IOSR Journal of Engineering (IOSRJEN) **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
Vol. 09, Issue 5 (May. 2019), ||S (IX) || PP 24-32

A Review Paper on Handwritten Character Recognition Using Machine Learning

Neha Chauhan[1], Prof. Sanjay Balwani[2], Prof. Mayuri Chawla[3]
[1,2,3] *Department of Electronics & Telecommunication,* [3]*Jhulelal Institute of Technology,* Nagpur, India

Abstract: *In this paper, we propose a new neural network architecture for state-of-the-art handwriting recognition, alternative to multi-dimensional long short-term memory (MD- LSTM) recurrent neural networks. The model is based on a convolutional encoder of the input images, and a bidirectional LSTM decoder predicting character sequences. In this paradigm, we aim at producing generic, multilingual and reusable features with the convolutional encoder, leveraging more data for transfer learning. The architecture is also motivated by the need for a fast training on GPUs, and the requirement of a fast decoding on CPUs. The main contribution of this paper lies in the convolutional gates in the encoder, enabling hierarchical context-sensitive feature extraction. The experiments on a large benchmark including seven languages show a consistent and significant improvement of the proposed approach over our previous production systems. We also report state-of-the-art result son line and paragraph level recognition on the IAM and Rimes databases.*

Keywords: *Machine learning, Handwriting recognition, neural networks, MD-LSTM, Character sequences, segmentation and recognition algorithms*

I. Introduction

Handwriting recognition has been one of the most fascinating and challenging research areas in field of image processing and pattern recognition in the recent years. It contributes immensely to the advancement of automation process and improves the interface between man and machine in numerous applications. In general, handwriting recognition is classified into two types as off-line and on-line handwriting recognition methods. The on-line methods have been shown to be superior to their off-line counter parts in recognizing handwritten characters due to the temporal information available with the former. However, in the offline systems, comparably high recognition accuracy level is obtained.

In on-line recognition multi-dimensional long short-term memory recurrent neural networks (MDLSTM- RNNs)[12],[13] became established as the state-of-the-art model for handwriting recognition with the help of neural networks. They were involved in all winning systems in international evaluations and are now a standard component of industrial systems. Yet, they have conceptual drawbacks. Most notably, the features computed by an MDLSTM layer at a given position may depend on quite "distant" input features. In other words, there is no explicit control of the input context. We observed that this lack of control of context dependency makes the features learnt on text lines images not generalizable to paragraph images. In the scope of handwriting recognition, we think that the context of the whole image is not relevant to predict a single character.

Instead, an area covering the character, and maybe surrounding ones, should be sufficient.

In the wake of the recent developments in deep learning, it would be beneficial to build a network that extracts reusable textual features. A current trend in machine learning is to obtain generic features with a neural network, trained on a lot of data for a well understood task. Such features can then be reused to train new models on different, and sometimes more difficult tasks with less data. Prominent examples can be found in computer vision, where AlexNet or VGGNet [11] are widely used as feature extractors for new tasks with visual inputs. In natural language processing, word embedding's are very popular. One of the goal of this work is to propose a network architecture that could be an AlexNet of document processing.

Nowadays, the automatic recognition of handwritten text is no longer the main bottleneck of production systems for document analysis and processing. Their performance is more limited by the line detection and extraction in complex documents, or language identification in multilingual streams. Deep learning opens new opportunities to solve these tasks, and we believe that learnt features that are good for text recognition could represent interesting cues to tackle these problems. To exploit the large quantity of data to improve the accuracy of models, we need a way to train the networks in days rather than weeks. On the other hand, speed requirements are crucial in production systems. MDL-STMs are not as easily parallelizable as convolutions are, for example, although optimized GPU implementations have been proposed. Bidirectional

LSTMs (BLSTMs) [14] are also faster, but require a sequential input. In the literature of neural networks, BLSTMs are often applied to language, and convolutions to images.

Following those observations, we propose a neural network architecture made of a convolutional encoder of the input image and a BLSTM decoder predicting the sequence of characters. Such architectures have been explored in the past but have not yet outperformed the state-of-threat MDLSTM. MDLSTMs include complex neurons, with an elaborated gating system, designed to control the information flow, and solving the vanishing and exploding gradient issues in recurrent networks. More than merely enabling an efficient recurrence, we believe that the gates are a crucial component in the success of those architectures, maybe more so than the recurrence itself. The main contribution of this paper is the addition of convolutional gates within the convolutional encoder of images. We evaluated our model on a big multilingual benchmark made of public and private data in seven languages, as well as on standard databases or dataset. We show that this new architecture outperforms the state-of-the art MDLSTM, and allows to extract generic features that are reusable across languages.

