Extraction of Handwritten Text using Word Beam Search Algorithm and Language Modeling

Kavitha Ananth[1], Kirubanand V B[2]

[1] Research Scholar, Department of Computer Science, CHRIST (Deemed to be University), Bengaluru, India

[2] Faculty of Department of Computer Science CHRIST (Deemed to be University), Bengaluru, India

Abstract: The challenge of recognizing handwriting in mortgage records is covered in this article. Businesses trying to digitize huge numbers of hand-marked scanned documents or reports have a significant challenge: offline handwritten text recognition from images. In order to translate a picture into a series of characters that match to the text that is contained in the image, this research suggests an innovative language model in combination with a deep convolutional network and a recurrent encoder-decoder network. Using the principles of Deep Learning and Word Beam Search, the complete model is trained as an end-to-end replacement for conventional handwriting recognition techniques. When the Connectionist Temporal Classification (CTC) loss function is trained on the digital form, an RNN is the result. Character probabilities are contained in this matrix for each discrete time step. By translating the character probabilities, a CTC decoding algorithm maps the final text. The token passing mechanism is used to create the recognized text from a list of dictionary words. We offer a novel and highly efficient method for developing restrictive models for classification which might associate entity names in accordance with the data contained in the article on entity types. A benchmark dataset predicated on the Mortgage domain is included. This Mortgage domain is evaluated in the presented model. We tested the model provided below using a set of benchmark mortgage datasets, which are published. The experimental outcomes were compared to the IAM and RIMES datasets, two openly accessible datasets. On the evaluation set of both datasets, word level precision at the cutting edge by 2.5% & 1.3%, respectively.

Keywords: CTC (Connectionist Temporal Classification), RNN (Recurrent Neural Networks), HTR(Handwritten Text Recognition), AI(Artificial Intelligence), HMM (Hidden Markov Model), RNN (Recurrent Neural Network), MDRNN (Multi-Dimensional Recurrent Neural Network), CNN (Convolutional Neural Network), NMT (Neural Machine Translation), LSTM (Long Short Term Memory), LM (Language Model), FSM (Finite State Machine)

1. INTRODUCTION

This paper is an extension and combined work about the original paper presented Turkish Journal of Computer and Mathematics Education, Handwritten Text Recognition using Deep Learning and Word Beam Search and the paper Building an Industry Standard Novel Language Model Using Named Entities. The

The process of converting handwritten characters or sentences into a language that a computer can understand is called handwriting text recognition (HTR). Since technological advancements in this area have made it possible to automate certain types of handwriting, a network of educational scholars has been actively investigating it. This paper explains how the AI tools are being employed in cloud computing environments. This artificial intelligence - driven businesses using cloud computing may grow efficient, planned, and knowledge-driven. AI in cloud computing can speed up enterprise digital transformation while also enhancing cloud performance and efficiency. To increase the performance, strategic, and knowledge-driven character of corporate operations while also supplying greater versatility, responsiveness, and financial savings, AI capabilities in the cloud computing environment are essential.

Given the potential benefit that may be obtained from removing the data from handwritten documents and combining it with modern AI technologies, Handwritten Text Recognition (HTR) has long been an important research topic [7] [28]. HTR is frequently divided into two types: offline recognition and online recognition. In this paper, we focus on the offline recognition problem, which is substantially more challenging than the online mode. This is unlike the online

mode, which additionally uses the text picture and properties such as stroke information and trajectory, the offline mode only has the image available for feature extraction.

HTR has traditionally been thought of as a pattern comparison issue where a sequence of features from the input data are matched to an output sequence made up of letters from the text. This is done mostly using Hidden Markov Models (HMM). The context information in a text sequence cannot be used by HMMs due to the Markovian assumption that every single observation depends only on the current state. Recurrent Neural Networks (RNN), which encode the context information in the hidden states, were used to overcome this restriction. However, because the RNN goal functions necessitate a distinct training signal at each time step, its utility was restricted to instances where each of the characters in a sequence could be segmented.

