Blog, Retrieved from https://www.aacsystems.co.uk/handwriting-recognition-challanges/

[7] Chuang, J., (2018), Handwriting recognition and language modeling with MXNet Gluon. Medium.com, Retrieved from https://medium.com/apache-mxnet/handwriting-ocr-handwriting-recognition-and-language-modeling-with-mxnet-gluon-4c7165788c67

[8] Wiles, R., (2019), Have we solved the problem of handwriting recognition?, Towards Data Science, Retrieved from https://towardsdatascience.com/https-medium-com-rachelwiles-have-we-solved-the-problem-of-handwriting-recognition-712e279f373b

[9] Handwritten recognition technology. MyScript. Retrieved from https://www.myscript.com/handwriting-recognition/

[10] Jin, L., Qian, X., Wang,Y. Image Classification: IAM Handwritting. GitHub. Retrieved from https://github.com/ljin75654/Final-Project-Group3/blob/6bbd31588caf0ca03551d3219a6464fe3e09f7fa/Final-Group-Presentation/Final%20Group%20Presentation.pdf

[11] Chuang, J. Delteil, T. Handwritten Text Recognition (OCR) with MXNet Gluon. GitHub. Retrieved from https://github.com/awslabs/handwritten-text-recognition-for-apache-mxnet

[12] Scheidl, H. Handwritten Text Recognition with TensorFlow. GitHub. Retrieved from https://github.com/githubharald/SimpleHTR

[13] Reddy, T. (2018) IAM Handwriting Top50: Offline IAM Handwriting Dataset's subset, w.r.t. the 50 most common writers. Kaggle. Retrieved from https://www.kaggle.com/tejasreddy/iam-handwriting-top50

[14] Scheidl, H. (2018) Build a Handwritten Text Recognition System using TensorFlow: A minimalistic neural network implementation which can be trained on the CPU. TowardsDataScience. Retrieved from https://towardsdatascience.com/build-a-handwritten-text-recognition-system-using-tensorflow-2326a3487cd5

[15] Dwivedi, P. English Deep Writer. GitHub. Retrieved from https://github.com/priya-dwivedi/Deep-Learning/blob/master/handwriting

[16] Dwivedi, P. (2018). Handwriting recognition using Tensorflow and Keras. TowardsDataScience. Retrieved from https://towardsdatascience.com/handwriting-recognition-using-tensorflow-and-keras-819b36148fe5

[17] Xing, L., Qiao, Y., Shenzhen key lab of Comp. Vis. Pat. Rec., Shenzhen Institutes of Advanced Technology, CAS, China, University of Chinese Academy of Sciences, China, The Chinese University of Hong Kong, Hong Kong. (2016). DeepWriter: A Multi-Stream Deep CNN for Text-independent Writer Identification. Retrieved from https://arxiv.org/pdf/1606.06472.pdf

[18] Keras CNN training to recognize captcha: Get low loss and get low accuracy. (2019). StackOverflow. Retrieved from https://stackoverflow.com/questions/53993955/keras-cnn-training-to-recognize-captcha-get-low-loss-and-get-low-accuracy