Universitat Autònoma de Barcelona

Robust Handwritten Text Recognition in Scarce Labeling Scenarios: Disentanglement, Adaptation and Generation

Lei Kang

Robust Handwritten Text Recognition in Scarce Labeling Scenarios: Disentanglement, Adaptation and Generation

A dissertation submitted by **Lei Kang** at Universitat Autònoma de Barcelona to fulfil the degree of **Doctor of Philosophy**.

Bellaterra, November 19, 2020

Director:	**Dr. Alicia Fornés** Autonomous University of Barcelona Computer Science Dept. & Computer Vision Center
Co-director:	**Dr. Marçal Rossinyol** AllRead MLT & Computer Vision Center Computer Science Dept., Autonomous University of Barcelona
Co-director:	**Dr. Mauricio Villegas** omni:us - Qidenus Group GmbH Berlin, Germany
Thesis Committee	**Prof. Dr.-Ing. Gernot A. Fink** Department of Computer science, TU Dortmund Dortmund, Germany **Dr. Oriol Ramos Terrades** Autonomous University of Barcelona Computer Science Dept. & Computer Vision Center **Dr. Verónica Romero Gómez** Computer science Dept., University of Valencia Valencia, Spain

This document was typeset by the author using LaTeX 2_ε.

The research described in this book was carried out at the Computer Vision Center, Universitat Autònoma de Barcelona.

ISBN ISBN "Pending"

Printed by Ediciones Gráficas Rey, S.L.

A mi mamá y mi papá,

Acknowledgements

First of all, I want to thank omni:us, who funded me to finish an industrial doctorate and introduced me a deep insight to real application scenarios of document analysis. I also want to thank Computer Vision Center, where provided me a relaxed atmosphere to make me enjoy the fun topics freely.

I am so grateful to Marçal Rossinyol, Alicia Fornés and Mauricio Villegas, who are my supervisors. Marçal always encouraged me to follow my heart to find the fun topics and gave me insightful suggestions when I got stuck on a problem. Marçal is also my life coach with his quote "El 'no' ya lo tenemos.", which gives me the great confidence to try out any new task. I will remember this all my life and share it with my future kids (maybe many years later). I would also like to thank Alicia, who guided me to express my ideas in a more natural flow for the articles. I also remembered the time when I was applying for the PhD position, Alicia kept reminding me of the call deadline and the required documents, so that I can now be an active researcher in the document analysis community. I would also like to show my gratitude to Mauricio, who is my industrial mentor from omni:us. I would say Mauricio is one of the best researchers in industry with both coding skills and insightful ideas. I can still remember the happy time with Mauricio in Berlin, where we enjoyed the live music and cocktails in summer, and glühwein in winter.

To Manuel Carbonell, actually I prefer to call him Manolo, my tough man, thank you so much for backing me up in both my research and everyday life. Thanks to him, I get to know more and more Spanish rap songs and amazing Catalan musicians. We both share the same life belief to "enjoy the moment" on either the current research topic no matter how hard it is or just chilling out with a beer in hand. I was, I am and I will always be a fan of your "Banda Cetamol", and wish your band a prosperous future! To Pau Riba, my "fourth" mentor, thank you so much for the intensive discussions for all my works. I can still recall the funny memory when Pau, Sounak and I explored the Berga. I cannot forget that night when I was for the first time amazed with full of stars in the peaceful sky. To Juan Ignacio Toledo, namely Nacho, thank you so much for guiding me to the right research line at the beginning of my PhD. Nacho also taught me many funny local phrases and kept showing me different traditional food in either Spain or Argentina. To Yaxing Wang, thank you so much for bringing me into the world of GANs. I would say Yaxing is one of the most hard-working researchers in CVC,

his passion can always inspire me in a positive way.

To my friends and colleagues in Document Analysis Group, Josep, Dimosthenis, Oriol, Lluis, Joan, Ali, Andres, Rubèn, Mohamed, Arnau, Sounak, Sanket and the rest of CVC, Kai, Fei, Shiqi, Shun, Chenshen, Yixiong, Yi and those who already graduated, Zhijie, Lichao, Xialei, Lu, Raúl, Edgar, Xim, Anjan, Carlos, Felipe, Gemma, I really enjoyed the good memories with all you guys, and for sure we will have some beers together from time to time. Also a big thanks to Montse, Gisele, Mireia and Claire for the constant help and patience for my administrative documents.

I would also thank my friends and colleagues in omni:us, Martin, Sofie, Harald, Eric, Vasanth, Nischal, Christopher, Anna, Diede, etc. I cannot include every one here, but I do remember all the happy moments that we shared together in Berlin.

Last but now least, I am grateful to my father and mother, who give me a strong support, so that I can choose whatever life style I want to lead. So let's celebrate after I obtain the PhD degree!

Salud!

Abstract

Handwritten documents are not only preserved in historical archives but also widely used in administrative documents such as cheques and claims. With the rise of the deep learning era, many state-of-the-art approaches have achieved good performance on specific datasets for Handwritten Text Recognition (HTR). However, it is still challenging to solve real use cases because of the varied handwriting styles across different writers and the limited labeled data. Thus, both exploring a more robust handwriting recognition architectures and proposing methods to diminish the gap between the source and target data in an unsupervised way are demanded.

In this thesis, firstly, we explore novel architectures for HTR, from Sequence-to-Sequence (Seq2Seq) method with attention mechanism to non-recurrent Transformer-based method. Secondly, we focus on diminishing the performance gap between source and target data in an unsupervised way. Finally, we propose a group of generative methods for handwritten text images, which could be utilized to increase the training set to obtain a more robust recognizer. In addition, by simply modifying the generative method and joining it with a recognizer, we end up with an effective disentanglement method to distill textual content from handwriting styles so as to achieve a generalized recognition performance.

We outperform state-of-the-art HTR performances in the experimental results among different scientific and industrial datasets, which prove the effectiveness of the proposed methods. To the best of our knowledge, the non-recurrent recognizer and the disentanglement method are the first contributions in the handwriting recognition field. Furthermore, we have outlined the potential research lines, which would be interesting to explore in the future.

Resum

Els documents escrits a mà no només es conserven en arxius històrics, sinó que també s'utilitzen àmpliament en documents administratius, com ara xecs o formularis. Amb l'auge de de l'anomenat aprenentatge profund (*Deep Learning*), s'ha aconseguit un bon rendiment en conjunts de dades específics per al reconeixement de text manuscrit. Tot i això, encara és difícil resoldre casos d'ús reals a causa de la variació entre estils d'escriptura de diferents escriptors i el fet de tenir dades etiquetades limitades. Per tant, es requereix explorar arquitectures de reconeixement d'escriptura més sòlides així com proposar mètodes per disminuir la bretxa entre conjunts de dades font i objectiu de manera no supervisada.

En aquesta tesi, en primer lloc, explorem noves arquitectures per al reconeixement de text manuscrit, un mètode *Sequence-to-Sequence* amb mecanisme d'atenció i un mètode basat en transformadors no recurrents. En segon lloc, ens centrem en la disminució de la bretxa de rendiment entre les dades d'origen i les de destinació de manera no supervisada. Finalment, proposem un grup de mètodes generatius per a imatges de text manuscrits, que es poden utilitzar per augmentar el conjunt d'entrenament per obtenir un reconeixement més robust. A més, simplement modificant el mètode generatiu i unint-lo amb un reconeixedor, acabem amb un mètode de desenredament eficaç per destil·lar contingut textual d'estils d'escriptura a mà per aconseguir un rendiment de reconeixement generalitzat.

Superem el rendiment dels reconeixedors de text manuscrit de l'estat de l'art en els resultats experimentals entre diferents conjunts de dades científics i industrials, que demostren l'eficàcia dels mètodes proposats. Tant ell reconeixement no recurrent com el mètode de desenredament són les primeres contribucions al camp del reconeixement d'escriptura a mà. A més, hem esbossat les línies de recerca potencials, que serien interessants explorar en el futur.

Resumen

Los documentos manuscritos no solo se conservan en archivos históricos, sino que también se usan ampliamente en documentos administrativos como cheques y reclamaciones. Con el auge de las redes neuronales profundas, muchas técnicas del estado del arte han obtenido un buen rendimiento en conjuntos de datos específicos para el reconocimiento de texto manuscrito (HTR). Sin embargo, los casos de uso reales todavía son un desafío debido a la variabilidad de estilos de escritura de diferentes escritores y la cantidad limitada de datos etiquetados. Por lo tanto, es necesario explorar tanto arquitecturas para reconocimiento de texto manuscrito más robustas como proponer métodos para disminuir la brecha entre los datos de origen y destino de una manera no supervisada.

En esta tesis, en primer lugar, exploramos arquitecturas novedosas para el HTR, desde el método secuencia-a-secuencia (Seq2Seq) con mecanismo de atención, hasta el método no recurrente basado en Transformers. En segundo lugar, nos centramos en reducir la brecha de rendimiento entre los datos de origen y de destino mediante métodos no supervisados. Finalmente, proponemos un grupo de métodos generativos para imágenes de texto manuscrito, que pueden usarse para aumentar el conjunto de entrenamiento y obtener un reconocedor más robusto. Además, simplemente modificando el método generativo y uniéndolo con un reconocedor, obtenemos un método eficaz para destilar el contenido textual de los estilos de escritura para lograr un rendimiento de reconocimiento generalizado.

En resultados experimentales obtenemos rendimientos en HTR que superan los del estado del arte en diferentes conjuntos de datos científicos e industriales, los cuales demuestran la efectividad de los métodos propuestos. Hasta donde sabemos, el reconocedor no recurrente y el método de para destilar son contribuciones originales en el campo de reconocimiento de texto manuscrito. Finalmente, hemos esbozado posibles líneas de investigación que sería interesante explorar en el futuro.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Handwritten Text Recognition and Its Limitations

Handwritten Text Recognition (HTR) has interested the Document Analysis and Pattern Recognition communities for several decades. Handwritten content is found in volumes of not only historic archives [128], but also contemporary administrative documents [114] such as invoices, cheques, tax forms, notes, accident claims, etc. Automatic reading systems are particularly interesting for document digitization processes where paper documents are converted into machine readable text. Thus, such extracted text can be further leveraged in distilling useful information for some computer applications, such as automatic decision making processes, document classification, automatic routing, etc. Even though research in HTR began in the early sixties [108], it is still considered as an unsolved problem. The main challenge is the huge variability and ambiguity of the strokes composing words encountered across different writers [29].

We humans, once we learn how to read scripted words, perform quite well at reading handwritten texts produced by individuals with handwriting styles that we have never seen before. However, computational models strive at being so generic unless they are supplied with huge amounts of training data coming from many different writers. But, gathering such huge annotated collections of training data quickly becomes too expensive. Although in the literature some publicly available benchmarking datasets have been established, such as IAM [103] or RIMES [5], their volumes are still far away from nowadays computer vision large scale datasets like ImageNet [32] or Open Images V4 [91], that contain millions of annotations. Without such large amount of training data, deep learning architectures for HTR are prone to overfit to the seen writers and not generalize well.

Moreover, besides the public benchmarking datasets, real use cases should also been taken into consideration when developing novel HTR methods. In industrial scenarios, the targeted documents are extremely varied across different document types and multiple writers. It is unrealistic to provide the groundtruth for each type of document and writer, so a robust recognizer that provides a generalized performance is demanded. Another important problem in industrial applications is how to transfer the learned knowledge from a largely owned dataset to a new target scenario, which might be small in quantity but presenting a large bias from the training data. Last but not least, the efficiency of both training and evaluation processes is also a concern, where simpler architectures and better utilization of parallel computing is desired.

In summary, even though we have seen an important performance improvement of HTR methods thorough the recent years, there are still some limitations. A great interest exists in developing a robust handwritten text recognizer methods aimed at dealing with varied handwriting styles but requiring limited labeled data. Even more, the proposed methods should work for not only in academic datasets but also in real industrial use cases.

1.2 State-Of-The-Art Methods

This thesis focuses on offline handwriting recognition in word and text-line levels. There are two groups of methods in this field, on the one hand, methods towards a robust recognizer, which are studied for exploring different neural network architectures; on the other hand, methods towards minimizing the bias between training and test data, which are in charge of boosting HTR performance without extra manual effort of labeling.

1.2.1 Recognition Methods

In the field of document analysis, there are three popular groups of methods achieving state-of-the-art recognition performance. Since the handwriting text is sequential in nature, the first applications with Hidden Markov Models (HMM) [12, 52] started to show a satisfactory performance on HTR tasks. To obtain a better feature representation from the handwritten images, some other Neural Network (NN) modules were equipped together with HMM to form the Hybrid HMM methods [19]. Recurrent Neural Networks (RNN) [96] has a stronger capability in dealing with long sequential dependencies. The join method of both RNN and Connectionist temporal classification (CTC) [57] has become the state-of-the-art approaches for HTR tasks [133, 152, 86, 123, 112] recently. Nearest neighbor search methods [125, 85, 2] maps all the handwritten words in a common embedding space and then the recognition is done by finding the nearest one along the feature distance. This group of methods has a severe limitation that Out-Of-Vocabulary

(OOV) words can not be predicted, which draws the main disadvantage for applications in the real use cases.