Innovations had been carried out in the manner of models with hybrid architectures that combined RNN and HMM [6] [4], but a significant advance was made when [12] suggested using Connectionist Temporal Classification (CTC) [11] in conjunction with RNN. CTC eliminates the requirement for segmented input by allowing the system to associate the input information sequence directly to a collection of output labels.. As the RNN-CTC model built its input sequence for the RNN using manually created features from the image, its performance was still constrained. The end-to-end model for HTR, Multi-Dimensional RNN (MDRNN) [10] was suggested. In order to learn long-term dependencies in both directions, it employs an ordered set of multi-dimensional RNN layers that process the input text picture along both axes. The objective is to capture both the spatial organization of the characters across the vertical axis and the sequence information throughout the horizontal axis. As shown in [24], which suggested a composite architecture that combines a deep one-dimensional RNN-CTC model and a convolutional neural network (CNN), and which maintains the state-of-the-art performance on common HTR benchmarks. Compared to conventional convolution processes that retrieve identical visual characteristics, this formulation requires more computing power.

In this article, we provide a different strategy that combines two recurrent networks for sequence matching on top of a convolutional network that acts as a feature extractor. The RNN-based Encoder-Decoder network [8] [27], which is frequently used for Neural Machine Translation (NMT) is used. This essentially accomplishes the task of creating the desired sequence from a source sequence. The architecture, training, and inference processes of our model have been improved with the addition of features like Batch & Layer Normalisation, Focal Loss, and Beam Search, to mention a few. The random distortions are included in the training to regularise the inputs. We particularly contribute significantly in the following areas:

a. A neural network design that can extract offline HTR from images and consists of convolutional and recurrent networks is shown.

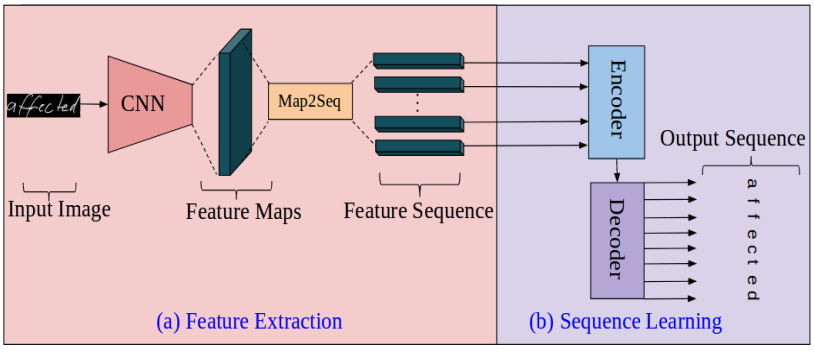

Figure 1a : Illustrates the model overview and how the convolutional feature maps are used to create the feature sequence.

An Encoder-Decoder network with Multi Head Attention which can provide significant boost in accuracy as compared to the standard RNN-CTC formulation for HTR is demonstrated.

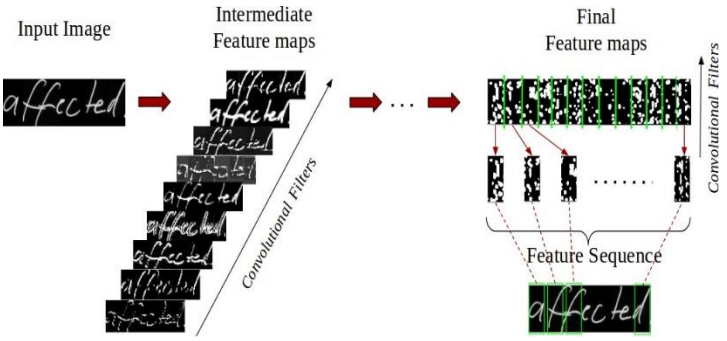

Figure 1b: Explains how to convert a string of output characters into a visual feature sequence.

We demonstrate that by sampling reduction of the images that are input to nearly one quarter of their

original size, it is possible to reduce calculations by 62% and the use of memory by 16% without affecting the model's overall accuracy.

2. PROPOSED METHOD

Our proposed model comprises of the following: (a) Image Acquisition Module which receives scanned image along with hand written text image and (b) Pre Processing Module performs normalization, noise removal and deslanting process and (c) Feature extraction and Sequence Learning module where each of the scaled sub-image is transformed into one or more feature maps by applying the CNN wherein the one or more feature contains visual features of a corresponding sub-image and (d) Post Processing module which maps the extracted value with a novel language module. Figure 1 presents the model's overall layout. Its distinct cognitive modules have an open interface, enabling quick and effective end-to-end training..

2.1 Image Acquisition Module

This system consists of an imaging system. The Imaging system is set up to capture a few text images such as information out of the document..