II. Proposed Model

A. Neural Network

Overview Following the observation that convolutional neural networks are widely used in computer vision, and bidirectional LSTM are very popular for language applications, we built a deep neural network that could conceptually be split into three main parts. The encoder of the input image is made of convolutional layers. It processes two-dimensional representations and provides 2D features maps. It contains only about 20% of the model's free parameters but represents the slowest component of the architecture ($\approx$ 80% of the computation).

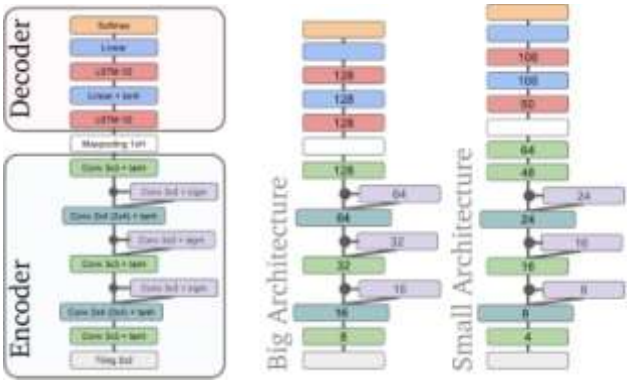

Fig 1. Proposed neural network

The goal is to make it as generic as possible to be reusable in order to factorize the processing time. The interface transforms the 2D image-like representation into the expected 1D representation (we predict sequences of characters). The decoder is a bidirectional LSTM RNN that processes feature sequences to predict sequences of characters.

B. Motivations

Several motivations led to the design of the proposed model. Leveraging research experience from other fields: a lot of research and good results were reported with convolutional neural networks on images (e.g. for object recognition) and LSTMs on language (e.g. speech recognition, machine translation). Since the inputs of our system are images, and the outputs are sentences, it makes sense to use the presented architecture.

i. Reusable features:

The lower layers of a neural network can be interpreted as a learnt feature extractor. In many applications, the lower layers of a network trained for a specific task are used to perform a new task. The problem with LSTMs is the lack of control on the locality of the extracted features, Using convolutions makes the result independent of the big picture (the same information is extracted at a given position, independently of the input being a word, line, and paragraph or page image). Since the extractedfeatures are used to recognize the text, they hold some textual-content information and could be useful for language identification, neural document layout analysis, attribute detection, attention models, and so on.

i. GPU training:

MDLSTMs do not lend themselves easily to GPUs (low expected speedup, hard to handle variable sized inputs, not implemented in most public neural net GPU libraries)[19]. If we are to train our models with a lot of data, CPU training will become prohibitive in training time. Finding good, GPU-compatible models was a pre-requisite to the Design of the model. Using components like convolutions, linear layers, and 1D recurrences was the main constraint.

ii. Tradeoff between speed, size, and accuracy:

One of the most important concern for production systems is to provide accurate systems, which are also fast and small to enable on device recognition. Analyzing the models and layers showed that convolutions are faster than LSTMs, and 1D LSTMs are faster than MDLSTM.

C. Multilingual System Traditionally:

We train our systems on language-specific data. For example, for the French neural network, we only use available data in French. For low resource languages, we fine-tune a neural network trained on another language, but still on language specific data. There is however a major issue with that approach: when we do not have much data for one language, it often means we only have one collection of documents. Thus, not only do we adapt the model to a new language, but we also adapt it to a specific collection, which might then be risky in production.

In our model, we first train the network on all available data, in all considered languages. Thus the encoder should extract features that are good across different collections. In a subsequent step, we fine-tune only the decoder to each language. This approach has several benefits. First, the encoder represents most of the computation time. Adapting only the decoder results in a very fast training. Since we keep a shared encoder, it does not specialize to a specific language or collection, so we can hope to have relatively generic features. Moreover, the model is now factorized, as its first part is shared across languages, which saves storing size.

In production setups, it might also be the case that the language is not known in advance. Since we trained a generic network first, it may be used in those in certain situations. Moreover, if the features of the encoder turn out to be generic enough, one may imagine using them for other purposes, such as document layout analysis or the rejection of incorrect segmentations.

Once again, the benefit could be a factorization of document processing.

III. Character Dataset

For applying machine learning algorithms, we need a dataset. To generate a dataset, we need to go through various stages (Fig. 1). As mentioned.

