Medical Handwritten Prescription Recognition Using CRNN

Roger Achkar
Computer & Communications Engineering
American University of Science and Technology
Beirut, Lebanon
rachkar@aust.edu.lb

Khodor Ghayad
Computer & Communications Engineering
American University of Science and Technology
Beirut, Lebanon
kghayad@aust.edu.lb

Rayan Haidar
Mechatronics Engineering
American University of Science and Technology
Beirut, Lebanon
rayan.m.haidar@gmail.com

Sawsan Saleh
Computer & Communications Engineering
American University of Science and Technology
Beirut, Lebanon
sa.0101@outlook.com

Rana Al Hajj
Mechatronics Engineering
American University of Science and Technology)
Beirut, Lebanon
rana_elhajj95@hotmail.com

Abstract*— Reading a doctor's handwritten prescription is a challenge that most patients and some pharmacists face; an issue that, in some cases, lead to negative consequences due to wrong deciphering of the prescription. Part of the reason why doctor's prescriptions are so difficult to decipher is that doctors make use of Latin abbreviations and medical terminology that most people don't understand. This paper demonstrates how Artificial Neural Networks (ANN) is used to develop a system that can recognize handwritten English medical prescriptions. Using the Deep Convolution Recurrent Neural Network to train this supervised system, input images are segmented and processed to detect characters and classify them into the 64 different predefined characters. The results show that the proposed system yields good recognition rates and an accuracy of %98.

Keywords: CRNN, RNN, ANN, prescriptions, handwritten text, OCR.

I. INTRODUCTION

Most Optical Character Recognition systems work at the link level by transforming the text-line image into a sequence of feature vectors using Recurrent Neural Networks (RNN) in order to recognize handwritten text [1]. There are some promising results of experiments where detection was executed at a paragraph level [2]. However, the best recognition is achieved at the line level [3]; hence, a solution to decipher a handwritten medical prescription correctly is still lacking. Medical errors due to wrong interpretation of prescription are common and can result in harm to patients and sometimes lead to death, especially when the wrong dosage or medicine is taken [4] 'To Err is Human' report by the Institute of Medicine (IoM) states that medical errors cause at least an estimated 44,000 preventable deaths annually in the United States of America alone, of which 7,000 deaths are attributed to illegible handwriting [5]. In many cases it is ideal to have prescriptions deciphered or translated into digital forms rather than be paper based, in order to make pharmacists' lives easier and minimize the risk of wrongly interpreted dosages and medication. In addition, illegible handwriting can lead to adverse medico-legal implications since sloppy handwriting can be interpreted by the jury as sloppy care [6]. Handwriting recognition is the ability of a program to receive input images and transform them into digital characters by detecting handwritten letters and map them into predefined characters list. Lexicon-Driven Handwritten Text Recognition is another form of recognition. It focuses on splitting words in case of spotting a spacing between them. This proved to be effective with 74.37% according to the paper "Handwritten character strings on medical prescription reading by using lexicon-driven" [7].

This paper describes how neural network or CRNN technology is used specifically to detect medical prescriptions and translate the handwritten text into a digital one. The paper includes the methods used to write the python-based code, the training process, and the results obtained. The training is done using short texts, since prescriptions usually consist of 2 or 3 words as shown in Figure 1.

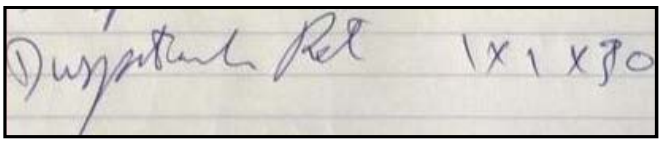

Figure 1. Prescription Sample

II. CRNN SYSTEM DISCRIPTION

This project is based on Convolutional Recurrent Neural Network (CRNN), which is inspired by VGG16 architecture [8] that is used for image recognition, noting that Convolutional Neural Networks differ from Recurrent Neural Networks and Traditional Neural Networks.

