Arabic Optical Character Recognition Using Attention Based Encoder-Decoder Architecture

Mohamed Sobhi
Researcher College of Computing and IT Arab Academy for Science, Technology and Maritime Transport, Egypt. +201020933303
chomskynet@gmail.com

Yasser Hifny
Associate professor University of Helwan, Ain Helwan, Cairo, Egypt. +15142437740
yhifny@fci.helwan.edu.eg

Saleh Mesbah
Associate Professor College of Computing and IT Arab Academy for Science, Technology and Maritime Transport, Egypt. +201005228715
saleh.mesbah@aast.edu

ABSTRACT

Optical character recognition (OCR) systems are used to convert scanned documents into text. Arabic OCR is an active area of research where high accuracy is demanding. This paper focuses on building a model for converting images that contain Arabic text into their corresponding text using a deep learning approach. This model does not require any knowledge of the underlying language and it is simply trained end-to-end on the KAFD dataset. It combines several standard neural components from vision and natural language processing. Features are extracted from images using Convolutional Neural Networks (CNNs) where the features are arranged in a grid. Each row is then encoded using a Recurrent Neural Networks (RNNs). An RNN decoder with a visual attention mechanism is used to generate the output text. Our preliminary experiments show that the presented approach is effective. The overall obtained accuracy is 89.82%. However, the individual results for some fonts are higher than this score.

KEYWORDS

Sequence to Sequence Model, Encoder-Decoder Architecture, Arabic OCR, Convolutional Neural Networks (CNNs), Recurrent Neural Network (RNN), Attention Mechanism, Long Short-Term Memory (LSTM)

ACM Reference Format:
Mohamed Sobhi, Yasser Hifny, and Saleh Mesbah. 2020. Arabic Optical Character Recognition Using Attention Based Encoder-Decoder Architecture. In *2020 2nd International Conference on Artificial Intelligence, Robotics and Control (AIRC'20), December 12–14, 2020, Cairo, MN, Egypt*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3448326.3448327

1 INTRODUCTION

Arabic OCR has many challenges such as the variations of Arabic characters according to their position in the word where every character may have two or four different forms. Thus, the number of classes will increase to be recognized from 28 to 100. Moreover the cursive nature of Arabic writing does not allow direct implementation of many algorithms formulated for othe [1]

This paper presents a model that is built to transfer the Arabic text images to their corresponding texts depending on the recent deep-learning techniques. The model uses different Arabic datasets. Section 2 reviews the related work of OCR. Section 3 explains the used model for supporting the current proposed work. Section 4 describes the used datasets and how these data sets are preprocessed to be used in the current model. Section 5 details the experiments used for building the model. Section 6 reviews the results of the proposed work and finally. Finally, Section 7 concludes the paper.

2 RELATED WORK

This section provides the appropriate background on previous work on image to text generation. Recently, numerous methods have been proposed for generating the text from image. Many of these methods are based on recurrent neural networks (RNN) and inspired by the successful use of sequence to sequence Models with neural networks.

A synthetic text generation engine to train three different models for scene text recognition is proposed: 90K-way dictionary encoding, character sequence encoding and bag-of-N-grams encoding. The experiments showed that the synthetic dataset is realistic and sufficient that it can efficiently replace real data [2]

A method to eliminate the need of pre-processing stages and the use of multiple neural networks in OCR by training a single neural network is propose [3] The network is used to detect and recognize text relying on semi-supervised learning. The proposed network was firstly evaluated on SVHN dataset, then it was evaluated on ICDAR 2013, SVT and IIIT5K datasets with accuracies 90.3%, 79.8% and 86% respectively. Finally, it was evaluated on FSNS dataset with 97% accuracy for character recognition and 71.8% accuracy for words.

A framework for lexicon-free OCR is propose [4] The proposed method mainly uses: recursive CNNs for image feature extraction, RNNs for character-level language modeling with soft-attention mechanism. However, the proposed method did not use LSTM. The framework was evaluated on ICDAR 2003, ICDAR 2013, Street View Text and IIIT5K datasets with accuracies 88.7%, 90%, 80.7% and 78.4% respectively for unconstrained text recognition.

In [5] they demonstrated the convenience of sequence to sequence learning in OCR by proposing a recurrent encoder-decoder framework. Attention based model was not used, but the paper shows the effect of LTSM in the sequence-to-sequence learning.