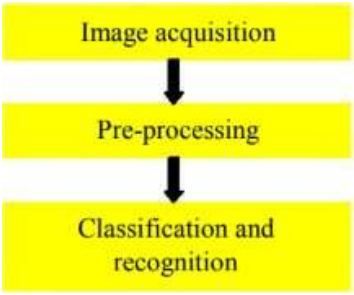

Fig 2. Overall procedure of implementation

1. Image acquisition

The first stage here is to obtain a clear image to an extent such that its zoomed view should not appear blurred. Image enhancement should not be needed, as an unclear image may not aid our task.

2. Pre-processing

If image is not enhanced in the previous step, we need to do it here. We apply spatial image filtering, global image thresholding processes, etc. Once this step is over, noise removal techniques are applied here (Fig. 3). The following algorithmic steps are used in pre-processing:

i. Noise removal :

Before sending the image for the further stages, we need to pre-process the image. Then we will be able to read the images clearly or else there will be more/less gaps between edges, characters, etc. Some parts of these techniques come under morphological operations.

ii. Colorization and background detection :

After noise removal, the image is sent further to detect various colors in and around the nameplate. Moreover, the background surrounding the characters is also detected and finally we have separated the characters from its surroundings.

iii. Edge detection:

In image processing, all the unnecessary data is removed and just the edges are kept. This helps in processing the given data.

iv. Plate segmentation :

This is an additional step and may not be used always. If the image is not processed in the background detection step, it is done here. As soon as characters are separated from nameplate, the job of this stage is over.

v. Character segmentation:

This is an important step because the characters are of different scripts. Some nameplates contain English, some Hindi and so on. In-depth processing is done here, as there are separate character segmentation of different languages.

vi. Erosion and dilation:

We need to increase or decrease the objects in size and that is done in this stage. The rule says that an image is made smaller by eroding the pixels on its edges while that same image is made larger by adding pixels around its edges.

vii. Skew detection/correction :

The image at this stage is skewed and rotated at an angle such that the image is viewed from a desired point. In some cases, the image appears to be laterally inverted and this stage operation is useful in such cases.

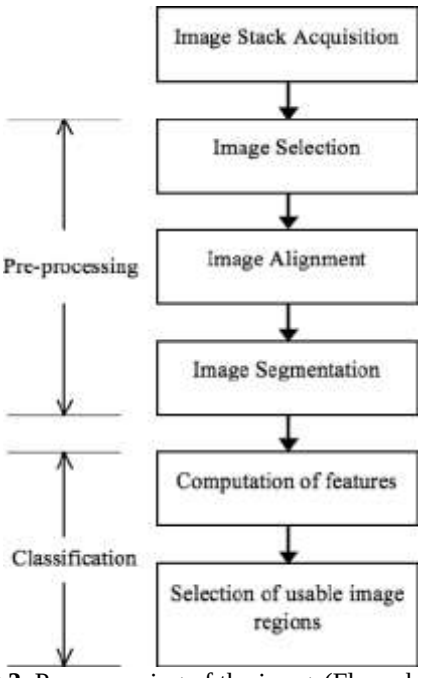

Fig 3. Preprocessing of the image(Flow chart)

IV. Gate Convolutions

In this section, we present the convolutional gates that we use in the encoder. The idea is depicted on Fig. 4. The gate controls the propagation of a feature to the next layer basically, the gate looks at the feature value at a given position, and at neighboring values, and decides whether that feature at that position should be kept or discarded. It allows to compute generic features across the whole image, and to filter when, according to the context, the features are relevant.

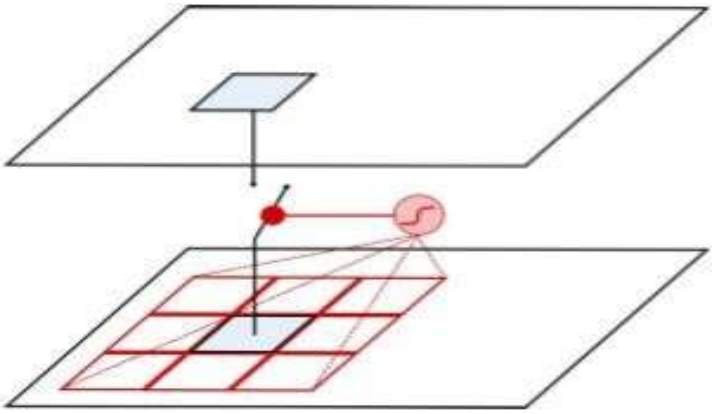

Fig 4. Convolutional Gate (in red): by applying a convolution filter, followed by a sigmoid activation, the system learns in which context a computed feature is relevant.