Traditional Neural Networks lack the use of memory, and use the result of previous training in order to predict the outcome, while Convolutional Neural Networks (CNN) have the ability or benefit to develop an internal representation of a two-dimensional image. This allows the network to detect the position and scale of letters in the input data images. As for Recurrent Neural Network (RNN), it works with sequence prediction problems, specifically the Long Short-Term Memory (LSTM), a type of RNN; it overcomes the problems of training a recurrent network that use long-term data dependency problem [9]. This network has the ability to remove or add information to the layer state by a decision made by a sigmoid layer; the output is to be based on said layer state. Finally, the output of the state is put through a *tanh* function and multiplied by the output of the sigmoid gate. A hybrid network model such as CRNN is the perfect

978-1-7281-1374-6/19/$31.00 ©2019 IEEE

choice of combining the benefits of both RNN and CNN networks to be used to tackle the problem of extracting letters in handwritten medical prescriptions. Figure 2 further illustrates how CNN, RNN and CRNN work.

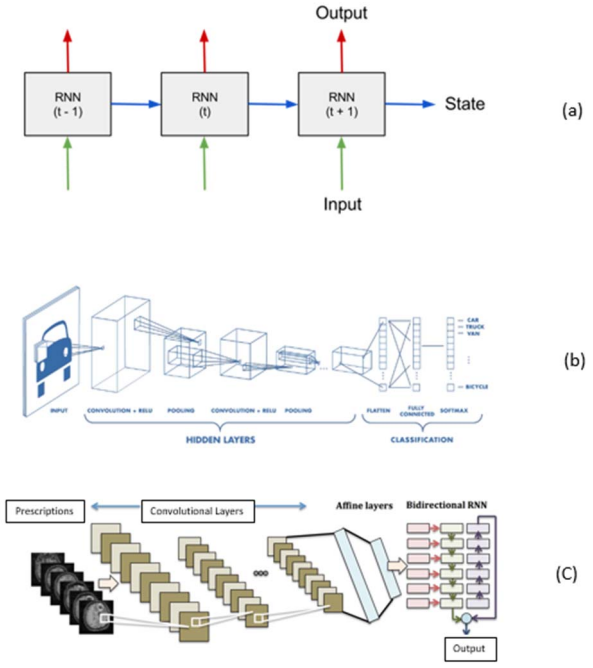

Figure 2. Illustration of (a) RNN, (b) CNN & (c) CRNN

The proposed network consists of a stack of 13 convolutional layers followed by three Bidirectional LSTM layers. To introduce non-linearity, an activation function called Rectified Linear Unit (ReLU) is used after every convolutional layer.

The input images are divided into 28 sub images or windows, where the height of the window is equal to the height of the text-line image; the vectors are extracted from the feature maps produced by the last convolutional layer and fed into the first LSTM layer. By combining the forward and backward outputs at the end of each Bidirectional LSTM layer rather than at the end of each of these layers, the weights are optimized faster.

Initially, the rate of the learning parameter used is 0.001, but later increased to speed up the training process that is explained in the following section.

III. TRAINING AND VALIDATION

The first step in the project training is defining which type of network is required, number of inputs, hidden layers, outputs (classes), and the preferred coding language. As stated earlier, this project focuses on CRNN using Python. The project is based on 66 different classes, including alpha-numeric characters, punctuations and spaces.

Initially, an image with multiple prescriptions, shown in Figure 3, is used. The problem faced is the division of the image of each prescription alone into small segments. The program is not able to detect the start and end of the prescription, so the text is required to be manually inserted on line-by-line basis, in other words, prescription by prescription. This will help the program to yield more accurate results with less computational time.

