Improving Word Recognition using Multiple Hypotheses and Deep Embeddings

Siddhant Bansal
CVIT, IIIT, Hyderabad, India
siddhant.bansal@students.iiit.ac.in

Praveen Krishnan
CVIT, IIIT, Hyderabad, India
praveen.krishnan@research.iiit.ac.in

C.V. Jawahar
CVIT, IIIT, Hyderabad, India
jawahar@iiit.ac.in

***Abstract*—We propose a novel scheme for improving the word recognition accuracy using word image embeddings. We use a trained text recognizer, which can predict multiple text hypothesis for a given word image. Our fusion scheme improves the recognition process by utilizing the word image and text embeddings obtained from a trained word image embedding network. We propose EmbedNet, which is trained using a triplet loss for learning a suitable embedding space where the embedding of the word image lies closer to the embedding of the corresponding text transcription. The updated embedding space thus helps in choosing the correct prediction with higher confidence. To further improve the accuracy, we propose a plug-and-play module called Confidence based Accuracy Booster (CAB). The CAB module takes in the confidence scores obtained from the text recognizer and Euclidean distances between the embeddings to generate an updated distance vector. The updated distance vector has lower distance values for the correct words and higher distance values for the incorrect words. We rigorously evaluate our proposed method systematically on a collection of books in the Hindi language. Our method achieves an absolute improvement of around 10% in terms of word recognition accuracy.**

***Index Terms*—Word recognition, word image embedding, EmbedNet**

I. Introduction

The task of word recognition involves converting the text in an image to a machine-readable format. Word recognition is an important use case of computer vision that finds various applications in digitizing old books, making self-driving cars understand signboard instructions, and creating assistive applications for people with special needs. All these tasks rely on accurate word recognition that is robust to extreme variations in lighting conditions, fonts, sizes and overall typography. To ensure the availability of the word recognizer to a broader audience, it should also be able to function for various languages and have low computational costs.

In this work, we focus on improving word recognition for the Hindi language, which is agglutinative and inflectional. Hindi contains 11 vowels and 33 consonants, and a horizontal line runs across the words, which is referred to as *Shirorekha*. If a consonant is followed by a vowel, the shape of the consonant is modified. Such characters are referred to as vowel modifiers. A compound character is formed when a consonant follows one or more consonants. Due to these modifiers and compound characters, the number of distinct shapes in Hindi is far more than that of the Latin scripts [1]. This makes word recognition for Hindi difficult, and hence, it is necessary to devise more intricate techniques which improve the word recognition accuracy for the Hindi language.

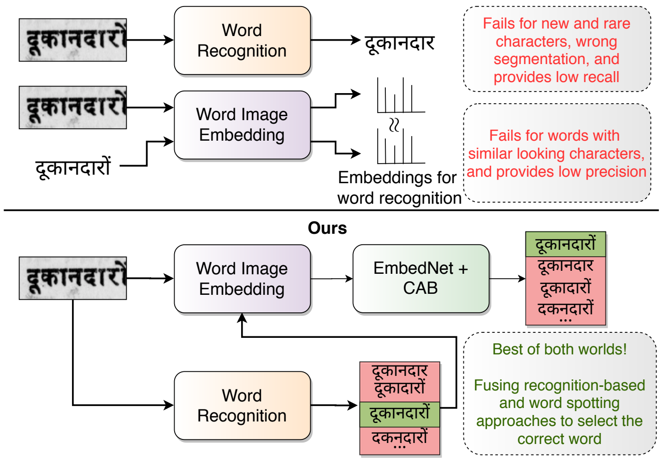

Word Recognition
दूकानदार
Fails for new and rare characters, wrong segmentation, and provides low recall
Word Image Embedding
दूकानदारों
≈
Embeddings for word recognition
Fails for words with similar looking characters, and provides low precision
Ours
Word Image Embedding
EmbedNet + CAB
दूकानदारों
दूकानदार
दूकादारों
दकनदारों
...
Word Recognition
दूकानदार
दूकादारों
दूकानदारों
दकनदारों
...
Best of both worlds!
Fusing recognition-based and word spotting approaches to select the correct word

Fig. 1. Word recognition based methods fail to perform when they encounter new and rare characters, wrong word image segmentation, and provide low recall. However, these methods excel in differentiating between visually similar characters, whereas it is vice-versa for methods using word image embeddings. We aim to use the methods proposed in this work - EmbedNet and CAB, for exploring the complementary properties of these methods. Diagram best viewed in color.

Traditionally, word recognition methods fall under two major categories: (a) methods directly converting a word image to its textual transcription [2]–[6], and (b) methods converting word images to embeddings and then performing recognition using these embeddings [7]–[10]. In this work, we will refer to methods in category (a) as word recognition and methods in category (b) as word image embeddings, respectively. We assume that the word images are already segmented. A word recognition based method aims at directly converting the word image to its corresponding textual transcription. Despite the wide availability of high-grade open-source OCR engines [11], using them with degraded images from historical documents is difficult. On the other hand, word image embedding methods focus on converting the word image and the corresponding text to a holistic representation where word image and its corresponding text lie closer to one another. After the projection of these images and texts to a learned representation space, they can be compared using an appropriate distance metric and perform recognition restricted to a lexicon [7].

Word recognition methods (OCR) perform reasonably well when the text present in the image is reasonably clean. However, if the OCR encounters a rare character or an image with higher degradation, it struggles to generate the correct prediction. In such cases, word image embedding methods prove to be much useful. The reason is that they do not identify each character but instead, focus on converting the word image to an embedding/representation where words with (visually) similar characters, lie closer in the embedding space, resulting in better predictions in challenging situations. However, word image embedding methods find it difficult to distinguish between two different words with an approximately similar set of characters, a task at which word recognition based approaches excel. Also, word recognition methods provide high recall, whereas, word image embeddings methods, provide high precision [12]. Inaccurate word segmentation degrades performance in word recognition based methods [13]. Inaccurate segmentation, however, does not hinder the performance of word image embedding as a slightly degraded word (due to cut) still lies closer to its textual transcription's embedding. Fig. 1, we show how we propose to use the complementary properties of both methods for improving word recognition.