In the fields of general computer vision and natural language processing, there are two more popular groups of methods apart from the three mentioned above. Sequence-to-Sequence (Seq2Seq) models follow an encoder-decoder paradigm using attention mechanism. This group of methods were first achieved the state-of-the-art performance in the fields of machine translation [135, 7], image captioning [155] and speech recognition [27, 8]. There are some early trials applying the Seq2Seq model to HTR tasks [17], but the performance is not satisfying. The Transformer architecture was first proposed by Vaswani *et al.* [146], which relies entirely on attention mechanisms without the need of recurrent nets. Recently, it has become the state-of-the-arts in natural language processing [33] and speech recognition [37]. The advantage of Transformer than Seq2Seq is the efficient utilization of parallel computing during training process. So it would also be interesting to apply this method to HTR tasks.

1.2.2 Bias Diminishing

As we have discussed before that there is always a bias between training and target data, so that the method cannot be well generalized to achieve the best performance. In document analysis community, one of the most popular methods is data augmentation techniques [152]. They include blurring/sharpening, elastic transforming, shearing, rotating, translating, scaling, gamma correcting and blending with synthetic background textures, which are performed by modifying the training data so as to improve the generalization. Another popular method is to make use of a large amount of synthetic data for pre-training and then fine-tune on the target data. Synthetic data [90] is created using TrueType fonts and data augmentation techniques. Given a text corpus, zillions of synthetic handwritten images could be rendered and used to enlarge the training set. Then, by accessing more data during training process, the recognizer could achieve a more generalized performance on HTR tasks.

There are also some other machine learning methods that are widely used in computer vision and natural language processing fields. In transfer learning [118], a model is firstly trained on a base dataset, and then transferred to a target dataset. The process works under the assumption that the learned features are general instead of specific to the base dataset. As the neural network methods always ask for a large amount of data for training, transfer learning is especially useful when the target data is few. Unsupervised domain adaptation [50] aims to classify unlabeled target domain by transferring knowledge from labeled source domain with domain shift. Based on the base model (e.g. a recognizer), an auxiliary domain classifier is equipped to discriminate between source and target data. The domain classifier is optimized by minimizing the loss on both source and target data during training process, while the base model is trained by maximizing the loss of domain

classifier. Thus, the domain-invariant features can be obtained with the Min-Max game. The disentanglement [104] offers a neural network model to disentangle a feature representation into different isolated factors. Recently, this is widely used in image-to-image translation [53]. Natural scene images can be disentangled into content (contour and shape) and style (visual appearance) features, which enable the generation of new images by permutation and combination of these content and style features.

1.3 Motivation and Research Questions

Based on the limitations of current HTR methods that they still lack the capacity to deal with the varied handwriting styles in a few data scenario, we would like to explore novel methods among the fields of document analysis, computer vision and natural language processing. In this thesis, we focus on the HTR problems. On the one hand, we explore different network architectures for the handwritten text recognizer, and achieve the state-of-the-art performance; on the other hand, we analyze the fundamental problem in HTR tasks and propose different approaches to reduce the gap between training and test sets. Through this thesis, we are interested in answering the following questions:

- *How long a sentence can be recognized properly with modern handwriting recognizer?*
- *Are Recurrent Neural Networks a must component in state-of-the-art handwriting recognizer?*
- *How a language model can help to improve the HTR performance?*
- *How could we reduce the gap between training set and test set in a target dataset or even between different datasets?*
- *How to disentangle textual content from handwriting styles in handwritten text images?*
- *How to generate artificial samples of handwritten text images with a particular content while mimicking a specific calligraphic style?*

1.4 Contributions

The main contributions of this thesis are:

1. We present a Seq2Seq model with location-based attention mechanism for HTR tasks, which is in an end-to-end fashion and achieves the state-of-the-art performance.

2. We present a novel language model, namely Candidate Fusion, to equip with Seq2Seq recognizer for the boosting of HTR performance, which is an effective way to incorporate external language knowledge while keeping the capacity in dealing with OOV words.

3. We present a non-recurrent model based on Transformer model for HTR tasks, which overcomes the drawback of Seq2Seq model with sequential pipelines. Thus, the proposed transformer recognizer, on the one hand, achieves a more effective training process with full parallelization in architecture; on the other hand, shows a better HTR performance on longer text images.

4. We propose a writer adaptation method to diminish the gap between synthetic and real datasets to reduce the demand for manually labeled training data. In addition, the adaptation method can also decrease the error rate gap between training and test sets in the real dataset, which is a further boost on the target data with the same amount of labeled training data.

5. We propose a disentanglement method to distill textual content from handwriting styles in an fashion of Image-to-Image translation, which is jointly trained with Seq2Seq recognizer with the shared textual content encoder. Thus, the HTR performance can be boosted with a more robust recognizer that is capable of extracting images features with exclusion of specific handwriting styles.

6. We propose a handwriting synthesis method using GANs to generate any text samples conditioned on text strings and handwriting style images, whose generated samples are indistinguishable by humans and can be utilized to boost the HTR performance.

We have categorized HTR methods in two main groups, either proposing new architectures or boosting the performance by other techniques, as shown in Table 1.1. Bold text indicates the proposed novel methods in this thesis.

1.5 Organization

The rest of this thesis is organized in six chapters.

In Chapter 2, we propose an attention-based Sequence-to-Sequence model for Handwritten Text Recognition. The proposed architecture has three main parts: an encoder, consisting of a CNN and a bi-directional GRU, an attention mechanism devoted to focus on the pertinent features and a decoder formed by a one-directional GRU, able to spell the corresponding word, character by character. We also propose a novel method to integrate an external language model to a Sequence-to-Sequence architecture, which provides suggestions from an external

Table 1.1: A general classification of HTR methods. Methods in bold text are proposed by this thesis. Corresponding contributions are highlighted in brackets.

Idea	Method
Architecture	HMMs [52, 19, 12] RNN+CTC [112, 123, 86, 152, 133] Nearest Neighbor Search [2, 85, 125] **Seq2Seq with Attention** [17, 134, 79] (contri. 1 & 2) **Transformers** [75] (contri. 3)
Boosting	**Synthetic Data of TrueType Fonts** [71, 1, 89, 76] (contri. 4) **Synthetic Data using GANs** [77] (contri. 6) **Writer Adaptation** [160, 78] (contri. 4) **Disentanglement** [74] (contri. 5)

language knowledge as a new input to the recognizer. Moreover, there are two improvements for the state-of-the-art language modeling methods: 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, both our proposed Seq2Seq recognizer and language modeling method achieve the state-of-the-art performance.

In Chapter 3, we propose a non-recurrent approach to recognize handwritten text by the use of transformer models. We propose a novel method that bypasses any recurrence. By using multi-head self-attention layers both at the visual and textual stages, we are able to tackle character recognition as well as to learn language-related dependencies of the character sequences to be decoded. Our model is unconstrained to any predefined vocabulary, being able to recognize out-of-vocabulary words, i.e. words that do not appear in the training vocabulary. We significantly advance over prior art and demonstrate that satisfactory recognition accuracies are yielded even in few-shot learning scenarios.

In Chapter 4, we propose an unsupervised writer adaptation approach that is able to automatically adjust a generic handwritten word recognizer, fully trained with synthetic fonts, towards a new incoming writer. We have experimentally validated our proposal using five different datasets, covering several challenges (i) the document source: modern and historic samples, which may involve paper degradation problems; (ii) different handwriting styles: single and multiple writer collections; and (iii) language, which involves different character combinations. Across these challenging collections, we show that our system is able to maintain its performance, thus, it provides a practical and generic approach to deal with new document collections without requiring any expensive and tedious manual

annotation step.

In Chapter 5, We propose a novel method that is able to produce credible handwritten text images by conditioning the generative process with both calligraphic style features and textual content. Our generator is guided by three complementary learning objectives: to produce realistic images, to imitate a certain handwriting style and to convey a specific textual content. Our model is unconstrained to any predefined vocabulary, being able to render whatever input word. Given a sample writer, it is also able to mimic its calligraphic features in a few-shot setup. We significantly advance over prior art and demonstrate with qualitative, quantitative and human-based evaluations the realistic aspect of our synthetically produced images. Extensive experiments have been done on making use of the generated samples to boost HTR performance. Thus, both qualitative and quantitative results have been carried out to demonstrate the advance among the state-of-the-art approaches.

In Chapter 6, we summarize the main contributions of this thesis and outline some possible future research lines on HTR tasks.

Chapter 2

Sequence-to-Sequence Approach for HTR

2.1 Introduction

The recognition of handwritten text was, in fact, one of the first application scenarios of convolutional neural networks, when LeCun *et al.* proposed in the late nineties such architectures [93] for recognizing handwritten digits from the MNIST dataset. In the literature, several other methods have been proposed for tackling the HTR task such as Hidden Markov Models (HMM) [12, 19, 52], Recurrent Neural Networks (RNN) and Connectionist Temporal Classification (CTC) [133, 152, 86, 123, 112], or nearest neighbor search methods in embedding spaces [125, 85, 2]. These methods have started to reach a satisfying performance in some specific and restricted use cases, but we are still far away from having a generic and robust system that is able to read any handwritten text.

Inspired in the latest advances in machine translation [135, 7], image captioning [155] or speech recognition [27, 8], we believe that sequence-to-sequence models backed with attention mechanisms [17, 134] present a significant potential to tackle HTR tasks. Recurrent architectures suit the temporal nature of text, written usually from left to right, and attention mechanisms have proven to be quite performant when paired with such recurrent architectures to focus on the right features at each time step. Sequence-to-Sequence (Seq2Seq) models follow an encoder-decoder paradigm. In our case, the encoder part consists of a Convolutional Neural Network (CNN) that extracts low-level features from the written glyphs, that are then sequentially encoded by an Recurrent Neural Network (RNN). The decoder is another RNN that will decode one character at each time step, thus spelling the whole text. An attention mechanism is introduced as a bridge between the encoder and the decoder, in order to provide a high-correlated

context vector that focuses on each character's feature at each decoding time step.

In the HTR tasks, text always follows a particular set of syntactic rules and presents a well defined morphological structure. Text recognition systems often integrate statistical language models [102, 87, 43] that are able to complement the optical part boosting the overall recognition performance. Language models for HTR tasks implemented as probability distributions over sequences of characters and words, aim to provide context to discern between sequences of characters that 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 [22] to neural network architectures [159].

However, in most of the state-of-the-art handwritten word recognition systems, including the recent Seq2Seq-based ones [134, 79], 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 corpus 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 [68, 143], or as an integrated guiding module, steering the decoding process towards the best fitting letter succession [61].

In this chapter, we present a novel Seq2Seq-based recognizer and a novel method of integrating an external language model for HTR. 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 corrects them. 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 2.11 the different transcription results that we are able to obtain with the proposed architecture.

2.2 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) [12, 19, 44, 52]. 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) [96] have started to become popular to deal with sequential data such as handwriting. For example, Bidirectional Long Short-Term Memory (BLSTM) [58] or Multidimensional Long Short-Term Memory (MDLSTM) [59] have been widely adopted by handwritten word recog-

nition community. Lately, these models have been discussed and improved. For example, Puigcerver [126] 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.* [141] 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 [57]. 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 CTC-based models need to be carefully designed base on the specific dataset, otherwise, a short handwritten image (e.g. a single punctuation mark) might end up with a even shorter sequence of features 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 as shown in Section 2.5.4.

Recently, inspired by machine translation [7, 135], speech recognition [8, 27] and image captioning [155], the sequence-to-sequence architecture [134, 79] 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.* [134] provided a sequence-to-sequence based handwriting recognizer, but they imposed a manually set sliding-window. We have analyzed various strategies to find a proper sequence-to-sequence based architecture for specifically targeting HTR tasks.

With the usage of RNNs, an implicit language model has been proved to help the recognition process in [129]. However, this internal language model is overfitted on the text of the training dataset. Among the popular handwritten dataset, 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 [102, 122, 147]. 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 [16, 113, 20, 98, 148, 126] have integrated character n-grams language

modelling into a RNN-CTC based handwriting recognizer. Jelinek *et al.* [72] provided a cache trigram language model, which can be adapted to the current document more closely. Della *et al.* [31] 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 [33], pretrained 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.* [162] propose a method to incorporate Bert into machine translation architecture, which is a good trial to utilize a powerful pretrained 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, it cannot be flexibly injected to the recognizer at each time step, because there is no recurrency in Bert-like LM that gains speed but loses flexibility.