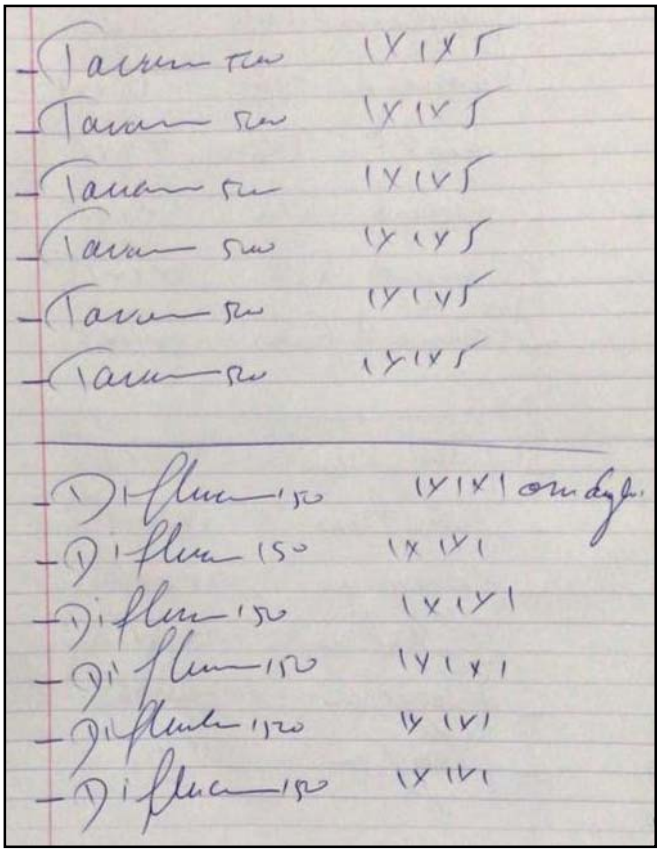

Figure 3. Multiple Prescriptions

Since prescriptions are usually small and don't include long sentences, the images need to be provided with 340x60 dimensions, shown in Figure 4.

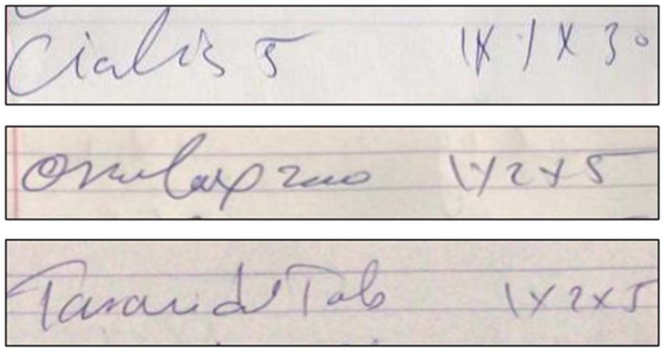

Figure 4. 340x60 Samples

The training focuses on image processing by using a sliding window of '2' width in order to check the written text by taking 2 or 3 characters at a time. The height of the window is equal to the height of the text-line image, which has been normalized to 64 pixels. The window overlap is equal to 2 pixels to allow continuous transition of the convolution filters. For each analysis window of 64x64 pixels in size, 16 feature vectors are extracted from the feature maps produced by the last convolutional layer and fed into the observation sequence. It is worth noting that the amount of feature vectors extracted from each sliding windows is important. The number must be reasonable as to provide a good sampling for the image.

In order to validate data, a minimum of 10,000 training samples need to be processed. There is also the need to define parameters, such as starting epoch (=0), learning parameter (=0.0005), and batch size (=10), which is the number of images processed at each iteration. The training could run for 1,000,000 epochs, and the model will be saved after 1 epoch. If 20 epochs passed with no improvement in the training or validation phase, the training would stop. In order to perform the training process and provide the system with the required input, the first step is to collect all the data needed to train this system. The input of the neural network is the image of different medical prescription; the desired output is specified in two ways. The first way is by creating a text with the output of every image. The second way is by creating a label for every character in the text.

To further enhance the performance of the system, the variability in the writing scale to augment the training set with text-line images at multiple scales is exploited. Based on a vertical scale score [10], the training lines are first classified into 3 classes (Large, Medium and Small); through this division the data volume per class become smaller. To address this problem, the training set for each class is expanded by adding synthetic data that have been augmented from scaling the other classes' data. For example, the first step is to reduce the large images, and stretch the small ones (by a predetermined factor for each class) to expand the number of medium sized images. Or to reduce the medium and large sized images to extend the set of small images [11].

When the training is done, a log file is created. This log file contains a detailed description of every epoch with the training loss, training error, and if the training has improved or not, in addition to a description of the with the validation loss and validation error. The training will automatically stop if the training data or the validation data didn't improve after 20 epochs [12].