AIRC'20, December 12–14, 2020, Cairo, MN, Egypt

ACM ISBN 978-1-4503-8926-6/20/12…$15.00
https://doi.org/10.1145/3448326.3448327

The proposed model outperforms LSTM with CTC output layer. However, it requires more memory space.

In [6] they proposed a method to identify multiple scripts at the same text-line level. The method is based on sequence learning model with LSTM capabilities, where a 1D-LSTM architecture is used. The model was evaluated on an English-Greek data and it gives 98.186% accuracy.

In [7] they presented WYGIYS (What You Get Is What You See) model for OCR of presentational markup. The model is based on convolutional networks with visual attention. They introduced a new dataset IM2LATEX-100K, besides a synthetic dataset of webpages paired with HTML, for training and evaluating the model. The model gives 75% accuracy

In [8] they proposed a model based on sequence-to-sequence architecture and a convolutional network to recognize handwritten text. The proposed model mainly consists of: A convolutional network for features extraction, an LSTM-based RNN for encoding and another LSTM-based RNN with attention for decoding. The model was evaluated on IAM and RIMES databases on which it gives 12.7% and 6.8% for word error rate, respectively.

In [9] they introduced ASTER (Attentional Scene Text Recognizer with Flexible Rectification) to recognize text with distortions and irregular layout. ASTER is an end-to-end model that includes both a rectification network and recognition network. The rectification network is based on Spatial Transformer Network, while the recognition network is based on attentional sequence-to-sequence model. The proposed model is trained on two synthetic datasets: Synth90K and Synth-Text, and is evaluated on: IIIT5K, SVT, ICDAR 2003, ICDAR 2013, SVTP and CUTE with accuracies 92.67%, 91.16%, 93.72%, 90.74%, 78.76% and 76.39%, respectively.

In [10] they proposed a system of three-neural network models for Arabic OCR. The first network is a three-layer neural network to recognize the text font size, then the text is normalized to 18pt font size to be fed to the second network. The second network model is a multi-channel neural network with three-window input. Finally, the third network is a two-layer convolutional network with two max pooling layers. The whole pipeline was evaluated on APTI dataset and gives an accuracy of 94.8%.

Calamari is an open-source line-based OCR system for contemporary and historical documents proposed In [11]. It supports pretraining and voting techniques besides using the Deep Neural Networks. The architecture of the proposed network is mainly composed of Convolutional Neural Networks (CNNs), pooling layers and Long-Short-Term-Memory (LSTM) layers. The network has a sequence-to-sequence task and is trained by CTC algorithm. UW3 and DTA19 datasets were used to evaluate Calamari and compare it with other systems such as OCRopy, OCRopus3 and Tesseract. Calamari outperforms the other existing systems using the Character Error Rate (CER) metrics, where it did not exceed 0.18% for the DTA19 dataset, and 0.11% for UW3 dataset. Moreover, the system is very fast in terms of the training and prediction times.

3 MODEL

This model is mainly based on combining a number of standard neural components for vision and natural language processing. Firstly,

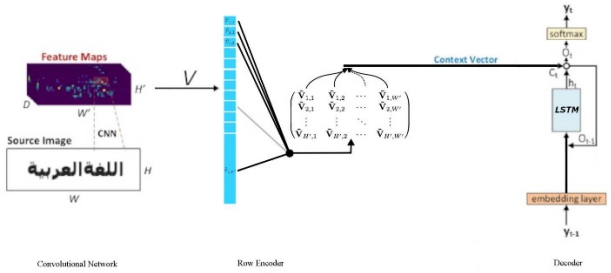

Figure 1: The full structure

it uses a Convolutional Neural Network (CNN) for features extraction from the image, which are then arranged in a grid. Secondly, it uses two Recurrent Neural Networks (RNN) for as an encoder-decoder. The first RNN encodes each row in the feature grid, then the second RNN, which is enhanced with a visual attention mechanism, decodes the encoded features. The full structure is illustrated in Figure1.

Model Structure:

- A CNN is used to extract a feature map V from the input image.
- Arrange the features in a grid.
- An RNN is used to encode spatial layout information for each row in the feature map.
- An RNN is used to decode with a visual attention mechanism to produce final outputs.

