Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2023.039528

ARTICLE

Text Extraction with Optimal Bi-LSTM

Bahera H. Nayef[1,*], Siti Norul Huda Sheikh Abdullah[2], Rossilawati Sulaiman[2] and Ashwaq Mukred Saeed[3]

[1]Computer Techniques Engineering Department, Ibn Khaldun University College, Baghdad, 10011, Iraq

[2]Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, Bangi, Selangor, 43600, Malaysia

[3]School of Electrical Engineering and Artificial Intelligence, Xiamen University Malaysia, Sepang, 43900, Malaysia

*Corresponding Author: Bahera H. Nayef. Email: bahera.hani@ik.edu.iq; bahera_hani@yahoo.com

Received: 03 February 2023 Accepted: 09 June 2023 Published: 08 October 2023

ABSTRACT

Text extraction from images using the traditional techniques of image collecting, and pattern recognition using machine learning consume time due to the amount of extracted features from the images. Deep Neural Networks introduce effective solutions to extract text features from images using a few techniques and the ability to train large datasets of images with significant results. This study proposes using Dual Maxpooling and concatenating convolution Neural Networks (CNN) layers with the activation functions Relu and the Optimized Leaky Relu (OLRelu). The proposed method works by dividing the word image into slices that contain characters. Then pass them to deep learning layers to extract feature maps and reform the predicted words. Bidirectional Short Memory (BiLSTM) layers extract more compelling features and link the time sequence from forward and backward directions during the training phase. The Connectionist Temporal Classification (CTC) function calcifies the training and validation loss rates. In addition to decoding the extracted feature to reform characters again and linking them according to their time sequence. The proposed model performance is evaluated using training and validation loss errors on the Mjsynth and Integrated Argument Mining Tasks (IAM) datasets. The result of IAM was 2.09% for the average loss errors with the proposed dual Maxpooling and OLRelu. In the Mjsynth dataset, the best validation loss rate shrunk to 2.2% by applying concatenating CNN layers, and Relu.

KEYWORDS

Deep neural network; text features; dual max-pooling; concatenating convolution neural networks; bidirectional long short memory; text connector characteristics

1 Introduction

Optical Character Recognition (OCR) can detect and categorize visual patterns into characters from a digital text image [1]. The pattern recognition process includes applying a set of image segmentation, feature extraction, and classification. OCR systems are a combination of pattern recognition and artificial intelligence techniques. OCR technology assists users in converting various types of documents, such as scanned papers, pdf files, and images taken by a digital camera, into editable data [2]. Since OCR technology became PC-based, users can perform OCR on the image to recognize the text using their computers, mobile phones, and tablets to select, copy, search, and edit the text [3]. There are two approaches to optical recognition, either offline or online. The offline recognition process is used for the preprinted characters.

Nevertheless, online recognition will activate its process while writing the characters. The quality of the recognition process relies on the quality of the input text (single character or script), either printed or handwritten [2]. The offline recognition system performance depends on static data such as bitmap images, making it more complicated than online recognition due to the need for image segmentation techniques [4]. According to [5], the full availability of online systems is easy to develop, has good accuracy, and is compatible with tablets and PDFs.

In correspondence to the input data type, the OCR can be classified into two types, the first is Machine printed recognition which the characters have uniform dimensions [4]. The second is Handwriting recognition which is more complicated due to various handwritten styles and different kinds of pen movements by a single user for a unique character [6].

Since the early nineties, a new era of recognition and classification research started by using deep learning networks (DNN) [7]. DNN performance depends on the initial initialization using unsupervised pre-training it will give outstanding results and acceptable running time [8]. DNN proved its superior in speech recognition, Natural language processing, and image classification over the classical state-of-the-art AI machine learning techniques [9–11]. The unsupervised pre-training approach is not the only method to train DNN efficiently.

The above-discussed studies revealed that the performance of deep learning applied in text extraction from images was outstanding. Our contribution is to improve the design of the deep learning model proposed by [12] by concatenating the first two CNN layers to extract the best features from both these layers individually and then merge them to be sent to the next layer. Also, propose dual max-pooling techniques for downsampling the extracted feature size and speeding up the training phase with maintaining good model performance. In addition, using OLRelu with CNN enhanced features extraction to include both the positive and negative features as well.

The proposed methodology aims to answer the following questions:

1- Does concatenate CNN layers improve text feature extraction from the word level?
2- How do downsample the size of the extracted features and speed up the training process?

This study aims to recognize handwritten characters from word images more accurately and reconstruct the words with minimum error per character. Also, the study introduces a model that reduces the number of epochs and the number of model parameters required to reach the best results for the training and validation error rates.

2 Related Work

Lately, the attention of researchers has been redirected toward the use of Deep Learning for digitizing handwritten documents in different languages. In the study [13], the researchers used Dropout to prevent overfitting and speedup running time and this leads to improving DNN performance. Convolutional Neural Networks are used by [14], Deep belief and Boltzmann NN were also conducted as an approach to overcome the overfitting issue [8,15]. The [16] and [17] studies discussed extracting handwritten characters of Urdu and Jawi languages from raw images using CNN and Recurrent Neural Networks (RNN). Other researchers, such as [18,19], discussed cursive and multi scales text detection in the natural scene images problem but using different approaches, where the study by

[18] discussed three neural network architectures, namely Visual Geometry Group (VGG16), Scalable Vector Graphics (SVG), and Residual Network (ResNet50), for feature extraction. The datasets used in their performance test of the proposed approaches were the Incidental Text proposed in the International Conference on Document Analysis and Recognition (ICDAR2015), (MSRA-TD500), the Focused Text ICDAR2013, Reading Chinese Text in the Wild (RCTW-17), Chinese scene text dataset (CASIA-10K), and Multi-lingual scene text (MLT-17) Also, while the latter [19] used only two datasets, the ICDAR2013, and ICDAR2015, to extract features from Region of Interest (ROI) only. This shift occurred due to cluster computing, GPUs, and better deep learning architecture performance, which embraces (RNN), CNN, and Long Short Term Memory networks (LSTM) [20]. An interesting study for multi scales object detection was proposed by [21]. The study framework is divided into two parts, the first part is concerned with extracting features from five CNN layers. These multi scale features are converted into multi-scale proposal generators by concatenating multiple Region Proposal Networks (RPN). The proposed methodology showd outstanding results for all datasets in the experiment. But, the proposed methods suffer from a high computation time and require a processor with high specifications.

The output of reviewed studies was outstanding, Albeit, the error rates for recognizing characters are still high. Especially when the text image contains noise such as shadows, cropped characters, and faded characters.

3 Text Extraction Using Deep Learning

A typical OCR system consists of several components, as shown in Fig. 1. The first step is to digitize the analog document using an optical scanner. Segmentation techniques can locate and extract each symbol from the region of interest. The extracted symbols will pre-process to remove noise to execute the feature extraction step. Recognizing each symbol is performed by comparing the extracted features to the learned symbol classes from the learning stage. The last step is to reconstruct the words and numbers of the original text using the context information.

The continuous development of deep learning shows an increasing advantage in the field of computer vision. Currently, the most popular methods are the top-down approaches [22] based on CNN and RNN represented by LSTM. After using deep learning methods, the accuracy of text detection is highly improved.

3.1 Long Short-Term Memory (LSTM)

The traditional recurrent is a one-direction layer that can use the previous context. Sequence labeling tasks require two directions for processing sequence. So, a bidirectional convolutional recurrent layer that uses two hidden layers is adopted. These two hidden layers enable the bidirectional convolutional layer to iterate from forward and backward correspondingly [23]. LSTM networks combine different gates and memory cells to overcome the vanishing gradient problem. LSTM can learn essential information and discards unwanted information [12].