Concurrently, Recurrent Neural Network Language Models (RNNLM) [110, 61, 68, 143] 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.* [61] 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.

2.3 Getting Enough Training Data

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 2.1 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 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.

Figure 2.1: 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.

2.3.1 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 [131], 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 2.1.

Figure 2.2 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 overfitting and leads to models that are able to generalize better than 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.

Table 2.1: Comparison of the data augmentation techniques among state-of-the-arts.

Methods	Dutta *et al.* [42]	Yousef *et al.* [157]	Proposed
Blurring/sharpening	–	–	✓
Elastic transformation	✓	✓	✓
Shear	✓	✓	✓
Rotation	✓	–	✓
Translation and scaling	✓	✓	✓
Gamma correction	–	–	✓
Blending with background	–	–	✓
Sign flipping	–	✓	–

Figure 2.2: 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.

2.3.2 Pre-training with Synthetic Data

Recently, it has become a common trend the use of synthetically generated images to magnify the training data volume [38]. 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 [88], 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 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 2.3.

Figure 2.3: 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.

2.4 Seq2seq Model with Attention Mechanism

Our attention-based seq2seq model consists of three main parts: an encoder, an attention mechanism and a decoder. Figure 2.4 shows the whole architecture proposed in this work. Let us detail each of the different parts.

2.4.1 Encoder

We start with a CNN to extract visual features. Since we believe that handwritten text images are not visually as complex as real world images, we choose a reasonable CNN architecture such as the VGG-19-BN [132] and initialize it with the pre-trained weights from ImageNet.

Then we introduce a multi-layered Bi-directional Gated Recurrent Unit (BGRU) which will involve mutual information and extra positional information for each

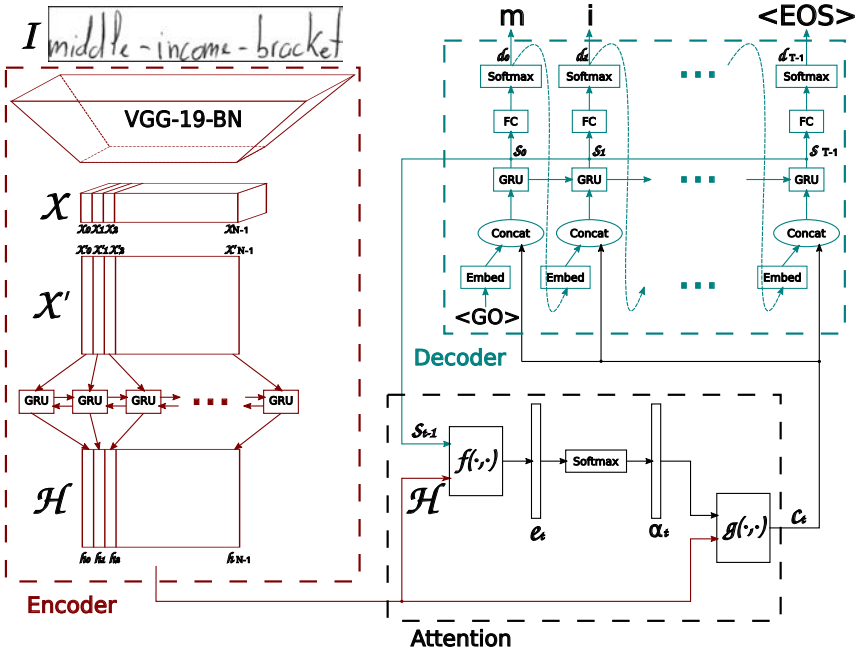


Figure 2.4: Architecture of the seq2seq model with attention mechanism.

column, and will encode the sequential nature of handwritten text. Considering the extra positional information, a simple positional encoding [146] is also another popular technique. In the experiments, we will test on the two methods for incorporating the extra positional information. So we use VGG+BGRU as an encoder to transfer the image $\mathcal{I}$ into an intermediate-level feature $\mathcal{X}$, which then is reshaped into a two-dimensional feature map $\mathcal{X}'$. The feature map $\mathcal{X}'$ can be referred as a sequence of column feature vectors $(x'_0, x'_1, \ldots, x'_{N-1})$, where N is the width of the feature map. $\mathcal{H}$ is the output of encoder which shares the same width of $\mathcal{X}'$. Each element $h_i \in \mathcal{H}$ is the output of BGRU at each time step, which will be further used to calculate attention.

2.4.2 Attention Mechanism

In this section we will discuss two main attention mechanisms, content-based attention and location-based attention.

Content-based Attention

The basic attention mechanism is content-based attention [7]. The intuition is to find the similarity between the current hidden state of the decoder and the

word image representation feature map, thus we can find the most correlated feature vectors in the feature map of the encoder, which can be used to predict the current character at the current time step. Let us define α_t as the attention mask vector at time step t, h_i as the hidden state of the encoder at the current time step $i \in \{0, 1, \ldots, N\text{-}1\}$, s_t as the hidden state of decoder at current time step $t \in \{0, 1, \ldots, T\text{-}1\}$, where T is the maximum length of decoding characters. Then,

$$\alpha_t = \text{Softmax}(e_t) \tag{2.1}$$

where

$$e_{t,i} = f(h_i, s_{t-1}) = w^T \tanh(Wh_i + Vs_{t-1} + b) \tag{2.2}$$

where w, W, V and b are trainable parameters. After obtaining the attention mask vector, the most relevant context vector can be calculated as:

$$c_t = g(\alpha_t, H) = \sum_{i=0}^{N\text{-}1} \alpha_{ti} h_i \tag{2.3}$$

Location-based Attention

The main disadvantage of content-based attention is that it expects positional information to be encoded in the extracted features. Hence, the encoder is forced to add this information, otherwise, content-based attention will never detect the difference between multiple feature representations of same character in different positions. To overcome it, we use an attention mechanism that takes into account the location information explicitly, *i.e.* location-based attention [27]. Thus, the content-based has been extended to be location-aware by making it take into account the alignment produced at the previous step. First we extract k vectors $l_{t,i} \in \mathbb{R}^k$ for every position i of the previous alignment α_{t-1} by convolving it with a matrix $F \in \mathbb{R}^{k \times r}$:

$$l_t = F * \alpha_{t-1} \tag{2.4}$$

And then, we replace Equation 2.2 by:

$$e_{t,i} = f'(h_i, s_{t-1}, l_t) = w^T \tanh(Wh_i + Vs_{t-1} + Ul_{t,i} + b) \tag{2.5}$$

where w, W, V, U and b are trainable parameters.

Attention Smoothing

In practice, the attended area is a little narrower than the target character area of the word image. Consequently, we can infer that the model can already get the correct prediction only focusing at the narrow area. However, from the viewpoint

of humans, a little wider covering area of the target character would be beneficial. For this reason, we propose to replace the Softmax Equation 2.1 with the logistic sigmoid σ proposed by [27]:

$$\alpha_{t,i} = \frac{\sigma(e_{t,i})}{\sum_{i=0}^{N} \sigma(e_{t,i})} \tag{2.6}$$

2.4.3 Decoder

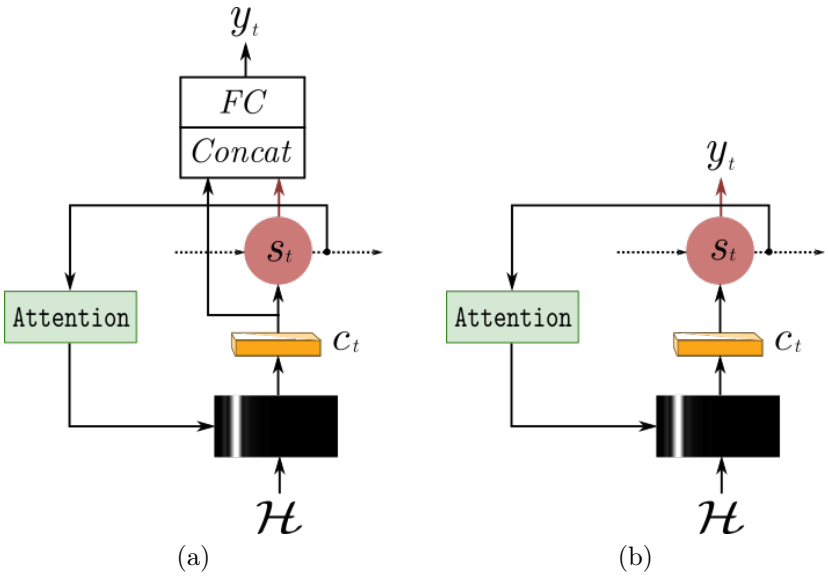


Figure 2.5: Architecture of the conventional decoder and our simplified decoder are shown in (a) and (b), respectively.

The decoder is a one-directional multi-layered GRUs. During each time step t, the concatenation of the embedding vector of the previous time step $\tilde{y}_{t-1}$ and the context vector c_t will be fed into the current GRU unit. The embedding vector for each character in the dataset's vocabulary comes from a look-up table matrix, which is randomly initialized and updated during the training process. 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 2.5. The experimental comparison between both two architectures will be detailed in Section 2.5.3. Thus, the prediction of each time step t is:

$$y_t = \arg\max(\omega(s_t)) \tag{2.7}$$

where $\omega(\cdot)$ is a linear layer. Then we use the index to fetch the corresponding embedding vector $\tilde{y}_t$ from the look-up table matrix:

$$\tilde{y}_t = \text{Embedding}(y_t) \tag{2.8}$$

The decoder always starts with the start signal $\langle GO \rangle$ as first input character and ends the decoding process when the end signal $\langle EOS \rangle$ occurs or until the maximum time step T.

The previous embedding vector and current context vector are concatenated to obtain s_t, the hidden state of decoder at current time step. Thus, at each time step of the decoding, the decoder GRU can take advantage of both the information of the previous character and the potentially most relevant visual features, which will benefit the model to make correct predictions. So,

$$s_t = \text{Decoder}([\tilde{y}_{t-1}, c_t], s_{t-1}) \tag{2.9}$$

where $[\cdot, \cdot]$ is the concatenation of two vectors. There are two techniques that we can adopt to improve the decoding process: multi-nomial decoding and label smoothing.

Multi-nomial Decoding

Inspired by [25], during the training process, instead of choosing the character that has the highest probability from the Softmax output d_t at time step t, multiple indices can be sampled from the multi-nomial probability distribution located in the Softmax output d_t. But to keep the model simple, here we sample only one index but in a random way based on the multi-nomial probability distribution, and this index corresponds to a specific character. Although only one index has been sampled, it allows the decoder to explore other alternative decoding paths towards the final word prediction, which could make the decoder more robust and lead to better performance, although it will absolutely take longer epochs to train.

Label Smoothing

Label smoothing [136] is a regularization mechanism to prevent the model from making over-confident predictions. It encourages the model to have higher entropy at its prediction, and therefore it makes the model more adaptable and improve generalization. We regularize the groundtruth by replacing the hard 0 and 1 classification targets with targets of $\frac{\varepsilon}{k}$ and $1 - \frac{k-1}{k}\varepsilon$. We choose the $\varepsilon = 0.4$.

2.4.4 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 [61], shown in Figure 2.6(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:

$$\hat{s}_t = \text{Decoder}([c_t, \tilde{y}_{t-1}, p_{t-1}^{lm}], \hat{s}_{t-1}) \tag{2.10}$$

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 2.5.3.

The difference between Equations 2.9 and 2.10 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 2.6 (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 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 2.6 (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 2.6 (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 pretrained on an external text corpus, and then plugged in the recognizer for a joint fine-tuning on the handwritten dataset. In the second 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 $y_{t-1}^l m$ 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 2.6 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 2.5 we will show the performance improvement on popular datasets.

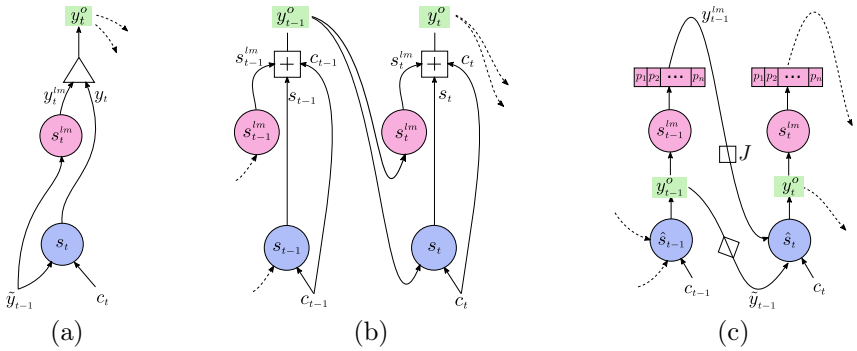


Figure 2.6: Architecture of the language models: (a) Shallow Fusion, (b) Deep Fusion, and (c) our proposed Candidate Fusion. The blue circles represent the hidden states of the decoder, the red circles represent the hidden states of the language model, and the green boxes are the final predictions at 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.

