Highlights

Candidate Fusion: Integrating Language Modelling into a Sequence-to-Sequence Handwritten Word Recognition Architecture

- A novel integrated language model for handwriting word recognition
- A sequence-to-sequence approach for handwritten word recognition
- Candidate fusion incorporates language statistics and commonly produced errors
- We achieve state-of-the-art performances challenging datasets

Candidate Fusion: Integrating Language Modelling into a Sequence-to-Sequence Handwritten Word Recognition Architecture

Lei Kang*†, Pau Riba*, Mauricio Villegas†, Alicia Fornés*, Marçal Rusiñol*
*Computer Vision Center, Barcelona, Spain
{lkang, priba, afornes, marcal}@cvc.uab.es
†omni:us, Berlin, Germany
{lei, mauricio}@omnius.com

Abstract

Sequence-to-sequence models have recently become very popular for tackling handwritten word recognition problems. However, how to effectively integrate an external language model into such recognizer is still a challenging problem. The main challenge while training a language model is to deal with the language model corpus which is usually different to the one used for training the handwritten word recognition system. Thus, the bias between both word corpora leads to incorrectness on the transcriptions, providing similar or even worse performances on the recognition task. In this work, we introduce Candidate Fusion, a novel way to integrate an external language model to a sequence-to-sequence architecture. Moreover, it provides suggestions from an external language knowledge, as a new input to the sequence-to-sequence recognizer. Hence, Candidate Fusion provides two improvements. On the one hand, the sequence-to-sequence recognizer has the flexibility to not only combine the information from itself and the language model, but also choose the importance of the information provided by the language model. On the other hand, the external language model has the ability to adapt itself to the training corpus and even learn the most common errors produced from the recognizer. Finally, by conducting comprehensive experiments, the Candidate Fusion proves to outperform the state-of-the-art language models for handwritten word recognition tasks.

Keywords: Handwritten Word Recognition, Sequence-to-Sequence Models, Language Model, Candidate Fusion

1. Introduction

Handwritten word recognition is the computer vision task that provide computers the ability to read handwritten text from images. Handwritten content is found in volume in both historic document collections [1], but also on current administrative documents [2] such as invoices, tax forms, notes, accident claims, etc. Automatic reading systems are particularly interesting for document digitization processes where paper documents are converted into machine encoded text. The information contained in such converted documents can be thus leveraged and used in any computer application, such as automatic decision making processes, document classification, automatic routing, etc. Unlike the recognition of typewritten text, handwritten word recognition is still a challenging research problem because of the large variability across different handwriting styles [3]. In the last few years, with the rise of deep learning architectures, some handwritten word recognition applications have started to reach a satisfying performance in some specific and restricted use cases [4, 5]. However, we are still far away from having a generic and robust system able to read any handwritten text.

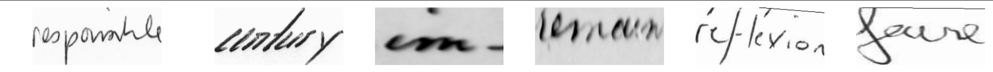
Automatic decoding textual information in images have several particularities. On the one hand, text is sequential in nature, coming in left to right order in latin languages. Most of the state of the art approaches are based on recurrent architectures [6] for leveraging this sequential information. On the other hand, text follows a particular set of syntactic rules and presents a well defined morphological structure. Text recognition systems often integrate statistical language models [7] that are able to complement the optical part boosting the overall recognition performance. Language models for handwritten word recognition implemented as probability distributions over sequences of characters and words, aim to provide context to discern between sequences of characters that might look similar, intending to resolve ambiguities from the optical recognition part. Different language model approaches have been proposed in the literature, from n-grams [8] to neural network architectures [9].

However, in most of the state-of-the-art handwritten word recognition systems, including the recent sequence-to-sequence-based ones [10, 11], the optical recognition part and the language models are seen as two separate and independent modules that are trained separately. Each of those modules are optimized separately using different data corpora, images of handwritten text on the one side, and a separate text corpora used to train the language statistics on the other. At the inference time, both modules are combined together. In that sense, language models are used either as a post-processing step, aiming at correcting recognition errors with the most likely sequence of characters [12], or as an integrated guiding module, steering the decoding process towards the best fitting letter succession [13].

In this paper we present a novel sequence-to-sequence-based handwriting recognition architecture that integrates the language model within the recognizer. Since the language model and the optical recognition parts are jointly trained and optimized, the language model does not just encode statistics about the language, but also models the most commonly produced errors from the optical decoder and how to correct them.

The handwriting word recognition architecture proposed in this paper significantly extends our preliminary work [11] by integrating a language model step within a sequence-to-sequence architecture. By incorporating the use of synthetic fonts and data augmentation strategies, we demonstrate the effectiveness and generality of our proposed approach in a significant amount of different public datasets and real industrial use cases. We exemplify in Figure 1 the different transcription results that we are able to obtain with the proposed architecture.

The rest of the paper is organized as follows. In Section 2, the state-of-the-art in handwritten word recognition is discussed. In Section 3, we present the data augmentation and pre-training processes leveraging the use of syn-



Syn.	`hesporuaklly`	`enterr`	`ma`	`eeree`	`RAlexiin`	`foure`
	↓	↓	↓	↓	↓	↓
+Tr.	`resporishle`	`unterry`	`cim`	`remann`	`réflévion`	`feure`
	↓	↓	↓	↓	↓	↓
+LM	**responsible**	**century**	**im**	**remain**	**réflexion**	**faire**

Figure 1: The improvements of performance from pre-training on only synthetic data, fine-tuning on training set of target dataset, to joint training with our proposed external language model, which are shown from top to bottom respectively indicating by arrows. The examples are from IAM, GW and Rimes datasets with two images per each from left to right respectively.

thetic handwriting-looking data. In Section 4, our attention-based sequence-to-sequence model for handwritten word recognition is described. In Section 5, we will focus on the proposed candidate fusion language model, while comparing it to two popular language models. In Section 6, the datasets, the full experimental setup, the ablation study, and the results on popular handwriting datasets are discussed in detail. Lastly, the conclusion is given in Section 7.

2. Related Work

Recognizing handwritten text has typically been addressed by combining computer vision and sequence learning techniques. The first handwritten word recognition approaches were based on Hidden Markov Models (HMMs) [14]. Such approaches used to be successful pioneers, while nowadays, they have been outperformed by Neural Networks-based architectures. With the rise of neural networks, Recurrent Neural Networks (RNNs) [15] have started to become popular to deal with sequential data such as handwriting. For example, Bidirectional Long Short-Term Memory (BLSTM) [16] or Multidimensional Long Short-Term Memory (MDLSTM) [17] have been widely adopted by handwritten word recognition community. Lately, these models have been discussed and improved. For example, Puigcerver [6] compared 1D-LSTM and 2D-LSTM layers to prove that multidimensional recurrent layers may not be necessary to achieve good accuracy for handwritten word recognition. Toledo *et al.* [18] provided an approach that combined character embeddings with a BLSTM decoding. Most of the handwritten word recognition approaches today are based on the use of a recurrent network with Connectionist temporal classification (CTC) layers [19]. However, CTC has two main drawbacks: First, the length of predicted sequence has always to be smaller than that of input sequence features. Thus, the number of pooling layers need to be carefully chosen in the CNN module of the encoder, so that the minimum width of image features would not be shorter than the maximum number of predicted characters. Second, the number of decoding time steps is dependent of input sequence features, i.e., the longer input handwritten image is, the longer the number of decoding time steps will be. Contrary, the number of decoding time steps is exactly the same as the maximum number of predicted characters in sequence-to-sequence-based approaches, because the attention mechanism could deal with the variable length visual features.

Recently, inspired by machine translation [20], speech recognition [21] and image captioning [22], the sequence-to-sequence architecture [10, 11] has started to be applied into handwritten word recognition tasks. These sequence-to-sequence approaches follow the architecture of encoder, decoder and attention mechanism. They present the advantage that by decoupling encoder and decoder, the output size is not determined by the input image width, so that the use of CTC can be avoided. For example, Sueiras *et al.* [10] provided a sequence-to-sequence based handwriting recognizer, but they imposed a manually set sliding-window. Our previous work [11] analyzed various strategies to find a proper sequence-to-sequence based architecture for specifically targeting handwritten word recognition tasks.