3.2 Connectionist Temporal Classification (CTC)

The output layer of deep learning architecture for text extraction is the CTC. It calculates the recognition loss and decodes the output of the previous layers as in Eq. (1) [24].

$$CTC_{loss} = -\log P(y|x), \tag{1}$$

where:

x: training sample, and

y: the produced sequence of labels.

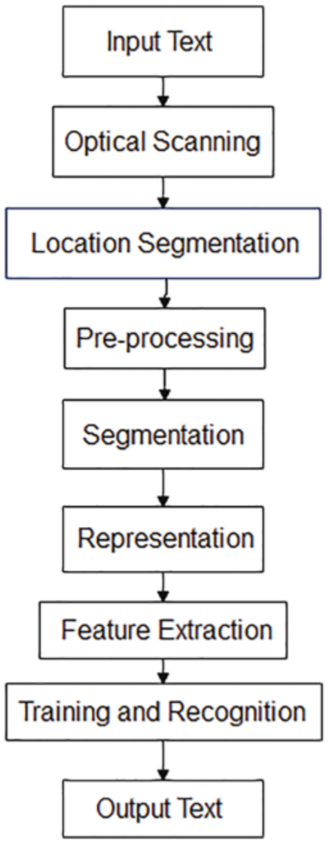


Figure 1: OCR stages [2]

4 The Proposed Methodology

The proposed architecture Fig. 2 comprises CNN block layers, two bidirectional Long Short-Term Memory (BiLSTM) layers, and Connectionist Temporal Classification (CTC) [25].

It starts with reading the images and appending them to the labels and dividing the dataset into training and validation sets. Next is generating sequences for mini-batches for both training and validation sets. The training set with size (128, 32) is passed to the first conv1 layer with kernel size (4, 4) to extract 32 feature maps. Relu or OLRelu from [26] is used to activate these features. The extracted text features are then passed to the second conv2 layer with kernel size [3, 3] to extract 32 feature maps and activate them with Relu or OLRelu. The extracted text features from the first and the second CNN layers are concatenated and pooled with the maxpooling1 layer. The output is then passed to conv3 and then conv4 layers. A second Maxpooling is applied to reduce the size of the features. The reduced output is convoluted with conv5 and conv6 and (3×3) kernel size to extract 64 text feature maps. The batch normalization layer follows conv5 and conv6.

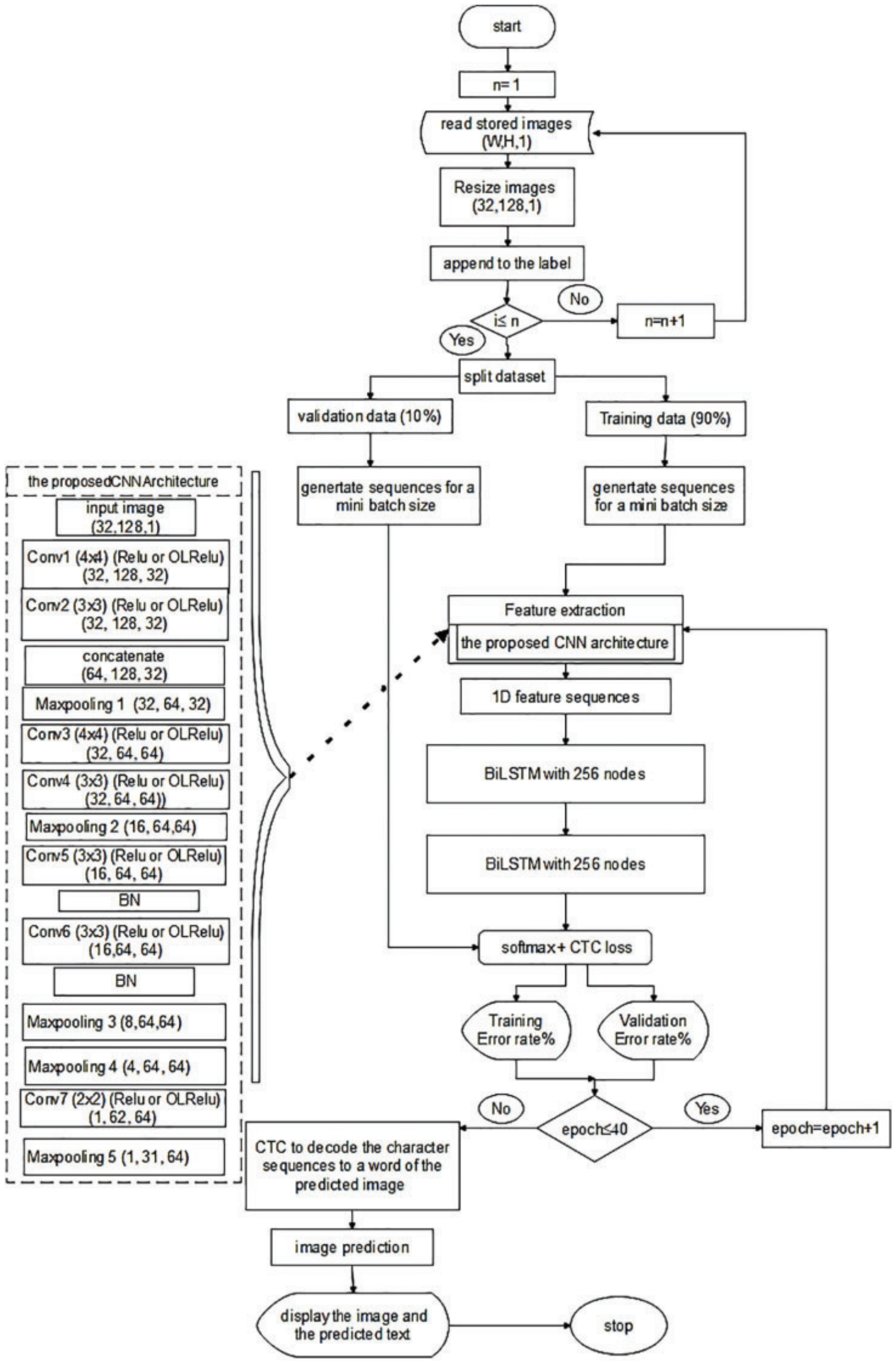

start
n= 1
read stored images (W,H,1)
Resize images (32,128,1)
append to the label
i≤ n
No
n=n+1
Yes
split dataset
validation data (10%)
Training data (90%)
genertate sequences for a mini batch size
genertate sequences for a mini batch size
the proposedCNN Architecture
input image (32,128,1)
Conv1 (4x4) (Relu or OLRelu) (32, 128, 32)
Conv2 (3x3) (Relu or OLRelu) (32, 128, 32)
concatenate (64, 128, 32)
Maxpooling 1 (32, 64, 32)
Conv3 (4x4) (Relu or OLRelu) (32, 64, 64)
Conv4 (3x3) (Relu or OLRelu) (32, 64, 64))
Maxpooling 2 (16, 64,64)
Conv5 (3x3) (Relu or OLRelu) (16, 64, 64)
BN
Conv6 (3x3) (Relu or OLRelu) (16,64, 64)
BN
Maxpooling 3 (8,64,64)
Maxpooling 4 (4, 64, 64)
Conv7 (2x2) (Relu or OLRelu) (1, 62, 64)
Maxpooling 5 (1, 31, 64)
Feature extraction
the proposed CNN architecture
1D feature sequences
BiLSTM with 256 nodes
BiLSTM with 256 nodes
softmax+ CTC loss
Training Error rate%
Validation Error rate%
No
epoch≤40
Yes
epoch=epoch+1
CTC to decode the character sequences to a word of the predicted image
image prediction
display the image and the predicted text
stop

Figure 2: Feature extraction flowchart

The output of the conv6 layer is reduced using dual Maxpooling layers (3 & 4). Next, conv7 with (2×2) kernel size is used to extract 64 text feature maps and then pass them to the last Maxpooling layer5 (2 × 1). The next step is applying two BiLSTM layers to label the sequences. The output from BiLSTM layers is passed to the CTC layer. CTC is used to manage the different alignments of the words. Moreover, it uses as a loss function when time is a variable quantity.