2.5 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.

2.5.1 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 2.2.

The standard Character Error Rate (CER) and Word Error Rate (WER) met-

rics are utilized to evaluate the system's performance. Formally,

$$CER = \frac{S + I + D}{N} \tag{2.11}$$

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.

$$WER = \frac{S_w + I_w + D_w}{N_w} \tag{2.12}$$

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. In the following experiments, the CER and WER metrics will range from [0-100]. Thus, a lower value indicates a better performance.

Table 2.2: Overview of the different datasets used in this work depicting its characteristics.

Dataset	Words	Writers	Period	Language
IAM [103]	115,320	657	Modern	English
GW [92]	4,860	1	Historical	English
Rimes [5]	66,978	1,300	Modern	French
CarCrash	24,492	640	Modern	German

2.5.2 Implementation Details

All these experiments were run using PyTorch [120] 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 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 [136] as a regularization mechanism.

All the images have been resized to a fixed height of 64 pixels while keeping the original ratio of the length/height. With the fixed height size of 64 pixels, we pad zeros to the right of every text image so as to meet the maximum length of images, 1011 and 2160 pixels, for word and text-line image respectively.

2.5.3 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

Table 2.3: 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

validation set of the IAM. The only one exception is Table 3.6, which is applied on the GW dataset.

Firstly, different popular CNN models have been evaluated in Table 2.3. Given that the VGG19-BN model obtains the best results, we have chosen it as the feature extractor in our architecture.

Secondly, we need to find out relatively perfect parameters for sizes of hidden state and hidden layers of both encoder and decoder. As the hidden state of the decoder should be initialized by the encoder, we always keep the size of the hidden state and the number of hidden layers the same for both the encoder and decoder. We tried 1, 2 and 3 layers, 128, 256, 512 and 1024 sizes, being a total of 12 experiments. From the results shown in Table 2.4, we can observe that the relatively best parameters are 2 layers and 512 size for both the encoder and decoder.

Table 2.4: Validation CER comparison changing the size of the hidden state and number of layers.

Size	Number of Layers		
	1	2	3
128	5.57	6.07	6.09
256	5.13	5.33	5.69
512	5.05	**5.01**	5.34
1024	5.19	5.03	5.10

Thirdly, we compare two different architectures of the encoder, as explained in Section 2.4.1. The CNN+BGRU architecture obtains better results than when using a CNN with positional encoding, as shown in the Table 2.5, because a trainable BGRU can provide not only the positional information, but also better mutual information among all the feature vector $\mathcal{H}$.

Fourthly, we compare our proposed decoder unit with the conventional decoder

Table 2.5: Comparison between positional encoding [146] and BGRU on IAM validation set.

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

unit explained in Section 2.4.3. From Table 2.6 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 2.6: 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**

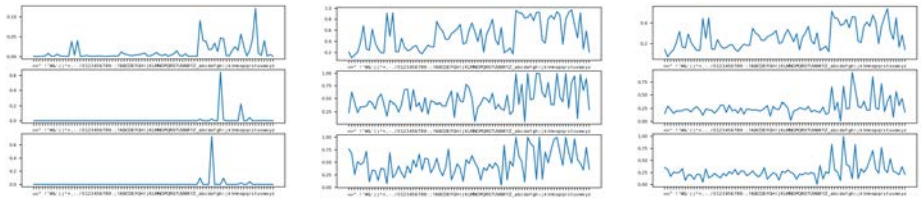
Table 2.7: Ablation study for the proposed model tested on the IAM dataset, character error rates are computed from validation set.

Attention	AttnSmooth	Multinomial	LabelSmooth	Valid-CER	Valid-WER
Content	–	–	–	5.79	15.91
	–	–	✓	5.08	13.88
Location	–	–	–	5.49	14.74
	–	–	✓	**5.01**	**13.61**
	–	✓	–	5.53	14.53
	–	✓	✓	5.03	13.66
	✓	–	–	5.72	15.92
	✓	–	✓	5.34	14.62
	✓	✓	–	5.84	15.85
	✓	✓	✓	5.56	14.85

As detailed in Section 2.4, we explored some techniques for potential improvements. Table 2.7, shows that the best performance was achieved using location-based attention and label smoothing. Studying the table, we can see that the label smoothing is really helpful. The location-based attention is just slightly better than the content-based one. The reason behind this little improvement is that the use of the BGRU in the encoder can already encode some positional information to the feature map. Contrary, once we encode the positional information explicitly, the result improves. In conclusion, the location-based attention still meets our expectation.

Concerning attention smoothing and multi-nomial decoding, they seem not helping our model. On the one hand, the original Softmax attention is already good (attention visualization can be found in Figure 2.8, 2.9 and 2.10), therefore smoothing the attention may introduce noise, which could harm the model. On the other hand, multi-nomial decoding enables the proposed approach to explore new decoding paths. This exploration was expected to make our model more robust, however, it has showed that this technique is still not able to outperform our best result in the table. This probably means that the multi-nomial decoding really makes our model harder to train.

Finally, to make the best of a language model, we have investigated which is the best way to inject the linguistic knowledge. In Table 2.8, we have applied different activation functions on the output of the language model using the GW dataset. The best performance has been achieved without the usage of activation 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 2.7. The softmax approach, as shown in Figure 2.7(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 2.7(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 2.7(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.



(a) (b) (c)

Figure 2.7: 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 2.8: 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**

2.5.4 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 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.

Baseline Model

Table 2.9 shows the most popular approaches on the IAM word-level dataset, however, most of them have applied different pre-processings on the original dataset. For HMM-based approaches, Gimenez *et al.* [52] corrected the slant in the image and made the gray level normalization. Bluche *et al.* [19] also corrected the slant in the image, enhanced the image contrast and added 20 white pixels on left and right to model the empty context. Bianne *et al.* [12] trained the model using all training and validation sets. These approaches have already been outperformed, since the RNN- and nearest neighbor- based approaches perform pretty well. In the case of RNN-based approaches, Mor *et al.* [112] filtered out punctuation and short words, and trained the model using training and validation sets. Krishnan *et al.* [86] has been pre-trained using synthetic data. Wiginton *et al.* [152] cleaned the punctuation and upper-cases, used the profile normalization and applied test augmentation.

Since the nearest neighbor-based approaches cannot work without lexicons, they cannot be widely used in daily or industrial use cases. In addition, Krishnan *et al.* [85] has also pre-trained using synthetic data, cleaned punctuation and upper-cases and applied test augmentation. Poznanski *et al.* [125] used a pre-trained model from synthetic data and applied test augmentation.

The bottom rows of Table 2.9 correspond to attention-based approaches, which

Table 2.9: Comparison with the state-of-the-art methods on IAM dataset of word-level.

Idea	Method	Lexicon[a]	LM	Proc.	Pre-train	CER	WER
HMMs	Gimenez *et al.* [52]	tr+va+te	✓	✓	–	–	25.80
	Bluche *et al.* [19]	te	✓	✓	–	–	23.70
	Bianne *et al.* [12]	tr+va+te	–	–	–	–	21.90
RNN + CTC	Mor *et al.* [112]	–	–	–	–	–	20.49
	Pham *et al.* [123]	–	–	–	–	13.92	31.48
	Krishnan *et al.* [86]	–	✓	–	Syn.	6.34	16.19
	Wiginton *et al.* [152]	–	–	✓	–	6.07	19.07
	Stunner *et al.* [133]	2.4M	✓	–	–	**4.77**	**13.30**
Nearest Neighbor Search	Almazan *et al.* [2]	te	–	–	–	11.27	20.01
	Krishnan *et al.* [85]	te+90K	–	–	Syn.	6.33	14.07
	Poznanski *et al.* [125]	tr+te	✓	✓	Syn.	**3.44**	**6.45**
Attention	Bluche *et al.* [17]	–	–	–	CTC	12.60	–
	Sueiras *et al.* [134]	–	–	✓	–	8.80	23.80
	Ours	–	–	–	–	**6.88**	**17.45**

[a]Vocabulary of all words occurring in training (tr), validation (va) and test set (te). 2.4 million (2.4M) and 90 thousand (90K) words lexicon.

are relatively new for handwriting recognition and have a significant potential for development. But Bluche *et al.* [17] has been pre-trained using CTC loss in order to get meaningful feature representation. Sueiras *et al.* [134] corrected the line skew and the slant in the images, normalized the height of the characters based on baseline and corpus line.

Among all those approaches, some of them have utilized language model (LM) explicitly. Even though no language model is used in our system, the RNN of the decoder might learn the relations between characters in the training vocabulary.

In summary, we can observe that our results are the best among the attention-based approaches and comparable to other state-of-the-art approaches especially with neither dataset pre-processing, model pre-training on synthetic dataset nor using CTC loss.

To further boost the HTR performance, we make use of the synthetic data to pre-train our recognizer as shown in Table 2.10. From the table, we observe that our recognizer achieves good performance on the IAM, GW and Rimes datasets. By introducing the synthetic pre-training, the performance is boosted from 6.88% to 5.79% for CER. We show some examples of the visualized attention maps on the IAM (Figure 2.8), GW (Figure 2.9) and Rimes (Figure 2.10). 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 2.10: 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.* [112]	–	20.49	–	–	–	11.95
Pham *et al.* [123]	13.92	31.48	–	–	8.62	27.01
Bluche *et al.* [17]	12.60	–	–	–	–	–
Wiginton *et al.* [152]	6.07	19.07	–	–	3.09	11.29
Sueiras *et al.* [134]	8.80	23.80	–	–	4.80	15.90
Kang *et al.* [79]	6.88	17.45	–	–	–	–
Krishnan *et al.* [87]	6.34	16.19	–	–	–	–
Toledo *et al.* [141]	–	–	7.32	–	–	–
Dutta *et al.* [42][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.

Figure 2.8: Visualization samples of attention on the IAM dataset.

Integration of the Language Model

In this subsection we evaluate the performance of our language model by expanding the baseline model shown in Table 2.10. The results are shown in Table 2.11. Our language model is pre-trained with a large corpus, detailed in Section 2.3.2, and fine-tuned with the training data within the whole sequence-to-sequence system during end-to-end training process. Thus, the language 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

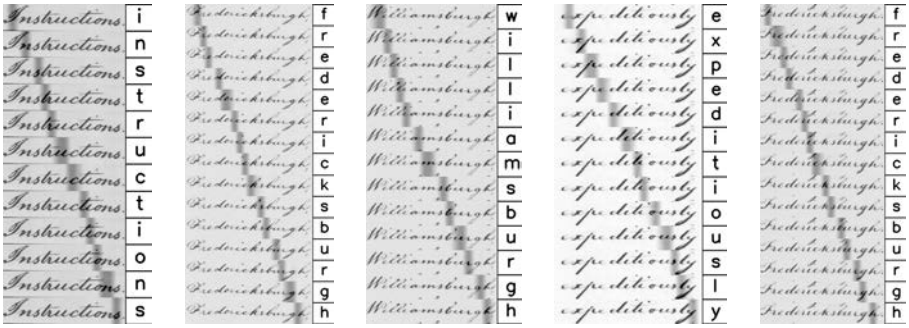


Figure 2.9: Visualization samples of attention on the GW dataset.

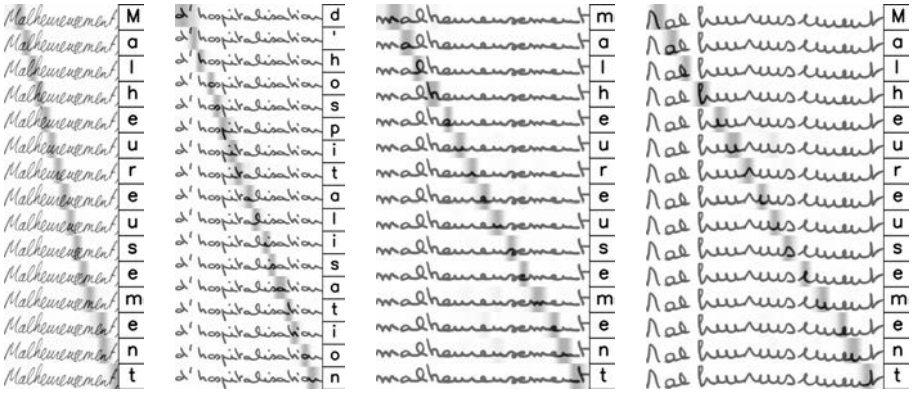
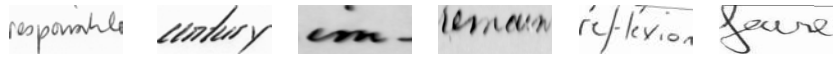


Figure 2.10: Visualization samples of attention on the Rimes dataset.