Designing a pipeline that can exploit the complementary properties provided by word recognition and word image embedding methods can further enhance word recognition. In our previous work [14], we propose to use the complementary information of both the methods to create a more reliable and robust word recognition algorithm. We propose to fuse multiple hypotheses generated by the word recognizer with the embeddings generated from the End2End network ('*E2E*') [15] for making use of the complementary information. Using the beam search decoding algorithm, we produce multiple (K) predictions for a word image from a CTC [16] based word recognizer, where K is the number of predictions generated for a word image. We show that as the value of K increases, the word recognition accuracy increases. Even though we proposed multiple rule-based methods for using this information and improving word recognition accuracy, we do not explore the learning-based techniques in [14].

In this work, we improve upon the methods presented in [14] and propose EmbedNet and a novel plug-and-play module called Confidence based Accuracy Booster (CAB) for improving word recognition. Fig. 3 presents the flowchart of the entire process which includes EmbedNet, CAB, and their roles in the word recognition pipeline. Here, EmbedNet attempts to learn an updated Euclidean space where the embeddings of the word image and its correct textual transcription lie closer together, while the incorrect ones lie farther away. The CAB boosts the word recognition accuracy by using the updated representation made available by EmbedNet. For accelerating future research, we release the code and models used in this work on our webpage[1].

[1]http://cvit.iiit.ac.in/research/projects/cvit-projects/word-recognition

II. Related Works

In this work, we are interested in devising deep learning methods for fusing the existing methods in the word recognition, and word image embedding realms also referred to as text recognition and word spotting, respectively. This section explores the previous work done in these domains.

A. Text Recognition

A typical text recognizer involves a feature extractor for the input image containing text, and a sequential encoder for learning the temporal information. Modern text recognizers use Convolutional Neural Networks (CNN) as a feature extractor and a Recurrent Neural Network (RNN) as a sequential encoder, which helps in modeling the text recognition problem as a Seq2Seq problem. Architectures using both CNN and RNN for this purpose are called Convolutional Recurrent Neural Network (CRNN) [2]. Previous works have used a wide range of recurrent networks for encoding the temporal information. Adak et al. [6] perform sequential classification using a RNN, whereas, [5] use Bi-directional Long-Short Term Memory (BLSTM) network for sequential classification using Connectionist Temporal Classification (CTC) loss [16]. Sun et al. [4] propose to use multi-directional LSTM as the recurrent unit, whereas, Chen et al. [3] use Separable Multi-Dimensional Long Short-Term Memory for the same. These methods attempt to address word recognition by undertaking the task of directly converting an input document to a machine-readable text.

B. Word Spotting

Word spotting [17] is an alternative for word recognition where we formulate a matching task. The fundamental problem in word spotting is about learning an appropriate feature representation for word images which is suitable for matching within the collections of document images. In this paper, we consider the word level segmentation to be available a-priori, and thereby limit our discussion on works which are in the domain of segmentation-based word spotting. An initial method [18] represents the word image using profile features and then uses different distance metrics for comparing them. Other works use handcrafted features [19], and Bag of Visual Words [20] for word spotting. Most of these early representations were learned in an unsupervised way. The later methods drifted towards learning in a supervised setting and presented robust representation schemes. One of the classical methods in this space is from Almazan et al. [21] which introduced an attributes framework referred to as Pyramidal Histogram of Characters (PHOC) for representing both images and text. More recently, various deep learning based approaches [22], [23] have improved word spotting. VGGNet [24] was adopted by Poznanski et al. [25] for recognising PHOC attributes. Many other methods in the word spotting domain successfully explored using PHOC as the embedding spaces through different CNN architectures [7]–[10]. In the Indian language document community, methods like [20], [26], [27] attempt word spotting methods on Indian texts.

In this work, we propose to combine a CNN-RNN architecture proposed in [28] with the embeddings generated from the *E2E* proposed in [15] using learning-based methods. By combining two different approaches, we aim at assimilating the best attributes of both the methods.

III. Methods for improving word recognition

In this section, we elaborate on the proposed method using EmbedNet and CAB . This section is divided as follows, Section III-A and III-B brief about CRNN [28] and the End2End network [15], respectively. In Section III-C, we motivate EmbedNet and Section III-D proposes a novel Confidence based Accuracy Booster (CAB), a plug-and-play module for boosting the word recognition accuracy.

A. Word Recognition

We use a standard CNN-RNN (CRNN) hybrid architecture that was proposed in [28]. The network converts the textual contents of an image to textual transcriptions. Fig. 3 shows the architecture of the CRNN architecture used in our work. It consists of a spatial transformer layer (STN) followed by the residual convolutional blocks. These blocks are responsible for learning a sequence of feature maps using ResNet18 [29]. These feature sequences serve as an input into a stacked bidirectional long short-term memory (BLSTM) network. Connectionist Temporal Classification (CTC) [16] is then used for decoding the target label sequence over all the frames.

B. Word Spotting

Fig. 3 shows the End2End network proposed in [15] which learns the textual and visual word image embeddings. The network consists of two major input streams: real and label. In the real stream, ResNet34 contributes by generating the features for the real images. The label stream further gets divided into: (a) synthetic image stream and (b) text stream. Synthetic image's feature extraction takes place with the help of a shallow CNN architecture, while, generation of textual features happens using a PHOC extractor. The features generated are then appended and treated as a conditional label and merged using a fully connected network. The features generated from these streams are appended and merged using a fully connected network which preserves information from both modalities. This fully connected network preserves information from both modalities. After this, the embedding layer projects the embeddings from the real and label stream to a common subspace.