4.1 The Proposed Concatenate CNN Layers and Features Extraction

CNN proposed architecture for text extraction features consists of seven CNN layers, five Maxpooling layers, and two batches of normalizing layers. The CNN output is used for the feature extraction step by concatenating CNN layers defined in Eqs. (2) & (3).

$$C_{l1}\left(i,j\right)=\sum_{m=1}^{M}\sum_{n=1}^{N}I\ \left(i+m,j+n\right)\ k_r\ \left(m,n\right), \tag{2}$$

$$C_{l2}\left(i,j\right)=\sum_{m=1}^{M}\sum_{n=1}^{N}I\ \left(i+m,j+n\right)\ k_r\ \left(m,n\right), \tag{3}$$

where l_1 *and* l_2 are representing the extracted text features from conv1 and conv2 layers. (M, N) represents the image width and height, (i,j) represents the position of the current feature map. The output of these two layers is concatenated as in Eq. (4).

$$C\left(i,j\right)=C_{l1}\left(i,j\right)+C_{l2}\left(i,j\right), \tag{4}$$

The convoluted features $C\left(i,j\right)$ are activated with either Relu or OLRelu as in Eqs. (5) & (6) in sequence:

$$f_{C(i,j)}=Max\ \left(x_{ij},0\right) \tag{5}$$

$$f_{C(i,j)}=Max\ \left(x_{ij},0\right)+xe^{-\propto}\min\left(x_{ij},0\right) \tag{6}$$

4.2 The Proposed Dual Max-Pooling

Another proposed approach applies dual Maxpooling layers for dual dimensionality reduction of the text features extracted from conv6. The output of the Maxpooling3 (MaxP3) is the input for the Maxpooling4 (MaxP4). The proposed dual Maxpooling is illustrated in Eqs. (7) and (8) [27].

$$\text{Max}pooling3\ \left(i,j\right)=max_{n=0,1,\ m=0,1,}\ f_{(i+n),\ (j+m)}^{BN}, \tag{7}$$

$$\text{Max}pooling4\ \left(i,j\right)=max_{n=0,1,\ m=0,1,}\ f_{(i+n),\ (j+m)}^{MaxP3}, \tag{8}$$

i and j represent the current index and f^{BN} and f^{MaxP3} represent the output features of conv6 after batch normalization and the output features of MaxP3, respectively.

4.3 The Proposed BiLSTM

The proposed architecture of CNN layers used for extracting text feature maps is presented in Fig. 2. The sequence of the feature maps is then passed to two cyclic layers of Bidirectional Long Short Term Memory (BiLSTM). BiLSTM will extract more text feature information from the received feature maps [28]. BiLSTM can learn bidirectional long-term dependencies between time steps of time series of sequence data. A single BILSTM comprises dual LSTM layers called causal and anti-casual counterparts. These two layers process time series in the same way, forward and backward in time but in opposite time order as shown in Fig. 3 [29].

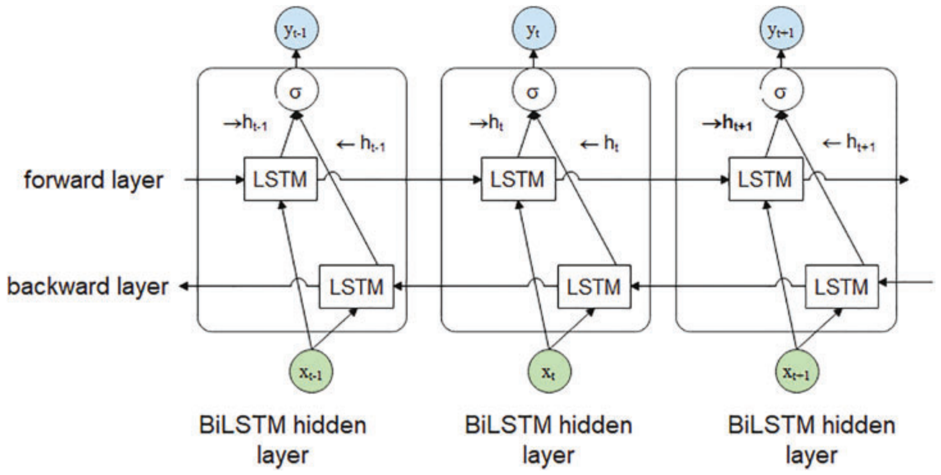


Figure 3: Bi-LSTM structure for three consequence steps. The previous $\rightarrow h_{t-1}$, *the next* $\leftarrow h_{t+1}$ and the current $\leftarrow h_t$. The arrows represent forward ($\rightarrow$) and backward ($\leftarrow$) time steps [12]

4.4 Text Connector

The CTC layer calculates the loss rate during the training and validation phases as Eq. (9). Also, it is used to output the prediction results in the testing phase. The dimension of the extracted text features is not appropriate to be fed directly to the CTC layer. The extracted text features from CNN layers have a 3D tensor (height, width, and number of feature maps). The input to the CTC layer should be a 2D tensor. So, an additional layer is required and called the transposition layer. Moreover, the CTC layer has another function decoding the output of the previous layers [28].

$$CTC_{loss} = -\sum\nolimits_{(y,x)\in s} \log P\left(y|x\right), \tag{9}$$

where:

s: training dataset of handwritten text images (Mjsynth or IAM),

$P\left(y|x\right)$: the probability of the ground truth y given to the training sample of handwritten text images x,

x: training sample of handwritten text images (Mjsynth or IAM), and

y: the produced sequence by the recurrent layers from x.

The CTC works in three main concepts, Encoding the text, Loss function, and Decoding text. When dividing the characters into 31-time steps as shown in Fig. 4, some characters take more than one time step, as shown in Fig. 4. The character "J" takes seven-time steps, "o" takes four-time steps, "y" takes five-time steps, " f " takes four-time steps, "u" takes four-time steps, and "l" takes two-time steps. So, the output from the network using CTC is "JJJJJJJooooyyyyyffffuuuull". CTC encodes it by adding a blank character denoted with "-" and then the word is encoded as "JJJJJJJ-oooo-yyyyy-ffff-uuuu-ll". Then the encoded text using CTC is trained using BiLSTM layers.

To train the BiLSTM, each image, and its label is given a calculated loss. The results will be a matrix of the probability (p) for each character at every time step, including the blank. The total probability equals 1, as shown in Fig. 5.