With the usage of RNNs, an implicit language model has been proved to help the recognition process in [23]. However, this internal language model is overfitted on the text of the training dataset. Among the popular handwritten dataset [24], there is a gap between training set and test set not only in the sense of handwriting styles, but also in the sense of text corpus. The main role of an external language model is to provide the knowledge learnt from an external text corpus, so that it can help to correct common errors made by the recognizer.

However, these sequence-to-sequence based handwriting recognizers do not integrate a language model in the whole system. Since the age of HMMs, there have been plenty of researches on the usage of linguistic knowledge to assist a HMM-based handwriting recognition process [25]. Later on, as the RNN-CTC model became the state-of-the-art on handwritten word recognition tasks, how to effectively integrate a language model into a recognizer has been a hot topic concurrent with the development of a handwriting recognizer. For instance [26, 27, 28, 6] have integrated character n-grams language modelling into a RNN-CTC based handwriting recognizer. Jelinek *et al.* [29] provided a cache trigram language model, which can be adapted to the current document more closely. Della *et al.* [30] proposed a minimum discrimination information model to adapt n-gram language model to a document. However, the n-gram model is just statistics on the co-ocurrence of characters computed over a text corpus and, even if they are helpful as an error-correcting post-processing step, they do not represent inherent language knowledge.

More recently, a Bert-like language model [31], pre-trained on plain text for masked word prediction and next sentence prediction tasks, has achieved state-of-the-art performance in many natural language understanding tasks. Zhu *et al.* [32] propose a method to incorporate Bert into machine translation architecture, which is a good trial to utilize a powerful pre-trained LM with a sequence-to-sequence model. However, Bert-like LM works at word- or wordpiece-level, which are restricted to a closed vocabulary so as to fail to predict OOV words. Moreover, since a Bert-like LM is trained and evaluated in parallel, it loses the sequential feature that can be split and injected into a Seq2Seq recognizer at each time step.

Concurrently, Recurrent Neural Network Language Models (RNNLM) [13, 12] have been developed prosperously among machine translation and speech recognition communities, because they can learn an effective representation of variable length characters and memory a long enough character history, outperforming the n-grams. Especially, Gulcehre *et al.* [13] provided two approaches: Shallow Fusion and Deep Fusion, which have been widely used and are the state-of-the-art RNNLMs in machine translation and speech recognition tasks. However, these RNNLMs are integrated into the sequence-to-sequence recognizer in a serial way. Both the sequence-to-sequence recognizer and the language models are trained separately and combined together in the final step. In that sense, the two different modules cannot properly benefit one from another and leverage the mutual information that both the optical recognizer and the language models could provide one to another. Our proposed candidate fusion language modelling is based on the idea that the optical part and the statistical character modelling shall communicate between each other, being able to jointly decode the most likely and most visually suitable character sequence.

3. Getting Enough Training Data

The first extension that we propose over our previous sequence-to-sequence architecture [11] is related to the training data pre-processing steps, namely the data augmentation step and the use of synthetic fonts.

A system able to effectively recognize handwritten words should be able to deal with the inherent deformations of handwriting text. These deformations not only come from the different writing styles across different individuals, but also in words written by the same person at different times. Figure 2 presents several real word images coming from different datasets and authors showing the huge variability in styles. Traditionally, to allow handwritten word recognition systems to generalize and prevent over-fitting, without having to manually annotate millions of word samples, data augmentation has been proposed in the literature. However, this technique is not able to increase the number of handwriting styles in the dataset. To solve this lack of diversity of handwriting styles, pre-training the recognition models with synthetic data is proposed. Intuitively, feeding more data that looks “realistic” as a pre-training provides a

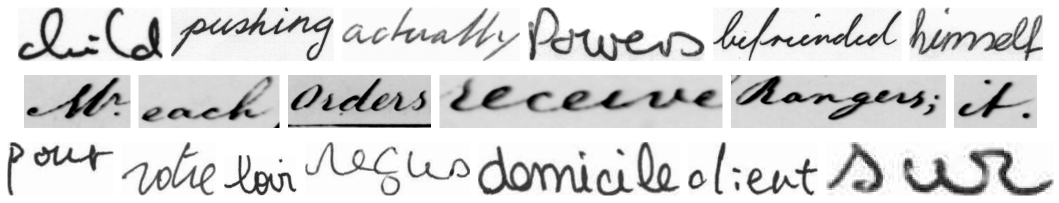


Figure 2: Real word samples in IAM, GW and Rimes datasets, from top to bottom, respectively. Each example has different characteristics such as shear, stroke width, language, etc.

pre-condition to our system, making it able to extract the general features required for handwriting recognition. Afterwards, a fine tuning with real data will adapt it to the desired use case. In this Section, both techniques are presented and adapted to handwritten words.

3.1. Data Augmentation

Having enough training data is crucial for the performance of deep learning frameworks. To tackle this problem, some data augmentation techniques have been proposed in the literature. Usually, random image transformations are applied to the training data in order to increase the diversity. In our sequence-to-sequence setting, these transformations are constrained to obtain a realistically looking image where the text is readable. In this work, we specially designed a pipeline able to capture the variability of real data in the document domain. These set of operations with random parameters are applied among all epochs and consist of a blurring / sharpening step, elastic transformations by using a mesh grid [33], shear, rotation, translation and scaling transforms, gamma correction and blending with synthetically generated background textures. The differences among the state-of-the-art data augmentation methods are shown in Table 1.

Figure 3 shows some examples that are used in training after the data augmentation module. Notice that the proposed operations introduce variations of word samples during training. This diversity helps to some extent to prevent over-fitting and leads to models that are able to generalize better than

Table 1: Comparison of the data augmentation techniques among state-of-the-arts.

Methods	**Dutta *et al.* [34]**	**Yousef *et al.* [35]**	**Proposed**
Blurring/sharpening	–	–	✓
Elastic transformation	✓	✓	✓
Shear	✓	✓	✓
Rotation	✓	–	✓
Translation and scaling	✓	✓	✓
Gamma correction	–	–	✓
Blending with background	–	–	✓
Sign flipping	–	✓	–

training just with the original set of images. However, the generated words are restricted to a fixed lexicon and the writing styles provided by the training set. Hence, the system is not able to extend the vocabulary which is a key feature in handwritten word recognition systems.

3.2. Pre-training with Synthetic Data

Recently, it has become a common trend the use of synthetically generated images to magnify the training data volume [36]. Instead of generating realistic images, the idea is to encode the necessary information required for a desired task. Available public datasets, such as the IIIT-HWS dataset [37], have already tackled the generation of synthetically generated handwriting word collections by the use of truetype electronic fonts. Such approach has the advantage that one can virtually generate an infinity of annotated training samples for free. However, the available datasets do not consider special characters (e.g. accents, umlauts, punctuation symbols, etc.) that may be required. Hence, we defined our own data generator able to be used to train several languages taking into account its own peculiarities.