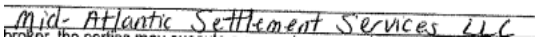

Figure 2: Sample Image of Handwritten Text

The Figure 2 explains the sample data. Document could be any form of document that contains various types of information, along with pictures, printed text, or handwritten (off-line) text. The capturing device may be set up to transmit the few captured images to the system

2.2 Pre Processing Module

By applying machine learning as a collection of non-linear transformations needed for a specific task. Convolutional Networks have shown to be highly good at extracting rich visual features from images. Our goal was to create a series of features that would preserve the spatial arrangement of the image's items while encoding local information in the image. To accomplish this, we convert the input image into a deep layer of feature maps using a regular CNN (excluding the fully-connected layers). By depth-wise detaching columns from the CNN, a layer called Sequence-to-Sequence [26] is applied on top to transform the feature maps into a series of feature vectors. This pre-processing module also scales the samples by maintaining the screen resolution. Almost every scaled sub-image has a predetermined height. The addition of extra pixels keeps the dimension constant throughout the sub-image. In other words, all of the feature maps' i-th columns are concatenated to create the i-th feature vector. Because convolution processes are translationally invariant, each column in Figure 2 depicts a vertical strip in the image (known as the Receptive field), moving from left to right.

Before being sent to the network, each image is scaled to a fixed height, with the aspect ratio of the image being preserved. This overcomes any length limits on the series and guarantees that each vector throughout the list of features conforms to the identical dimensionality.

2.3 Feature Extraction and Sequence Learning

The Feature extraction module is a CNN accustomed to transforming every one of a few scaled sub-images into a few characteristic maps, where the few distinctive maps contain visual characters among the appropriate sub-image. The CNN derives rich visual information from each single scaled sub-image and in the pattern of less characteristic maps by acquiring a set of rules instantly significant non-linear transformations. Local properties in the suitable sub-image can be encoded using the retrieved rich visual qualities, while the spatial relationships between objects in the corresponding sub-image are kept. The proposed structure consists of three primary components in a unified platform: the encoder, decoder, and multi-head attention mechanism. The input handwritten (off-line) image is inverted into a sub-image, making more use of pre-processing image module.

The Seq2Seq module is equipped to change the situation few additional characteristic maps using a listing of the proposal's realization invention of distinct vectors, which is made possible by separating depth-wise columns from a few typical maps and concatenating relevant columns from few characteristic maps of the related sub-image. Concatenating the columns of a few specific maps, for example, produces a collection of distinct vectors. A context vector c that serves as a representation of the complete sequence can be created, for instance, by concatenating the ith columns of a few typical maps. This is done by employing an RNN with the hidden state ht = g(x_t, h_{t1}) at each time step t and the final state c = s (h_1,..., h_{Ts}), where g and s are some non-linear functions.

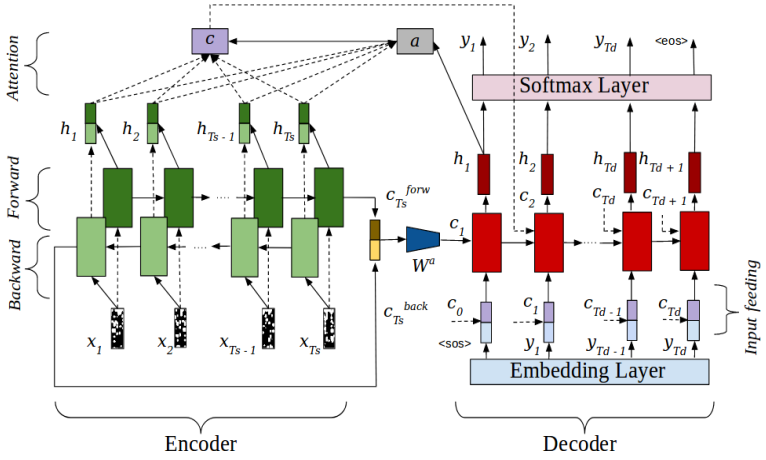

Figure 3: *Encoder - Decoder* System using Multi Head Attention layer with a Softmax Layer on top.

Such a formulation using a simple RNN cell turns out to be worthless for learning even reasonably long sequences due to the vanishing gradient phenomenon [14][5] caused by repeated multiplications of gradients in an unfolded RNN. Instead, because of their superior capacity to model and learn long-term dependencies, we use the Long Short Term Memory (LSTM) [13] cells considering.the presence of a memory cell c ∈ R^n.