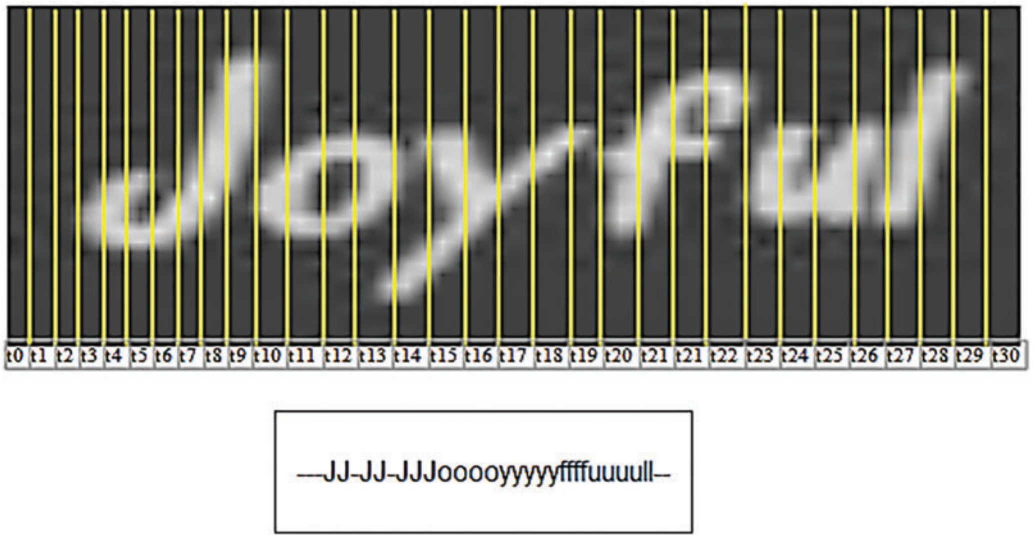

t0 t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t13 t14 t15 t16 t17 t18 t19 t20 t21 t21 t22 t23 t24 t25 t26 t27 t28 t29 t30
---JJ-JJ-JJJooooyyyyyffffuuuull--

Figure 4: Dividing the word "joyful" image sample from Mjsynth Dataset into 31-time steps

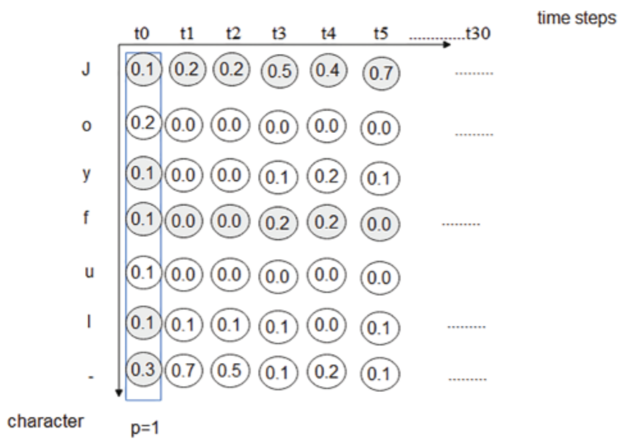

time steps
t0 t1 t2 t3 t4 t5 ....t30
J 0.1 0.2 0.2 0.5 0.4 0.7
o 0.2 0.0 0.0 0.0 0.0 0.0
y 0.1 0.0 0.0 0.1 0.2 0.1
f 0.1 0.0 0.0 0.2 0.2 0.0
u 0.1 0.0 0.0 0.0 0.0 0.0
l 0.1 0.1 0.1 0.1 0.0 0.1
- 0.3 0.7 0.5 0.1 0.2 0.1
character
p=1

Figure 5: Matrix of the probability for each character at every time-step

The corresponding character probabilities are multiplied together to get the probability for a single path. For example the path "j–" the probability is $(0.1 \times 0.3 \times 0.7) = 0.021$, and for the path "jjjj" the probability is $(0.1 \times 0.2 \times 0.2 \times 0.5 = 0.002)$. To get the corresponding probability for the given ground truth, the probabilities of all possible paths are summed up $(0.021 + 0.002 = 0.023)$. The loss can be calculated by applying the negative logarithm of the probability. The loss can be backpropagated and trained in the network. The output of the trained BiLSTM is the unseen text images. Then choose the best path by considering the character with maximum probability at every time step. For example, to t0 the max probability is for "-" and the same for t1, and t2. So, the output text is "-" For t3, t4, and t5 the max probability is for character "j" which means the output text is "j". The CTC merges the duplicated characters and removes the blanks to get the final decoded text. According to the example in Fig. 4 the output is "Joyful".

5 The Experiments and Results

This section presents and discusses all the experimental results, analyze the results, and conducts comparisons.

5.1 Experiment 1: Extracting Feature with Individual Max-Pooling

This experiment tests the model's performance with a set of CNN, Relu (OLRelu), and Max-pooling blocks. The total number of parameters is 759,103, and the number of trained parameters is 759,103. The results of ten runs are presented in Table 1 and Fig. 6. The results validation averages for Mjsynth data with OLRelu and Relu are 4.5 and 3.0664, with standard deviations of 0.23 and 0.288 in sequence. While with the IAM dataset, the Validation loss averages are 3.167 and 2.5, with standard deviations of 0.181 and 0.569 for Relu and OLRelu in sequence. Also, the performance of OLRelu is better than Relu due to the cropped characters resulting from segmenting the image with a line of words into separated images with single words. This indicator has increased the ambiguity of the characters.

Table 1: The train and validation loss rates for individual Maxpooling with Relu and OLRelu for ten runs

Data	Mjsynth				IAM dataset			
	Relu		OLRelu		Relu		OLRelu	
	Train loss	Val loss	Train loss	Val loss	Train loss	Val loss	Train loss	Val loss
Min	2.467	4.2153	3.2674	4.5589	3.501	2.9208	2.2442	2.2568
Max	3.7936	4.9406	5.1372	5.4325	4.2523	3.4787	3.0082	2.639
Avg	3.0664	4.5002	3.9762	5.0096	3.8923	3.1678	2.5885	2.4148
Std	0.4179	0.2395	0.5223	0.2884	0.2291	0.1814	0.2493	0.1310

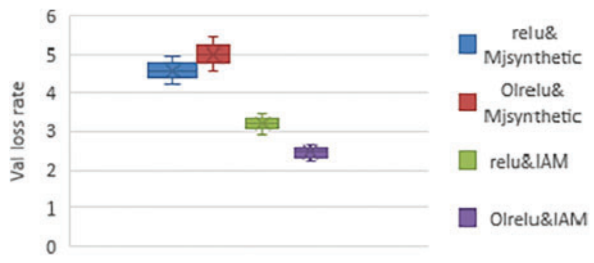


Figure 6: The average loss rates for individual Maxpooling with Relu and OLRelu for words extraction from Mjsynthatic and IAM images

The performance of Relu showed less error rate than with OLRelu using the Mjsynth dataset. Nevertheless, the validation loss rate with OLRleu is less than with Relu using the IAM dataset. According to Fig. 7, the loss rates using Relu and OLRelu with the IAM dataset are less than with the Mjsynth dataset. Due to the structure of the Mjsynth data, preprocessing techniques are used to create it. Such as adding noise shadows and using different image resolutions. It increases the negative feature maps. Relu works by eliminating all nodes with negative values. On the other hand, OLRelu considers both positive and negative feature maps, increasing the probability of the character's misclassification.