IV. RESULTS

As afore-stated, at the conclusion of the training, a log file is generated including the number of epochs, error, batch, training error, training loss, validation error, validation loss and whether or not the training has improved as shown in Figure 5.

```
Training Data
Epoch 0, Batch: 0, Loss: 448.299225, Error: 2.154589, Good
Epoch 0, Batch: 1, Loss: 282.747925, Error: 1.063636, Good
Training loss: 365.523560, Training error: 1.609113
Training imporving.

Training Data
Epoch 63, Batch: 0, Loss: 0.563016, Error: 0.103604, Good
Epoch 63, Batch: 1, Loss: 0.619156, Error: 0.121951, Good
Training loss: 0.591086, Training error: 0.112777
Training imporving.

Training Data
Epoch 198, Batch: 0, Loss: 0.001130, Error: 0.110599, Bad
Epoch 198, Batch: 1, Loss: 0.000699, Error: 0.114286, Bad
Training loss: 0.000915, Training error: 0.112442
Training not imporving.
```

Figure 5. Log File

An epoch is the iteration in which a weight is updated. The training took 203 epochs to finish; it only stops when training has not shown improvement after 20 consecutive epochs. A batch is the number of samples included in each epoch. For example, assume there are 100 images available for training and 10 images are sent by iteration, then, the batch is equal to 10. The error displayed is the difference between the actual output and desired output, while the training error is the average between errors of 2 batches. If training loss is greater than validation loss, then the result is an under-fitting network which requires a change in the parameters, meaning the network is not working properly. If the training loss is less than validation loss, then the result is an over-fitting, which requires a slight change in the parameters, meaning the network is "overqualified." However, the sought desired result or output is to have training loss equal to validation loss. The first testing results, Figure 6, show that the validation error is higher than the training error, meaning this system is over-fitting. This implies that the learning process needs to be slowed. The spike shown in epoch 33 is due to noise done during the training. The first training was done using normal handwritten texts and took 16 hours to finish.

The results are summarized in Table 1 and Figure 6.

Table 1. Training 1

Epoch	Training Error 2	Training Loss 2	Validation error 2	Validation Loss 2
1	1.851534	842.072388	2.007543	874.261353
2	0.825247	418.674988	0.940522	424.992218
3	0.810121	299.525055	0.790954	266.163208
4	0.799893	243.815811	0.739319	217.40451
5	0.74414	206.582031	0.749828	193.463547
6	0.733371	186.056427	0.704516	174.002289
7	0.695369	169.987061	0.666132	156.48764
8	0.662191	155.00943	0.645494	138.080338
.....				
.....				
134	0.012348	0.033322	0.012261	0.041365
135	0.013331	0.21817	0.012261	0.052948
136	0.012212	0.058335	0.012261	0.047167
137	0.014376	0.156609	0.021552	2.065261

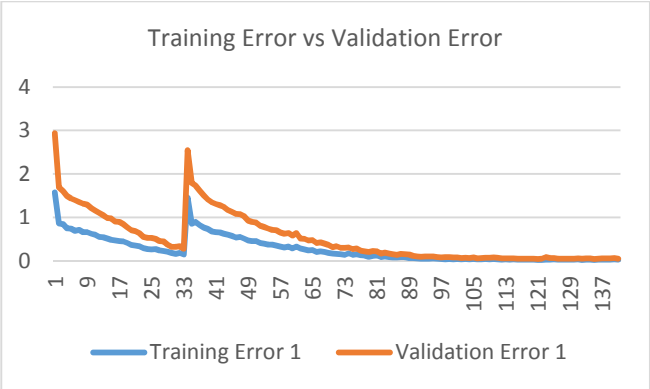

Figure 6. Training 1 (Normal Handwritten Text)

In the second training, cursive handwritten text is used. The learning parameter is also changed from 0.0001 to 0.0005 in order to check the difference between training error and validation error. Since the final input is prescriptions, the input is inserted line by line instead of complete paragraphs. The graph, shown in Figure 7, reflects more stable results, but with more epochs. This is normal due to the increase of the

learning parameter. The results are summarized in Table 2 and Figure 7.