Convolutional Network: A Convolutional Neural Network (CNN, or ConvNet) is a sort of deep learning, feedforward Artificial Neural Network used extensively in analyzing visual image recognition tasks. It is mainly composed of an input layer, an output layer, and multiple hidden layers: convolutional layer, pooling layer, loss layer (drop out), fully connected layer (dense layer – Multilayer Perceptron) [12]The visual feature extraction in this model is achieved through a multi-layer convolutional neural network interleaved with max-pooling layers. This network is considered a standard architecture nowadays. The convolution layer neurons execute a dot product between the image pixels and a filter. Then a Rectified Linear Unit (ReLU) and pooling layer follow each convolution layer. The ReLU is a non-linear activation function performed on each element to detect the negative values and threshold them to zero, such as max(0,x). For example: the output of ReLu (2, -3) will be (2,0). The pooling layer mainly reduces the dimensions of the image as the computation moves towards successive layers, and its output is then the input for down-sampling. The image-dimensions reduction results in reducing the number of parameters which helps in controlling the problem of overfitting. The CNN is designed such that pooling layers are interleaved in-between successive convolutional layers. The pooling layer acts on each feature map independent of others and resizes it by performing a MAX operation. For example, in Figure 2, an image of dimensions 4*4 will be reduced by a pooling layer having a filter of dimensions 2*2 and stride 2. The output image from this layer will be 2*2, thus it has been reduced to 50% of its previous size. The max is performed

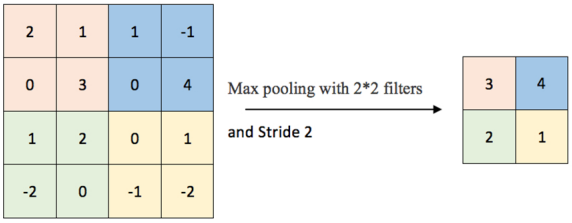

Figure 2: Pooling layer operation [13]

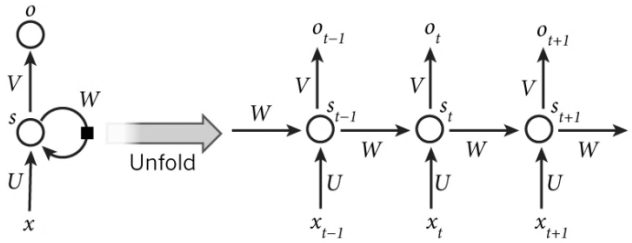

Figure 3: RNN Architecture [14]

to get the greatest number from the numbers that are in the filter's window [13]

max (2, 1, 0, 3) = 3

In this model, as we use visual attention model, we have to preserve the locality of CNN features, so final fully-connected layers are not used. The CNN accepts the raw input $R^{H\times W}$ and outputs a grid of features V of size $D \times H^1 \times W^1$, where D is the number of channels, H^1 is the reduced-sized grid height and W^1 is its width.

Row Encoder: Unlike image captioning, where the CNN features are used as they are, in OCR the encoder has to localize the relative positions within the image. This is achieved through running RNNs over each row of the CNN features.

A recurrent neural network (RNN) is a neural network characterized by the concept of 'internal memory', which differentiate it from other feedforward Networks the RNN internal memory is created by cyclic connections through its units. RNN is composed of three input layers (X_{t-1}, X_t, X_{t+1}), three hidden layers (h_{t-1}, h_t, h_{t+1}), three output layers (y_{t-1}, y_t, y_{t+1}), and weight matrices (W, U, V). The RNN accepts one input at time, for example, at time t the **network** is fed with input X_t, then it goes through the hidden layers for output prediction. The hidden layers represent the core of the network, the reason behind their importance is that they keep track of previous inputs. Each hidden layer accepts input from its previous hidden layer, in addition to its input unit, to predict the output. The weigh matrix of the hidden layer must be squared to maintain the same number of inputs as there are outputs.

The weight matrices (W, U, V) are initially set randomly. As for the first hidden layer (h_{t-1}), it is initialized by the dot product performed between its current input X_{t-1} at time t-1 and its weight matrix U, since it has no prior hidden layer and thus no contributions for the output prediction. The activation function such as sigmoid function then accepts the **resulted** dot product to generate values for the first hidden layer. Generally, data is processed in RNN by accepting an input X_t at time t, multiplying it with the weight matrix U and then passing it through hidden layer h_t. The hidden layer h_t also accepts another input from its prior hidden layer h_{t-1}, which has been parameterized with the weight matrix W. These two inputs contribute in predicting the output y_t, where a dot product is performed on the hidden layer h_t and the weight matrix V. This flow is repeated until all layers are covered to predict the final output.

