Unconstrained Arabic Handwritten Text Recognition Using Convolutional Recurrent Neural Network

Ahmad AbdulQadir AlRababah[1], **Mohammed Khalid Aljahdali**[2]
2

, Abdulrahim Abdulhamid Al jahdali[3], and Mohammed Saleh AlGhanmi[4]
[1, 2, 3, 4]Department of Computer Science
Faculty of Computing and Information Technology in Rabigh, King Abdulaziz University
Rabigh 21911, Saudi Arabia

Abstract
Arabic text is cursive unlike many of most common languages, also, letters have different shapes depending on position of the letter, the shape of the first letter depends on what is after it, the shape of middle letters depend on what is before it and after it, and the shape of the last letter depends on what is before it. Moreover, different letters have very similar shapes. These properties make Arabic text recognition a difficult computer vision task. In this paper we try to solve the Arabic text recognition task without being constrained by a dictionary or a language model. We propose a new neural network architecture that achieves state of the art results in the unconstrained recogni-tion task. Our architecture is convolutional neural network with residual connections, followed by Bi-directional Long Short-Term Memory (BLSTM) layer, then finally a fully connected layer.

Keywords
deep learning; text recognition; convolutional neu-ral network; recurrent neural network; computer vision

1. Introduction

Most of the text that we use in our daily lives is digital. However, handwritten text is present in notes, documents, letters, and historical scripts to name a few, which are still an important part of our society. The Digitalization of hand-written text is very important to allow us to use it with our technologies.

Handwritten text recognition is an open research problem in the field of Computer Vision, there is much research on the topic. Moreover, Arabic handwritten text could be considered a more difficult task than Latin languages, due to the nature of the cursive text, similar characters, and the different ways of writing characters see Fig.1. Also, Arabic handwritten text has much less available datasets.

The emergence of Deep Learning techniques in the field of Computer Vision after the amazing breakthrough on ImageNet by [1]. Since then Deep Learning-based models dominated Computer Vision tasks [2] achieving a higher result than any other state of the art methods. As for text recognition, the trend continued Recurrent Neural Networks and Convolutional Neural Networks based models achieved state-of-the-art re-sults surpassing all other methods.

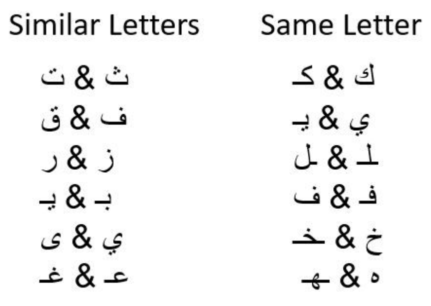


Fig. 1. Difficulties of the Arabic Language Different letters have very similar shapes, and the same letter could have very different shapes.

However, most of the current state-of-the-art results, in the arabic handwritten text recognition are constrained by a limited number of words, and their accuracies are supported by decoders that use the available dictionary of words [3] [4]. This encourages us to try to approach the problem in an unconstrained way, where our model does not require a decoder that limits its output to the available dictionary.

Our method considers images as a sequence of width pixels. We extract the features of the image, this output feature map would be the number of filters in the final convolution layer by the width of the input image downsampled by 4. This sequence is then processed to the BLSTM layer, then finally the output of the BLSTM is decoded using a fully connected layer that outputs the probability of all classes for each point in the sequence. We use the Connectionist temporal classification (CTC) [5] loss function which allows us to calculate a loss for the predicted probabilities of our model based on the ground-truth target, which might be of different length than the predicted sequence. This approach allows us to handle input images of arbitrary sizes. However, the limitation of our method is that the downsampled width length must be equal to or greater than the target sequence length.

Manuscript received January 5, 2025
Manuscript revised January 20, 2025
https://doi.org/10.22937/IJCSNS.2025.25.1.15

This paper proposes a convolutional recurrent neural net-work architecture, that achieves state-of-the-art results in the unconstrained Arabic handwritten text recognition task. Section II gives an overview of the related work in the text recognition task. Section III describes the dataset, preprocessing and augmentation. Section IV describes the proposed architecture. Finally, section V describes the system that is used for training, the experiments with the hyperparameter choices, and the results.