The gate (g) is implemented as a convolution layer with sigmoid activation. It is applied to the input feature maps x. The output of the gating mechanism is the point wise multiplication of the input with the output of the gate:

$y = g(x).x$ (1)

Where

$g(xij)=\sigma(w00xi-1,j-1 + w01xi-1,j + w02xi-1,j+1 + w10xi,j-1 + w11xi,j + w12xi,j+1 + w20xi+1,j-1 + w21xi+1,j + w22xi+1,j+1)$

Which is reminiscent of the input or output gates in LSTMs, except that the context is in the input space rather than coming from outputs of the layer at neighboring positions. In that sense, it is also quite similar to the gating mechanism proposed in Quasi-RNNs[15].

Fig. 4, is actually a simplistic representation of the gate. In fact, the gate computes a value in [0, 1] for each feature (i.e. each dimension of the feature vectors at each position), based on the whole feature vectors at neighboring positions. Thus the context taken into account to decide whether a given value is relevant is relatively rich.

In Fig. 5, we show a real example of what the gate does. In the example, the gate is applied to intermediate feature maps computed from the image of the word "television". We display the feature maps before and after the gate, as well as the maps of gate values.

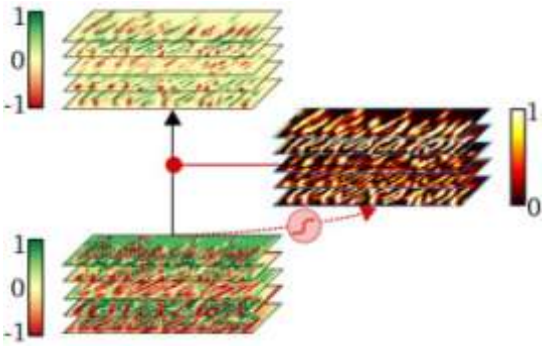

Fig 5. Visualization of the effect of a convolutional gate.

That example illustrates what, in our opinion, is very interesting with gates. First, we see that there is an effective selection or filtering of features. The gating system allows a feature to be excitatory, inhibitory, or absent, whereas with a tanh it has either a positive or a negative effect, and with a sigmoid it is either present of absent.

V. Visualization Of The Dataset And Classification

Now the process of applying classification algorithms comes into picture. The characters are identified using Gated Convolutional Recurrent Neural Networks.

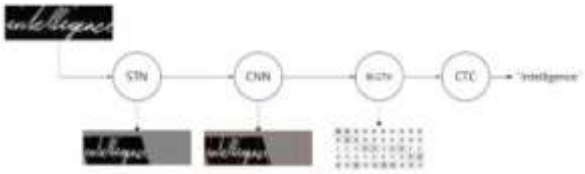

Fig 6. Processing of the image back end process

In this section, we present the convolutional gates that we use in the encoder. The idea is depicted on Fig. 6. The gate controls the propagation of a feature to the next layer.

Basically, the gate looks at the feature value at a given position, and at neighboring values, and decides whether that feature at that position should be kept or discarded. It allows to compute generic features across the whole image, and to filter when, according to the context, the features are relevant.

VI. Experimental Setup

To train our model, we used as much data as possible, from various sources. We did not find any public data for some languages, so we collected data from other sources to build private collections. For several collections, we did not have a line-level ground-truth.

VII. Architecture Details

We built neural networks according to the principles previously described. To optimize different objectives (accuracy, speed and size), we designed two architectures. Both are made of a convolutional encoder, a max-pooling across the vertical dimension and a recurrent decoder. The encoder consists of a 3x3 convolutional layer with 8 (resp. 4) features, a 2x4 convolutional layer with 16 (resp. 8) features, a 3x3 convolutional gate, a 3x3 convolutional layer with 32 (resp. 16) features, a 3x3 convolutional gate, a 2x4 convolutional layer with 64 (resp. 24) features, a 3x3 convolutional layer with 128 (resp. 32) features, for the big (resp. small) network. The encoder of the small network has an additional 3x3 convolutional layer with 64 features. The decoders are made of 2 bidirectional LSTM layers of 128 (resp. 50 and 100) units, with a linear layer of 128 (resp. 100) neurons in between.