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

Figure 2.11: 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.

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.

Table 2.11: 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.

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

Table 2.12: 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**

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 2.13. The performance of joining the candidate fusion LM is proved to be statistically significant with threshold P-value 0.05. Even though the results on text-line Rimes data are acceptable and the Candidate Fusion LM shows a great boost on the recognition performance, the CERs are not as good as those in word level. We do believe that this is the limitation of Seq2Seq method, whose main problem is the gradient vanishing for long sequences such as text-line texts. This problem motivates us to propose a non-recurrent method for HTR in next Chapter.

Table 2.13: Results at text-line level on the Rimes dataset. Both of the methods are pre-trained with French synthetic data.

Method	LM	CER	WER
Seq2Seq	–	8.33	25.31
Seq2Seq	Candidate Fusion	**6.87**	**21.14**

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 2.12. 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 a well-known CTC-based approach [126], our proposed approach achieves better performance, as shown in Table 2.14. This results suggests that our model has a good generalization ability.

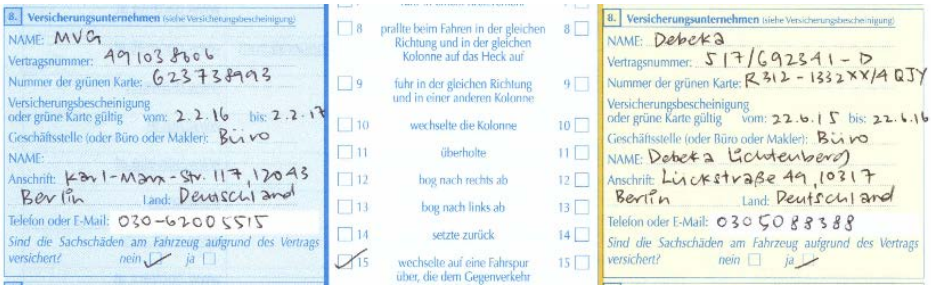


Figure 2.12: A cropped area of the real industrial use case dataset.

Table 2.14: Results of a real use case.

Method	CER	WER
CTC-based [126]	5.6	7.4
Proposed	**3.7**	**4.5**

2.6 Conclusion

In this chapter, we have presented an attention-based Seq2Seq model and a novel way to integrate an external language model into the Seq2Seq model for HTR. The extensive evaluation, including an ablation study as well as comparisons with state-of-the-art approaches, demonstrates the effectiveness of our approach. Indeed it not only outperforms the existing approaches on public scientific datasets, but it also shows its robustness on a real industrial use case.

However, this Seq2Seq-based method has gradient vanishing problem especially when dealing with long sequences because of the recurrent nets architecture. In addition, this recurrency hinders the training speed without the full use of parallel computing.

Chapter 3

Transformer-based approach

3.1 Introduction

As introduced in the previous Chapter, we know that Seq2Seq approaches, conformed by encoder-decoder networks led by attention mechanisms, have become the state-of-the-art methods for HTR [109]. These methods are not only a good fit to process images sequentially, but also have, in principle, the inherent power of language modelling, *i.e.* to learn which character is more probable to be found after another in their respective decoding steps. Nonetheless, this ability of language modelling has proven to be limited, since recognition performances are in most cases still enhanced when using a separate statistical language model as a post-processing step [138].

Despite the fact that attention-based encoder-decoder architectures have started to be used for HTR with impressive results, one major drawback still remains. In all of those cases, such attention mechanisms are still used in conjunction with a recurrent network, either BLSTMs or Gated Recurrent Unit (GRU) networks. The use of such sequential processing deters parallelization at training stage, and severely affects the effectiveness when processing longer sequence lengths by imposing substantial memory limitations.

Motivated by the above observations, we propose in this chapter a non-recurrent method to replace Seq2Seq one for HTR tasks. We got inspiration from Vaswani *et al.*, who proposed in [146] the seminal work on the Transformer architecture. Transformers rely entirely on attention mechanisms, relinquishing any recurrent designs. Stimulated by such advantage, we propose to address the HTR problem by an architecture based on transformers, which dispenses of any recurrent network. By using multi-head self-attention layers both at the visual and textual stages, we aim to tackle both the proper step of character recognition from images, as well as to learn language-related dependencies of the character sequences to be

decoded.

The main contributions of our work are summarized as follows. ***i***) For the first time, we explore the use of transformers for the HTR task, bypassing any recurrent architecture. We attempt to learn, with a single unified architecture, to recognize character sequences from images as well as to model language, providing context to distinguish between characters or words that might look similar. The proposed architecture works at character level, waiving the use of predefined lexicons. ***ii***) By using a pre-training step using synthetic data, the proposed approach is able to yield competitive results with a limited amount of real annotated training data. ***iii***) Extensive ablation and comparative experiments are conducted in order to validate the effectiveness of our approach. Our proposed HTR system achieves new state-of-the-art performance on the public IAM dataset.

3.2 Related Work

Vaswani *et al.* presented in [146] the Transformer architecture. Their proposal relies entirely on the use of attention mechanisms, avoiding any recurrent steps. Since the original publication, the use of transformers has been popularized in many different computer vision and natural language processing tasks such as automatic translation [33] or speech-to-text applications [37]. Its use has started to eclipse recurrent architectures such as BLSTMs or GRUs for such tasks, both by being more parallelizable, facilitating training, and by having the ability to learn powerful language modelling rules of the symbol sequences to be decoded.

However, to the best of our knowledge, the transformer architecture has not yet been used to tackle the handwriting recognition problem. It has been nonetheless used lately to recognize text in natural scenes [99]. In such works, the original transformers architecture, often applied to one-dimensional signals (*i.e.* text, speech, etc.), has been adapted to tackle two-dimensional input images. Image features are extracted by the use of CNNs [130], two-dimensional positional encodings [94, 14] or additional segmentation modules [9] help the system locate textual information amidst background clutter. However, all such works present some limitations when dealing with handwritten text lines. On the one hand, all such architectures work with fixed image size whereas for handwriting recognition we have to face variable length inputs. On the other hand, they work at individual word level, whereas in handwriting recognition we have to face much longer sequences. Finally, despite also having its own great variability, scene text is often much legible than cursive handwriting, since in most of the cases words are formed by individual block letters, which, in turn, are easier to synthesize to obtain large training volumes.

Summarizing, state-of-the-art handwriting recognition based on deep recurrent networks have started to reach decent recognition results, but are too computationally demanding at training stage. Moreover, albeit they shall have the ability

to model language-specific dependencies, they usually fall short of inferring adequate language models and need further post-processing steps. In this chapter we propose, for the first time, the use of transformers for the HTR task, bypassing any recurrent architecture. A single unified architecture, both recognizes long character sequences from images as well as models language at character level, waiving the use of predefined lexicons.

3.3 Proposed Method

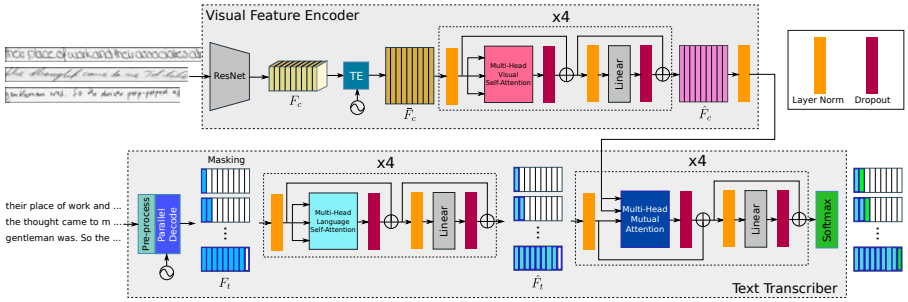


Figure 3.1: Overview of the architecture of the proposed method. Text-line images are processed by a CNN feature extractor followed by a set of multi-headed self-attention layers. During training, the character-wise embeddings of the transcriptions are also encoded by self-attentions and a final mutual attention module aims to align both sources of information to decode the text-line character by character.

3.3.1 Problem Formulation

Let $\{\mathcal{X}, \mathcal{Y}\}$ be a handwritten text dataset, containing images $\mathcal{X}$ of handwritten textlines, and their corresponding transcription strings $\mathcal{Y}$. The alphabet defining all the possible characters of $\mathcal{Y}$ (letters, digits, punctuation signs, white spaces, etc.), is denoted as $\mathcal{A}$. Given pairs of images $x_i \in \mathcal{X}$ and their corresponding strings $y_i \in \mathcal{Y}$, the proposed recognizer has the ability to combine both sources of information, learning both to interpret visual information and to model language-specific rules.

The proposed method's architecture is shown in Figure 3.1. It consists of two main parts. On the one hand a visual feature encoder aimed at extracting the relevant features from text-line images and at focusing its attention at the different character locations. Subsequently, the text transcriber is devoted to output the decoded characters by mutually attending both at the visual features as well as the language-related features. The whole system is trained in an end-to-end fashion, learning both to decipher handwritten images as well as modelling language.

3.3.2 Visual Feature Encoder

The role of the visual feature encoder is to extract high-level feature representations from an input handwritten image $x \in \mathcal{X}$. It will encode both visual content as well as sequential order information. This module is composed by the following three parts.

CNN Feature Encoder

Input images x of handwritten text-lines, which might have arbitrary lengths, are first processed by a Convolutional Neural Network. We obtain an intermediate visual feature representation F_c of size f. We use the ResNet50 [66] as our backbone convolutional architecture. Such visual feature representation has a contextualized global view of the whole input image while remaining compact.

Temporal Encoding

Handwritten text images are sequential signals in nature, to be read in order from left to right in Latin scripts. The temporal encoding steps are aimed to leverage and encode such important information bypassing any recurrency.

In a first step, the three-dimensional feature F_c is reshaped into a two-dimensional feature by keeping its width, *i.e.* obtaining a feature shape $(f \times h, w)$. This feature map is later fed into a fully connected layer in order to reduce $f \times h$ back to f. The obtained feature $F_c^{'}$, with the shape of (f, w), can be seen as a w-length sequence of visual vectors.

However, we desire that the same character appearing at different positions of the image has different feature representations, so that the attention mechanisms are effectively and unequivocally guided. That is, we want that the visual vectors $F_c^{'}$ loose their horizontal shift invariance. Following the proposal from Vaswani *et al.* [146], a one-dimensional positional encoding using sine and cosine functions is applied.

$$\begin{aligned} TE(pos, 2i) &= \sin\left(\frac{pos}{10000^{2i/f}}\right) \\ TE(pos, 2i+1) &= \cos\left(\frac{pos}{10000^{2i/f}}\right), \end{aligned} \tag{3.1}$$

where $pos \in \{0, 1, 2, \ldots, w-1\}$ and $i \in \{0, 1, 2, \ldots, f-1\}$.

$F_c^{'}$ and TE, sharing the same shape are added along the width axis. A final fully connected layer produces an abscissa-sensitive visual feature $\bar{F}_c$ with shape (f, w).

Visual Self-Attention Module

To further distill the visual features, self-attention modules are applied four times upon $\bar{F}_c$. The multi-head attention mechanism from [146] is applied using eight heads. This attention module takes three inputs, namely the query Q_c, key K_c and value V_c, where $Q_c = K_c = V_c = \bar{F}_c$. The correlation information is obtained by:

$$\hat{v}_c^i = \text{Softmax}\left(\frac{q_c^i K_c}{\sqrt{f}}\right) V_c, \tag{3.2}$$

where $q_c^i \in Q_c$ and $i \in \{0, 1, \ldots, w-1\}$. The final high-level visual feature is $\hat{F}_c = \{\hat{v}_c^0, \hat{v}_c^1, \ldots, \hat{v}_c^{w-1}\}$.

3.3.3 Text Transcriber

The text transcriber is the second part of the proposed method. It is in charge of outputting the decoded characters, attending to both the visual features as well as the language-specific knowledge learnt form the textual features. It is worth to note that unlike translation of speech-to-text transformer architectures, our text transcriber works at character level instead of word-level. It will thus learn n-gram like knowledge from the transcriptions, *i.e.* predicting the next most probable character after a sequence of decoded characters. The text transcriber consists of three steps, the text encoding, the language self-attention step and the mutual-attention module.

Text Encoding

Besides the different characters considered in alphabet $\mathcal{A}$, we require some symbols without textual content for the correct processing of the text-line string. Special character $\langle S \rangle$ denotes the start of the sequence, $\langle E \rangle$ the end of the sequence, and $\langle P \rangle$ is used for padding. The transcriptions $y \in \mathcal{Y}$ are extended to a maximum length of N characters in the prediction.