As a text corpus, several books written in English and French have been used. These books are freely available on the Internet and will model the language character distribution. From these books, over 250.000 unique words were collected. Afterwards, we randomly render those words with 387 freely available

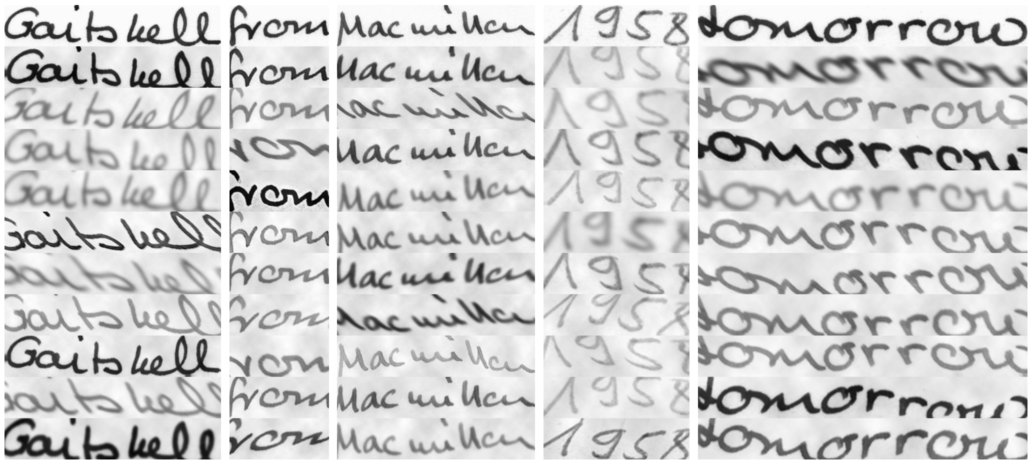


Figure 3: Examples of data augmentation on real handwritten words. The first row shows real word samples, then followed by 10 variations of such word after random data augmentation.

electronic fonts that imitate cursive handwriting. However, for a given font, all of the instances of a character will always look the same. In order to overcome such limitation, the same data augmentation pipeline previously presented has been applied. This augmentation step is applied online within the data loader, so that each batch is randomly augmented. Some samples of synthetic words are shown in Figure 4.

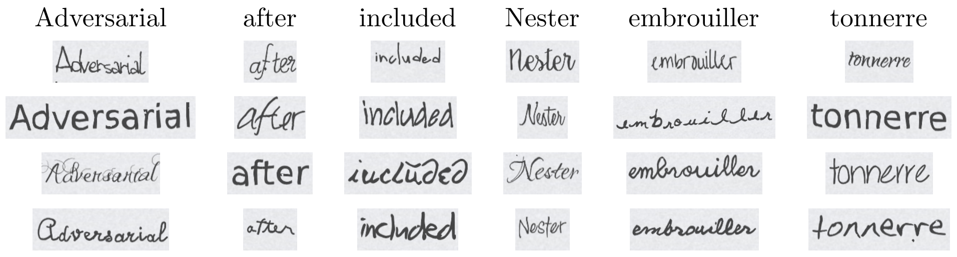


Figure 4: Examples of synthetic data generation. The first row is the given word from a public dictionary, then followed by 10 rendered image samples with different electronic fonts and random augmentation.

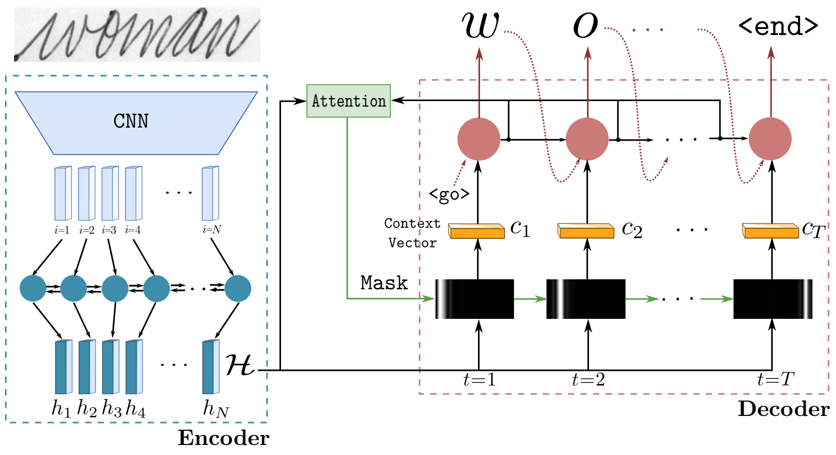


Figure 5: Architecture of the proposed sequence-to-sequence based recognizer.

4. Sequence-to-Sequence Word Recognizer

Our baseline handwritten word recognizer, previously introduced in [11] follows an encoder-decoder architecture. An attention mechanism is used to help the system focus at some spatial locations of the image when decoding character by character.

Figure 5 introduces the whole pipeline of the proposed sequence-to-sequence architecture. The proposed model has three components: encoder, attention and decoder. Each part will be detailed in the following sections.

4.1. Encoder

The first component of our architecture is an encoder, shown in the blue box in Figure 5, whose objective is to extract high-level features from handwriting word images, that hopefully will discriminate between different glyphs. These features should be able to codify the contents of the image as well as the order. In this work, we tested two different approaches for the encoding part. On the one hand a Convolutional Neural Network with positional encoding [38], and, on the other hand a CNN followed by a Recurrent Neural Network architecture, specifically, a Bidirectional Gated Recurrent Unit (BGRU).

Both the positional encoding and the recurrent network modules play the role to provide positional information. The idea is to help the decoder to follow the proper order during the decoding process. Different CNN architectures will be further discussed in Section 6.3. We denote $h_i \in \mathcal{H}, i \in \{1, 2, \ldots, N\}$ as the output sequence of the encoder, where $\mathcal{H}$ is the final feature representation of the whole word and N is the length of $\mathcal{H}$. Note that different word lengths will yield different feature sizes N. Figure 5 presents the encoder with a RNN.

4.2. Attention Mechanism

The basic idea of attention mechanisms is to focus the decoder part of the network to specific regions containing relevant information at each time step. In our particular scenario, what we expect is that the attention mask focuses at each different character at each decoding step following the reading order. By using attention mechanisms, the decoder task becomes much more simple since it should only focus on individual character recognition rather than obtaining also the proper order and feature alignment.

In our work we compare and discuss two popular attention mechanisms applied to the specific task of recognizing handwritten words. A detailed performance evaluation is provided in Section 6.3.

4.2.1. Content-based Attention

Attention mechanisms were first proposed in the machine translation field to help the decoder in deciding which words from the source sentence the network shall focus on to predict the proper output word in the target language. In this setting Bahdanau *et al.* [20] proposed an attention model which deals with the content of the input features. In this case, the order is not taken into account in this model and should be processed in the encoder step either by using positional encoding or a recurrent architecture. This occurs because the proposed mechanism deals with the feature contents rather than using a combination with the previous attention mask.

In this setting, we define α_t as the mask vector of the attention at time step t and s_t as the hidden state of the decoder at current time step $t \in \{1, 2, \ldots, T\}$, where T is the maximum length of the transcription. The mask vector of the attention α_t is calculated by:

$$\alpha_t = \text{Softmax}(e_t) \tag{1}$$

where $e_{t,i} = f(h_i, s_{t-1})$ is a learnable function. In this model $f(h_i, s_{t-1}) = w^T \tanh(Wh_i + Vs_{t-1} + b)$ where w, W, V and b are trainable parameters.

We denote as context vector the features that will be fed to the decoder at each time step. These features are generated from the attention mask α_t and the encoder feature representation H. Thus, the context vector at the current time step is obtained by,

$$c_t = g(\alpha_t, H) = \sum_{i=0}^{N\text{-}1} \alpha_{ti} h_i \tag{2}$$

4.2.2. Location-based Attention

Despite the use of positional encoding or recurrent networks, it is still a challenging task to track the attention steps across the word image. In this sense, Chorowski *et al.* [21] introduced a new attention method that makes use of the previous attention mask. In practice, this is done by adding an extra location term l_t, which is calculated by:

$$l_t = F * \alpha_{t-1} \tag{3}$$

where $F \in \mathbb{R}^{k \times r}$ is a trainable parameter, $*$ is a convolutional operation. Thus, the location-based attention can be written as:

$$e_{t,i} = f'(h_i, s_{t-1}, l_t) = w^T \tanh(Wh_i + Vs_{t-1} + Ul_{t,i} + b) \tag{4}$$

where w, W, V, U and b are trainable parameters. In this way, we have explicitly included the location information into the attention mechanism.