Among the different variants of RNN, long short-term memory networks (LSTMs) [15] have shown effectiveness in the majority of NLP tasks. Thus, in this model the experiments use RNN with LSTM networks.

The long-short-term memory (LSTM) neural network is an extended version of simple RNN, where the past and current memories are related through linear dependence, instead of the non-linear connection between the past and current layers' activity. Furthermore, and most importantly, an LSTM has an introduced forget gate to adjust each of the past memory elements to contribute to the current memory cell.

Generally, each layer of the LSTM network accepts three inputs: X_t, h_{t-1} and C_{t-1}, which denotes the input at time t, the output of the previous LSTM unit, and the memory of the previous unity, respectively. The memory unit represents the most important unit in LSTMs, as it differentiates them from RNNs. The current layer has output h_t and the current unit has memory C_t. LSTMs have shown their effectiveness in captioning long term temporal dependencies. This obviously helped in improving the state-of-the-art for many difficult problems including: handwriting recognition and generation, language modeling and translation, acoustic modeling of speech, speech synthesis, protein secondary structure prediction and analysis of audio and video data among others [16In this model, the new feature grid **V** is created from **V** by running an RNN across each row of that input. **Recursively** for all rows h ∈ {1, . . . , H} and columns w ∈ {1, . . . , W }, the new features are defined as

$$\mathbf{V}hw = \mathbf{RNN}\,(\mathbf{V}h,w-1\,,\,\mathbf{V}hw\,) \quad (1)$$

In order to capture the sequential order information in vertical direction, i use a trainable initial hidden state $\mathbf{V}h,0$ for each row, which we refer to as positional embeddings.

This model uses an attention-based encoder-decoder composed of a pair of recurrent neural networks (RNNs). As for the encoder, it maps the variable-length input sequence to a vector. The decoder then maps the vector back to a variable length output sequence. To achieve maximum conditional probability of an output sequence given an input sequence, the two networks are trained jointly [17]**Decoder** Only considering the grid **V**, the decoder predicts the tokens of output text $\{y_t\}$. The decoder is trained as a conditional language model in order to predict the probability of the following token taking in consideration the history and annotations. The mentioned language model is described on top of a decoder RNN,

$$p(yt+1|y1\,,\ldots,\,yt\,,\,\mathbf{V}) = \text{softmax}(\mathbf{W}out\;\mathbf{o}t\,) \quad (2)$$

Where $\mathbf{o}t = \tanh(\mathbf{W}c\,[\mathbf{h}t\,;\mathbf{c}t\,])$ and $\mathbf{W}out\,,\mathbf{W}c$ are learned linear transformations. The vector h_t is used to summarize the decoding history: $\mathbf{h}t = \mathbf{RNN}\,(\mathbf{h}t-1\,,[yt-1\,;\mathbf{o}t-1\,])$. The context vector $\mathbf{c}t$ is used to capture the context information from the annotation grid.

Table 1: Details of the third experiment

Font Name	Train	Val	Test	All
AGA Kaleelah Regular	9958	3218	3244	16420
Al-Mohand	9132	3162	2813	15107
AL-Qairwan	9152	2928	3132	15212
Andalus	8801	2966	2920	14687
Arabic Transperent	12574	4176	4226	20976
Arabic Typesetting	11480	3676	3832	18988
Arabswell	7946	2578	2578	13102
Arial	11178	3752	3560	18490
Diwani Letter	7078	2208	2388	11674
Maghribi Assile	12209	3783	4070	20062
Midan	11930	3836	3608	19374
Simplified Arabic	8200	2715	2772	13687
Tahoma	11592	3804	3852	19248
Times New Roman	11510	3714	3678	18902
Traditional Arabic	8826	2716	2920	14462
Total	151566	49232	49593	250391

Attention Model: This model follows the mechanism of highly resolving a certain image region and fading the surrounds to focus on the specified region. Using the attention method eases the long-term dependencies modeling by adjusting the focal point over the period. This mechanism arises direct dependence between the model states at different time, which consequently introduces a hidden state h_t at each time step [18]