2.4 Post Processing Module

Post-processing module consists of an encoder-decoder system which is built using Modern RNN based encoder-decoder system unit is set up to generate a few sets of characters by mapping the characteristics related with each series of characteristic vectors. The encoder system begins with a deep CNN, which is beneficial at obtaining the attributes. Thus it creates a succession of distinct visual vectors from segmented images. It connects three interlaced sampling regions with fully connected layers that reduce the image's spatial dimensions while boosting the representative depth of the characteristic vectors simultaneously.

A deep RNN examines a succession of convolutional properties and encodes temporal context in a few images. The BLM layers merge two LSTM layers that execute the series differently to encode both the forward and backward intrinsic relationship of written text. Three BLSTM layers make up the encoder RNN. The encoder-CNN converts an input picture I into an intermediate-level characteristic map X. This is a unique one where a group of column vectors X = (X1,..., XM), in which M represents the duration of the input stream that has also been subsampled. The BLM layers then analyze the order and generate the total of upstream and downstream suppressed states as the combined feature pattern. The final encoded characteristic map H = (H1,.., HM), which models the images information by feeding

into it. The temporal context of the group is created by superimposing a 2D fully connected layer on top of the third BLSTM layer's state sequence. The sub-image generated at step is analyzed with the series of characteristic vectors. The encoder module generates a stable output of the context vector. The decoder unit triggers a collection of encoded symbols or machine-readable text for the matching sub-image by applying to a response given the values similar to accurate identification. The decoder-RNN consists of a unidirectional LSTM layer system in one instance to generate the intended data block. Everything and every consequent is represented by p (yt | y1,..., yt-1, ct) = softmax(f (yt-1, St-2, ct)), where f is the LSTM and st-1 is the preceding concealed state. The decoder, in particular, must collect facts regarding the output sequence's history and save it in its internal state of remembrance. Instead of the encoder BLSTM's end state, the decoder LSTM is initialized with a zero state. The decoder is adjusted at each decoding time interval and uses an attention mechanism to concentrate mostly on a significant portion of the encoded characteristic representation.

Attention mechanisms in one embodiment dynamically adjust depending on the vector representation for each stage decoder using the key, value, and query parameters to the encoder. In most cases, a linear combination of the succession of encoded order or sequence is used. ct = $\sum_{j=1}^{M}$ αt,jhj yields the context vector ct at decoding time step t. The attention weights t,j are normalized attention scores affected by the activity of the multi-head attention function that is being employed.αt,j = Att (st, hj , αt-1).

In one form, residual connections are used to configure the RNN-based encoder-decoder units to allow gradient passage to a following recurrent layer via recurrent units. A dropout mechanism is used with a depth-wise link to capture long-term dependency. Hence the recurrent connection in the decoder unit is not modified. An MHAM is accustomed to aligning the decoder unit's secret states with the encoder unit's hidden conditions.

In one form, normalizing is implemented in the layers for recurrent activations in the RNN based encoder-decoder units to moderate hidden state dynamics and enhance convergence during system 100 training. A linear transformation to generate logits, W256×N, is applied to the decoder unit, whereas N is considered as the output vocabulary size. Logits is used at each time step of the Softmax operation. This helps to establish a probabilistic model across the generated vocabulary.

The Back Propagation Through Time (BPTT) is accustomed to performing gradient descent across the RNN-based encoder-decoder units. The network weights and biases are then updated using the Back Propagation Through Time into the CNN.

In one example, the Beam Search Decoding Algorithm is employed to determine by maximizing a joint distribution, and we can find the best sequence across a beam of hypotheses. In an example, the image post-processing module is utilized to create the output file by assembling the group of characters to generate both digital and handwritten text recognized for each sub-image through making use of the Modern RNN based encoder-decoder unit along with Multi-Head attention mechanisms.

Based on the words already observed in the sequence, a novel Language Model can estimate the likelihood of the next word in the line. This language model was proposed from textual data from the mortgage industry. The (k + 1) grams are collected from flowing text to train a k-order language model, and the (k + 1) word is handled as the supervision signal. As a result, a vast amount of training data is produced from many data sources. Rather than only predicting the token with the highest score, a record of k hypotheses is kept (for example, k=5 is the beam size). Then we will have V potential tokens representing the hypotheses at each new time step. This results in a total of 5V new predicted tickets. Then we keep the top five, and so on. Define H_t as the collection of hypotheses decoded at time step t in formal terms.

$$H_t:=\{(w1_1,\ldots,w1_t),\ldots,(wk_1,\ldots,wk_t)\}$$

For example, if k=2, one possible H2 is H2: = (identified word1) and (identified word2)

When all hypotheses have reached the eos> token, the theory with the highest score is returned. All conceivable H_t is mapped to the respective Language Model (LM) to generate the proper term.