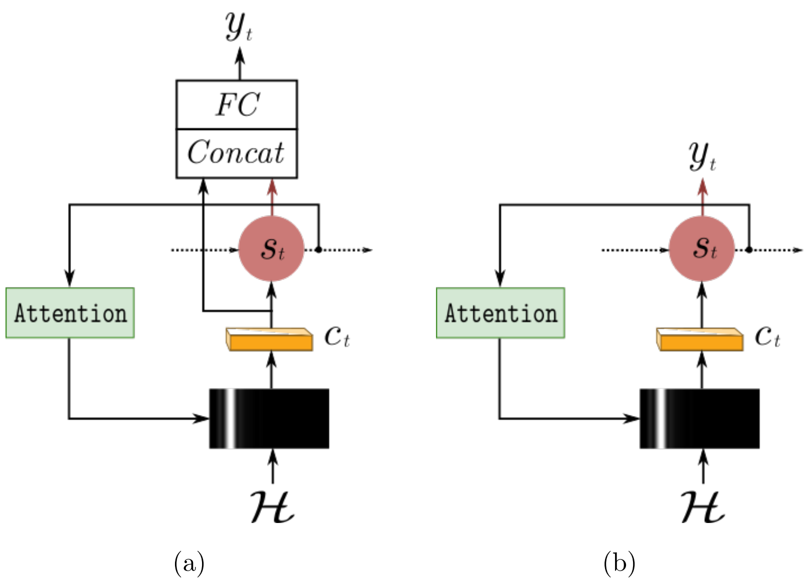


Figure 6: Architecture of the conventional decoder and our simplified decoder are shown in (a) and (b), respectively.

4.3. Decoder

Given the corresponding context vector and a $\langle go \rangle$ symbol, the decoder should be able to start the decoding process of the image text. The decoder is implemented as a unidirectional multi-layered GRU, which has enough capacity to predict a character at each time step. This character is then fed to the next iteration until an $\langle end \rangle$ symbol appears.

The decoder's output are the different predicted characters $\mathbf{y} = \{y_1, y_2, \ldots, y_T\}$ at each time step, where T is the maximum length of the final transcription. Our proposed decoder unit is different from the conventional decoder unit that has been utilized in other sequence-to-sequence approaches as shown in the red box in Figure 5. Since the decoder unit itself has enough ability to produce a proper character output, we can reduce the extra injection of context vector c_t. Thus, We simplify the procedure between the decoder hidden state s_t and the output logit y_t at current time step t as shown in Figure 6. The experimental comparison between both two architectures will be detailed in Section 6.3.

The predicted character at the current time step t is calculated by:

$$y_t = \arg\max(\omega(s_t)) \tag{5}$$

where $\omega(\cdot)$ is a linear operation to match the size of logit output to the size of possible characters. And then, to keep it simple and efficient, we just pick up the character with the highest probability and transfer it to a corresponding embedding vector $\tilde{y}_t$ by a embedding layer:

$$\tilde{y}_t = \text{Embedding}(y_t) \tag{6}$$

The input of each decoder unit consists of the previous prediction $\tilde{y}_{t-1}$ and the context vector c_t, so each hidden state of the decoder s_t is calculated by:

$$s_t = \text{Decoder}([c_t, \tilde{y}_{t-1}], s_{t-1}) \tag{7}$$

where $[\cdot, \cdot]$ is the concatenation of two vectors.

At the beginning of the decoding process, we always feed into the start signal $\langle go \rangle$ as the first input character, while the prediction process ends when the end signal $\langle end \rangle$ occurs or until the maximum time step T is reached.

5. Candidate Fusion Language Model

In this section, we propose a novel way to integrate language models into sequence-to-sequence models for handwritten word recognition tasks, that we coined as candidate fusion. The main idea is that the we first train a very simple language model with just text corpora (no images) with a recurrent neural network that given a sequence of characters is able to predict which is the most likely character to come next. This would be a similar idea of the well known word2vec models (e.g. skipgram) that are able to deduce most likely words within context, but for characters. Once this language model is pre-trained, now we can combine it with the optical decoder, so that the input to the decoder are not only the attended visual features at each particular time step, but also which is the most likely characters to be decoded given the ones that have been decoded so far.

Unlike the popular Shallow Fusion and Deep Fusion language models [13], shown in Figure 7(a) and (b) respectively, the final prediction is not decided by merging the outputs of the recognizer and the language model. The role of our language model is to provide other probabilities among all the characters, as indicated by y_t^{lm}, where t is the current time step during the decoding stage. This language model information will be injected into the decoder as one of the inputs. So the new hidden state of the decoder $\hat{s}_t$ is calculated by:

$$\hat{s}_t = \text{Decoder}([c_t, \tilde{y}_{t-1}, p_{t-1}^{lm}], \hat{s}_{t-1}) \tag{8}$$

where p_{t-1}^{lm} is the output of the language model from s_{t-1}^{lm} at the previous time step t−1. The effect of the linguistic knowledge will be extensively analyzed in Section 6.3.

The difference between Equations 7 and 8 is that we add now a second "adviser" p_{t-1}^{lm} into our decoder. Thus, the decoder can learn a trade-off between its output $\tilde{y}_{t-1}$ and that of an additional language model. We call it "adviser" because the decoder unit could choose to take into account the information from the "adviser" or totally ignore it. In this way, the explicit language model will never make the recognition performance worse than the baseline that is trained without language model at all. The reason is that, in an extreme case, if the bias of the language knowledge between the training data and the outside corpus is too high, the decoder can be adapted to predict transcriptions by ignoring the language model and just relying on the optical part.

Shallow Fusion directly applies a language model to the final prediction of the decoder by simply summing up both the probabilities of the recognizer y_t and the language model y_t^{lm}, as shown in Figure 7 (a). Because of the bias of the language knowledge between the training corpus and the outside corpus, summing up the probabilities of both the recognizer and language model modules may produce incorrect final transcriptions. Therefore, to make full use of Shallow Fusion, one must carefully select a corpus that shares most of the words within the target dataset and tune the weight hyper-parameter that is in charge of the trade-off between both the probabilities of the recognizer and the

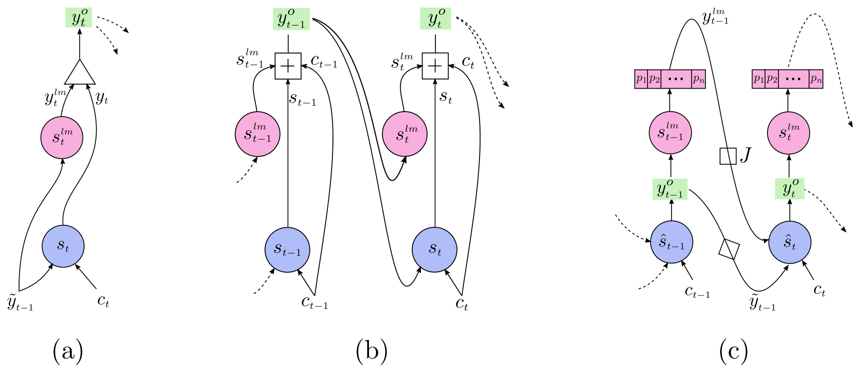


Figure 7: Architecture of the language models: (a) Shallow Fusion, (b) Deep Fusion, and (c) our proposed Candidate Fusion. The blue circles represent the hidden states of the decoder, the red circles represent the hidden states of the language model, and the green boxes are the final predictions at each time step. Especially in (a), the rectangle represents the summation between the predictions of the language model and the decoder; in (b), the crossed rectangle represents the concatenation process among the hidden state of the language model and that of the decoder and context vector; in (c), J is the injection function, which could be a softmax activation, sigmoid activation, embedding or direct connection without activation.

language model.

Deep Fusion shares the same language model as Shallow Fusion, but it goes one step further to merge both information from the recognizer and the language model in the feature level as shown in Figure 7 (b). The decoder of the recognizer and the language model are two independent pipelines, while the hidden state of the recognizer s_t, the hidden state of the language model s_t^{lm} and the context vector c_t at time step t merge together by concatenating them. Afterwards, the merged feature goes through a fully connected layer and an activation layer to generate final prediction y_t^o. Both the recognizer and the language model contribute to the final prediction, but they are still independent from each other. Thus, this method still can not handle the bias of the language knowledge between both the training corpus and the outside corpus. In any case, as it can be jointly fine-tuned, the performance could be better than the Shallow Fusion case.