C. EmbedNet

In this work, we propose EmbedNet for projecting the embeddings to an updated embedding space and CAB for boosting the word recognition accuracy. We use a set of n word images for which we want to get the textual transcriptions. As shown in Fig. 3, these n images are passed through the real stream of *E2E* to generate embeddings represented by $\phi_i \; \forall i \in 1, \ldots, n$. K predictions generated for each of the n word image are converted to embeddings symbolized by $\psi_i^j \; \forall i \in 1, \ldots, n; \forall j \in 1, \ldots, K$ using the *E2E*'s label stream. We set the value of K equal to 20 throughout the paper, unless otherwise specified.

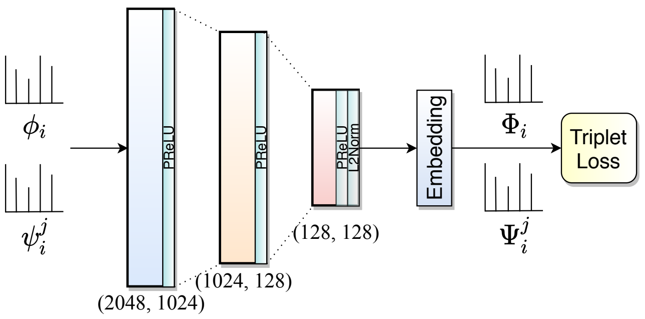


Fig. 2. An EmbedNet, during training, takes in a word image's embedding (ϕ_i^a), correct text's embedding (ψ_k^+) and incorrect text's embedding (ψ_l^-) one at a time. Corresponding output embeddings are passed through the triplet loss for training. Once trained, it takes in ϕ_i and ψ_i^j and generates Φ_i and Ψ_i^j, respectively. Tuples underneath the blocks represent the input and output size of the corresponding block. See text for notation.

We generate Φ_i and Ψ_i^j by providing ϕ_i and ψ_i^j as inputs to the EmbedNet, respectively. Fig. 2 shows the EmbedNet architecture; it projects the embeddings from $\Re^{2048}$ to $\Re^{128}$ using a 2048 dimensional linear input layer and 128 dimensional linear output layer and a hidden layer in between. We add PReLU activation function after each layer; it helps in introducing non-linearity to the model. L2 normalization is performed on the final layer's output to project the embedding on a 128-dimensional hyper-sphere. We train the EmbedNet for 200 epochs with early stopping acting as a regularizer and use the Adam optimizer with a constant learning rate of 0.0001.

Let EmbedNet be a function f_{en} defined as $f_{en}(\phi_i, \psi_i^j) = \Phi_i, \Psi_i^j$; it learns a compact Euclidean space where the correct Ψ_i^j lies closer to Φ_i, and incorrect Ψ_i^j lies farther away from Φ_i. We achieve the compact Euclidean space by training the EmbedNet using the triplet pairs as originally proposed in [30]. The pairs constitute of three different embeddings; the first one is the embedding of the word image from the train set for which we want to generate the textual transcription; we refer to them as the anchor denoted as $\phi_i^a \; \forall i = 1, \ldots, n$. The second one is the embedding of c words for which we have correct textual transcription; we call them positive denoted as $\psi_k^+ \; \forall k \in 1, \ldots, c$. The third one is the embedding of w words with incorrect textual transcription; we refer to them as negative denoted as $\psi_l^- \; \forall l \in 1, \ldots, w$. We sample the anchor from ϕ_i, and ψ_i^j is sampled for generating positive and negative.

The triplets are further classified into:

a) Hard Negatives: Equation 1 shows the condition for hard negatives; here, the Euclidean distance between the anchor and positive is greater than the distance between the anchor and negative. Due to this, they contribute the most while training the EmbedNet.