For each architecture, two networks are created: a slow network operating on original images, and a fast network operating on downscaled images to about 70% of their original size. We will refer to these networks as Accurate (A) and Fast (F) for the two instances of the big architecture, and Fast Small (S) and Faster Small (X) for the two instances of the small architecture. Overall, the big architecture contains about 750k parameters and the small one 300k. After weight quantization to 12 bits (which does not hurt the performance), these networks occupy respectively 1.1MB and 500kB of disk space.

VIII. Training

In this section, we present the results of the networks trained on all the available data, in all languages. We will refer to these networks as Generic models, since they are not specific to a language. We compare the four networks (Accurate, Fast, Fast Small and Faster Small) to the previous version of A2iA Text Reader.

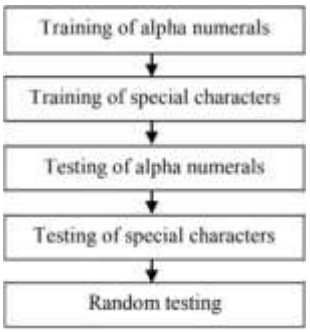

Fig 7. Machine learning procedure

We trained the network to minimize the Connectionist Temporal Classification [20] objective function. We performed the optimization with stochastic gradient descent, using the RMS Prop method with a base learning rate of 0.0004 and mini batches of 8 examples.

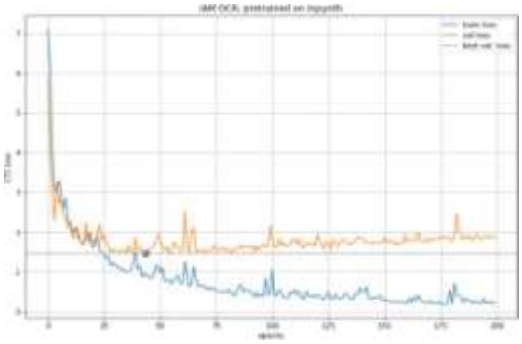

Fig 8. CTC loss Vs Epochs graph

Following the motivations previously exposed, we first train each of the four models on all the available data, to obtain generic Latin-script neural networks. The decoder part is subsequently adapted to each language, using the features extracted with the encoder. The encoder is not fine-tuned to maintain generic, language and collection-independent features. For comparison purposes, we also adapted to whole architecture to each language, and trained neural networks without gates.

IX. Review

In this paper, we propose a new neural network architecture for state-of-the-art handwriting recognition, alternative to multi-dimensional long short-term memory (MD-LSTM) recurrent neural networks. The model is based on a convolutional encoder of the input images, and a bidirectional LSTM decoder predicting character sequences. In this paradigm, we aim at producing generic, multilingual and reusable features with the convolutional encoder, leveraging more data for transfer learning. The architecture is also motivated by the need for a fast training on GPUs, and the requirement of a fast decoding on CPUs. The main contribution of this paper lies in the convolutional gates in the encoder, enabling hierarchical context-sensitive feature extraction.

The experiments on a large benchmark including seven languages show a consistent and significant improvement of the proposed approach over our previous production systems. We also report state-of-the-art resultson line and paragraph level recognition on the IAM and Rimes databases.

A. Language-Specific Adaptations

To improve the performance of the models, we subsequently adapt the generic models on language-specific data. As previously mentioned, we believe that keeping a shared encoder brings many advantages, such as ensuring that the features are not collection-specific. However, we need to make sure that we do not lose too much potential accuracy by fine tuning only the decoder. In Table II, we see that adapting only the decoder already brings a lot of improvement, making all models better than their counterpart in the previous version of A2iA Text ReaderTM, by 10 to 40%. Adapting the whole network is rarely helpful, especially for languages with less data. In these cases, the error rates may even be higher than those obtained when only the decoder is

adapted. Those results validate our approach and intuition.

X. Discussion

In the proposed architecture, we have conceptually decoupled an "image-level" model made of convolutions and a "language-level" model made of recurrent layers. By training the encoder on a big amount of data, in several languages and coming from different collections, we aimed at learning generic textual features, which can be used to recognize several languages, but also in a transfer learning scenario to learn different tasks. We have seen that even with no fine-tuning to a specific language, the whole network gives competitive results, and could be applied when the language in the document is not known in advance, or when storage space requirements are small and one model per language do not fit.