Our Candidate Fusion language model, shown in Figure 7 (c), is designed to further boost the performance. During training, it treats each independent word as a sequence of characters for input and tries to generate a shifted prediction, which has no $\langle go \rangle$ symbol at the beginning but with extra $\langle end \rangle$ symbol to the end. The language model is first pre-trained on an external text corpus, and then plugged in the recognizer for a joint fine-tuning on the handwritten dataset. In the fine-tuning process, the input of the language model is the prediction of the recognizer at the previous time step, which takes into account both information of the recognizer and the language model. Note that the language model is also fine-tuned on the text of training dataset, which could further reduce the gap between the external text corpus and the text of target dataset. The intuition behind is to take advantage of the mutual information from both the optical recognizer and the morphology of the tackled language. This means that the decoder incorporates information both from the attended visual features and the language knowledge at each time step, and, at the same time, the language model itself can also adapt to the most common mistakes made by the recognizer. To do this, at each time step $t{-}1$, the language model takes the final prediction y_{t-1} as input and outputs a corrected version y_{t-1}^{lm} utilizing the learnt knowledge from the outside corpus. Then, at the next time step t, the recognizer takes the previous prediction of the recognizer y_{t-1}, the corrected version of the language model $y_{t-1}^{l}m$ and the current context vector c_t as inputs to generate the final prediction y_t. At Figure 7 we can see our difference that the final prediction is taken from the recognizer and the language model is highly integrated into the recognizer system as a candidate prediction, that is why we denote this method Candidate Fusion. We believe that it is a natural way to integrate a language model. In Section 6 we will show the performance improvement on popular datasets.

6. Experiments

In this section we present the extensive evaluation of our proposed approach. First, we perform several ablation studies on the key components to analyze the most suitable architecture. Second, we compare our recognizer with the state-of-the-art models on handwritten word recognition. Next, we analyze the performance of the most popular language models and compare with the proposed candidate fusion approach. Further, we apply a simple edit-distance-based lexicon to evaluate how the use of a closed lexicon can boost the performance. Finally, we provide an experiment on an industrial use case.

6.1. Datasets and Metrics

We will use several datasets for the experimental evaluation. They have been selected because of their different particularities: single or multiple writers, modern or historical documents or written in different languages. The IAM, George Washington (GW) and Rimes datasets are publicly available, whereas CarCrash is a private dataset. The details of these datasets are shown in Table 2.

The standard Character Error Rate (CER) and Word Error Rate (WER) metrics are utilized to evaluate the system's performance. Formally,

$$CER = \frac{S + I + D}{N} \tag{9}$$

where S, I, D are the number of character substitutions, insertions and deletions, respectively. N is the total number of characters in the groundtruth transcription.

$$WER = \frac{S_w + I_w + D_w}{N_w} \tag{10}$$

The WER metric is computed similar to CER. In this case, S_w, I_w, D_w and N_w refer to words instead of characters. Thus, a lower value indicates a better performance.

6.2. Implementation Details

All these experiments were run using PyTorch [41] on a cluster of NVIDIA GPUs. The training was done using the Adam optimizer with an initial learning rate of $2 \cdot 10^{-4}$ and a batch size of 32. We have set the dropout probability to

Table 2: Overview of the different datasets used in this work depicting its characteristics.

Dataset	**Words**	**Writers**	**Period**	**Language**
IAM [39]	115,320	657	Modern	English
GW [40]	4,860	1	Historical	English
Rimes [24]	66,978	1,300	Modern	French
CarCrash	24,492	640	Modern	German

be 50% for all the GRU layers except the last layer of both the encoder and the decoder. There is a probability of 50% to apply data augmentation on the training set, and we use label smoothing [42] as a regularization mechanism.

6.3. Ablation Study

The first experiment corresponds to an ablation study, which has been performed using the IAM dataset. The CER (%) and WER (%) shown correspond to the validation set of the IAM. The only one exception is Table 9, which is applied on the GW dataset.

Firstly, different popular CNN models have been evaluated in Table 3. Given that the VGG19-BN model obtains the best results, we have chosen it as the feature extractor in our architecture. The experiments are performed by choosing the CNN+BGRU architecture in the encoder part and location-based attention with label smoothing in the decoder part.

Secondly, we compare two different architectures of the encoder, as explained in Section 4.1. The CNN+BGRU architecture obtains better results than when using a CNN with positional encoding, as shown in the Table 4, because a trainable BGRU can provide not only the positional information, but also better mutual information among all the feature vector $\mathcal{H}$.

Thirdly, we compare our proposed decoder unit with the conventional decoder unit explained in Section 4.3. From Table 5 we notice that the proposed decoder unit has a similar and even slightly better performance even without the post feeding of the context vector. Thus, we opt to keep the simpler version of the decoder architecture.

Table 3: Comparison among the popular CNN models on IAM validation set.

Model	CER	WER	Model	CER	WER
VGG11-BN	7.35	20.91	ResNet101	5.38	14.34
VGG13-BN	6.85	19.76	ResNet152	5.13	13.89
VGG16-BN	6.57	17.04	SqueezeNet 1.0	6.82	17.35
VGG19-BN	**5.01**	**13.61**	SqueezeNet 1.1	8.25	20.56
ResNet18	6.72	16.13	Densenet121	5.29	13.79
ResNet34	5.51	14.25	Densenet169	5.30	14.23
ResNet50	5.27	13.95	Densenet201	5.37	13.79

Table 4: Comparison between positional encoding [38] and BGRU on IAM validation set.

Encoder	CER	WER
Pos. enc.	5.67	14.79
CNN+BGRU	**5.01**	**13.61**

Table 5: Comparison between the conventional decoder unit and the proposed simplified decoder unit on IAM validation set.

Decoder Unit	CER	WER	Time/Batch (s)
Conventional	5.06	13.91	0.236
Proposed	**5.01**	**13.61**	**0.229**

Table 6 shows the comparison between the two attention methods, as detailed in Section 4.2. We can observe that location-based attention with label smoothing obtains the best performance.

Finally, to make the best of a language model, we have investigated which is the best way to inject the linguistic knowledge. In Table 7, we have applied different injection functions on the output of the language model using the GW dataset. The best performance has been achieved without the usage of acti-

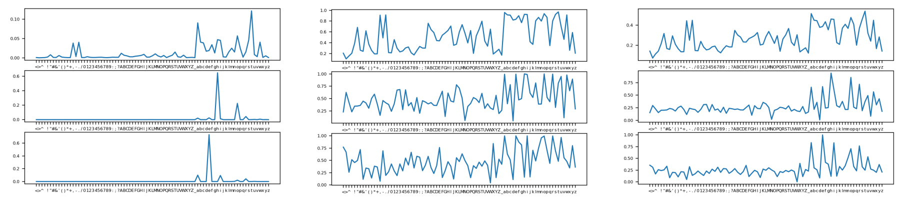
Table 6: Comparison between content and location-based attentions on IAM validation set.

Attention	**LabelSmooth**	**CER**	**WER**
Content	–	5.79	15.91
	✓	5.08	13.88
Location	–	5.49	14.74
	✓	**5.01**	**13.61**

vation function while doing batch normalization on the concatenation of the three components: the prediction of external language model, the embedding of the character that is predicted by the decoder at previous time step, and the current context vector. Different activations have been visualized in Figure 8. The softmax approach, as shown in Figure 8(a), is not working well because it gives too strong hypothesis to only one specific character in the available list. On the contrary, the sigmoid approach, as shown in Figure 8(b), gives independent probabilities across the available character list, but it also highlights the unrelated characters. The embedding approach selects the best hypothesis from the language model and feeds its embedded format into the decoder. This can help because the embedding process has projected the relevant linguistic characters into a common latent space, which gives the decoder an opportunity to select a possible character in a closed range in that space, but the embedding process loses some useful information. Thus, the best way is to use what it is provided from the language model without any activation function as shown in Figure 8(c), while batch normalizing the three inputs of the decoder can further improve the performance because of the similar value range for the three different vectors.