$$||\phi_i^a - \psi_k^+||_2^2 > ||\phi_i^a - \psi_l^-||_2^2 \tag{1}$$

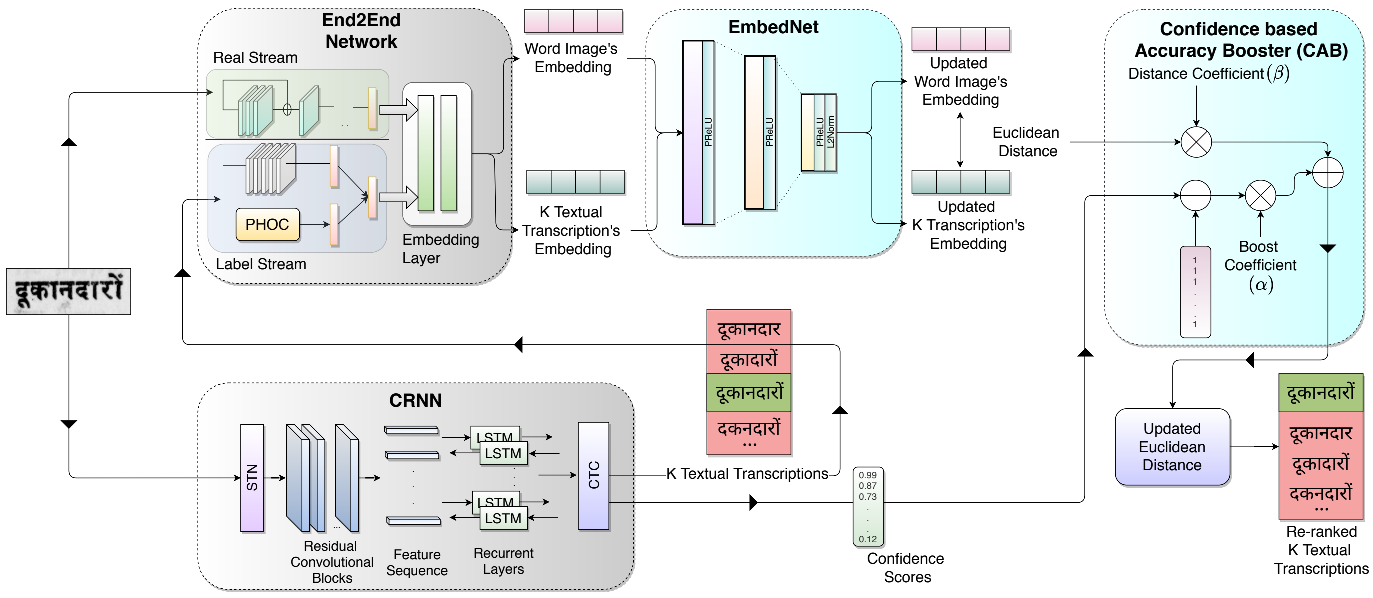


Fig. 3. For generating the textual transcription, we pass the word image through the CRNN [28] and the End2End network (‘*E2E*’) [15], simultaneously. The CRNN generates multiple (K) textual transcriptions for the input image, whereas the *E2E* network generates the word image’s embedding. The K textual transcriptions generated by the CRNN are passed through the *E2E* network to generate their embeddings. We pass these embeddings through the EmbedNet proposed in this work. The EmbedNet projects the input embedding to an updated Euclidean space, using which we get updated word image embedding and K transcriptions’ embedding. We calculate the Euclidean distance between the input embedding and each of the K textual transcriptions. We then pass the distance values through the novel Confidence based Accuracy Booster (CAB), which uses them and the confidence scores from the CRNN to generate an updated list of Euclidean distance, which helps in selecting the correct prediction. Diagram best viewed in color.

b) Semi-hard Negatives: Equation 2 defines the condition for semi-hard negatives; it relies on the margin (γ). Here the Euclidean distance between the anchor and negative is less than the distance between the anchor and positive + γ but higher than the Euclidean distance between the anchor and positive.

$$||\phi_i^a - \psi_k^+||_2^2 < ||\phi_i^a - \psi_l^-||_2^2 < ||\phi_i^a - \psi_k^+||_2^2 + \gamma \quad (2)$$

c) Easy Negatives: Equation 3 shows the condition for easy negatives; here, the Euclidean distance between the anchor and positive + γ is less than the distance between the anchor and negative.

$$||\phi_i^a - \psi_k^+||_2^2 + \gamma < ||\phi_i^a - \psi_l^-||_2^2 \quad (3)$$

Easy negatives do not contribute while training the EmbedNet as the condition of Euclidean distance between the anchor and positive example being less than the distance between the anchor, and negative is already satisfied. We train the EmbedNet using the Triplet loss, it is defined as:

$$L(\phi_i^a, \psi_k^+, \psi_l^-) = max(||\phi_i^a - \psi_k^+||_2^2 - ||\phi_i^a - \psi_l^-||_2^2 + \gamma, 0) \quad (4)$$

here ϕ_i^a, ψ_k^+ and ψ_l^- are anchor, positive and negative embeddings respectively and γ is the margin.

The triplet pairs are updated after every epoch. For updating, we pass ϕ_i and ψ_i^j through the EmbedNet, identify anchors, positives, and negatives. They are then further divided into hard negatives, semi-hard negatives, and easy negatives using equations 1, 2, and 3, respectively.

After training the EmbedNet, we generate ϕ_i and ψ_i^j for word images in the test set and pass them through the EmbedNet to generate Φ_i and Ψ_i^j; these updated embeddings help in selecting the correct predictions with much higher confidence. The reason behind this is, in the updated embedding space the correct text’s embeddings lie closer to the input word image’s embedding, and the wrong text’s embeddings lie farther away from the input word image’s embedding. For predicting the text in a given word image i, we query $\Psi_i^j \ \forall j \in 1, \ldots, K$ using Φ_i to generate a ranked list of predictions in increasing order of Euclidean distance. We consider the word with the least Euclidean distance as the new prediction.

D. Confidence based Accuracy Booster (CAB)

As shown in Fig. 4, CAB uses a vector of length K consisting of confidence scores, which are a measure of confidence of the CRNN for that particular prediction. Authors in [14] sum this confidence score with the Euclidean distances to improve the word recognition accuracy. We improve on it and introduce a novel Confidence based Accuracy Booster (CAB) as a plug-and-play module. Mathematically, CAB can be defined as:

$$f_{cab}(\overrightarrow{c}, \overrightarrow{e_d}) = |\overrightarrow{1} - \overrightarrow{c}| \times \alpha \oplus \beta \times \overrightarrow{e_d} \quad (5)$$

here, f_{cab} is the function for CAB, $\overrightarrow{c}$ denotes the vector of confidence scores of length K, $\overrightarrow{e_d}$ denotes the vector of Euclidean distances of length K, $\overrightarrow{1}$ denotes the vector of ones of length K, α is the boost coefficient, β is the distance coefficient, and $\oplus$ is the element wise addition operation. α and β are fixed to a constant value. f_{cab} takes in $\overrightarrow{c}$ and $\overrightarrow{e_d}$

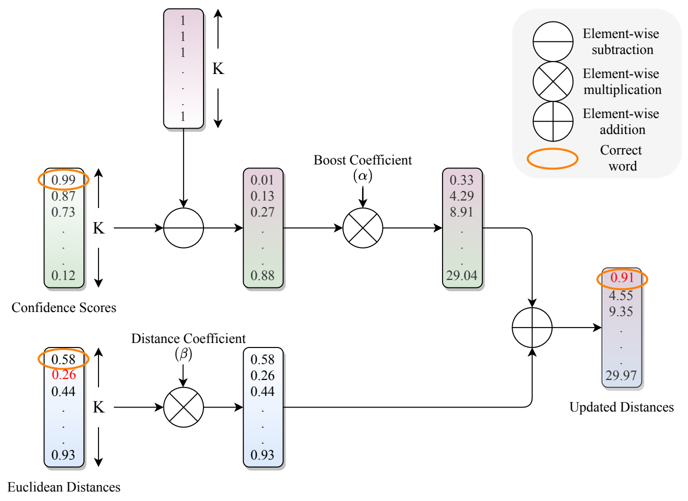


Fig. 4. Confidence based Accuracy Booster (CAB) takes in the confidence scores and the Euclidean distances and generates the updated distances. The number shown in red indicates the word with the lowest Euclidean distance with ϕ_i. The number circled in orange shows the correct word, which ideally should have the lowest Euclidean distance. According to the original Euclidean distances, the correct word is at position 2, whereas, the correct word is actually at position 1 (circled in orange). At position 1 in the confidence scores vector, we have a value of 0.99. After using CAB we get updated distances where the distance for the word at position 1 is the least. Therefore, CAB helps in incorporating the confidence scores and getting reliable distance values. Diagram best viewed in color.

and generates an updated list of distance where embeddings of words with higher confidence scores have a smaller distance value from ϕ_i. Using CAB, we achieve the highest word recognition accuracy on the validation set when we set the value of α and β equal to 33 and 1, respectively. Therefore, we fix the value of α and β to 33 and 1, unless otherwise stated.