The solution has been proposed in the unstructured document data extraction consisting of digital and handwritten complete systems for Industry built using Modern neural networks and a novel Language Model. Per the present

situation, this disclosure is explained utilizing the input handwritten text recognition for single or multiple unstructured scanned documents consisting of digital and handwritten text. The proposed innovation and methods, on the contrary, have the ability to recognize handwritten text from numerous scanned unstructured documents composed of digital and handwritten data by mapping the Softmax output towards a novel Language Model which identifies the appropriate handwritten word.

3. IMPLEMENTATION DETAILS

3.1 Image Preprocessing

This framework receives images containing a single phrase of text written by hand, which could or could not represent a full sentence. A single channel with 256 levels of intensity makes up each image. We flip the images before training, which makes it a little bit simpler to perform the CNN triggers to learn because the primary focus is made up of higher intensity on a dark background. The input images have been compressed by a typical height of 128 pixels to 32 pixels, as indicated in Table 1, while keeping the aspect ratio of the original image. The images are buffered with pixels from the background equivalent to the width of the batch's widest image on the left and right side. The consistency in the batch's size is retained when utilizing small batch training. Early tests showed that our model tended to over predict from the initial set of data. As an additional regularization step to avoid this, we added random distortions [24] to the training photos, such that the framework will, ideally, transform a group of inputs that had not been seen before in each iteration.

Table 1: Image down sampling's impact on a model's training performance

Size	Computations	Memory
	Tflops	(GB
128 ×W	1.5×10^4	9.5
32 ×W	5.9×10^3	7.9

Translation, rotation, shear, and scaling are the four procedures that each training batch is put through. Each operation's parameters are independently sampled from a Gaussian distribution. At the start of the experiments, the procedures and the underlying distribution were selected by viewing a few cases and were then fixed.

Table 2: Impact of Image Downsizing on Training Model Performance

Size	Computations (Tflops)	Memory (GB)
128 × W	1.5×10^4	9.5
32 × W	5.9×10^3	7.9

3.2 Convolutional Layer

Seven convolutional layers are stacked serially in the implemented model, and Leaky ReLU [20] activations are used. The top layer has a kernel size of 2x2 pixels without input padding. The bottom six we used a kernel size of 3x3 pixels along with input padding which is 1 pixel wide. Kernel strides in both the vertical and horizontal axes are 1 pixel. To avoid internal covariate shift and accelerate training, triggers of the convolutional layers are batch normalised [15] before propagating to the subsequent layer. To make the input less dimensional, pooling techniques are used to some convolutional layers' activations. Four max-pooling layers make up our model, including two which have a 2x1 kernel size to preserve a horizontal spatial distribution of text and the other two utilising traditional 2x2 non-overlapping kernels.

Table 3 Depicts the details of the convolution layer used for the network configuration

Configuration	Layers 1 – 2 – 3 – 4 – 5 – 6 – 7
Conv. filters	16 32 64 64 128 128 128
Maxpool(2x2)	✓ - ✓ - X- X- X- X- X
Maxpool(2x1)	X- X- X- X- ✓ - ✓ - X

3.3 RNN Encoder-Decoder

Utilizing the LSTM cells consisting of 256 hidden units, and the encoder and decoder. To increase both networks' ability for learning, we let them both extend to a depth of two layers. To enable gradient transfer over the recurrent units to the layers below, residual connections [16] are made.