6.4. Main Results

In this section, firstly, we describe the comprehensive experiments that have been conducted to explore the best sequence-to-sequence architecture for handwritten word recognition tasks. Secondly, based on the baseline model that has



(a) (b) (c)

Figure 8: Visualization of the probability distributions among characters when using different injection functions for the output of language model. The groundtruth is "the", where predicted probability distributions are shown from top to bottom corresponding to "t", "h" and "e", respectively. (a) is of softmax, (b) is of sigmoid, and (c) is without activation.

Table 7: Comparison of the injection functions to inject the external language model on GW validation set.

Injection function	**CER**	**WER**
Baseline	2.82	7.13
Softmax	2.78	7.13
Sigmoid	2.81	7.04
Embedding	2.78	7.13
No activation	2.58	6.78
No activation + batch norm.	**2.52**	**6.61**

been selected, we carry on further experiments with the sequence-to-sequence model equipped with the different language models to prove their different effectiveness and robustness. Finally, a real industrial use case is shown to demonstrate its applicability to industry.

6.4.1. Baseline Model

We would like to analyze the performance of our sequence-to-sequence recognizer without any assistance from external language model nor a lexicon. So, in Table 8, we have listed all the comparable results achieved by the state-of-the-art handwriting recognizers. From the table, we observe that our recognizer achieves good performance on the IAM, GW and Rimes datasets. We show some examples of the visualized attention maps on the IAM, GW and Rimes (Figure 9). From those examples, we observe that the attention is able to attend each character at its corresponding time step. In addition, it can adapt itself to change its focus depending on the varied width of each character.

Table 8: Comparison with the state-of-the-art handwritten word recognition works, without language model nor lexicon. Results are evaluated on test sets of IAM, GW and Rimes datasets.

	IAM		GW		Rimes	
Method	CER	WER	CER	WER	CER	WER
Mor *et al.* [4]	–	20.49	–	–	–	11.95
Pham *et al.* [43]	13.92	31.48	–	–	8.62	27.01
Bluche *et al.* [5]	12.60	–	–	–	–	–
Wiginton *et al.* [44]	6.07	19.07	–	–	3.09	11.29
Sueiras *et al.* [10]	8.80	23.80	–	–	4.80	15.90
Kang *et al.* [11]	6.88	17.45	–	–	–	–
Krishnan *et al.* [45]	6.34	16.19	–	–	–	–
Toledo *et al.* [18]	–	–	7.32	–	–	–
Dutta *et al.* [34][a]	**4.88**	**12.61**	4.29	12.98	**2.32**	**7.04**
Proposed	5.79	15.15	**2.82**	**7.13**	2.59	8.71

[a]This work provides the results using **Test-time Augmentation**, which are not directly comparable with other results.

6.4.2. Integration of the Language Model

In this subsection we evaluate the performance of our language model by expanding the baseline model shown in Table 8. The results are shown in Table 9. Our language model is pre-trained with a large corpus, detailed in Section 3.2, and fine-tuned with the training data within the whole sequence-to-sequence system during end-to-end training process. Thus, the language

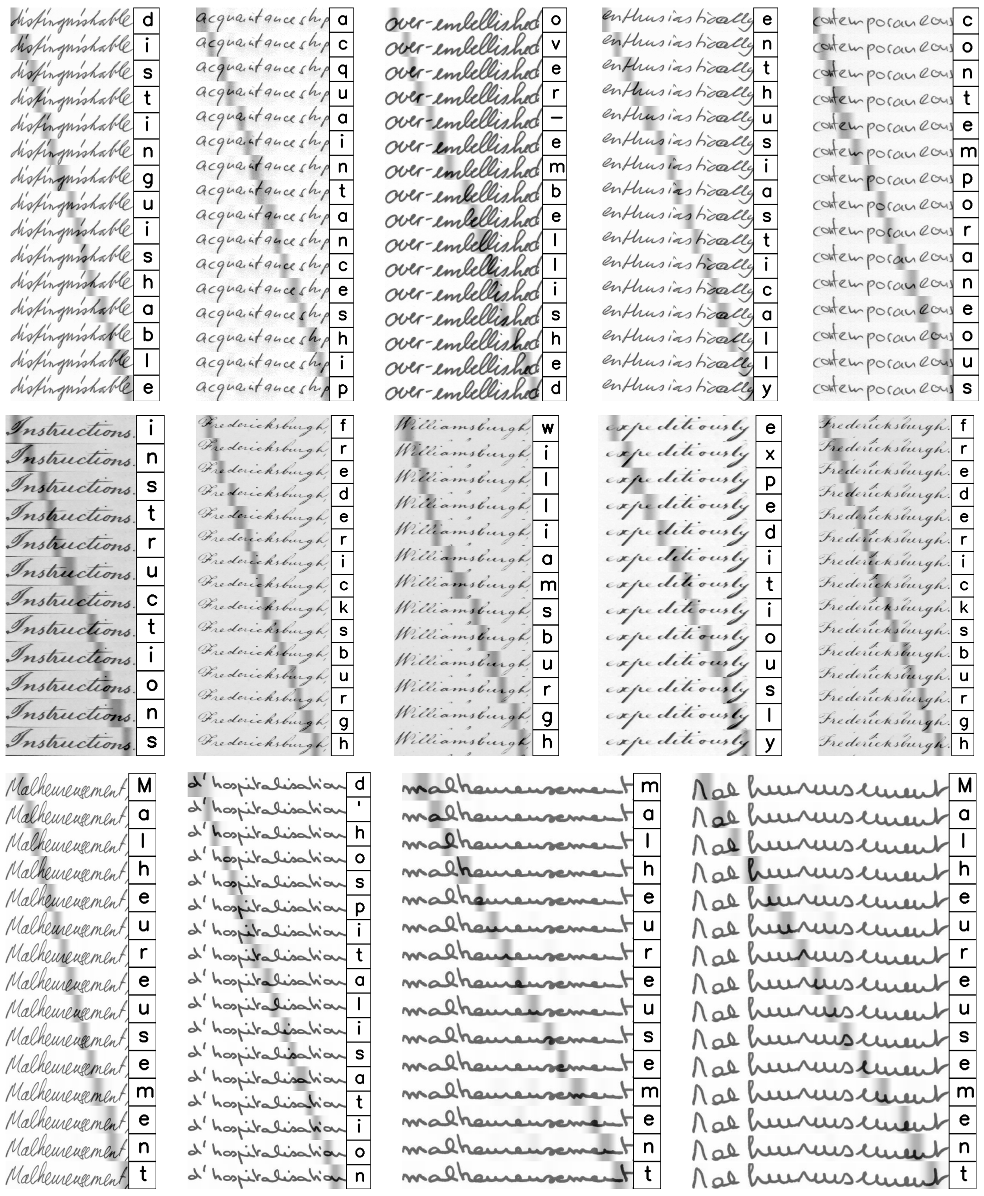

d i s t i n g u i s h a b l e
a c q u a i n t a n c e s h i p
o v e r – e m b e l l i s h e d
e n t h u s i a s t i c a l l y
c o n t e m p o r a n e o u s
i n s t r u c t i o n s
f r e d e r i c k s b u r g h
w i l l i a m s b u r g h
e x p e d i t i o u s l y
f r e d e r i c k s b u r g h
M a l h e u r e u s e m e n t
d ' h o s p i t a l i s a t i o n
m a l h e u r e u s e m e n t
M a l h e u r e u s e m e n t

Figure 9: Visualization samples of attention on IAM dataset (top), GW dataset (middle) and Rimes dataset (bottom).

model can adapt itself to a specific dataset corpus while keeping the capacity of generalization. To make a fair comparison, we have tuned the trade-off weight between language model and recognizer to achieve the best performance in the case of shallow fusion. While in the case of deep fusion, the results are obtained with early stopping on validation set.