2. Related Work

As for text recognition recently an architecture of BLSTM followed by CTC loss function and used Token Passing and Word Beam Search (WBS) decoder [4]. They also proposed a novel algorithm for adaptive data augmentation (ADA). They Explore different architectural choices, the best performing is BLSTM, CTC loss, WBS, and ADA algorithm. They achieve 95.19% CAR and 96.19% WAR using characters as models, WBS decoder, and ADA algorithm. Moreover, they achieve 86.70% CAR and 83.90% WAR using characters as models and the WBS decoder, without the ADA algorithm. However, these high results come with the help of search decoders which depends on a dictionary of vocabulary.

Another approach for text recognition et al Poznansk [6] developed a CNN architecture, it uses the VGG [7] style which consists of 9 convolution layers, 3 fully-connected layers, and the max-out for the activation layers. Batch normalization used after each convolution, and before each max-out activation. The novelty of the approach was the use of multiple separate and parallel fully connected layers, where each layer leads to a separate group of predictions. Tested on IFN/ENIT they scored a word accuracy rate of 99.29% for the abc-d configuration and 97.07% for the abcd-e configuration.

For the unconstrained text recognition [8] proposed a simple neural network architecture. The architecture contains mostly depthwise convolutions instead of regular convolutions, as well as gate blocks which is the use of attention gates to control the interlayer Information flow, and filter out the insignificant signals. They experimented with many datasets from different languages dataset. they achieved remarkable results of 8.7% CER on the KHATT dataset.

In the task of holistic word classification, [10] proposed a CNN architecture for holistic Arabic handwritten name classifications, they used SUST-ARG which is a dataset of Arabic handwritten names. Their proposed architecture is a CNN consisting of convolutional layers, RuLE activation function, batch normalization, and max-pooling layers. They achieved an accuracy of 99% on 20 classes (names).

For the task of Arabic characters classification a CNN architecture was proposed by [11], also they introduced a new Arabic Character dataset named Hijja. They used a CNN architecture consisting of convolution layers, max-pooling layers, and finally, a fully connected layers. They achieved an accuracy of 88%, precision of 87.88%, recall of 87.81%, and an F1 score of 87.8%.

We used the IFN/ENIT dataset [12], which is considered a benchmark in the field of Arabic handwritten text recognition. It is composed of 946 Tunisian town/village names, written by more than 400 people. The dataset is split into 5 subsets: a, b, c, d, and e, and there are 3 train/test configurations which are: abc/d, bcd/a, and abcd/e.

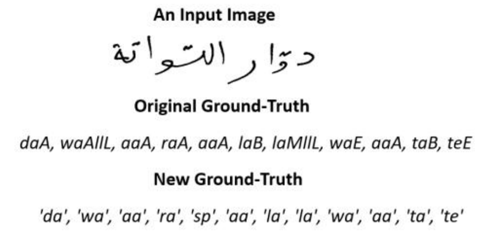

An Input Image
Original Ground-Truth
daA, waAllL, aaA, raA, aaA, laB, laMllL, waE, aaA, taB, teE
New Ground-Truth
'da', 'wa', 'aa', 'ra', 'sp', 'aa', 'la', 'la', 'wa', 'aa', 'ta', 'te'

Fig. 2. The difference between the original provided ground-truth and our new ground-truth

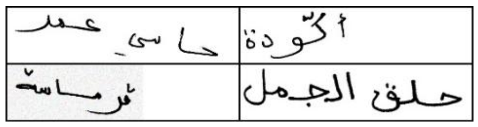


Fig. 3. An example of a batch of images, with different widths. All the images are padded with white space on the right to have the same width as the widest image.

B. Preprocessing

The ground-truth preprocessing [13] is done to reformat the ground-truth files. In the case of the IFN/ENIT dataset, the ground truth is provided in terms of character shapes as modeling units and not in terms of character as modeling units, also in their labeling they do not consider space between two words as a character or a modeling unit. For instance, in an image of a town name that has more than one word, there is no indication that a word ended and another has started. Moreover, there is an addition of 'llL' on any ground truth character shape that has a 'shadda' see Fig.2. We change this format into character as a modeling unit, add a space between words, and remove the 'shadda' indicators.