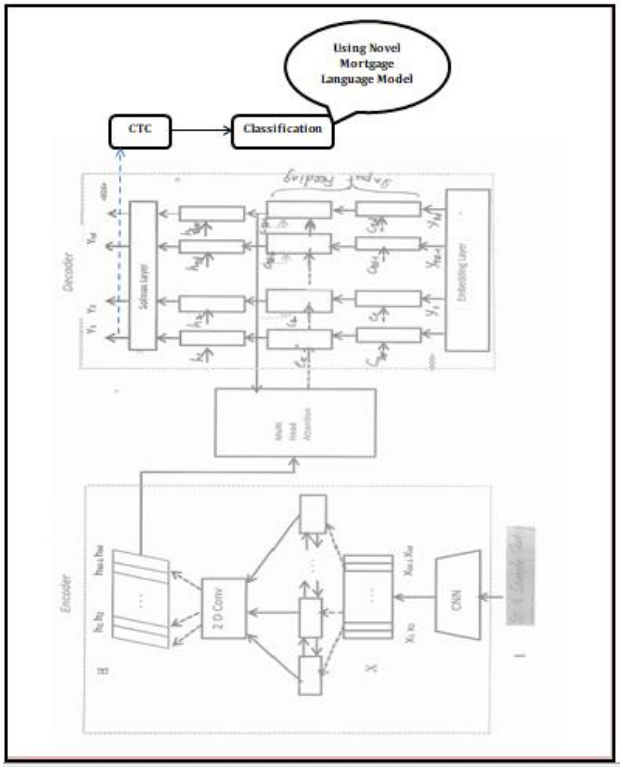

Figure 4: End to End Implementation Model

Additionally, we regularise the network using dropout [23] along depth links without changing the recurrent connections, maintaining the network's ability to record long-term dependencies in the process. The activity of individual neurons are Layer Normalised [2] to avoid covariate shift owing to mini batch training, which has proven to be extremely efficient in stabilizing the network's hidden state dynamics. We use a linear transformation for the final forecast $W \in R^{256xN}$ on the RNN predictions, where N is used to create the logits, or output vocabulary size. To establish a distribution of probability over the outcome vocabulary at each timestep, the softmax operation is applied to the logits.

4. THE TRAINING & INFERENCE

In our tests, the batch size is set at 12 during training. With a learning rate of 0.001, we optimize using the Adam [17] algorithm. To get the highest validation accuracy, the model underwent training for 25 intervals. We employ a beam size for the inference that is proportional to the total amount of categories in the output.

5. DATASET

We test our strategy using the following publicly accessible dataset. The IAM Handwriting Database v3.0 (English) [22] has 1538 pages of text. These are broken up into writer-independent training, validation, and test sets, respectively. These sets each contain 6161, 940, and 1861 segmented lines. The repository contains handwriting samples from 657 distinct authors. The average height and width of the line images are 124 pixels and 1751 pixels, respectively. The database has 79 distinct characters total, not counting whitespace.

6. EXPERIMENTS

The mean Character Error Rate (CER) and mean Word Error Rate (WER), which become performance metrics determined as the mean over all text lines, serve as indicators to examine our model on the evaluation partition of both datasets. They are defined as,

$$CER = \frac{\text{No of char s(c) in a sentence wrongly classified}}{\text{Total chars in a sentence}}$$

$$WER = \frac{\text{No. of words(w) in a sentence spelled incorrectly}}{\text{Total words (w) in a sentence}}$$

We used AWS Cloud computing for the tests we conducted. The model's time for inference is 2.5 seconds

7. RESULTS

Table 3 displays the effect of Beam Search, Focal Loss, and Layer Normalisation (LN) on the basic model. The effectiveness of the base model was enhanced by LN by roughly three percent. The most significant benefit was achieved by transitioning from greedy decoding to beam search, that enhanced the model's precision by 4-5%. Focal loss was also utilised, which increased accuracy by 1-3%.

Table 4: Impact of The model Efficiency on Focal Loss, Beam Search, and Layer Normalisation

System	IAM		RIMES	
	CER(%)	WER(%)	CER(%)	WER(%)
Baseline	17.3	25.4	11.5	18.1
+ LN	12.1	22.8	9.6	15.7
+ LN + Focal Loss	11.3	20.1	7.2	13.4
+ LN + Focal Loss + Beam Search	8	16.6	3.4	9.5

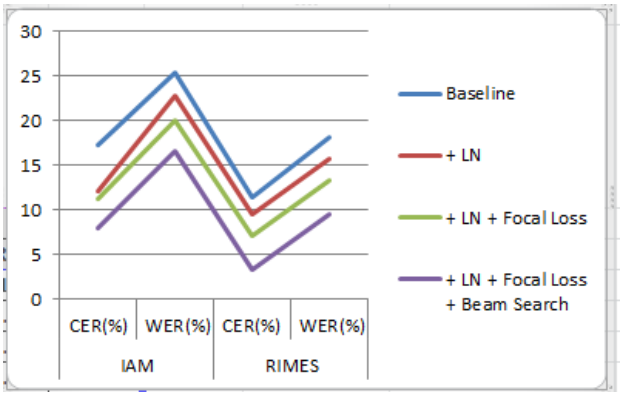

Figure 5: Graphical representation of Model Performance on Layer Normalisation, Focal Loss, and Beam Search Effect

8. COMPARISON TO THE MOST RECENT TECHNOLOGY

We compare our method's effectiveness to the cutting-edge in Tables 4 and 5 according to the maximum GPU memory usage and the total amount of trainable parameters. We contrast the precision of our approach with that of previously published methods.