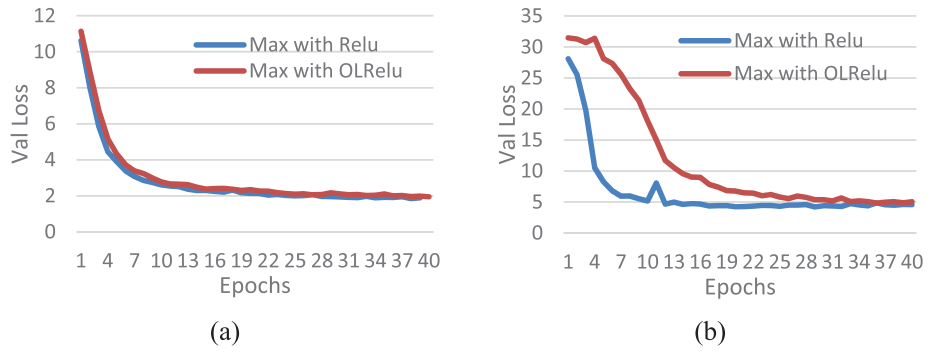
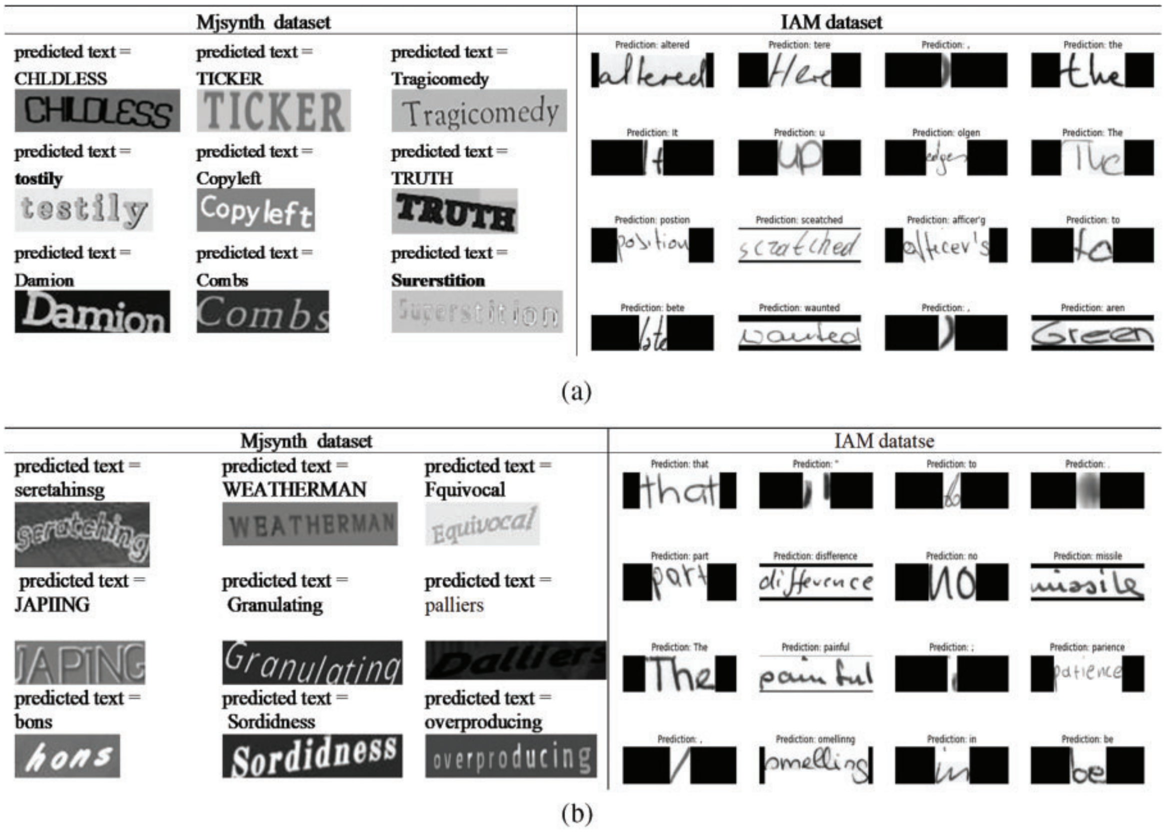


Figure 7: The performance of Relu and OLRelu with (a) IAM dataset and (b) Mjsynth dataset with individual Maxpooling and no CNN concatenate

For more illustrations, this study chose the best run from both datasets and presented the results per epoch as in Fig. 8. With the IAM dataset, there was a slight difference in the loss rates between Relu and OLRelu. On the contrary, the performance of Relu is better than OLRelu with Mjsynth data. It shows fewer loss rates between epochs 1–32. Then equal to OLRelu for the rest of the epochs.



Mjsynth dataset
IAM dataset
predicted text = CHLDLESS
predicted text = TICKER
predicted text = Tragicomedy
predicted text = tostily
predicted text = Copyleft
predicted text = TRUTH
predicted text = Damion
predicted text = Combs
predicted text = Surerstition
(a)
Mjsynth dataset
IAM datatse
predicted text = seretahinsg
predicted text = WEATHERMAN
predicted text = Fquivocal
predicted text = JAPIING
predicted text = Granulating
predicted text = palliers
predicted text = bons
predicted text = Sordidness
predicted text = overproducing
(b)

Figure 8: Samples from the predicted words using the proposed model with individual Maxpooling with (a) Relu. (b) OLRelu

The model's performance with Relu using MJsyenthtic data showed slower divergence than with Relu. The loss rate reached ten at epoch 5 with Relu and OLRelu at epoch 14. Nevertheless, at epoch 33, the loss rates of both Relu and OLRelu are matched.

Fig. 8a shows samples from the predicted words using Rule and individual Maxpooling for a run. From the predicted words, we can conclude the following: with the Mjsynth dataset, the word images are either horizontally, with an angle, or curved aligned. When created, many different text effects were used, such as adding a shadow, dark or light brightness, thin or thick font, and different font styles. We noticed that the model could correctly predict most word characters in datasets like "Copyleft" and "Tragicomedy". While it missed one or two characters in some words, such as the predicted word "tostily" and the text in the image is "testily" also the "Surerstition" and in the image, it should be "Superstition" for the Mjsynth dataset. Similar to the IAM dataset, some words and characters are correctly predicted like "the", "to", and "altered" but some wrongly predicted characters such as "posstion" instead of "position", "aren" instead of "Grean", "olgen" instead of "edges" and more.

Fig. 8b shows samples from the predicted images for both datasets using the OLRelu activation function. With the Mjsyenthtic dataset, some characters are mispredicted, such as "JAPIING" instead of "JAPING", and "bons" instead of "hons". At the same time, some characters are correctly predicted, such as "Granulating", and "Fquivocal". IAM data also found some correctly extracted characters, and some are wrong as shown in Fig. 8b for IAM data. The correct extracted words are "part", "that", "No", "missile", and painful. Some extraction exhales wrong characters such as "omelling" instead of "smelling", disfference" instead of "difference", and "parience" instead of "patience".

5.2 *Experiment 2: Extracting Feature with Dual Max-Pooling*

This experiment is conducted to study the effect of applying dual Maxpooling layers on eliminating the low-valued features with Relu and OLRelu activation functions. The total number of parameters is 361,696 and the trainable parameters are 361,696.

The results of the validation loss rate for both activation functions of ten runs are presented in Table 2 and Fig. 9. The resulting validation loss rate for the Mjsynth dataset with Relu is worse than with individual Maxpooling layers due to the construction of the dataset. The IAM dataset obtains better results than individual Maxpooling layers. On the other hand, the OLRelu with dual Maxpooling performed better than with individual pooling layers for both datasets. The validation loss rate decreased for both datasets with OLRelu.

This can be explained as follow: the dual reduction for the dimensionality of the IAM samples led to the extraction of the best text feature maps with high values. These features increased the rate of distinguishing character classes and decreased the loss rates with Relu and OLRelu. While with Mjsynth data, dual reduction for the dimensionality caused losing important feature maps, which led to an increase in the loss rates with both Relu and OLRelu.

For further explanation, Fig. 10 presents the performance of both activation functions with the two datasets for the best run. The performance of Relu with the Mjsynth dataset and dual Maxpooling showed a high validation loss rate than with individual max-pooling. However, the dual Maxpooling with Relu and OLRelu enhanced gradient divergence and speed error rates. Also, with the IAM dataset, the loss rate gradient diverged smoothly with Relu and OLRelu.