A primary motivation behind creating and using CAB is to incorporate the confidence scores generated by the CRNN. As the value of K increases, noise in the predictions increases, which leads to lower confidence score values. Thus, by updating the distance values using the confidence scores, we can filter out the noisy predictions and select more relevant predictions.

IV. Experiments and Results

A. Dataset and evaluation metric details

TABLE I
The dataset consists of pages from the books in the Hindi language. The pages are annotated at word-level. The annotated words are further divided into train, test, and validation sets.

Language	Annotated	# Pages	# Word Images		
			Train	Validation	Test
Hindi	Yes	402	72,000	8,000	25,475

We perform all the experiments on books in the Hindi language, sampled and annotated from the DLI [31] collection. These books range from different periods and consist of a variety of font, font sizes, and a few degraded pages. As summarised in Table I, the sampled books consist of 402 pages containing $1,05,475$ words. We further divide these words into train, validation, and test sets for training and testing the EmbedNet. We use a pre-trained word recognizer (CRNN [28]) and an End2End ('*E2E*') network for all the experiments. We report the word recognition accuracy (WRA) for all the experiments performed. WRA is defined as

$$WRA = \frac{n_r}{n} \times 100, \tag{6}$$

where n_r represents the number of correctly recognised words, and n is the total number of words. WRA for methods using K hypotheses is calculated after generating the re-ranked list of predictions. This list is arranged in the increasing order of Euclidean distance with respect to the query. The word at the first position of the list is used for calculating the WRA.

TABLE II
Summary of the number of triplet pairs generated for different values of γ; we further classify them into hard, semi-hard, and easy negatives. We use hard and semi-hard negatives for training the EmbedNet. The number of hard, semi-hard, and easy negatives changes as the training progresses.

γ	# triplets		
	$15,71,820$		
	# hard negatives	# semi-hard negatives	# easy negatives
0.2	$7,894$	$1,04,468$	$14,59,458$
0.4	$7,894$	$4,24,590$	$11,39,336$
0.8	$7,894$	$11,52,568$	$4,11,358$
1	$7,894$	$11,64,065$	$3,99,861$

As described in Section III-C, we generate the triplets and categorize them for training the EmbedNet. Table II summarises the number of hard, semi-hard, and easy negatives for different margins (γ). With an increase in margin, we observe an increase in the number of semi-hard negatives and a decrease in easy negatives. It is beneficial to have a larger value for γ, as it maximizes the number of hard negative and semi-hard negative samples. Easy negatives do not contribute to EmbedNet's training, so we do not use them for training the network. While training the EmbedNet, the number of hard, semi-hard, and easy negatives changes. Initially, as the network starts from a random initialization, the number of easy negatives is the least while the number of hard and semi-hard negatives are more. As the training progresses, the number of easy negatives starts to increase while the other two categories decrease.

B. Selection of the best value for the margin

TABLE III
EmbedNet's performance for various values of γ.

Sr. No.	γ	Highest WRA (at K)
1.	0.2	83.116 (2)
2.	0.4	83.116 (2)
3.	0.8	83.226 (2)
4.	**1**	**83.242 (2)**

We train and validate multiple EmbedNets for different γ using the train and validation set defined in Table I. The aim here is to select the best value of γ. For that, we perform the experiments on four different values of γ. The results are reported in Table III. EmbedNet with γ equal to 1 has the highest WRA on the validation set as compared to a lower values of γ. The reason for this is that a small value of γ has a low count of semi-hard negatives (Table II), which results in reduced triplets for training the EmbedNet. For the rest of the paper, we consider the value of γ equal to 1 unless otherwise stated.

C. Results and Comparison with various methods

This section presents baseline methods used for assessing the improvement after using the EmbedNet with and without CAB. We also compare the WRA between the baselines and the methods proposed in this work. Baseline methods are:

a) Open-source OCR: For the first baseline, we use a pre-trained open-source word recognizer: Tesseract [11]. The motive here is to compare with an OCR which is not trained on noisy document images.

b) CRNN: The second baseline score shows the performance of the CRNN [28] trained using the best path decoding algorithm. It was trained on $Dataset1$ defined in [14]. Here, we generate a single prediction for each test image.

c) E2E+C: We generate the third baseline score using the method proposed in [14]; for that, we use the embeddings generated from *E2E* and multiple (K) hypotheses generated from the CRNN. For calculating the WRA of a given word image i, we perform a nearest neighbor's search on $\psi_i^j\ \forall j \in 1, \ldots, K$ using ϕ_i and add the confidence information to the distances obtained after the nearest neighbor's search; this provides us a re-ranked list, from which we consider the word with the least Euclidean distance as the new prediction. We refer to this method as '*E2E+C*'.

d) Multilayared Perceptron: For calculating the last baseline score, we train a Multi-Layered Perceptron (MLP) on the train data defined in Table I. MLP is a function f_{mlp} defined as $f_{mlp}(\phi_i) = \hat{\phi_i}$; it projects ϕ_i to an updated embedding space where the Euclidean distance between $\hat{\phi_i}$ and correct ψ_i^j is less than the distance between ϕ_i and correct ψ_i^j. MLP consists of three layers, the initial and final layers have the input and output dimensions of 2048, respectively; the hidden layer has the input and output dimensions of 256 and 128, respectively. The ReLU activation function follows each layer to introduce non-linearity. Mean Squared Error (MSE) is used as a loss function for training the MLP. We train the network for 150 epochs with early stopping acting as a regularizer and use the Adam optimizer with a constant learning rate of 0.0001. For calculating the word recognition accuracy, we query ψ_i^j using $\hat{\phi_i}$ for a given value of i and $j \in 1, \ldots, K$ to get a ranked list of predictions in increasing order of Euclidean distance. We consider the word with minimum Euclidean distance as the new prediction.