A character-level embedding is performed by means of a fully-connected layer that maps each character from the input string to an f-dimensional vector. The same temporal encoding introduced in eq. 3.1 is used here to obtain

$$F_t = \text{Embedding}\,(y) + TE, \tag{3.3}$$

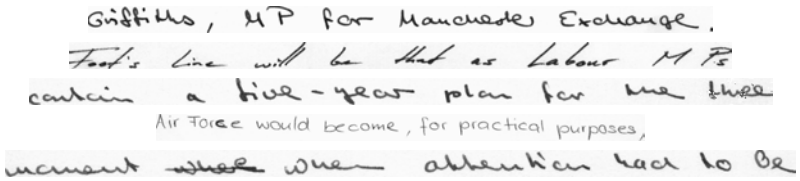
where F_t has the shape of (f, N).

In the decoding step of recurrent-based HTR approaches [79, 109] every decoded character is iteratively fed again to the decoder, to predict the next character, thus inhibiting its parallelization. Contrary, in the transformer paradigm,

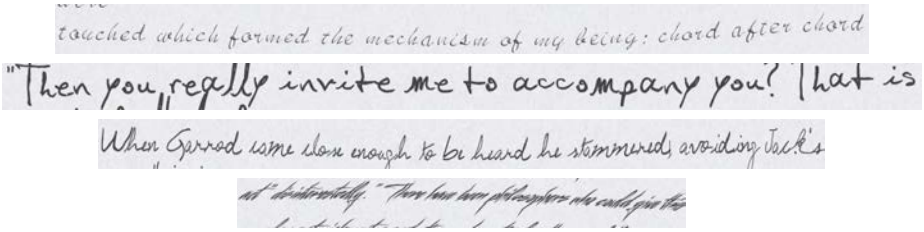
all possible decoding steps are fed concurrently at once with a masking operation [146]. To decode the j-th character from y, all characters at positions greater than j are masked so that the decoding only depends on predictions produced prior to j. Such a parallel processing of what used to be different time steps in recurrent approaches drastically reduces training time.

Language Self-attention Module

This module follows the same architecture as in Section 3.3.2 and aims to further distill the text information and learn language-specific properties. $\hat{F}_t$ is obtained after the self-attention module implicitly delivers n-gram-like features, since to decode the j-th character from y all character features prior to j are visible.



a) Real data from IAM dataset



b) Synthetically rendered text-lines with truetype fonts

Figure 3.2: Examples of real and synthetic training handwritten text-line images.

Mutual-attention Module

A final mutual self-attention step is devoted to align and combine the learned features form the images as well as from the text strings. We follow again the same architecture from Section 3.3.2, but now the query Q_t comes from the textual representation $\hat{F}_t$ while the key K_c and value V_c are fed with the visual representations $\hat{F}_c$

$$\hat{v}_{ct}^{i} = \text{Softmax}\left(\frac{q_t^j K_c}{\sqrt{f}}\right) V_c, \tag{3.4}$$

where $q_t^j \in Q_t$ and $j \in \{0, 1, \ldots, N-1\}$. The final combined representation is $\hat{F}_{ct} = \{\hat{v}_{ct}^0, \hat{v}_{ct}^1, \ldots, \hat{v}_{ct}^{N-1}\}$.

The output $\hat{F}_{ct}$ is expected to be aligned with the transcription Y. Thus, by feeding the $\hat{F}_{ct}$ into a linear module followed by a softmax activation function, the final prediction is obtained.

3.3.4 Inference on Test Data

When evaluating on test data, the transcriptions $\mathcal{Y}$ are not available. The text pipeline is initialized by feeding the start indicator $\langle S \rangle$ and it predicts the first character by attending the related visual part on the input handwritten text image. With the strategy of greedy decoding, this first predicted character is fed back to the system, which outputs the second predicted character. This inference process is repeated in a loop until the end of sequence symbol $\langle E \rangle$ is produced or when the maximum output length N is reached.

3.4 Experimental Evaluation

3.4.1 Dataset and Performance Measures

We conduct our experiments on the popular IAM handwritten dataset [103], composed of modern handwritten English texts. We use the RWTH partition, which consists of 6482, 976 and 2914 lines for training, validation and test, respectively. The size of alphabet $|\mathcal{A}|$ is 83, including special symbols, and the maximum length of the output character sequence is set to 89. All the handwritten text images are resized to the same height of 64 pixels while keeping the aspect ratio, which means that the textline images have variable length. To pack images into mini-batches, we pad all the images to the width of 2227 pixels with blank pixels.

Character Error Rate (CER) and *Word Error Rate* (WER) [47] are used for the performance measures. The CER is computed as the Levenshtein distance which is the sum of the character substitutions (S_c), insertions (I_c) and deletions (D_c) that are needed to transform one string into the other, divided by the total number of characters in the groundtruth (N_c). Formally,

$$CER = \frac{S_c + I_c + D_c}{N_c} \tag{3.5}$$

Similarly, the WER is computed as the sum of the word substitutions (S_w), insertions (I_w) and deletions (D_w) that are required to transform one string into the other, divided by the total number of words in the groundtruth (N_w). Formally,

$$WER = \frac{S_w + I_w + D_w}{N_w} \tag{3.6}$$

3.4.2 Implementation Details

Hyper-Parameters of Networks

In the proposed architecture, the feature size f is 1024. We use four blocks of visual and language self-attention modules, and each self-attention module has eight heads. We use 0.1 dropout setting for every dropout layer. In the text transcriber, all the transcriptions include the extended special symbols $\langle S \rangle$ and $\langle E \rangle$ at the beginning and at the end, respectively. Then, they are padded to 89 length with a special symbol $\langle P \rangle$ to the right, which is the maximum number of characters in the prediction N. The output size of the softmax is 83, which is the size of the alphabet $\mathcal{A}$, including upper/lower cased letters, punctuation marks, blank space and special symbols.

Optimization Strategy

We adopt label smoothing mechanism [136] to prevent the system from making over-confident predictions, which is also a way of regularization. As the ground-truth are one-hot vectors with binary values, label smoothing is done by replacing the 0 and 1 with $\frac{\varepsilon}{|\mathcal{A}|}$ and $1 - \frac{|\mathcal{A}| - 1}{|\mathcal{A}|}\varepsilon$, where ε is set to 0.4. We utilize Adam optimizer [81] for the training process with an initial learning rate of $2 \cdot 10^{-4}$, while reducing the learning rate by half every 20 epochs. The implementation of this system is based on PyTorch [120] and performed on a NVIDIA Cluster. The code will be publicly available.

3.4.3 Pre-training with Synthetic Data

Deep learning based methods need a large amount of labelled training data to obtain a well generalized model. Thus, synthetic data is widely used to compensate the scarcity of training data in the public datasets. There are some popular synthetically generated handwriting datasets available [88, 76], but they are at word level. For this reason we have created our own synthetic data at line level for pre-training. First, we collect a text corpus in English from online e-books and end up with over $130,000$ lines of text. Second, we select 387 freely available electronic cursive fonts and use them to randomly render text lines from the first step. Finally, by applying a set of random augmentation techniques (blurring/sharpening, elastic transforming, shearing, rotating, translating, scaling, gamma correcting and blending with synthetic background textures), we obtain a synthetic dataset with $138,000$ lines. The comparison between the synthetic data and the real data is shown in Figure 3.2.

3.4.4 Ablation Studies

In the ablation studies, all the experiments are trained from scratch with the IAM training set at line-level, and then early-stopped by the CER of the validation set, which is also utilized as an indicator to choose the hyper-parameters as shown in Table 3.1 3.2 3.4.

Architecture of CNN Feature Encoder

We have explored different popular Convolutional Neural Networks for the feature encoder detailed in Section 3.3.2. The best results were obtained with ResNet models. We modified the original ResNet architecture to slighty increase the final resolution of the features, by changing the stride parameter from 2 to 1 in the last convolutional layer. From Table 3.1, the best performance is achieved with a modified version of ResNet50.

Table 3.1: Ablation study on Convolutional architectures. * indicates modified architectures.

CNN	CER (%)	WER (%)
ResNet34	6.33	22.63
ResNet34*	5.44	20.13
ResNet50	5.49	20.93
ResNet50*	**4.86**	**18.65**

Function of Temporal Encoding

In both the visual feature encoder and the text transcriber, we have used temporal encoding in order to enforce an order information to both visual and textual features. Nonetheless we want to analyze its impact. In Table 3.2, it is clear that using temporal encoding at text level boosts the performance drastically from 7.72% to 4.86%, and from 6.33% to 5.52%, depending on whether we use it at image level or not. The best performance is reached when using the temporal encoding step both for image and text representations.

Role of Self-Attention Modules

Self-attention modules have been applied in both image and text levels. In Table 3.4 we analyze their effect in our system. We observe that the visual self-attention module barely improves the performance. Nonetheless, for the language self-attention module, it really plays an important role that improves the performance from 7.71% to 4.86%, and from 7.78% to 4.89%, with and without the

Table 3.2: Ablation study on the use of temporal encoding in image and text levels.

Image level	Text level	CER (%)	WER (%)
–	–	6.33	21.64
✓	–	7.72	24.70
–	✓	5.52	20.72
✓	✓	**4.86**	**18.65**

Table 3.3: Fine-tuning with different portions of real data (line-level test set with greedy decoding).

	20%		**40%**		**60%**		**80%**		**100%**	
	CER	WER	CER	WER	CER	WER	CER	WER	CER	WER
Seq2Seq	20.61	56.50	16.15	46.97	15.61	46.01	12.18	38.11	11.91	37.39
+ Synth	18.64	51.77	13.01	39.72	13.00	39.34	12.15	37.43	10.64	33.64
Ours	73.81	132.74	17.34	42.57	10.14	30.34	10.11	29.90	7.62	24.54
+ Synth	**6.51**	**20.53**	**6.20**	**19.69**	**5.54**	**17.71**	**4.90**	**16.44**	**4.67**	**15.45**

visual self-attention module, respectively. Our intuition is that the language self-attention module actually does learn language-modelling information. This implicitly learned language model is at character level and takes advantage of the contextual information of the whole text-line, which not only boosts the recognition performance but also keep the capability to predict out-of-vocabulary (OOV) words.

Table 3.4: Ablation study on visual and language self-attention modules.

Image level	Text level	CER (%)	WER (%)
–	–	7.78	29.78
✓	–	7.71	28.50
–	✓	4.89	18.57
✓	✓	**4.86**	**18.65**

We showcase in Figure 3.4 some qualitative results on text-line recognition, where we visualize the attention maps as well. The attention maps are obtained by averaging the mini attention maps across different layers and different heads. Those visualizations prove the successful alignment between decoded characters and images.

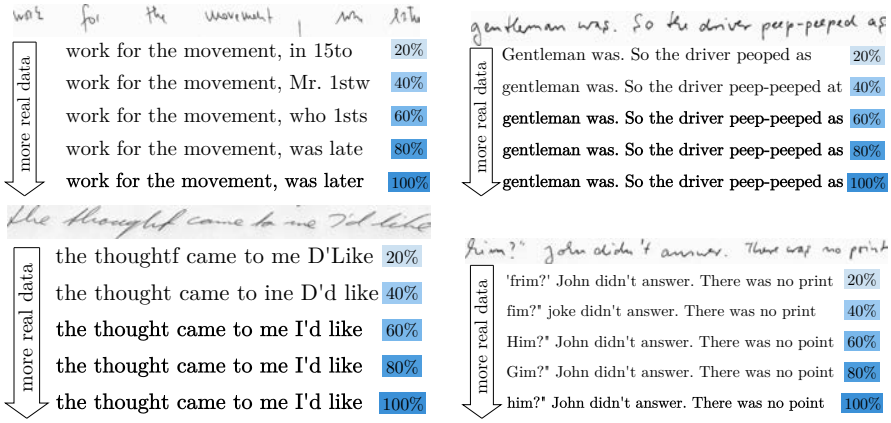


Figure 3.3: Performance of the transformer-based decodings for different amounts of real training data.