Table 2. Training 2

Epoch	Training Error 2	Training Loss 2	Validation error 2	Validation Loss 2
1	1.851534	842.072388	2.007543	874.261353
2	0.825247	418.674988	0.940522	424.992218
3	0.810121	299.525055	0.790954	266.163208
4	0.799893	243.815811	0.739319	217.40451
5	0.74414	206.582031	0.749828	193.463547
6	0.733371	186.056427	0.704516	174.002289
7	0.695369	169.987061	0.666132	156.48764
8	0.662191	155.00943	0.645494	138.080338
.....				
.....				
134	0.012348	0.033322	0.012261	0.041365
135	0.013331	0.21817	0.012261	0.052948
136	0.012212	0.058335	0.012261	0.047167
137	0.014376	0.156609	0.021552	2.065261
138	0.021552	2.009428	0.012261	0.13757
139	0.014152	0.376548	0.015383	0.386125
140	0.014304	0.536242	0.015383	0.80391
141	0.019738	1.424158	0.01744	0.895046
142	0.014361	0.425639	0.014393	0.314056
143	0.012244	0.224846	0.012261	0.177649
144	0.012255	0.158993	0.012261	0.12861
145	0.016576	0.376977	0.013327	0.284227

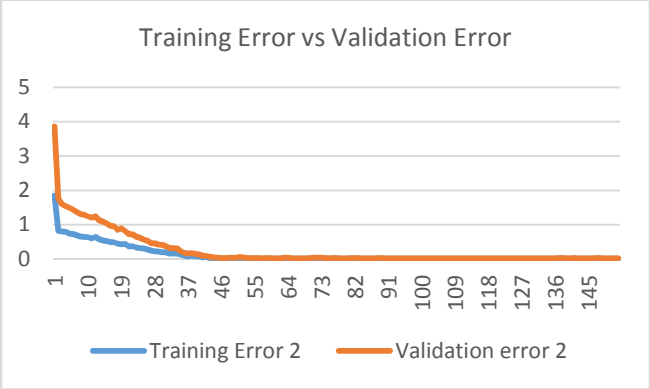

Figure 7. Training 2 (Cursive Handwritten Text)

In third and final training, doctors' prescriptions are used. The results, shown in Figure 8, took only 8 hours to appear, and they are astonishing. As shown in the graph, the training finished after 201 epochs. The error decreased smoothly until finally reaching 0.003. The results are summarized in Table 3 and Figure 8.

After finding the results of Trainings 1, 2 and 3, the code is able to calculate each error in order to find the best training set.

Table 3. Training 3

Epoch	Training Error 3	Training Loss 3	Validation Error 3	Validation Loss 3
1	1.609113	365.52356	1.678601	352.795959
2	0.801075	144.051056	0.795095	140.049988
3	0.688139	87.52652	0.712	73.876724
4	0.676442	60.524696	0.630014	52.349949
5	0.604186	44.38446	0.586983	42.711258
6	0.59019	37.628136	0.521823	35.113335
7	0.503411	31.636316	0.56325	34.263344
8	0.508083	28.495663	0.45004	25.957809
.....				
.....				
191	0.112413	0.000661	0.113045	0.001292
192	0.112482	0.000845	0.113045	0.001283
193	0.112413	0.000815	0.113045	0.00126
194	0.112777	0.000891	0.113045	0.001252
195	0.112413	0.000819	0.113045	0.001266
196	0.112418	0.000621	0.113045	0.001252
197	0.112442	0.001422	0.113045	0.00127
198	0.112487	0.000608	0.113045	0.001218
199	0.112442	0.000915	0.113045	0.001258

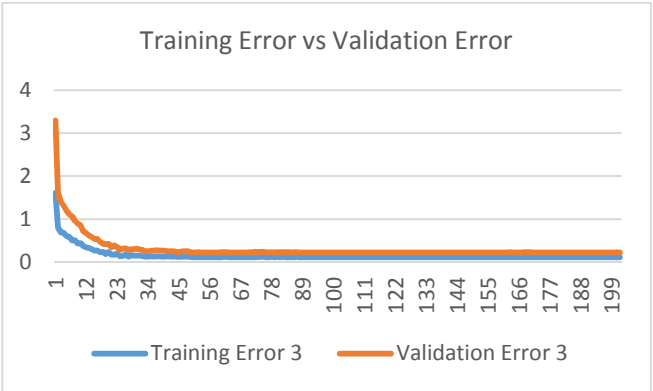

Figure 8. Training 3 (Doctor's Prescription Text)

Finally, the results are tested using different prescriptions than the ones used in training. A total of 50 prescriptions were used with 45 correct results, with a 10% error. The decoded file after finalized testing is shown in Figure 9. The decoded file includes the image name on the left and its decoded output on the right. The decoded file is able to get the output and compare it with the desired output in order to calculate the efficiency of the system.