Table 2: The train and validation loss rates with Relu and OLRelu for 10 runs with dual Maxpooling

	Mjsynth				IAM			
	Relu		OLRelu		Relu		OLRelu	
	Train loss	Val loss	Train loss	Val loss	Train loss	Val loss	Train loss	Val loss
Min	4.897	5.443	2.618	4.078	1.497	1.891	1.897	2.019
Max	5.222	5.979	3.903	4.465	1.863	2.501	2.066	2.127
Avg	5.056	5.656	3.359	4.317	1.741	2.271	1.984	2.091
Std	0.125	0.179	0.398	0.133	0.102	0.270	0.051	0.032

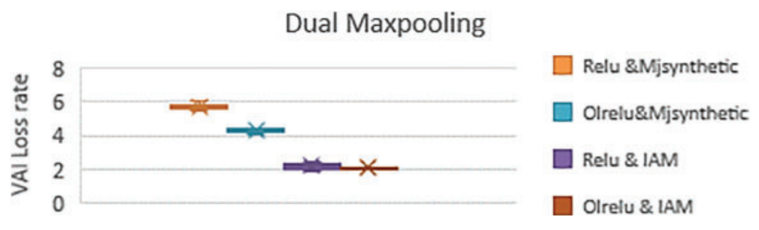
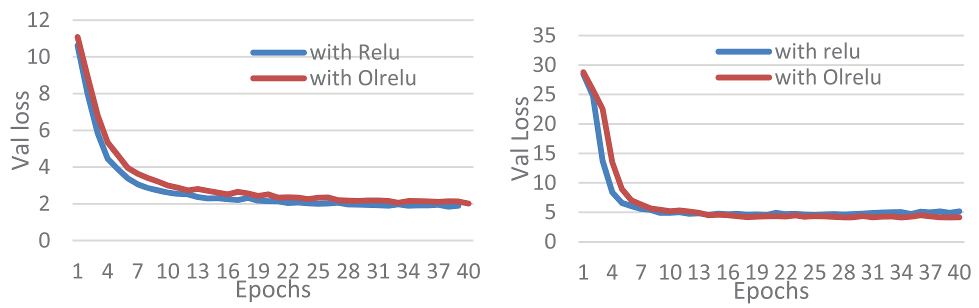


Figure 9: The validation loss rates with Relu and OLRelu using dual Maxpooling with Mjsynth and IAM datasets



(a) (b)

Figure 10: Relu and OLRelu performance with dual Maxpooling for 40 epochs for (a) Mjsynth data (b) IAM data

Fig. 11a presents a set from the predicted words for both datasets using Relu. As noticed from the images, some words are horizontal, and curved, with clockwise and anticlockwise angles, some images are dark, and some are light with shadows. The model performance is enhanced by overcoming the overfitting problem that combined Relu and individual Maxpooling with the Mjsynth dataset. Also, we still have missed characters from words such as “Obressionaly” instead of “Obsessionally”. On the other hand, some characters are extracted correctly even from unclear or curved images such as “BEHARPEN” and “commodiously”.

Regarding the IAM dataset with Relu, the model performance is enhanced. The loss rates decreased noticeably. Some of the missed extracted characters with individual Maxpooling and Relu are correctly extracted using dual Maxpooling, such as the words “wanted”, “Here”, “up”, and "officer's". Yet, we still have missed extracted characters such as the words "bte" instead of "late", "aljers" instead of "edges", and "sceukcked" instead of "sczatched".

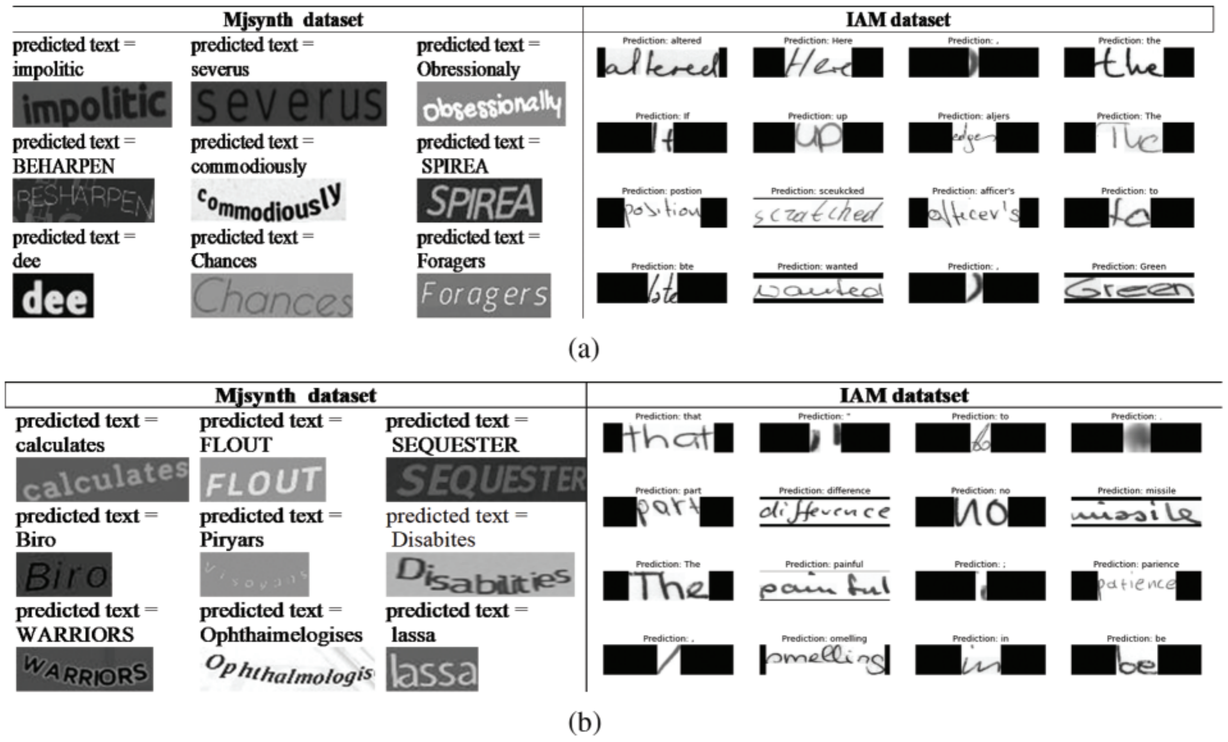


Figure 11: Samples from the predicted words using the proposed model with dual Maxpooling with (a) Relu. (b) OLRelu

Fig. 11b shows some correctly and incorrectly extracted characters with OLRelu activation functions with Mjsynth and IAM datasets. For the Mjsynth dataset, it shows better performance with most characters, but it shows the disability to extract characters that have a similar form like "i" and "l" in the word "Disabites" and they stack to each other. It considers them as three similar characters and replaces them with one character. Also, it shows more correctly extracted words such as "Biro", "SEQUESTER", "KHUFU", and " GREAIPES". With the IAM dataset and OLRelu, the words extracted with incorrect characters with Relu are correctly extracted with OLRelu such as "painful", and "differences". But we still have miss extracted characters such as the words "omelling" must be "smelling", and "parience" instead of "patience".

5.3 *Experiment 3: Extracting Text Feature with Concatenating CNN Layers*

This experiment was conducted to evaluate the performance of the proposed model with Relu, OLRelu and concatenate the first two CNN layers. The total number of parameters is 422,839 and the trainable parameters are 422,839. The experiment aims to improve the value of the extracted text feature maps to enhance the model performance. The loss results of ten runs are presented in Table 3 and Fig. 12.

The results showed a better performance for the Relu with Mjsyenth data than with the IAM dataset. But in contrast with OLRelu, it showed better performance with the IAM dataset than with Mjsynth data. We believe the reason is related to the function of Relu with the negative text feature maps. Also, it is due to how the dataset set is collected and processed. The IAM dataset words are originally segmented from a line-based text. So, some words were cropped from more than one side. Concatenating CNN layers with Mjsenthetic data enhanced the extracted high values text features maps. On the other hand, the IAM data concatenating led to an increase in the unimportant features with negative values, leading to needing more epochs to improve the model performance.