Table 5: Comparison of accuracy with earlier techniques

Methods	IAM		RIMES	
	CER(%)	WER(%)	CER(%)	WER(%)
2DLSTM [10], reported by [24]	8.2	27.4	3	17.6
CNN-1DLSTM-CTC [24]	6.1	20.1	2.5	10.6
Our method	8	16.6	3.4	9.5

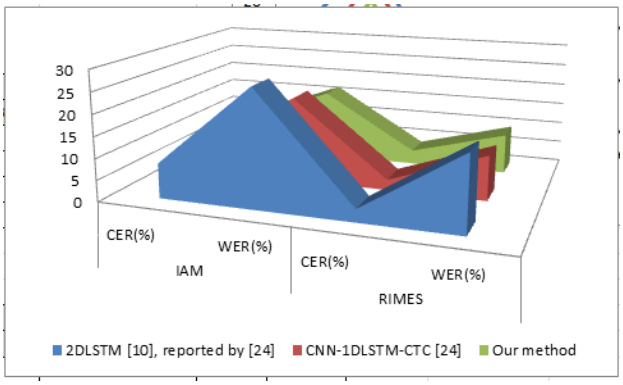

Figure 6: Graphical representation of comparison accuracy

Table 6: Efficiency Comparison in relation to state-of-art

Methods	Memory (GB)	# of Parameters (Mi)
CNN-1DRNN-CTC [24]	10.6	9.2
Our method	**7.8**	**4.5**

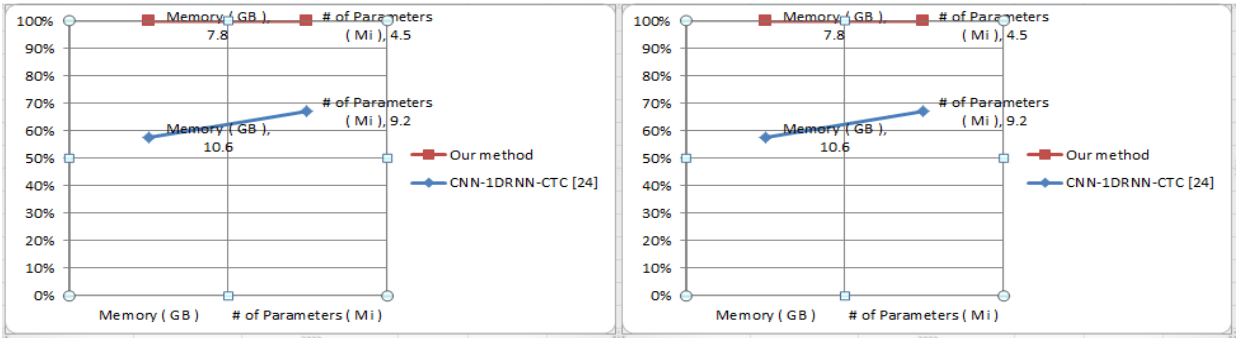

Figure 7: Representation of efficiency in relation to state-of-art

Our character level accuracy is just slightly better than the state-of-the-art [24], despite the fact that we exceeded it at the word level. It suggests that while our model makes less errors overall when looking at words as a whole, it likelihood to make further errors in terms that already contain mislabeled letters. This results from our model's inference behavior, which draws on earlier predictions to produce the present output. The outcome is, a past error may set off a chain of subsequent errors. However, better word accuracy demonstrates that our model typically correctly guesses each word in a line. In essence, this prototype recognizes words reasonably accurately, however, if an error occurs, the word level prediction is off by an increased amount of letters.

9. CONCLUSION

We suggest a novel language model for accurately recognizing handwritten text that combining the entity model and boosting the word beam search. Using a open source, our model performs much better than all the prior techniques, and on a private dataset, it outperforms them by a respectable margin. Despite the model's satisfactory performance on benchmark sampling data, we plan to do additional analyses to see how it performs in entirely unrestricted scenarios with a increased categories in Language Model.