$$output\ error\ (e_{output}) = \ actual\ output - predicted\ output \quad (1)$$

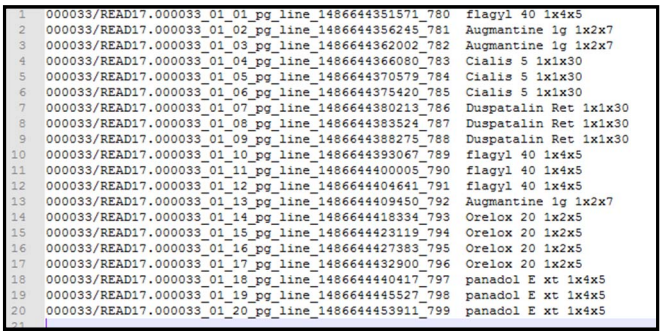

Figure 9. Decoded File

V. CONCLUSION

In this paper, a state-of-the-art CRNN system for text-words recognition of doctors' prescriptions is presented. It shows how to train such system with few labeled input data. Precisely, 10% labeled examples were provided for training. Finally, a normalized-scale for the samples is proposed in order to get better and more accurate results. This combination of normal handwriting and that of doctors' prescriptions generated accurate results. This program achieved a score of 95% accuracy, which is really great, taking into account training time and amount of data input. In order to improve these results, more work is needed on input handling. The algorithm needs to learn how to read hard paragraphs and automatically segment them into small texts. Although the network is complex, a small change in the initial parameters leads to great differences in the results. There is also a great difference between using clear texts and handwritten doctors' prescriptions in adjusting the weights. Having clear text in the training phase doesn't yield correct results while testing prescriptions, and vice-versa. Some might insist that this work is mainly focused on character recognition; however, it focuses on *doctors'* character recognition and not any simple handwritten text. For the future work, this project could be implemented in a form of mobile application that can be accessible to various users, including patients, pharmacists and people responsible for double-checking doctors' prescriptions.

ACKNOWLEDGMENT

We would like to thank Mrs. Henriette Skaff, chief editor at AUST for editing the article.

REFERENCES

[1] B. Moysset, C. Kermorvant, and C. Wolf, "Full-page text recognition: Learning where to start and when to stop," arXiv preprins arXiv: 1704, 08628, 2017.

[2] T. Bluche, "Joint line Segmentation and transcription for end-to-end handwritten paragraph recognition," in Advances in Neural Information Processing Systems, 2016, pp. 838-846.

[3] P. Voigtlaender, P. Doetsch, and H. Ney. "Handwriting recognition with large multidimensional long short-term memory recurrent neural networks," in Frontiers in Handwriting Recognition (ICFHR), 2016 15th international Conference on. IEEE, 2016, pp. 228-233

[4] Daniel K Sokol and Samantha Hettige. Poor handwriting remains a significant problem in medicine. J R SOC Med. Dec 2006; 99(12): 645-646. PMCID: PMC1676338

[5] Dr. Nomal Chandra Borah, Chairman & Managing Director, Guwahati Neurological Research Centre. Doctor's illegible handwriting creating confusion, causing death. April 20, 2015

[6] Stephens E. Medical-legal liability in emergency medicine. 2005.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv: 1409.1559, 2014.

[8] Colah. Understanding LSTM Networks. Posted on github on August 27, 2015.

[9] Roger Achkar, Mustafa El-Halabi, Elie Bassil, Rayan Fakhro, and Marny Khalil. Voice identity finder using the back propagation algorithm of an artificial neural network. Procedia Computer Science, 95:245–252, 2016.

[10] Youssef Harkouss, Souhad Mcheik, and Roger Achkar. Accurate wavelet neural network for efficient controlling of an active magnetic bearing system. 2010.

[11] Chafic Saide, R´egis Lengelle, Paul Honeine, Cedric Richard, and Roger Achkar. Nonlinear adaptive filtering using kernel-based algorithms with dictionary adaptation. International Journal of Adaptive Control and Signal Processing, 29(11):1391–1410, 2015.

[12] George Abou Kassm and Roger Achkar. Lpr cnn cascade and adaptive deskewing. Procedia Computer Science, 114:296–303, 2017