Table 3: The model performance with Concatenate CNN layers

	Mjsynth				IAM			
	Relu		OLRelu		Relu		OLRelu	
	Train loss	Val loss	Train loss	Val loss	Train loss	Val loss	Train loss	Val loss
Min	3.453	1.902	2.547	3.221	4.068	4.055	2.301	1.672
Max	3.773	2.962	3.765	3.489	4.321	4.219	2.922	2.841
Avg	3.590	2.222	3.025	3.369	4.201	4.150	2.605	2.165
STD	0.102	0.305	0.450	0.078	0.085	0.047	0.243	0.430

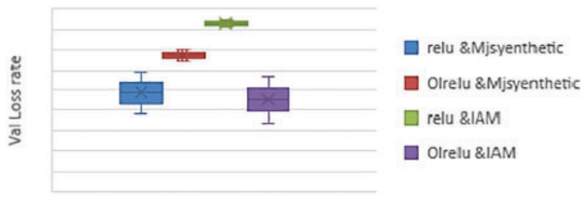


Figure 12: Min, max, and average loss rates of concatenate CNN layers

The model's performance with both Relu and OLRelu per epoch with the Mjsynth dataset is shown in Fig. 13 below. As clear from Fig. 13, the OLRelu was slower than Relu in the early epochs, whereas, with Relu, the validation loss rate reached 5 at epoch five, while with OLRelu, the validation loss reached five at epoch 9. The model performance with Relu and OLRelu are matched in the later epochs.

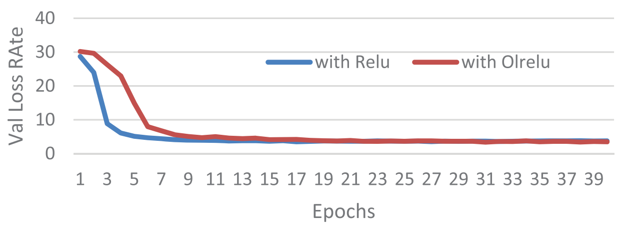


Figure 13: The performance of the model with concatenating CNN and dual Maxpooling layers per epoch with the Mjsynth dataset

The results for the model with Relu and OLRelu for the IAM dataset are presented in Fig. 14. The performance of the model with OLRelu showed better results than those with Relu. We need to run over 40 epochs to gain fewer validation loss rates.

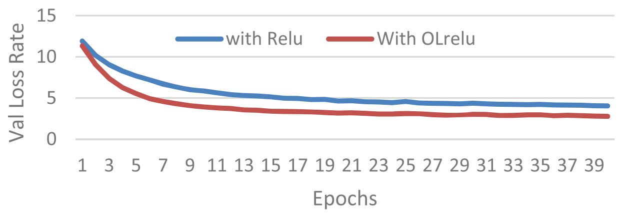


Figure 14: The performance of the model with concatenating CNN and dual Maxpooling layers per epoch with the IAM dataset

6 Comparison with State of the Art

Table 4 contains a list of the recent studies' results regarding text extraction from images problem. Our proposed method showed better results than the state-of-the-art studies from the presented results.

Table 4: Comparison with state-of-the-art

Study	Techniques	Dataset	Error per character (EPC)	STD
[24]	CNN_BLSTM + CTC	IAM	7.9%	0.12
		RIMES	4.4%	0.18
[30]	CNN_BLSTM + CTC	IAM	17.4%	-
		RIMES	24.3%	-
[31]	CNN + attention-CTC	ADOCR (synthetic)	3.57%	-
[32]	CNN-LSTM-CTC	IAM	13.77%	-
	CNN-LSTM-CTC + geometric augmentation		9.81%	
Our proposed model	CNN + BLSTM + CTC	**Mjsynth with Relu** concatenate CNN	2.2221	0.3050
		Mjsynth with OLRelu concatenate CNN	3.36867	0.0780
		IAM with Relu dual max-pooling	2.2707	0.2562
		IAM with OLRelu dual max-pooling	2.09109	0.0319

Table 5: Comparision between individual and dual max in terms of the number of epochs

Mjsynth data		
Method	No. of Epochs	Avg. val loss
Individual max-pooling with Relu	300	6.48
Individual max-pooling with Relu	200	4.5

(Continued)

Table 5 (continued)

Mjsynth data		
Method	No. of Epochs	Avg. val loss
Dual max-pooling with Relu	40	5.65
CNN Concatenate with Relu	40	2.22
Dual max-pooling with OLRelu	40	4.31
CNN Concatenate with OLRelu	40	3.37

7 Conclusion

The study was proposed to examine using dual Maxpooling for dimensionality reduction. In addition to using concatenate CNN layers to enhance the feature maps extraction from images. The highly valued extracted features are passed to two layers of BLSTM with 50 units to extract more features and find the time sequence between the word characters. The CTC loss function is performed to calculate the training and validation loss rates by updating the training parameters. The experiments were conducted with individual max-pooling, dual max-pooling, and concatenated CNN layers. Two datasets were used; the first is the Mjsynth dataset, and the second is the IAM dataset. The results were compared to the state of art studies. The proposed approach achieved better results than the state-of-the-art. Samples reduction, concatenation, and merging of the extracted features from CNN layers led to activating more related features from the images. The number of parameters of the proposed model with dual MaxPooling and concatenating CNN layers (361,696 and 422,839 in sequence) is less than that of using individual MaxPooling (759,103).

The difference between individual and Dual Maxpooling in terms of the number of epochs required to reach the reported error rate for the Mjsynth dataset is shown in Table 5. The reported validation error rate for individual Maxpooling for all experiments was only for the last 40 epochs of 300 and 200 epochs. Since the results kept repeating, we consider only the last 40 epochs for comparison. As shown dual Mapooling with Relu takes 40 epochs to reach an error rate that is equivalent to the error rate with individual Maxpooling and Relu with 100 and 150 epochs. This shows that Dual Maxpooling speeds up the training and reduces the number of epochs. The results are related to the Mjsynth dataset because it was the first data used to test the proposed method and then apply the same settings for the IAM dataset.

The study presents samples from the predicted words combined with the original image of these words. In general, the proposed methods showed better performance in terms of validation loss rate with OLRelu and IAM datasets than with the Mjsynth dataset. The Mjsynth dataset was created using a special application and contains shadows and noise. In addition, the created words were rotated at different angles. These data augmentation techniques led to reducing the value of the image features and increasing misclassification rates.

The resulting loss (Error Per character) of the state of art studies [24,30,32] with IAM dataset using CNN. BLSTM and CTC were 7.9%. 17.4% and 13.77%, respectively. While the proposed CNN concatenation and dual max-pooling showed less loss rate (2.091%, 2.165% in turn) than the mentioned state of art studies. The resulting loss rate of the study [31] with the Syenth dataset was 3.57%. while with the proposed CNN concatenating the results showed better results (2.22%). The proposed Concatenating CNN led to improving the quality of the activated features by eliminating the noise data represented by the ambiguous samples that contain shadow or mixed text with different resolutions. But since the number of extracted features increased the time for training the model per epoch also increased. Also, to speed up training the model a high-specification GPU processor is requested. With handwritten character recognition problems, handwritten datasets contain different handwritten styles and sizes. This increases the difficulty of recognizing characters. for future work, more image enhancement techniques will be applied to improve the quality of the images for better recognition.

Acknowledgement: We would like to convey our gratitude to the research team members at the Digital Forensic Lab and Medical and Health Informatics Lab at the Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, who contributed to this project. Also our thanks to Ms. Manal Mohamed for her support.