As for input images preprocessing, we normalize (Normal-ization of images is making their pixel values between 1 and 0) images by dividing their pixel values by 255, this helps the model learn faster and better; because, neural networks process inputs using small weight values, and inputs with large values can disrupt or slow down the learning process [14]. Also, we resize the images to a fixed height and dynamic width to preserve their aspect ratio. When we feed a batch of images to the word recognition model we change the width of all the images in the batch to have the same width as the widest image by adding white padding to the smaller images see Fig.3.

C. Augmentation

As for input augmentation, we use a general geometric augmentation for text images that was proposed by [15], this augmentation helps create more images that realistically appear as written by a different writer see Fig.4. Furthermore, we use arithmetic image augmentations [16], by adding a random value to each pixel, inverting the pixel values, or multiplying each pixel with a random value, these arithmetic operations result in changing the background color, text color, or both of them see Fig.5. Also, we use standard image

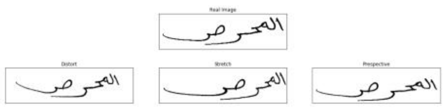


Fig. 4. Real image vs augmented images by general geometric augmentation for text images.

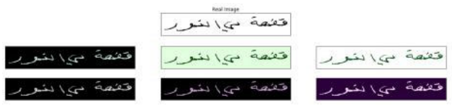


Fig. 6. Real image vs augmented images by Gaussian noise, Poisson noise, rotation, and shear Augmentations.

augmentations Gaussian noise, Poisson noise, rotating, and shearing see Fig.6. Overall, we use these augmentations to improve the generalization capability of the model and reduce overfitting.

3. Methods

Our proposed architecture is Convolutional layers with residuals connections [17], [18], followed by a BLSTM [19]– [21], then fully connected layer to decode the output of the BLSTM see Fig.7, then softmax activation to convert the output into probabilities, and the CTC loss function. Most of the current methods approach this task directly using a recurrent neural network [4]. We think this task is a visual task, thus Convolutional Neural Network (CNN) is needed; however, the content of images are arabic words, arabic letters unlike other languages, depends on what is before and what is after; therefore, using Recurrent Neural Network (RNN) is also appropriate. Consequently, we use the two architectures to solve this task. First we use the CNN to extract the features, downsample the height dimension to 1, and downsample the width dimension by 4. The output feature map becomes the downsampled width dimension by the number of output channels. After that, we treat the width as a sequence and feed it to the BLSTM, each element of the output sequence is decoded by the fully connected layer followed by softmax activation. Then, the final output dimensions will be the downsampled width by the number of classes, meaning that for each width we will have probabilities for each letter. In the case of training, we send the probabilities to the CTC loss function with the ground-truth label, to calculate a loss value for training the model. In the other case, which is the inference, we use greedy decoder where we take the highest probable letter at each width. Because we are using the CTC loss function we will have a blank class beside the rest of the letters, and we have to apply the CTC decoding which is done by two steps, see Fig.8. Firstly, removing all the repetitions of the letters. Secondly, removing all the blank letters. Finally, by applying the CTC decoding we will end up with our predicted sequence of letters.

To discuss our proposed architecture further, the CNN is composed of two building blocks which are the residual blocks and the layers. The residual block contains two convolutional layers each followed by a batch normalization layer and a ReLU activation function, and before the last ReLU activation we apply the residual connection y = f (x) + x. Equation 1 requires that the dimensions of x and f(x) matches, and in the case that they do not match we apply 1x1 convolution to the x, to match their dimensions. The layers are 4 stacked residual blocks, with a dropout layer after each two residual blocks. The CNN is 3 layers, each followed by max-pooling layer, expect the last which is followed by adaptive average pooling, which reduce the height of the feature map to 1, and keep the width, unless the width is more than 256 it will be reduced to 256 by the adaptive average pooling layer. As for the BLSTM layer, it is composed of 2 stacked layers, input dimensions same as the last layer in the CNN, and the hidden dimensions of it is a hyperparameter that we set. Finally we have a fully connected layer, which takes an input size of the BLSTM hidden dimensions, and output size of the number of characters.

$$y = f(x) + x \tag{1}$$

4. Experiment & Results

As for experiments we built a whole system for training, testing, and inference. This system helped us conduct many experiments. In following two sections we will discuss the system, and our results

A. Experiments System

This system uses PyTorch [22] for the neural networks, and PyTorch Lightning [23] to integrate callbacks (which allows us to add features like changing the images size during training), training loggers [24], and 16 bit precision training. Our system has many features that allowed us to experiment easily, which are:

Multiple weight initializers

Increasing the size of images during training
Different learning rate schedulers.