A training technique that optimizes loss of character according to the correctness of the entire sequence instead of the total loss for specific characters would be developed with the goal to improve the current method. The model

would act similarly during training and inference as a result. So as to improve the effectiveness of the representation and correct errors, particularly for unusual phrases or sequences, the specific domain language model can be added to the training plan.

Data Availability

The data will be available from the author upon request (kp.kavitha@res.christuniversity.in)

Conflicts of Interest

No conflicts of interest are disclosed by the authors.

ACKNOWLEDGEMENT

Under the supervision of Dr. Kirubanand V.B. Assistant Professor, as a part of Computer Science Department, Christ (Deemed to be University).

10. REFERENCES

[1] S. Wiseman, S. M. Shieber, and A. M. Rush, “Challenges in Data-to-Document Generation,” 2017, Accessed: Feb. 19, 2022. [Online]. Available: https://github.com/harvar.
[2] K. C. Arnold, K.-W. Chang, and A. T. Kalai, “Counterfactual Language Model Adaptation for Suggesting Phrases.” [Online]. Available: https://www.yelp.com/dataset_.
[3] “An Improved Error Model for Noisy Channel Spelling Correction.”
[4] R. Collobert, J. Weston, J. Com, M. Karlen, K. Kavukcuoglu, and P. Kuksa, “Natural Language Processing (Almost) from Scratch,” 2011.
[5] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, “Neural Architectures for Named Entity Recognition.” [Online]. Available: https://github.com/.
[6] X. Ma and E. Hovy, “End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF.”
[7] J. Devlin, M.-W. Chang, K. Lee, K. T. Google, and A. I. Language, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” [Online]. Available: https://github.com/tensorflow/tensor2tensor.
[8] M. E. Peters, M. Neumann, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, “Deep contextualized word representations.” [Online]. Available: http://allennlp.org/elmo.
[9] V. Athavale, S. Bharadwaj, M. Pamecha, A. Prabhu, and M. Shrivastava, “Towards Deep Learning in Hindi NER: An approach to tackle the Labelled Data Scarcity.” [Online]. Available: https://github.com/3Top/word2vec-api.
[10] W. Gunawan, D. Suhartono, F. Purnomo, and A. Ongko, “Named-Entity Recognition for Indonesian Language using Bidirectional LSTM-CNNs,” *Procedia Computer Science*, vol. 135, pp. 425–432, Jan. 2018, doi: 10.1016/J.PROCS.2018.08.193.
[11] N. Abinaya, N. John, B. H. B. Ganesh, A. M. Kumar, and K. P. Soman, “AMRITA_CEN@FIRE-2014: Named Entity Recognition for Indian Languages Using Rich Features,” in *Proceedings of the Forum for Information Retrieval Evaluation*, 2014, pp. 103–111, doi: 10.1145/2824864.2824882.
[12] F. Morin and Y. Bengio, “Hierarchical Probabilistic Neural Network Language Model,” in *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, 2005, vol. R5, pp. 246–252, [Online]. Available: https://proceedings.mlr.press/r5/morin05a.html.
[13] Y. Ji, C. Tan, S. Martschat, Y. Choi, N. A. Smith, and P. G. Allen, “Dynamic Entity Representations in Neural Language Models,” Association for Computational Linguistics.
[14] O. Vinyals, G. Brain, M. Fortunato, and N. Jaitly, “Pointer Networks.”
[15] J. Gu, Z. Lu, H. Li, and V. O. K. Li, “Incorporating Copying Mechanism in Sequence-to-Sequence Learning.”
[16] S. Wiseman, S. M. Shieber, and A. M. Rush, “Challenges in Data-to-Document Generation,” Association for Computational Linguistics. [Online]. Available: https://github.com/.
[17] P. Yin and G. Neubig, “A Syntactic Neural Model for General-Purpose Code Generation,” pp. 440–450, doi: 10.18653/v1/P17-1041.
[18] S. Merity, N. Shirish Keskar, and R. Socher, “Regularizing and Optimizing LSTM Language Models,” 2017.
[19] B. Ray, D. Posnett, V. Filkov, and P. T. Devanbu, “A large scale study of programming languages and code quality in github,” *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2014.
[20] J. Pennington, R. Socher, and C. D. Manning, “GloVe: Global Vectors for Word Representation,” 2014.
[21] M. Rabinovich, M. Stern, and D. Klein, “Abstract Syntax Networks for Code Generation and Semantic Parsing,” pp. 1139–1149, doi: 10.18653/v1/P17-1105.

DOI: https://doi.org/10.15379/ijmst.v10i2.2970