As we can see in this table, our language model can boost the performance on all the three datasets, achieving better results than the Shallow Fusion and Deep Fusion language models. In fact, the Shallow Fusion makes the performance to decrease on all the three datasets, because it is too sensitive that any peaky probability distribution from both the outputs of the decoder and the external language model can ruin the final result. The Deep Fusion model behaves quite well on the GW and Rimes datasets, being able to improve the results a little bit compared to the baseline. In conclusion, our proposed Candidate Fusion is better than the Shallow and Deep Fusion approaches, because it is trainable and flexible to assist the recognizer during decoding. In addition, it does not need to manually tune the trade-off between the outputs of decoder and language model. In fact, in our Candidate Fusion architecture, the role of an external language model is to provide an extra predicted transcription based on the recognizer's prediction and its own language knowledge, while at the same time, the external language model can be adapted to the most common errors made by the sequence-to-sequence optical recognizer.

6.4.3. Restriction with a Close Dictionary

In all the experiments shown above, we never restrict the recognizer to a specific lexicon, which means the recognizer can predict out-of-vocabulary (OOV) words. Indeed, a generic handwritten word recognizer should not be restricted to closed lexicon in industrial use cases. However, since the use of closed lexicons is also a common practice, we have also tested how it can improve the overall performance. Thus, in Table 10, we have applied a simple edit-distance method to find the closest word in three lexicons: the brown lexicon with the lexicon of the test set (te+brown), the lexicon from the target dataset (tr+va+te), and

Table 9: Comparison with the state-of-the-art handwritten word recognition with language model, but not constrained by a lexicon. Results are evaluated on test sets of IAM, GW and Rimes datasets.

	IAM		GW		Rimes	
Method	CER	WER	CER	WER	CER	WER
Baseline no LM	5.79	15.15	2.82	7.13	2.65	8.71
Shallow Fusion LM	6.14	16.12	2.95	7.73	3.63	12.29
Deep Fusion LM	5.91	15.45	2.72	6.79	2.54	8.20
Candidate Fusion LM	**5.47**[a]	**14.51**[a]	**2.51**	**6.62**	**2.26**[a]	**7.47**[a]

[a]Statistically significant with threshold P-value 0.05.

only the lexicon of the test set (te). As expected, a lexicon can always improve the performance.

6.4.4. *Application to Text-line Level*

Our proposed method is not restricted to word level data. Thus, we propose an experiment to apply the Candidate Fusion LM based recognizer to text-line level Rimes dataset as shown in Table 11. The performance of joining the candidate fusion LM is proved to be statistically significant with threshold P-value 0.05.

6.4.5. *Application to a Real Industrial Use Case*

Finally, we evaluate our recognizer in a real world scenario for recognizing handwritten fields in car crash statement forms, which is an in-house private dataset. Due to the privacy protection, we could only show a cropped image of the real dataset in Figure 10. In this industrial dataset, the texts to be recognized are names, telephone numbers, emails, addresses and even check-boxes, which are way more challenging than the popular scientific datasets and would be unfeasible to be included in a vocabulary. Thus, we do not use explicit language model for both seq2seq- and CTC- based methods. Compared with

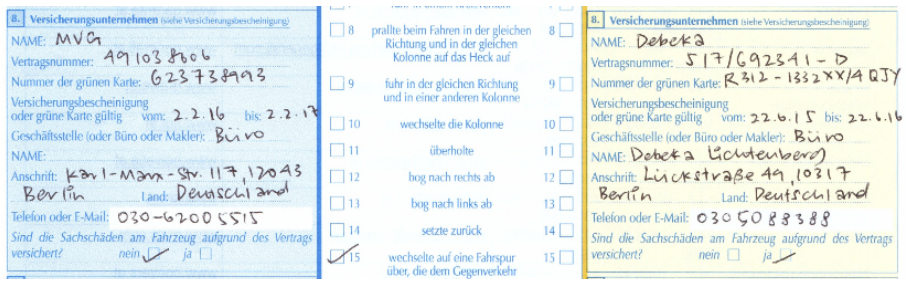
Table 10: Applying a simple edit-distance based lexicon constraint, the results are evaluated on test sets of IAM, GW and Rimes datasets.

	IAM		GW		Rimes	
Lexicon	CER	WER	CER	WER	CER	WER
Baseline	5.47	14.51	2.51	6.62	2.26	7.47
te+brown	4.97	10.30	2.29	4.90	1.82	4.59
tr+va+te	4.47	8.83	1.79	3.95	1.67	4.44
te	**4.15**	**8.11**	**1.63**	**3.44**	**1.47**	**3.81**

Table 11: Results at text-line level on the Rimes dataset.

Method	CER	WER
Baseline (w/o LM)	8.33	25.31
+ Candidate fusion LM	**6.87**	**21.14**

a well-known CTC-based approach [6], our proposed approach achieves better performance, as shown in Table 12. This results suggests that our model has a good generalization ability.


8. **Versicherungsunternehmen** (siehe Versicherungsbescheinigung)
NAME: MVG
Vertragsnummer: 49103 8606
Nummer der grünen Karte: 623738993
Versicherungsbescheinigung oder grüne Karte gültig vom: 2.2.16 bis: 2.2.17
Geschäftsstelle (oder Büro oder Makler): Büro
NAME:
Anschrift: Karl-Marx-Str. 117, 12043 Berlin Land: Deutschland
Telefon oder E-Mail: 030-6200 5515
Sind die Sachschäden am Fahrzeug aufgrund des Vertrags versichert? [✓ nein] ja

8 prallte beim Fahren in der gleichen Richtung und in der gleichen Kolonne auf das Heck auf 8
9 fuhr in der gleichen Richtung und in einer anderen Kolonne 9
10 wechselte die Kolonne 10
11 überholte 11
12 bog nach rechts ab 12
13 bog nach links ab 13
14 setzte zurück 14
[✓ 15] wechselte auf eine Fahrspur über, die dem Gegenverkehr 15

8. **Versicherungsunternehmen** (siehe Versicherungsbescheinigung)
NAME: Debeka
Vertragsnummer: 517/692341-D
Nummer der grünen Karte: R312-1332XX/4QJY
Versicherungsbescheinigung oder grüne Karte gültig vom: 22.6.15 bis: 22.6.16
Geschäftsstelle (oder Büro oder Makler): Büro
NAME: Debeka Lichtenberg
Anschrift: Lückstraße 49, 10317 Berlin Land: Deutschland
Telefon oder E-Mail: 030 5088388
Sind die Sachschäden am Fahrzeug aufgrund des Vertrags versichert? nein [✓ ja]

Figure 10: A cropped area of the real industrial use case dataset.

Table 12: Results of a real use case.

Method	CER	WER
CTC-based [6]	5.6	7.4
Proposed	**3.7**	**4.5**

7. Conclusion

In this paper we have presented a novel integration of an external language model into a sequence-to-sequence model for handwritten word recognition. Our proposed Candidate Fusion language model is trained and optimized together with the optical recognizer, avoiding biases between different training corpora. In addition, it has the advantage that the language model guides the decoding according to the most likely character sequence. The extensive evaluation, including an ablation study as well as comparisons with state-of-the-art approaches, demonstrates the effectiveness of our approach. Indeed our approach not only outperforms the existing approaches on public scientific datasets, but it also demonstrates its robustness on a real industrial use case.

Acknowledgments

This work was supported by the Spanish projects TIN2017-89779-P and RTI2018-095645-B-C21, and grants 2016-DI-087, FPU15/06264 and RYC-2014-16831. Titan GPU was donated by NVIDIA.

References

[1] V. Romero, A. Fornés, N. Serrano, J. A. Sánchez, A. H. Toselli, V. Frinken, E. Vidal, J. Lladós, The esposalles database: An ancient marriage license corpus for off-line handwriting recognition, Pattern Recognition 46 (6) (2013) 1658–1669.

[2] B. Moysset, R. Messina, Are 2D-LSTM really dead for offline text recognition?, International Journal on Document Analysis and Recognition 22 (3) (2019) 193–208.