Different optimizers

- Changing learning rate schedulers during training
- Support multiple datasets, by defining a class for that dataset
- Support multiple architectures
- Many hyperparameters choices via CLI
- Logging experiment name, hyperparameters, and results in Weights and Biases
- Many other features

B. Experiments

In these experiments we used 1 GPU, which is the RTX 2080, also we used 16 bit precision for training, which allowed

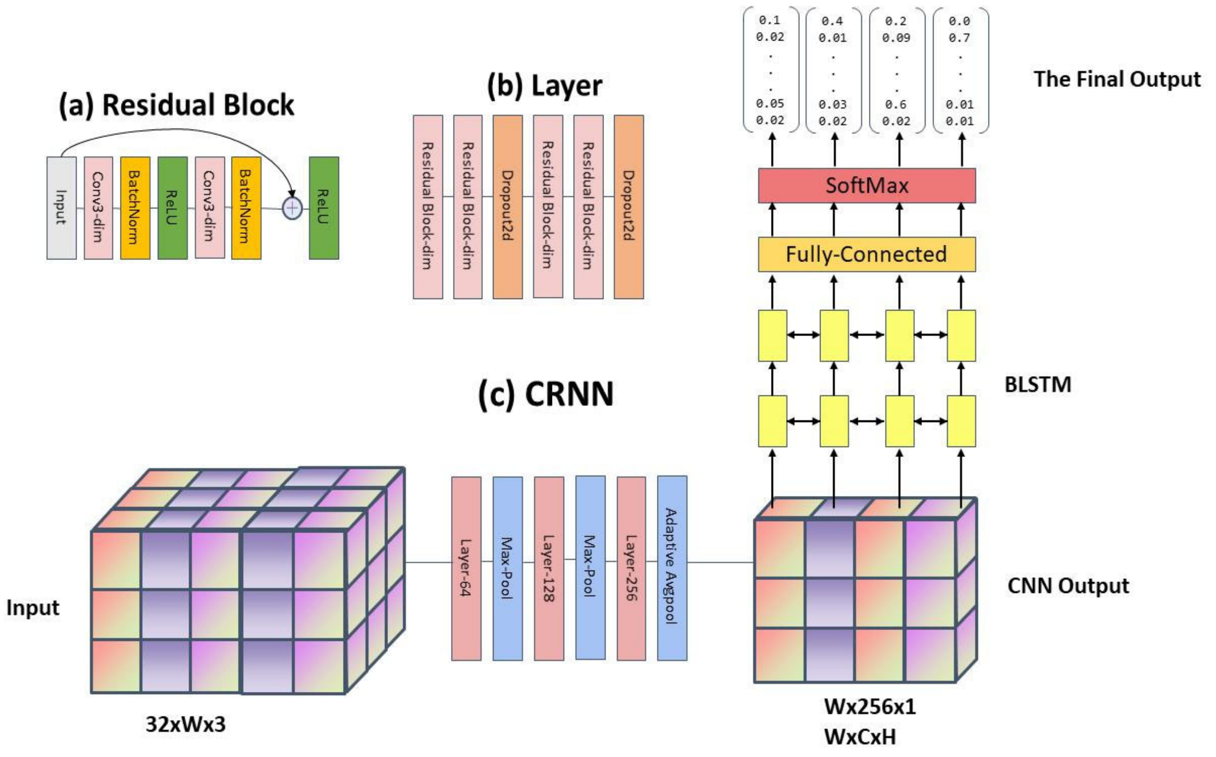
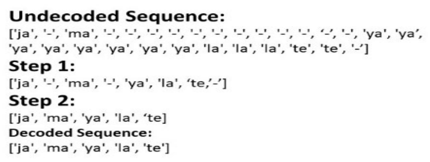


Fig. 7. This figure describes our full CRNN architecture. (a) Is the smallest block in the CNN. (b) Layer is composed of 4 residual blocks and two dropout layers. (c) Is the whole architecture, which consist of a CNN that have 4 Layers, then followed by BLSTM which takes the the output feature map of the CNN as a sequence, than the fully connected layer will output a probability distribution of all the classes for each point in the sequence.