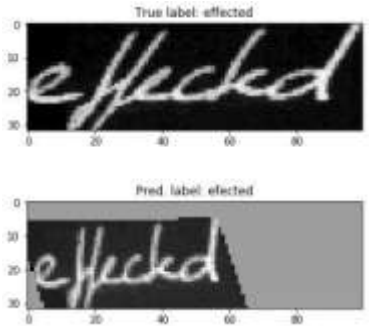

Fig 9. Example of features extracted by the encoder.

The fine-tuning of the decoder yields good results in every tested language, allowing to share a significant part of the network across languages. The processing is factorized since all decoders operate with a common encoder. When inspecting the features learnt by the encoder, we observed that most of them were indeed generic textual features. For example, we see in Fig. 9, that one seems to detect accents and i-dots while another responds to n-like shapes We have carried out several experiments such as language classification and wrong line segmentation rejection from these features and obtained promising results, which are beyond the scope of this paper, but confirm the interest of this approach.

XI. Future Scope

Going by the issues addressed in this paper, we conclude That there is a huge scope to extend this work forward. We can go further in recognizing characters in a more challenging scenario.
Machine learning has been applied to a number of applications. Some of the literatures covering these are languages other than English, namely, Latin, Cyrillic, Arabic, Hebrew, Indic, Bengali (Bangla), Devanagari, Tamil, Chinese, Japanese, Korean, etc. Highlight in 1950's, applied throughout the spectrum of industries resulting into revolutionizing the document management process. Optical Character Recognition or OCR has enabled scanned documents to become more than just image files, turning into fully searchable documents with text content recognized by computers.

XII. Conclusion

The paper addresses and implements a useful method of character recognition. Handwritten recognition was the main aim, as printed text recognition did not require many datasets for the machine to be trained. A dataset of around 200 samples was applied for those 36 alpha numerals and further 100 for special characters. Gated Convolutional Recurrent Neural Network & machine learning proves to be useful as classification labels can be obtained from the features. Ultimately, the data is stored in excel sheet for further analysis.

References

[1]. Renuka Kajale1, "Supervised machine learning in intelligent character recognition of handwritten and printed nameplate"2017 International Conference on Advances in Computing, Communication and Control (ICAC3), 2017*)*
[2]. Subodh L. Wasankar, "Machine Learning with Text Recognition" 2010 IEEE International Conference on Computational Intelligence and Computing Research,2010.I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
[3]. Mithili N Mayekar, "Implementation Of Machine Learning Algorithm For Character Recognition On GPU"Proceedings of the IEEE 2017 International Conference on Computing Methodologies and Communication (ICCMC), 2017.
[4]. -Dippal Israni, "Vehicle Classification and Surveillance using Machine Learning Technique"2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)..

[5]. Moriwake Ryo, ”Character Recognition in Road Signs Using a Smartphone” 2017 6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI).M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.
[6]. Introduction to machine learning by Nils .J. Nilssons
[7]. Machine learning for absolute beginner by Oliver theobald
[8]. https://www.researchgate.net/publication/320442536_A_Survey_on_ Optical_Character_Recognition_System
[9]. https://www.researchgate.net/publication/320442536_A_Survey_on_ Optical_Character_Recognition_System
[10]. https://ieeexplore.ieee.org/Xplore/home.jsp
[11]. https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg- googlenet-resnet-and-more-666091488df5
[12]. https://ieeexplore.ieee.org/document/8313738/authors#authors
[13]. Vishal Chavan, “Printed text recognition using BLSTM and MDLSTM for Indian languages” IEEE Xplore: 12 March 2018.
[14]. https://medium.com/@kangeugine/long-short-term-memory-lstm- concept-cb3283934359
[15]. https://www.kaggle.com/bkkaggle/keras-quasi-recurrent-neural- network
[16]. https://books.google.co.in/books?id=YrU8DwAAQBAJ&pg=PA955 &lpg=PA955&dq=synthetic+line+snippets&source=bl&ots=hD6Baa 5REX&sig=ACfU3U3PbhlMHvqkr1KRWXf1npRt7wyizA&hl=en&sa=X&ved=2ahUKEwiN_- yL2aLhAhXKfH0KHZpSBXkQ6AEwAXoECAgQAQ#v=onepage& q=synthetic%20line%20snippets&f=false
[17]. https://www.a2ia.com/en/research-development
[18]. Steve Renals, “Multilingual training of deep neural networks” 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Electronic ISBN: 978-1-4799-0356-6.
[19]. https://cloud.google.com/ml-engine/docs/tensorflow/using-gpus.
[20]. https://en.wikipedia.org/wiki/Connectionist
[21]. temporal_classification