Table IV contrasts the WRA of all the baselines with the methods proposed in this work. We observe the lowest WRA for the methods not using multiple hypotheses, i.e., methods for which we have a maximum value of K equal to 1. Using [11], we achieve a WRA of 35.435 on the test set. As the training of the open-source OCR does not take place on the noisy documents that we are using, it performs the worst; this shows that the data that we are using contains highly degraded word images which are difficult to understand. On the other hand, we train a CRNN [28] on the train split defined in Table I and achieve a WRA of 81.543 on the test set.

TABLE IV
COMPARISON BETWEEN THE WRA OF ALL THE BASELINES AND THE METHODS PROPOSED IN THIS WORK. K_{high} SIGNIFIES K'S VALUE AT WHICH WE ACHIEVE THE HIGHEST WRA; (K) SIGNIFIES THE MAXIMUM VALUE OF K FOR THAT EXPERIMENT.

Sr. No.	Method	WRA	$K_{high}(K)$
1.	Tesseract [11]	35.435	1 (1)
2.	CRNN [28]	81.543	1 (1)
3.	*E2E+C* [14]	83.062	2 (20)
4.	*E2E+C* + CAB (ours)	84.358	11 (20)
5.	MLP (ours)	83.259	3 (20)
6.	EmbedNet (ours)	83.216	2 (20)
7.	MLP + CAB (ours)	84.782	20 (20)
8.	**EmbedNet + CAB (ours)**	**85.364**	**20 (20)**

We observe an improved WRA for the methods using multiple hypotheses, i.e., methods for which we have a maximum value of K higher than 1; in all the experiments, we have the maximum value of K equal to 20. We observe the WRA plateauing on the validation data for higher values of K; due to this, we choose to limit the highest value of K at 20. *E2E+C* achieves the maximum WRA of 83.062. However, as we observe in Fig. 5, it achieves a maximum WRA at a small value of K (2); WRA begins to decrease as we increase K. So, when using *E2E+C*, one cannot use K higher than two, making it impractical to use. We add CAB to *E2E+C* and observe a performance gain and more consistent WRA values for a higher value of K. Using *E2E+C* + CAB, we achieve the highest WRA of 84.358 at $K = 11$. Fig. 5 shows the change in WRA on increasing the value of K; we observe a steady increase till $K = 11$, after which the WRA starts to decrease. However, this decrease in the WRA is very small as compared to *E2E+C* without CAB. Even at $K = 20$, *E2E+C* + CAB achieves 8.267 more WRA as compared to *E2E+C* at $K = 20$. The reason for such stability is the usage of the confidence scores. As K increases, the noise present in the OCR's predictions also increases, leading to a lower confidence score for the noisy predictions. CAB uses this fact and results in better and consistent WRA.

We observe an improvement in the WRA for MLP and EmbedNet without CAB as compared to *E2E+C*, CRNN, and Tesseract [11]. MLP and EmbedNet achieve the highest WRA of 83.259 and 83.216, respectively. As observed in the case of *E2E+C* and shown in Fig. 5, the WRA starts to decrease as the value of K is increased, making them impractical to use. Upon using CAB with MLP and EmbedNet, we observe high gains in the WRA for large K as shown in Fig. 5. MLP + CAB attains the highest WRA at $K = 20$, equal to 84.782,