4 DATA DESCRIPTION AND PREPROCESSING

The used datasets in the current work were selected from KAFD Arabic text database [19] is including texts from a variety of subjects: religious, medicine, science, history, etc., where these texts include names, places, cities, numbers, etc. Each text is uniquely constructed in one of 40 fonts. The dataset texts are varying between the most commonly used fonts in arabic texts, including fonts that are similarly alike like the four fonts (Arabic Transparent, Times New Roman, Simplified Arabic and Arial). Each text is then represented in 10 different font-sizes: 8, 9, 10, 11, 12, 14, 16, 18, 20 and 24 points, then each size is represented in the four styles of Arabic texts: normal, bold, italic and bold-italic. Finally, for each font style, texts are organized in three sets: training, validation and test sets with percentage 60%, 20% and 20%, respectively.

The process of constructing KAFD dataset starts by collecting and constructing texts in total of 28,767 pages, followed by printing the texts using HP Laser jet 600-M601 printer with 1200 x 1200 dpi resolution. The printed papers are then scanned in grayscale using Rioch IS760D. Scanning is performed on different resolutions: 100, 200, 300 and 600 dpi resulting in 115,068 scanned pages saved in files with '.tif' extension.

KAFD has two level variations: page and line levels. This enables researchers to the dataset in either level as preferred. All text images undergo a segmentation process to provide the line level variation of the dataset. This process results in 2,576,024-line images. For training the proposed model, we use the line level variation.

Before beginning the experiments, the dataset is preprocessed as following:

- A python script is used to generate A vocab list of the set of the date (unique characters).
- Spaces in each line is marked by "<SP>" and space is added after each character to be used in the proposed model.
- Another python script is used to generate the corresponding image of each line with different Dots Per Inch (DPI) to have different image qualities.

5 EXPERIMENTS

The modeling is done using the Open NMT (Neural Machine Translation) system [20] It is based on the a PyTorch [21]machine learning library. These experiments are run on a 24GB NVIDIA Tesla K80 GP [22]

In the first experiment, 14462 images of the Traditional Arabic font are used; 8826 images for training, 2716 images for validation and 2920 images for testing, the type of the images is tif and it has a 300 DPI. For the second experiment, the type of images is png and 96 DPI. In addition, the third experiment is a large scale system based on 250391 images representing 15 types of Arabic fonts. Table 1 shows the distribution of these images.

6 RESULTS

The performance measures used to evaluate the system is the Word Error Rate (WER); A python script is developed to calculate the WER of the system depending on the Levenshtein distance [23] It used to calculate the insertion, deletion and substitution percentage. In the first experiment, the WER rate was 20.2%. In the second experiment the rate yielded 17.8%. Hence, it becomes clear to us that the use of the PNG image type with a 96 dpi, has recorded a higher accuracy than the TIf file type with an accuracy of 300 dpi. In the third experiment based on 15 fonts large dataset, the WER was 10.1. Based on these experiments, we found that the large dataset is important for the encoder-decoder approach. In addition,

Table 2: Details of the Font Type

Font Name	WER	Accurse
AGA Kaleelah Regular	0.035	96.495
Al-Mohand	0.032	96.794
AL-Qairwan	0.242	75.831
Andalus	0.152	84.781
Arabic Transperent	0.081	91.920
Arabic Typesetting	0.147	85.258
Arabswell	0.078	92.189
Arial	0.061	93.927
Diwani Letter	0.258	74.182
Maghribi Assile	0.046	95.422
Midan	0.073	92.729
Simplified Arabic	0.053	94.696
Tahoma	0.099	90.124
Times New Roman	0.108	89.193
Traditional Arabic	0.084	91.578
Total	0.102	89.829

the 96 dpi are more suitable for our modeling approach. Table 2 shows WER for each Font Type.

7 CONCLUSION

This paper introduces a deep learning framework for converting the scanned images that contain Arabic text into their corresponding text. This model does not require any knowledge of the underlying language. It is simply trained end-to-end on different font types from the KAFD dataset. Convolutional Neural Networks (CNNs) are used to extract salient features from images and an RNN decoder with a visual attention mechanism is used to generate the output text. The preliminary experiments show that the presented approach is effective. The overall obtained accuracy is 89.82%. However, the individual results for some fonts are higher than this score.