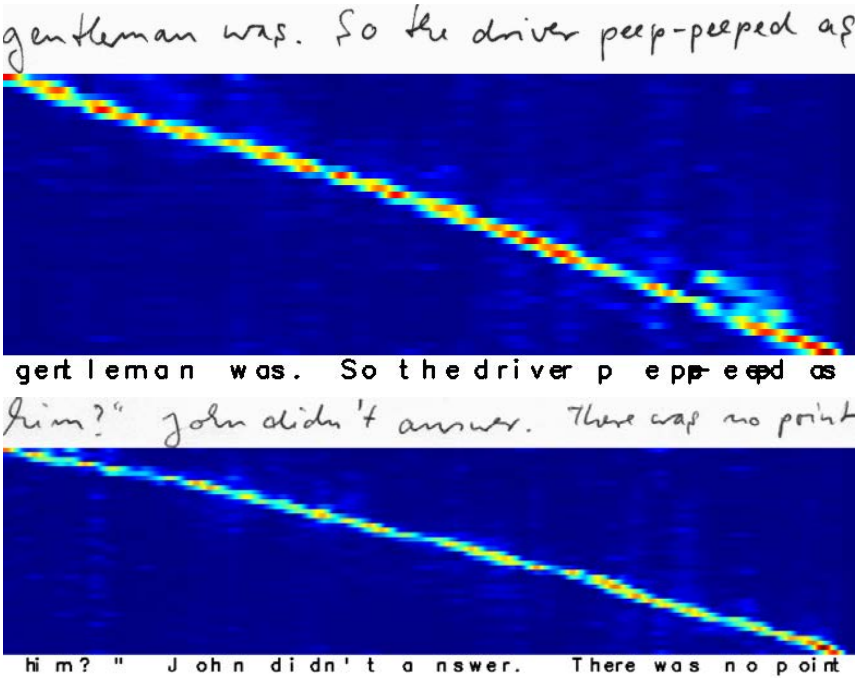


Figure 3.4: Qualitative results on text-line recognition and visualization of attention maps that coarsely align transcriptions and corresponding image characters.

3.4.5 Detailed Comparison with Seq2Seq Model

In order to provide a fair comparison between the proposed architecture and recurrent-based solutions, we re-implemented a state-of-the-art recurrent handwriting recognition pipeline, and we train and evaluate those under the exact same circumstances. Following the methods proposed in Chapter 2, we built a sequence-to-sequence recognizer composed of an encoder, a decoder and an attention mechanism. The encoder consists of a VGG19-BN [132] and a two-layer Bidirectional Gated Recurrent Units (BGRU) with feature size of 512. The decoder is a two-layer one directional GRU with feature size of 512, and we power the architecture with a location-based attention mechanism [27]. All the dropout layers are set to 0.5. Label smoothing technique is also used during the training process. The maximum number of predicted characters is also set to 89. All the hyper-parameters in this sequence-to-sequence model are also exhaustively validated by ablation studies with validation data.

We first provide in Table 3.5, the CER and WER rates on the IAM test set both when training the networks from scratch and just using the IAM training data, and when pre-training the networks with synthetic data for a later fine-tuning step on real data. We also provide the model size and the time taken per epoch during training. While the sequence-to-sequence model has much less parameters, it still takes longer to train than the transformers-based one. We also observe that both models benefit from the use of synthetic pre-training, improving the final error rates quite noticeably for the transformers model, although such boost is not so drastic for the sequence-to-sequence approach.

Table 3.5: Comparison between Recurrent and Transformers.

Method	CER (%)	WER (%)	Time(s)	Param(M)
Seq2Seq	11.91	37.39	338.7	37
+ Synth	10.64	33.64	338.7	37
Ours	7.62	24.54	202.5	100
+ Synth	**4.67**	**15.45**	**202.5**	**100**

3.4.6 Few-shot Training

Due to the scarcity and the cost of producing large volumes of real annotated data, we provide an analysis on the performance of the proposed approach when dealing with a few-shot training setup, when compared again with the sequence-to-sequence approach. To mimic a real scenario in which only a small portion of real data is available, we randomly selected 20%, 40%, 60% and 80% of the IAM training set.

As shown in Table 3.3, both sequence-to-sequence and transformer-based ap-

proaches follow the same trend. The more real training data is available, the better the performance is. Overall, the transformer-based method performs better than the sequence-to-sequence, except for the extreme case of just having a 20% of real annotated training data available. The transfomer approach, being a much larger model, struggles at such drastic data scarcity conditions. However, when considering the models that have been pre-trained with synthetic data, the transformer-based approach excels in few-shot setting conditions. We provide in Figure 3.3 some qualitative examples of the transcriptions provided by different models trained with reduced training sets. All of the models were pre-trained with synthetic data.

3.4.7 Language Modelling Abilities

In order to validate whether the proposed approach indeed is able to model language-specific knowledge besides its ability to decode handwritten characters, we propose to test whether using a state-of-the-art language model as a post-processing step actually improves the performance. We implement a shallow fusion [61] language model, consisting of a recurrent network with $2,400$ LSTM units. It has been trained on $130,000$ English text-lines. The additive weight for the shallow fusion is set to 0.2.

We observe in Table 3.6, that the use of such language modelling post-processing is useless, somehow indicating that the proposed approach already incorporates such language-specific contextual information within the language self-attention module.

Table 3.6: Effect of using a post-processing langauge model.

Method	CER (%)	WER (%)
Ours	4.67	**15.45**
+LM	**4.66**	15.47

3.4.8 Comparison with the State-Of-The-Art

Finally, we provide in Table 3.7 and extensive performance comparison with the state of the art. Different approaches have been grouped into a taxonomy depending on whether they are based on HMMs or early neural network architectures, whether they use recurrent neural networks (usually different flavours of LSTMs) followed by a Connectionist Temporal Classification (CTC) layer, or if they are based on encoder-decoder sequence-to-sequence architectures. Within each group, we differentiate results depending on whether they make use of a closed vocabulary of size Ω or they are able to decode OOV words. Bluche *et al.* [18] achieves the best result among the methods using a closed lexicon, while our proposed method

Table 3.7: Comparison with the State-Of-The-Art approaches on IAM line level dataset.

System	Method	Ω (k)	CER (%)	WER (%)
HMM/ANN 2008 - now	Almazán *et al.* [2]	–	11.27	20.01
	España *et al.* [44]	–	9.80	22.40
	Dreuw *et al.* [40]	50	12.40	32.90
	Bertolami *et al.* [10]	20	–	32.83
	Dreuw *et al.* [41]	50	10.30	29.20
	Zamora *et al.* [159]	103	7.60	16.10
	Pastor *et al.* [119]	103	7.50	19.00
	España *et al.* [44]	5	6.90	15.50
	Kozielski *et al.* [35]	50	5.10	13.30
	Doetsch *et al.* [34]	50	4.70	12.20
RNN+CTC 2008 - now	Chen *et al.* [24]	–	11.15	34.55
	Pham *et al.* [123]	–	10.80	35.10
	Krishnan*et al.* [87]	–	9.78	32.89
	Wigington *et al.* [153]	–	6.40	23.20
	Puigcerver [126]	–	5.80	18.40
	Dutta *et al.* [42]	–	5.70	17.82
	Graves *et al.* [58]	20	18.20	25.90
	Pham *et al.* [123]	50	5.10	13.60
	Puigcerver [126]	50	4.40	12.20
	Bluche *et al.* [18]	**50**	**3.20**	**10.50**
Seq2Seq 2016 - now	Chowdhury [28]	–	8.10	16.70
	Bluche [15]	–	7.90	24.60
	Bluche [15]	50	5.50	16.40
Transf.	**Ours**	–	**4.67**	**15.45**

obtains the best result among the methods without using a closed lexicon, while still competing with most of the closed-vocabulary approaches.

3.5 Conclusion

In this chapter, we have proposed a novel non-recurrent and open-vocabulary method for handwritten text-line recognition. As far as we know, it is the first approach that adopts the transformer networks for the HTR task. We have performed a detailed analysis and evaluation on each module, demonstrating the suitability of the proposed approach. Indeed, the presented results prove that our method not only achieves the state-of-the-art performance, but also has the capability to

deal with few-shot training scenarios, which further extends its applicability to real industrial use cases. Finally, since the proposed approach is designed to work at character level, we are not constrained to any closed-vocabulary setting, and transformers shine at combining visual and language-specific learned knowledge.

However, by witnessing the experimental results, we find that the performance on training data always performs better than that of test data, *i.e.* there is a gap between training and test data. It would be interesting to define this gap and reduce it, so that a better performance on test data can be obtained.

Chapter 4

Unsupervised Writer Adaptation

4.1 Introduction

In the first two chapters, we have proposed two recognition methods, a Seq2Seq method and a transformer-based method. And in the experimental results, we have witnessed a gap between the training and test data, so that the recognition performance on test set can never be as good as the training one. We define it as *domain gap*, which is caused of different handwriting styles between training and test data. Thus, we further define two domains: source and target domain, where the source domain contains the labeled images referred as training set while the target domain consists of unlabeled images referred as test set. Thus, the source domain can be either the real training data or the huge amount of synthetically rendered data (refer to Chapter 2).

For the real dataset, the different handwriting styles between training and test sets produce the domain gap. Similarly, even if the synthetic fonts are carefully selected, the extracted visual features will most likely differ from the ones one might find when dealing with real handwritten text. In that sense, a final adjustment step is needed in order to bridge the representation gap between the synthetic and the real samples.

Such issue raised awareness of the document analysis community, that has researched on the topic of writer adaptation since the early nineties [105, 124, 51]. The main motivation of such applications, consist in adapting a generic writer-independent HTR system, trained over a large enough source dataset, towards a new distribution of a particular writer. Especially interesting are the approaches that are able to yield such writer adaptation step in an unsupervised manner, that is, without needing any ground-truth labels from the new target writer.

Our main application contribution stems for the use of unsupervised domain

adaptation to forge an annotation-free handwriting recognition system. Our proposed approach is fully trained with synthetically generated samples that mimic the specific characteristics of handwritten text. Later, it is unsupervisedly adapted towards any new incoming target writer. In particular, the system produces transcriptions (without the need of labelled real data) that are competitive even compared to supervised methods. Text being a sequential signal, several temporal pooling alternatives are proposed to redesign current domain adaptation techniques so that they are able to process variable-length sequences. All in all it represents a step towards the practical success of HTR in unconstrained scenarios.

We show some examples of the results obtained after such writer adaptation in Fig. 4.4. We observe that even though the synthetically trained model outputs gibberish text, the committed errors are quite understandable, since the confusion is between letters and glyphs that are visually close. Once the unsupervised writer adaptation is applied, the text is correctly transcribed in all those cases. Our proposal is validated by using five different datasets in different languages, showing that our handwritten word recognizer is adapted to modern and historic samples, single and multi-writer collections. Our proposed adaptable handwritten word recognition model outperforms the state of the art, and compares quite favourably to supervised fine-tuning methods while not needing any manually annotated label.

4.2 Related Work

Inspired by the speech recognition community, writer adaptation techniques have been applied to modify early handwritten text recognition models based on *Hidden Markov Models* [51, 127, 1]. Once an omni-writer model has been trained, the model parameters, consisting of the Gaussian mixture means and variances, can be modified to better fit the target data distribution. Other early works proposed an *Expectation-Maximization* strategy [116, 137] over a set of different character recognizers. The main advantage of such techniques was that the adaptation procedure to unseen target writers was done in an unsupervised manner, without needing any target labelled data.

With the rise of deep learning, the use of *Long Short-Term Memory* (LSTMs) architectures became established for HTR. Such data hungry approaches have been commonly trained with the largest publicly available datasets, and then fine-tuned to the target collection to be recognized. Such tuning strategies [4, 55, 115] guarantee that the neural networks can be properly trained, ending up extracting relevant features from handwriting strokes, that are later revamped to the target collection. But fine-tuning presents the downside of needing manual annotations both from the source and target datasets. In order to alleviate such pain, the use of synthetically generated texts as source data has lately surfaced [89, 62, 11]. By the use of synthetic fonts, overfitting is avoided at no labelling cost. However, HTR models fully trained on synthetically generated data still need to be grounded

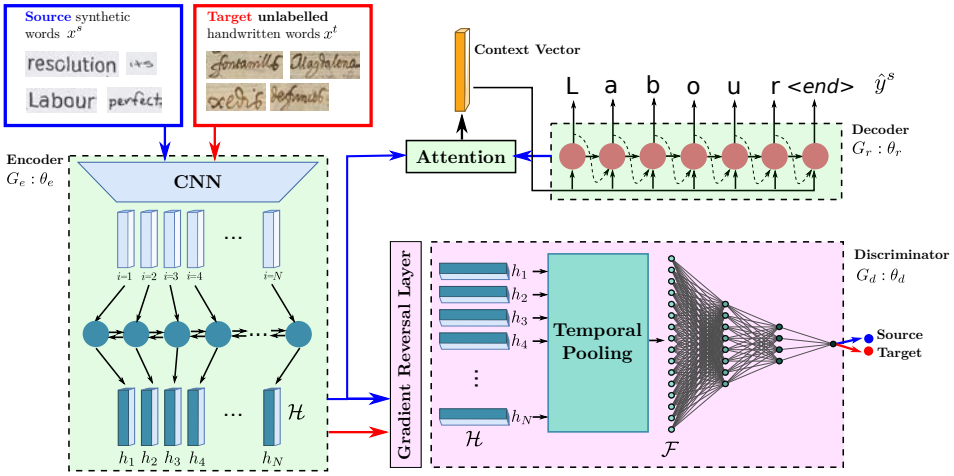


Figure 4.1: Architecture of the adaptable handwritten word recognizer. The model consists of an encoder, a decoder and a discriminator. The discriminator incorporates a temporal pooling step to be able to adapt to variable-length sequences. The blocks corresponding to the handwriting recognizer, and therefore used for inference, are highlighted in light green; the block responsible for the unsupervised domain adaptation during training is highlighted in light magenta (best viewed in color).

with real data in order to be effective, and thus target labels are still needed.