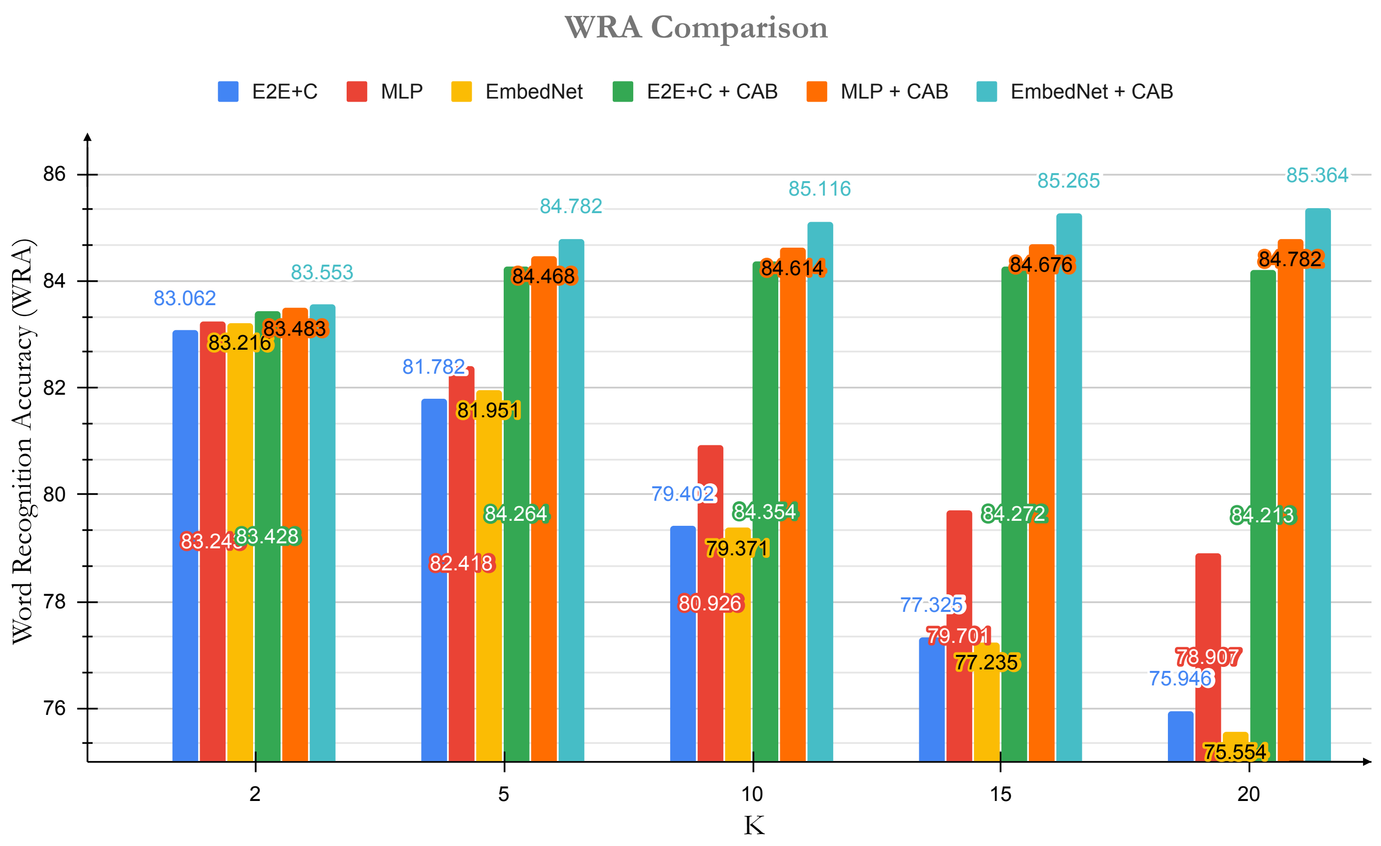
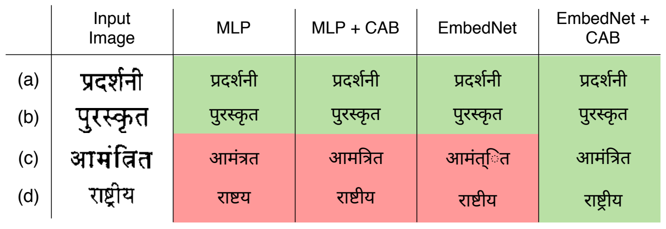


Fig. 5. Comparison between the WRA for *E2E+C*, MLP, and EmbedNet with and without CAB. For the experiments not using CAB, the WRA first increases and then starts to decrease. The reason for such a trend is, as K increases, the noise in the CRNN's predictions increases leading to lower WRA. However, using CAB helps avoid this issue, as it uses the confidence scores from the CRNN, which decreases as the noise increases. We achieve the highest WRA of 85.364 using EmbedNet + CAB at $K = 20$.

which is 8.836 more than the MLP without CAB at $K = 20$. As pointed out in Section III-C, EmbedNet not only helps in bringing correct Ψ_i^j closer to Φ_i but also pushes incorrect Ψ_i^j farther away from Φ_i. CAB utilizes this fact, and as we can see, we obtain a WRA of 85.364, which is 9.418 more than the *E2E+C* without CAB at $K = 20$.

Hence, by using the CAB, we observe substantial gains in WRA as K increases, and we also see a more steady WRA for all the values of K; this enables us to freely choose any value of K without any loss of WRA, which was not possible while using *E2E+C*, MLP, and EmbedNet without CAB.

D. Qualitative Results



	Input Image	MLP	MLP + CAB	EmbedNet	EmbedNet + CAB
(a)	प्रदर्शनी	प्रदर्शनी	प्रदर्शनी	प्रदर्शनी	प्रदर्शनी
(b)	पुरस्कृत	पुरस्कृत	पुरस्कृत	पुरस्कृत	पुरस्कृत
(c)	आमंत्रित	आमंत्रत	आमत्रित	आमंत्ि‌त	आमंत्रित
(d)	राष्ट्रीय	राष्टय	राष्टीय	राष्टीय	राष्ट्रीय

Fig. 6. Qualitative results on randomly chosen word images after processing using MLP and EmbedNet with and without CAB. Diagram best viewed in color.

Fig. 6 shows qualitative results on some randomly chosen words. Words in Fig. 6 (a) and (b) are long and contain characters and contain half consonants. Both of the words are recognised perfectly by MLP and WordNet with and without CAB. However, words in Fig. 6 (c) and (d) contains rare characters and are distorted, due to this MLP with and without CAB, and EmbedNet fail to predict the correct word. EmbedNet + CAB performs well for these cases and is able to predict the correct word. This shows the ability of EmbedNet to use the complementary information provided by word recognition and word image embedding methods.

E. Computational Costs

Table V shows the time taken for various processes done in the entire pipeline. All the experiments are performed on Intel Xeon E5-2640 v4 processors with 32 GB RAM on NVIDIA GEFORCE GTX 1080 Ti GPU. For calculating the time taken, we run the experiments 10 times and average the time taken in all the runs. The process of calculating the word accuracies for all the values of K is parallelizable; this reduces the time taken by $\frac{1}{K}$.

There are two modes in which the majority of our experiments take place. The first mode is the offline mode, which involves computations required only once. It includes time taken

TABLE V
TIME TAKEN BY VARIOUS PROCESSES IN THE WORD RECOGNITION PIPELINE. TIME FOR THE METHODS DEPENDENT ON K IS CALCULATED FOR $K = 20$. VALUES ARE REPORTED FOR A SINGLE WORD'S IMAGE/TEXT.