ACKNOWLEDGMENTS

The authors would like to thank Bibliotheca Alexandrina for granting the permission to use its computational resources provided by the High-Performance Computing (HPC) Department. (hpc.bibalex.org)

REFERENCES

[1] A. M. Zeki and M. S. Zakaria, 2004, "Challenges in Recognizing Arabic Characters,".
[2] M. Jaderberg, K. Simonyan, A. Vedaldi and A. Zisserman, 2014, "Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition," arXiv preprint arXiv:1406.2227.
[3] C. Bartz, H. Yang and C. Meinel, 2017, "STN-OCR: A single Neural Network for Text Detection and Text Recognition," arXiv preprint arXiv:1707.08831.
[4] C.-Y. Lee and S. Osindero, 2016, "Recursive Recurrent Nets with Attention Modeling for OCR in the Wild," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
[5] D. K. Sahu and M. Sukhwani, 2015, "Sequence to Sequence Learning for Optical Character Recognition," arXiv preprint arXiv:1511.04176.
[6] A. Ul-Hasan, M. Z. Afzal, F. Shafait, M. Liwicki and T. M. Breuel, 2015, "A sequence learning approach for multiple script identification," in 2015 13th International Conference on Document Analysis and Recognition (ICDAR).
[7] Y. Deng, A. Kanervisto and A. M. Rush, 2016, "What You Get Is What You See: A Visual Markup Decompiler.," arXiv: Computer Vision and Pattern Recognition.
[8] J. Sueiras, V. Ruiz, Á. Sánchez and J. F. Vélez, 2018, "Offline continuous handwriting recognition using sequence to sequence neural networks," Neurocomputing, vol. 289, pp. 119-128.
[9] B. Shi, M. Yang, X. Wang, P. Lyu, C. Yao and X. Bai, 2019, "ASTER: An Attentional Scene Text Recognizer with Flexible Rectification," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 41, no. 9, pp. 2035-2048.
[10] M. A. Radwan, M. I. Khalil and H. M. Abbas, 2018, "Neural Networks Pipeline for Offline Machine Printed Arabic OCR," Neural Processing Letters, vol. 48, no. 2, pp. 769-787.
[11] C. Wick, C. Reul and F. Puppe, 2018, "Calamari – A High-Performance Tensorflow-based Deep Learning Package for Optical Character Recognition," arXiv preprint arXiv:1807.02004.
[12] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, 2001, "Gradient-based learning applied to document recognition," Intelligent Signal Processing, pp. 306-351.
[13] A. Karpathy, 2018, "Stanford university cs231n: Convolutional neural networks for visual recognition,". [Online]. Available: http://cs231n. stanford. edu/syllabus. html/.
[14] D. Britz, 2015, "Recurrent neural networks tutorial, part 1–introduction to rnns.,". [Online]. Available: http://www. wildml. com/2015/09/recurrent-neural-networkstutorial-part-1-introduction-to-rnns/. [Accessed 1 February 2019].
[15] S. Hochreiter and J. Schmidhuber, 1997, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735-1780.
[16] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink and J. Schmidhuber, 2017, "LSTM: A Search Space Odyssey," IEEE Transactions on Neural Networks, vol. 28, no. 10, pp. 2222-2232.
[17] K. Cho, B. v. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio, 2014, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).
[18] C. Raffel and D. P. W. Ellis, 2015, "Feed-Forward Networks with Attention Can Solve Some Long-Term Memory Problems," arXiv preprint arXiv:1512.08756.
[19] H. Luqman, S. A. Mahmoud and S. Awaida, 2014, "KAFD Arabic font database," Pattern Recognition, vol. 47, no. 6, pp. 2231-2240.
[20] G. Klein, Y. Kim, Y. Deng, J. Crego, J. Senellart and A. M. Rush, 2017, "OpenNMT: Open-source Toolkit for Neural Machine Translation," arXiv preprint arXiv:1709.03815.
[21] R. Collobert, K. Kavukcuoglu and C. Farabet, 2011, "Torch7: A Matlab-like Environment for Machine Learning," in BigLearn, NIPS Workshop.
[22] "nvidia Web Site," [Online]. Available: https://www.nvidia.com/en-gb/data-center/tesla-k80/. [Accessed 1 February 2019].
[23] K. U. Schulz and S. Mihov, 2002, "Fast string correction with Levenshtein automata," International Journal on Document Analysis and Recognition, vol. 5, no. 1, pp. 67-85.