Undecoded Sequence:
['ja', '-', 'ma', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', 'ya', 'ya', 'ya', 'ya', 'ya', 'ya', 'ya', 'ya', 'la', 'la', 'la', 'te', 'te', '-']
Step 1:
['ja', '-', 'ma', '-', 'ya', 'la', 'te,'-']
Step 2:
['ja', 'ma', 'ya', 'la', 'te]
Decoded Sequence:
['ja', 'ma', 'ya', 'la', 'te']

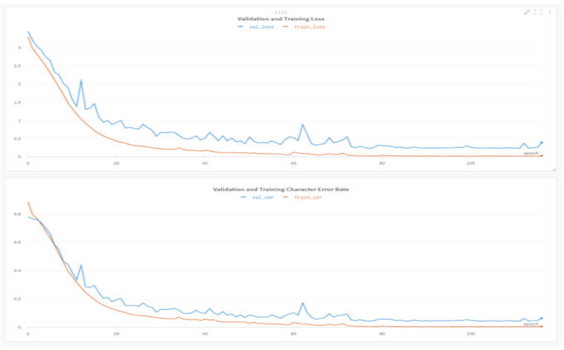


Fig. 8. Example of CTC decoding. Which first removes all the repeations, then remove all the blank labels, resulting in a decoded sequence.

us to train faster and use less GPU memory. As for abc/d configuration of dataset, we used 64 as the initial number of channels in our CNN, and increased the number by factor of 2 in the following layers, we had two dropout layers with 0.55 probability in each layer as seen in Fig.7, we used BLSTM with 2 layers, 256 hidden size, and 0.2 dropout probability. We used the Stochastic Gradient Descent (SGD) optimizer, with a learning rate of 0.025, momentum of 0.9, and weight decay of 1e-4 (0.0001); furthermore, we started with constant learning rate, then used a callback to use exponential learning rate decay policy from the 40
epoch with a factor of 0.965. We used the default weight initialization provided by PyTorch. As for the data related hyperparameters, a batch size of 8 was used, and the initial images height was 32, we used a callback to increase the height of the images by 8, and 16 respectively, which leads in an increase to the width to preserve the aspect ratio, the height increases was in the 123, and 137 epoch respectively. We trained for 149 epochs, which took 1 day and 15 hours. In Fig.9 we see the the loss and the Character Error Rate (CER) plotted over epochs during the training process.

As for the abcd/e configuration, we trained with almost the same hyperparameters as the abc/d configurations expect the height increase was by 8 and 16 in the 120 and 132 epoch respectively, finally the model was trained for 142 epochs.

TABLE I
OUR CRNN MODEL RESULTS, ALSO COMPARED TO OTHER RESULTS.

Methods		CER		
		Configurations		
		abc/d	abcd/e	bcd/a
Ours		1.99	7.27	2.61
BLSTM^{1}	[4]	6.9	11.84	8.59
BLSTM^{2}	[4]	4.81	8.79	6.67

For the final configuration bcd/a, we had to train a new model, where we used also the same hyperparameters as the last two configurations, expect the height increase by 8 and 16 in the 120 and 145 epoch respectively, finally the model was trained for 155 epoch.

C. Results

Overall, we achieve state-of-the-art results in the recognition task without the use of language models, dictionaries, and search-based decoders on the IFN/ENIT dataset; our results can be seen in Table.I. Unfortunately, to the best of our knowledge, there is not any published work that does not use language models, dictionaries, and search-based decoders on the IFN/ENIT dataset. Therefore, we had to compare our results to other methods that use language models, dictionaries, or search-based decoders, which is not a fair comparison, but it also shows us that we do not need to constrained the output of the recognition models to a limited dictionary to achieve the highest results.