Mode	Process	Average time (in milliseconds)	Dependent on
Offline	Text from CRNN	580	K
	Text embeddings' generation	18.2	K
	Image embeddings' generation	23.89	K
Online	EmbedNet pass	0.21	Network's size
	MLP pass	0.22	Network's size
	WRA calculation	0.24	K
	WRA calculation with CAB	0.27	K

in generating the OCR output for all the n word images and the time taken by the End2End network in generating ϕ_i and ψ_i^j. Second is the online mode, which includes computations that are required every time we calculate WRA. It includes time taken in passing the embeddings through the MLP and EmbedNet. It also includes the time taken in calculating WRA with and without CAB.

V. CONCLUSION

To summarise, in this work, we aim at fusing the word recognition and word image embedding approaches for word recognition. For achieving this, we propose EmbedNet, which helps in learning an updated Euclidean space. We also propose CAB for using the updated Euclidean space and boosting the WRA by approximately 10% at $K = 20$. We show that learning based approaches for fusion show more promising results than rule-based fusion. As a future task, we plan to develop an end-to-end architecture capable of fusing word recognition and word image embedding approaches.

REFERENCES

[1] K. Dutta, P. Krishnan, M. Mathew, and C. V. Jawahar, "Towards Accurate Handwritten Word Recognition for Hindi and Bangla," in *Computer Vision, Pattern Recognition, Image Processing, and Graphics*, 2018.

[2] 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*, 2017.

[3] Z. Chen, Y. Wu, F. Yin, and C. Liu, "Simultaneous script identification and handwriting recognition via multi-task learning of recurrent neural networks," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2017.

[4] Z. Sun, L. Jin, Z. Xie, Z. Feng, and S. Zhang, "Convolutional multi-directional recurrent network for offline handwritten text recognition," in *Conference on Frontiers in Handwriting Recognition (ICHFR)*, 2016.

[5] U. Garain, L. Mioulet, B. B. Chaudhuri, C. Chatelain, and T. Paquet, "Unconstrained Bengali handwriting recognition with recurrent models," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2015.

[6] C. Adak, B. B. Chaudhuri, and M. Blumenstein, "Offline Cursive Bengali Word Recognition Using CNNs with a Recurrent Model," in *International Conference on Frontiers in Handwriting Recognition (ICHFR)*, 2016.

[7] P. Krishnan, K. Dutta, and C. V. Jawahar, "Deep Feature Embedding for Accurate Recognition and Retrieval of Handwritten Text," in *International Conference on Frontiers in Handwriting Recognition (ICHFR)*, 2016.

[8] S. Sudholt and G. A. Fink, "PHOCNet: A Deep Convolutional Neural Network for Word Spotting in Handwritten Documents," in *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016.

[9] T. Wilkinson and A. Brun, "Semantic and Verbatim Word Spotting Using Deep Neural Networks," in *International Conference on Frontiers in Handwriting Recognition (ICHFR)*, 2016.

[10] S. Sudholt and G. Fink, "Attribute CNNs for Word Spotting in Handwritten Documents," *International Journal on Document Analysis and Recognition (IJDAR)*, 2017.

[11] R. Smith, "An Overview of the Tesseract OCR Engine," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2007.

[12] P. Krishnan, R. Shekhar, and C. Jawahar, "Content level access to Digital Library of India pages," in *ACM International Conference Proceeding Series (ICPS)*, 2012.

[13] A. Gordo, J. Almazán, N. Murray, and F. Perronin, "LEWIS: Latent Embeddings for Word Images and Their Semantics," in *International Conference on Computer Vision (ICCV)*, 2015.

[14] S. Bansal, P. Krishnan, and C. V. Jawahar, "Fused Text Recogniser and Deep Embeddings Improve Word Recognition and Retrieval," in *Document Analysis Systems (DAS)*, 2020.

[15] P. Krishnan, K. Dutta, and C. V. Jawahar, "Word Spotting and Recognition Using Deep Embedding," in *IAPR International Workshop on Document Analysis Systems (DAS)*, 2018.

[16] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks," in *International Conference on Machine Learning (ICML)*, 2006.

[17] R. Manmatha, C. Han, and E. M. Riseman, "Word spotting: A new approach to indexing handwriting," in *Computer Vision and Pattern Recognition, CVPR*, ser. CVPR '96, 1996, p. 631.

[18] T. Rath and R. Manmatha, "Word spotting for historical documents," in *International Journal of Document Analysis and Recognition (IJDAR)*, 2007.

[19] A. Balasubramanian, M. Meshesha, and C. V. Jawahar, "Retrieval from document image collections," in *Document Analysis Systems (DAS)*, 2006.

[20] R. Shekhar and C. V. Jawahar, "Word Image Retrieval Using Bag of Visual Words," in *Document Analysis Systems (DAS)*, 2012.

[21] J. Almazán, A. Gordo, A. Fornés, and E. Valveny, "Word spotting and recognition with embedded attributes," *PAMI*, 2014.

[22] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Synthetic data and artificial neural networks for natural scene text recognition," in *Workshop on Deep Learning, NIPS*, 2014.

[23] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Deep Features for Text Spotting," in *European Conference on Computer Vision (ECCV)*, 2014.

[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[25] A. Poznanski and L. Wolf, "CNN-N-Gram for Handwriting Word Recognition," in *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[26] A. Bhardwaj, S. Kompalli, S. Setlur, and V. Govindaraju, "An OCR based approach for word spotting in Devanagari documents," in *Document Recognition and Retrieval Conference (DRR)*, 2008.

[27] S. Chaudhury, G. Sethi, A. Vyas, and G. Harit, "Devising interactive access techniques for Indian language document images," in *International Conference on Document Analysis and Recognition (ICDAR).*, 2003.

[28] K. Dutta, P. Krishnan, M. Mathew, and C. V. Jawahar, "Improving CNN-RNN Hybrid Networks for Handwriting Recognition," in *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2018.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *CoRR*, 2015.

[30] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[31] V. Ambati, N. Balakrishnan, R. Reddy, L. Pratha, and C. V. Jawahar, "The Digital Library of India Project: Process, Policies and Architecture," in *Second International Conference on Digital Libraries (ICDL)*, 2007.