Funding Statement: The Ministry of Higher Education, Malaysia, which supported this project under the Fundamental Research Grant Scheme (FRGS) FRGS/1/2019/ICT02/UKM/02/9 entitled "Convolution Neural Network Enhancement Based on Adaptive Convexity and Regularization Functions for Fake Video Analytics". This grant was received by Prof. Assis. Dr. S. N. H. Sheikh Abdullah, https://www.ukm.my/spifper/research_news/instrumentfunds.

Author Contributions: The authors confirm their contribution to the paper as follows: study conception and design: Dr. B. H. Hani, Prof. Assis. Dr. S. N. H. Sheikh Abdullah; data collection: Dr. B. H. Hani; analysis and interpretation of results: Dr. A. Saeed, Dr. R. Sulaiman; draft manuscript preparation: Dr. B. H. Hani, Prof. Assis. Dr. S. N. H. Sheikh Abdullah. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The readers can obtain all datasets by sending a reasonable request to the corresponding author (bahera_hani@yahoo.com).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] J. Céline Mancas-Thillou, "Natural scene text understanding," in *Segmentation and Pattern Recognition,* InTech, Vienna, Austria, pp. 123–142, 2007. https://doi.org/10.5772/4966

[2] A. Chaudhuri, K. Mandaviya, P. Badelia and S. K. Ghosh, "Optical character recognition systems," in *Optical Character Recognition Systems for Different Languages with Soft Computing*, vol. 352. United States Patent: Springer, pp. 9–41, 2017.

[3] H. S. Ackley, "Methods for optical character recognition (OCR)," Google Patents No. 10,621,470, 2020.

[4] N. Islam, Z. Islam and N. Noor, "A survey on optical character recognition system," Arxiv Preprint, arXiv:1710.05703, 2017.

[5] M. T. Qadri and M. Asif, "Automatic number plate recognition system for vehicle identification using optical character recognition," in *2009 Int. Conf. on Education Technology and Computer*, Singapore, IEEE, pp. 335–338, 2009.

[6] S. Sen, S. Chowdhury, M. Mitra, F. Schwenker, R. Sarkar *et al.,* "A novel segmentation technique for online handwritten Bangla words," *Pattern Recognition Letters*, vol. 139, pp. 26–33, 2020. https://doi.org/10.1016/j.patrec.2018.02.008

[7] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015. https://doi.org/10.1016/j.neunet.2014.09.003

[8] G. E. Hinton, S. Osindero and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[9] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed *et al.,* "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012. https://doi.org/10.1109/MSP.2012.2205597

[10] P. Goyal, S. Pandey and K. Jain, *Deep Learning for Natural Language Processing*. Berkeley, CA: Apress, Springer, pp. 138–143, 2018.

[11] Z. Chen, "Deep-learning approaches to object recognition from 3D data," M.S. Dissertation. Case Western Reserve University, Cleveland, Ohio, USA, 2017.

[12] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[13] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus *et al.,* "Overfeat: Integrated recognition, localization and detection using convolutional networks," Arxiv Preprint arXiv:1312.6229, 2013.

[14] R. Salakhutdinov and G. Hinton, "An efficient learning procedure for deep boltzmann machines," in *Artificial Intelligence and Statistics*, Hilton Clearwater Beach Resort, Clearwater Beach, Florida, USA, pp. 448–455, 2009.

[15] A. A. Chandio, M. Asikuzzaman, M. R. Pickering and M. Leghari, "Cursive text recognition in natural scene images using deep convolutional recurrent neural network," *IEEE Access*, vol. 10, pp. 10062–10078, 2022. https://doi.org/10.1109/ACCESS.2022.3144844

[16] A. H. Hasan, K. Omar and M. F. Nasrudin, "Multi-classifier Jawi handwritten sub-word recognition," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 8, no. 4-2, pp. 1528–1533, 2018.

[17] W. He, X. Y. Zhang, F. Yin and C. L. Liu, "Multi-oriented and multi-lingual scene text detection with direct regression," *IEEE Transactions on Image Processing*, vol. 27, no. 11, pp. 5406–5419, 2018.

[18] F. Liu, C. Chen, D. Gu and J. Zheng, "FTPN: Scene text detection with feature pyramid based text proposal network," *IEEE Access*, vol. 7, pp. 44219–44228, 2019.

[19] J. Memon, M. Sami and R. A. Khan, "Handwritten optical character recognition (OCR): A comprehensive systematic literature review (SLR)," Arxiv Preprint arXiv:2001.00139, 2020.

[20] Z. Liu, W. Zhou and H. Li, "Scene text detection with fully convolutional neural networks," *Multimedia Tools and Applications*, vol. 78, no. 13, pp. 18205–18227, 2019.

[21] S. D. Khan, L. Alarabi and S. Basalamah, "A unified deep learning framework of multi-scale detectors for Geo-spatial object detection in high-resolution satellite images," *Arabian Journal for Science and Engineering*, vol. 47, no. 8, pp. 9489–9504, 2022. https://doi.org/10.1007/s13369-021-06288-x

[22] M. Elsaraiti and A. Merabet, "Application of long-short-term-memory recurrent neural networks to forecast wind speed," *Applied Sciences*, vol. 11, no. 5, pp. 2387, 2021.

[23] Z. Cui, R. Ke, Z. Pu and Y. Wang, "Stacked bidirectional and unidirectional LSTM recurrent neural network for forecasting network-wide traffic state with missing values," *Transportation Research Part C: Emerging Technologies*, vol. 118, pp. 102674, 2020.

[24] X. Huang, L. Qiao, W. Yu, J. Li and Y. Ma, "End-to-end sequence labeling via convolutional recurrent neural network with a connectionist temporal classification layer," *International Journal of Computational Intelligence Systems*, vol. 13, no. 1, pp. 341–351, 2020.

[25] B. Shi, X. Bai and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 11, pp. 2298–2304, 2016.

[26] B. H. Nayef, S. N. H. S. Abdullah, R. Sulaiman and Z. A. A. Alyasseri, "Optimized leaky ReLU for handwritten arabic character recognition using convolution neural networks," *Multimedia Tools and Applications*, vol. 81, no. 2, pp. 2065–2094, 2022.

[27] L. G. Hafemann, R. Sabourin and L. Oliveira, "Learning features for offline handwritten signature verification using deep convolutional neural networks," Arxiv Preprint arXiv:1705.05787, 2017.

[28] H. Zhan, S. Lyu and Y. Lu, "Handwritten digit string recognition using convolutional neural network," in *2018 24th Int. Conf. on Pattern Recognition (ICPR)*, Beijing, China, IEEE, pp. 3729–3734, 2018. https://doi.org/10.1109/ICPR.2018.8546100

[29] H. ElMoaqet, M. Eid, M. Glos, M. Ryalat and T. Penzel, "Deep recurrent neural networks for automatic detection of sleep apnea from single channel respiration signals," *Sensors*, vol. 20, no. 18, pp. 5037, 2020.

[30] C. Wick, J. Zöllner and T. Grüning, "Rescoring sequence-to-sequence models for text line recognition with CTC-prefixes," Arxiv Preprint arXiv:2110.05909, 2021.

[31] B. H. Belay, T. Habtegebrial, M. Liwicki, G. Belay and D. Stricker, "A blended attention-CTC network architecture for amharic text-image recognition," in *ICPRAM*, Vienna, Austria, SciTePress, pp. 435–441, 2021. https://doi.org/10.5220/0010284204350441

[32] A. Sulaiman, K. Omar and M. F. Nasrudin, "Two streams deep neural network for handwriting word recognition," *Multimedia Tools and Applications*, vol. 80, no. 4, pp. 5473–5494, 2021.