5. Conclusion

In conclusion, we propose a neural network architecture that uses CNN and RNN, to solve the Arabic text recog-nition task without constraints. We change the ground-truth format to characters as modeling unit, and apply 3 types of augmentations. Our approach is to recognize letters regardless of their shapes, and also our model accepts inputs of any size. We achieve state-of-the-art results without the use of language models, dictionaries, or search-based decoders. We almost beat the current state-of-the-art results that use language models, dictionaries, or search-based decoders.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Communications of the ACM, vol. 60, no. 6, pp. 84–90, 2017.
[2] A. A. AlRababah, "Neural networks precision in technical vision sys-tems," IJCSNS, vol. 20, no. 3, p. 29, 2020.
[3] G. A. Abandah, F. T. Jamour, and E. A. Qaralleh, "Recognizing hand-written arabic words using grapheme segmentation and recurrent neural networks," International Journal on Document Analysis and Recognition (IJDAR), vol. 17, no. 3, pp. 275–291, 2014.
[4] M. Eltay, A. Zidouri, and I. Ahmad, "Exploring deep learning ap-proaches to recognize handwritten arabic texts," IEEE Access, vol. 8, pp. 89 882–89 898, 2020.
[5] A. Graves, "Connectionist temporal classification," in Supervised Se-quence Labelling with Recurrent Neural Networks. Springer, 2012, pp. 61–93.
[6] A. Poznanski and L. Wolf, "Cnn-n-gram for handwriting word recog-nition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2305–2314.
[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

[8] M. Yousef, K. F. Hussain, and U. S. Mohammed, “Accurate, data-efficient, unconstrained text recognition with convolutional neural net-works,” Pattern Recognition, vol. 108, p. 107482, 2020.

[9] S. A. Mahmoud, I. Ahmad, W. G. Al-Khatib, M. Alshayeb, M. T. Parvez, V. Margner,¨ and G. A. Fink, “Khatt: An open arabic offline handwritten text database,” Pattern Recognition, vol. 47, no. 3, pp. 1096–1112, 2014.

[10] M. E. Mustafa and M. K. Elbashir, “A deep learning approach for handwritten arabic names recognition,” 2020.

[11] N. Altwaijry and I. Al-Turaiki, “Arabic handwriting recognition system using convolutional neural network,” Neural Computing and Applica-tions, pp. 1–13, 2020.

[12] M. Pechwitz, H. El Abed, and V. Margner,¨ “Handwritten arabic word recognition using the ifn/enit-database,” in Guide to OCR for Arabic Scripts. Springer, 2012, pp. 169–213.

[13] A. A. Q. AlRababah, “On the associative memory utilization in english-arabic natural language processing,” International Journal of Advanced and Applied Sciences, vol. 4, pp. 14–18, 2017.

[14] T. Jayalakshmi and A. Santhakumaran, “Statistical normalization and back propagation for classification,” International Journal of Computer Theory and Engineering, vol. 3, no. 1, pp. 1793–8201, 2011.

[15] C. Luo, Y. Zhu, L. Jin, and Y. Wang, “Learn to augment: Joint data augmentation and network optimization for text recognition,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 13 746–13 755.

[16] A. B. Jung, K. Wada, J. Crall, S. Tanaka, J. Graving, C. Reinders, S. Ya-dav, J. Banerjee, G. Vecsei, A. Kraft, Z. Rui, J. Borovec, C. Vallentin, S. Zhydenko, K. Pfeiffer, B. Cook, I. Fernandez,´ F.-M. De Rainville, C.-H. Weng, A. Ayala-Acevedo, R. Meudec, M. Laporte et al., “imgaug,” https://github.com/aleju/imgaug, 2020, online; accessed 01-Feb-2020.

[17] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[18] ——, “Identity mappings in deep residual networks,” in European conference on computer vision. Springer, 2016, pp. 630–645.

[19] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[20] A. Graves, M. Liwicki, S. Fernandez,´ R. Bertolami, H. Bunke, and J. Schmidhuber, “A novel connectionist system for unconstrained hand-writing recognition,” IEEE transactions on pattern analysis and machine intelligence, vol. 31, no. 5, pp. 855–868, 2008.

[21] A. Graves and J. Schmidhuber, “Offline handwriting recognition with multidimensional recurrent neural networks,” Advances in neural infor-mation processing systems, vol. 21, pp. 545–552, 2008.

[22] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in Advances in Neural Information Processing Systems 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alche´-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[23] W. Falcon, “Pytorch lightning,” GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning, vol. 3, 2019.

[24] L. Biewald, “Experiment tracking with weights and biases,” 2020, software available from wandb.com. [Online]. Available: https://www.wandb.com/