[3] J.-P. Crettez, A set of handwriting families: style recognition, in: Proceedings of the International Conference on Document Analysis and Recognition, 1995.

[4] N. Mor, L. Wolf, Confidence prediction for lexicon-free OCR, in: Proceedings of the IEEE Winter Conference on Applications of Computer Vision, 2018.

[5] T. Bluche, J. Louradour, R. Messina, Scan, attend and read: End-to-end handwritten paragraph recognition with MDLSTM attention, in: Proceedings of the International Conference on Document Analysis and Recognition, 2017.

[6] J. Puigcerver, Are multidimensional recurrent layers really necessary for handwritten text recognition?, in: Proceedings of the International Conference on Document Analysis and Recognition, 2017.

[7] S. Dutta, N. Sankaran, K. P. Sankar, C. Jawahar, Robust recognition of degraded documents using character n-grams, in: Proceedings of the International Workshop on Document Analysis Systems, 2012.

[8] H. Bunke, S. Bengio, A. Vinciarelli, Offline recognition of unconstrained handwritten texts using hmms and statistical language models, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (6) (2004) 709–720.

[9] F. Zamora-Martinez, V. Frinken, S. España-Boquera, M. J. Castro-Bleda, A. Fischer, H. Bunke, Neural network language models for off-line handwriting recognition, Pattern Recognition 47 (4) (2014) 1642–1652.

[10] J. Sueiras, V. Ruiz, A. Sanchez, J. F. Velez, Offline continuous handwriting recognition using sequence to sequence neural networks, Neurocomputing 289 (2018) 119–128.

[11] L. Kang, J. I. Toledo, P. Riba, M. Villegas, A. Fornés, M. Rusinol, Convolve, attend and spell: An attention-based sequence-to-sequence model for handwritten word recognition, in: Proceedings of the German Conference on Pattern Recognition, 2018.

[12] S. Toshniwal, A. Kannan, C.-C. Chiu, Y. Wu, T. N. Sainath, K. Livescu, A comparison of techniques for language model integration in encoder-decoder speech recognition, in: Proceedings of the IEEE Spoken Language Technology Workshop, 2018.

[13] C. Gulcehre, O. Firat, K. Xu, K. Cho, L. Barrault, H.-C. Lin, F. Bougares, H. Schwenk, Y. Bengio, On using monolingual corpora in neural machine translation, arXiv preprint arXiv:1503.03535 (2015).

[14] S. Espana-Boquera, M. J. Castro-Bleda, J. Gorbe-Moya, F. Zamora-Martinez, Improving offline handwritten text recognition with hybrid hmm/ann models, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (4) (2010) 767–779.

[15] Z. C. Lipton, J. Berkowitz, C. Elkan, A critical review of recurrent neural networks for sequence learning, arXiv preprint arXiv:1506.00019 (2015).

[16] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, J. Schmidhuber, A novel connectionist system for unconstrained handwriting recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (5) (2008) 855–868.

[17] A. Graves, J. Schmidhuber, Offline handwriting recognition with multidimensional recurrent neural networks, in: Proceedings of the Conference on Neural Information Processing Systems, 2009.

[18] J. I. Toledo, S. Dey, A. Fornés, J. Lladós, Handwriting recognition by attribute embedding and recurrent neural networks, in: Proceedings of the International Conference on Document Analysis and Recognition, 2017.

[19] A. Graves, S. Fernández, F. Gomez, J. Schmidhuber, Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks, in: Proceedings of the International Conference on Machine Learning, 2006.

[20] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, arXiv preprint arXiv:1409.0473 (2014).

[21] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, Y. Bengio, Attention-based models for speech recognition, in: Proceedings of the Conference on Neural Information Processing Systems, 2015.

[22] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, Y. Bengio, Show, attend and tell: Neural image caption generation with visual attention, in: Proceedings of the International Conference on Machine Learning, 2015.

[23] E. Sabir, S. Rawls, P. Natarajan, Implicit language model in lstm for ocr, in: Proceedings of the International Conference on Document Analysis and Recognition, 2017.

[24] E. Augustin, M. Carré, E. Grosicki, J.-M. Brodin, E. Geoffrois, F. Prêteux, Rimes evaluation campaign for handwritten mail processing, in: Proceedings of the International Conference on Frontiers in Handwriting Recognition, 2006.

[25] U.-V. Marti, H. Bunke, Using a statistical language model to improve the performance of an hmm-based cursive handwriting recognition system, in: Hidden Markov models: applications in computer vision, 2001, pp. 65–90.

[26] T. Bluche, J. Louradour, M. Knibbe, B. Moysset, M. F. Benzeghiba, C. Kermorvant, The a2ia arabic handwritten text recognition system at

the open hart2013 evaluation, in: Proceedings of the International Workshop on Document Analysis Systems, 2014.

[27] B. Moysset, T. Bluche, M. Knibbe, M. F. Benzeghiba, R. Messina, J. Louradour, C. Kermorvant, The a2ia multi-lingual text recognition system at the second maurdor evaluation, in: Proceedings of the International Conference on Frontiers in Handwriting Recognition, 2014.

[28] P. Voigtlaender, P. Doetsch, H. Ney, Handwriting recognition with large multidimensional long short-term memory recurrent neural networks, in: Proceedings of the International Conference on Frontiers in Handwriting Recognition, 2016.

[29] F. Jelinek, B. Merialdo, S. Roukos, M. Strauss, A dynamic language model for speech recognition, in: Proceedings of the Speech and Natural Language Workshop, 1991.

[30] S. A. Della Pietra, V. J. Della Pietra, R. L. Mercer, S. Roukos, Adaptive language modeling using minimum discriminant estimation, in: Proceedings of the Speech and Natural Language Workshop, 1992.

[31] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).

[32] J. Zhu, Y. Xia, L. Wu, D. He, T. Qin, W. Zhou, H. Li, T.-Y. Liu, Incorporating bert into neural machine translation, Proceedings of the International Conference on Learning Representations (2020).

[33] P. Y. Simard, D. Steinkraus, J. C. Platt, et al., Best practices for convolutional neural networks applied to visual document analysis., in: Proceedings of the International Conference on Document Analysis and Recognition, 2003.

[34] K. Dutta, P. Krishnan, M. Mathew, C. Jawahar, Improving cnn-rnn hybrid networks for handwriting recognition, in: Proceedings of the International Conference on Frontiers in Handwriting Recognition, 2018.

[35] M. Yousef, K. F. Hussain, U. S. Mohammed, Accurate, data-efficient, unconstrained text recognition with convolutional neural networks, Pattern Recognition (2020).

[36] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, V. Koltun, CARLA: An open urban driving simulator, in: Proceedings of the Annual Conference on Robot Learning, 2017.

[37] P. Krishnan, C. Jawahar, Generating synthetic data for text recognition, arXiv preprint arXiv:1608.04224 (2016).

[38] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Proceedings of the Conference on Neural Information Processing Systems, 2017.

[39] U.-V. Marti, H. Bunke, The IAM-database: an english sentence database for offline handwriting recognition, International Journal on Document Analysis and Recognition 5 (1) (2002) 39–46.

[40] V. Lavrenko, T. M. Rath, R. Manmatha, Holistic word recognition for handwritten historical documents, in: Proceedings of the International Workshop on Document Image Analysis for Libraries, 2004.

[41] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch, in: Proceedings of the Conference on Neural Information Processing Systems Workshop, 2017.

[42] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[43] V. Pham, T. Bluche, C. Kermorvant, J. Louradour, Dropout improves recurrent neural networks for handwriting recognition, in: Proceedings of the International Conference on Frontiers in Handwriting Recognition, 2014.

[44] C. Wigington, S. Stewart, B. Davis, B. Barrett, B. Price, S. Cohen, Data augmentation for recognition of handwritten words and lines using a CNN-LSTM network, in: Proceedings of the International Conference on Document Analysis and Recognition, 2017.

[45] P. Krishnan, K. Dutta, C. Jawahar, Word spotting and recognition using deep embedding, in: Proceedings of the International Workshop on Document Analysis Systems, 2018.