In order to discard target labelled data, unsupervised domain adaptation techniques have been proposed in the literature. Given a labeled source dataset and an unlabeled target dataset, their main goal is to adjust the recognition model so that it can generalize to the target domain while taking the domain shift across the datasets into account. A common approach to tackle unsupervised domain adaptation is through an adversarial learning strategy [49, 50, 121, 145], in which the discrepancy across different domains is minimized by means of jointly training a recognizer network and a domain discriminator network. The recognizer seeks to correctly recognize the labeled source domain data, whereas the discriminator has to distinguish between samples drawn either from source or target domains. The adversarial model is trained jointly in a min-max fashion, in which the aim is to minimize the recognition loss while maximizing the discriminator loss. For instance, Ganin *et al.* [50] adapted a digit recognizer trained on handwritten digits from MNIST to tackle other target digit datasets such as MNIST-M or SVHN; or Yang *et al.* [156], who proposed an unsupervised domain adaptation scheme for Chinese characters across different datasets. Such strategy has been proven to be effective when dealing with classification problems, where the source and target domains share the same classes. However it can not be straightforwardly applied to HTR applications, where, instead of a classification problem, the input

and output signals are sequential in nature.

We propose to integrate this adversarial domain adaptation for the recognition of cursive handwriting recognition using an encoder-decoder framework. Thus, both the inputs and outputs of our system are variable-length signals formed by a sequence of characters. Although the same character set has to be used for both source and target domains, the proposed method is not restricted to a particular output lexicon nor language. We incorporate a temporal pooling step aimed at adjusting the adversarial domain adaptation techniques to problems having variable-length signals. To the best of our knowledge, just the recent parallel work of Zhang *et al.* [160] proposes a similar idea. However, they propose that both the recognition and discrimination steps focus on character level. By disentangling the recognition and discrimination processes, one working at character and the other at word level respectively, we significantly outperform their approach. In addition, by synthetically rendering the source words with truetype fonts, our system does not require any manually generated label, and is trained "for free", not requiring any real annotated training data to be used as source domain.

4.3 Adaptable Handwritten Word Recognition

4.3.1 Problem Formulation

Our main objective is to propose an adaptable handwritten word recognizer application that is initially trained by synthetically generated word images, and then adapted to a specific handwriting style in an unsupervised and end-to-end manner. Our architecture, depicted in Fig. 4.1, consists of two interconnected branches, the handwriting recognizer and the discriminator, in charge of the adaptation process. By means of a gradient reversal layer, the two blocks will play an adversarial game in order to obtain an intermediate feature representation that is indistinguishable whether it is generated from a real or synthetic input, while being representative enough to yield good transcription performances.

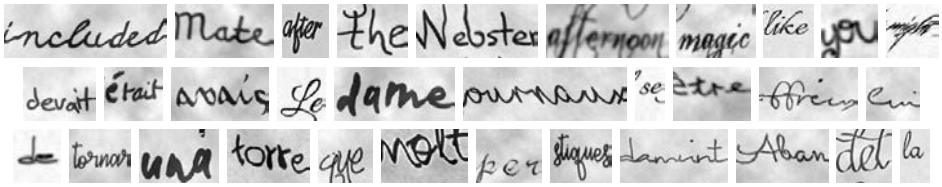


Figure 4.2: Synthetically generated words in English (top), French (mid) and Catalan (bottom) used during training.

In the proposed framework, two different flows are followed. The synthetically generated source words $x_i^s \in \mathcal{D}_s$ do come with their associated transcriptions

$y_i^s \in \mathcal{Y}_s$, and enter both the recognizer and the discriminator branches. Contrary, the real target word images $x_i^t \in \mathcal{D}_t$, being unlabelled, are just processed through encoding and the discriminator block. At inference time, the prediction of the target texts $\hat{y}_i^t$ is not bounded by any previously defined lexicon, being totally independent of $\mathcal{Y}_s$.

4.3.2 Rendering Synthetic Sources

The use of synthetically generated word collections that look like real handwriting to magnify training data volumes has become a common practice. Although several public datasets, such as the IIIT-HWS dataset [88], exist, we decided to create our own, in order to include special characters (e.g. accents, umlauts, punctuation symbols, etc.) that we want our recognizer to tackle. 387 freely available electronic fonts that imitate cursive handwriting were selected. A text corpus consisting of over 430,000 unique words was collected from free ebooks written in English, French, Catalan and German languages. By randomly rendering those words with the different electronic fonts, we ended up with more than 5.6 million word images. In order to add more variability to the synthetic collection and to act as a regularizer, we incorporate a data augmentation step, specifically tailored to produce realistic deformations that one can find in handwritten data. This augmentation step is applied online within the data loader, so that each batch is randomly augmented. Pixel-level deformations include blurring, gamma, brightness and contrast adjustments or Gaussian noise. Geometric transformations such as shear, rotation, scaling and an elastic deformation are also randomly applied. Finally a model generating random background textures that simulate paper surface is applied. Some samples of synthetic words are shown in Fig. 4.2.

4.3.3 Handwritten Word Recognition Framework

We use a sequence-to-sequence architecture topped with an attention mechanism to be our handwritten word recognition branch, which follows the same method in Chapter 2. Such architectures are able to process and output variable length data, and thus are not restricted to work with a predefined vocabulary. It consists of two main parts: the encoder and the attention-based decoder.

Encoder

The aim of the encoder is to extract high-level features given a word image, which can be further adapted in the same feature hyperspace. In this work we define the encoder as a Convolutional Neural Network feature extractor followed by a Recurrent Neural Network. The initial CNN is in charge of extracting visual features that characterize the handwritten words. This encoder leads to the final feature representation $\mathcal{H}$. We denote $h_i \in \mathcal{H}, i \in \{1, 2, ..., N\}$ as the output sequence of

the encoder. N is the length of $\mathcal{H}$, which varies according to the lengths of the input word images. Thus, we denote $G_e : \mathcal{I} \rightarrow \mathbb{R}^{D \times N}$ as the encoder function given an image $I \in \mathcal{I}$ with parameters θ_e.

Attention-based Decoder

The decoder is a one-directional multi-layered GRU, which predicts one character $\hat{y}^s_{i,k}$ at each time step k until reaching the maximum number of steps T or meeting the end of sequence symbol $\langle \text{end} \rangle$. Thus, let G_r denote the decoder function given the output of encoder $\mathcal{H} \in \mathbb{R}^{D \times N}$ with parameters θ_r, and its output is a sequence of characters $\hat{y}^s_i$, which is the concatenation of $\hat{y}^s_{i,k}$, where $k \in \{1, 2, ..., T\}$. Location-based attention mechanism [27] is chosen to give a constraint that the decoding process should be done from left to right of the images.

4.3.4 Temporal Pooling for Unsupervised Writer Adaptation

Text being a sequential and variable-length signal, state-of-the-art adversarial domain adaptation methods can not be straightforwardly used, since they all rely on having fixed length feature vectors. We propose to explore several *Temporal Pooling* strategies in order to transfer the variable length feature representation $\mathcal{H}$ into a fixed size feature representation $\mathcal{F}$ within the discriminator module:

Column-wise Mean Value (CMV) treats $\mathcal{H}$ as a column-wise sequence feature. The mean value is calculated as follows

$$\mathcal{F} = \frac{1}{N} \sum_{i=1}^{N} h_i. \tag{4.1}$$

Spatial Pyramid Pooling (SPP) [65] is a flexible solution for handling different scales, sizes and aspect ratios of images. It severs the images into divisions from finer to coarser levels and aggregates local features in a fixed-size feature vector.

Temporal Pyramid Pooling (TPP) [151] is an one-directional SPP. It is considered to be more suitable for handwriting recognition tasks, because words are composed of a sequence of characters, and they are read in a specific direction.

Gated Recurrent Unit (GRU) are used to process the sequential signal $\mathcal{H}$ to output a fixed size feature representation $\mathcal{F}$. In our model, we simply apply a 2-layered one-directional GRU.

Once we have obtained a fixed representation, $\mathcal{F}$ is fed into the domain classifier, which consists of three fully connected layers with batch normalization and ReLU activation. θ_d is used to represent the parameters of the discriminator G_d.

The output of G_d is binary, either predicting that the features $\mathcal{F}$ come from source or target samples.

4.3.5 Learning Objectives

Until now, we have a recognition loss L_r from the decoder and a discriminator loss L_d from the domain classifier. Since our model is trained in end-to-end fashion, the overall loss for the training scheme is defined as

$$\begin{aligned} L(\theta_e, \theta_r, \theta_d) = & \sum_{x_i \in \mathcal{D}_s} L_r\left(G_r\left(G_e(x_i)\right), y_i\right) - \\ & \lambda \sum_{x_j \in \mathcal{D}_s \cup \mathcal{D}_t} L_d\left(G_d\left(G_e(x_j)\right), d_j\right), \end{aligned} \tag{4.2}$$

where λ is a hyper-parameter to trade off the two losses L_r and L_d. In Section 4.4.2, different λ methods have been studied.

As stated before, the source data consists in synthetic word images plus their corresponding labels. The target data corresponds to real word images, but without labels.

The parameters of the discriminator are randomly initialized during the writer adaptation process. For the forward pass, the synthetic word images can be transferred through both the recognizer and the discriminator, while the real word images can only contribute to the discriminator loss. The backward propagation follows the same but reverse flow of the model by applying a *Gradient Reversal Layer* (GRL) [49] between the encoder and the discriminator. This layer applies the identity function during the forward pass but during the backward pass it multiplies the gradients by the parameter $-\lambda$. Thus, this layer reverses the gradient sign that flows through the model. By doing so, the model can be trained in a min-max optimization fashion. Minimizing the discriminator loss means to train a better discriminator for distinguishing between the synthetic and real data. In contrast, maximizing the discriminator loss for the encoder means to eliminate the differences of data feature distribution between the synthetic and real data. The goal of the optimization process is to find a saddle point that

$$\hat{\theta}_e, \hat{\theta}_r = \arg\min_{\theta_e, \theta_r} L(\theta_e, \theta_r, \theta_d) \tag{4.3}$$

$$\hat{\theta}_d = \arg\max_{\theta_d} L(\theta_e, \theta_r, \theta_d). \tag{4.4}$$

In short, synthetic data contributes to both the recognizer and the discriminator, whereas real data only contributes to the discriminator.

Table 4.1: Overview of the different datasets used in this work depicting its characteristics.

Dataset	Words	Writers	Period	Language
GW [92]	4,860	1	Historic	English
IAM [103]	115,320	657	Modern	English
Rimes [5]	66,978	1,300	Modern	French
Esposalles [46]	39,527	1	Historic	Catalan
CVL [83]	99,902	310	Modern	English/German

4.4 Experiments

In order to carry our writer adaptation experiments, we will use five different publicly available datasets with different particularities: single or multiple writers, coming from historic or modern documents or written in English, French, Catalan or German. We provide the details of such datasets in Table 4.1. To evaluate the system's performance, we will use the standard *Character Error Rate* (CER) and *Word Error Rate* (WER) metrics. In the tables, these values are in percentage ranging from [0-100].

4.4.1 Implementation Details

All our experiments were run using PyTorch [120] 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 be 50% for all the GRU layers except the last layer of both the encoder and decoder. During training, we have kept a balance in the total number of samples shown for both synthetic source words and real unlabelled target data. However, the training set is shuffled at each epoch and source and target data balancing is not guaranteed within a batch.

4.4.2 Ablation Study

Before assessing the performance of the proposed unsupervised writer adaptation model, we want to validate the adequacy of several hyper-parameters involved in our system. The following experiments are carried out using the IAM validation set as target dataset, except the last one, where the GW dataset was used instead. First, we evaluate which is the best temporal pooling strategy to recast the variable-length features of the encoder to the fixed-length features needed by the discriminator. In Table 4.2, we observe that the GRU achieves the best performance. The GRU module has trainable parameters, so, contrary to the other aggregation strategies, it can learn how to effectively pool the variable-length features into a meaningful fixed-length representation, and consequently, obtain a

better performance. For the rest of experiments we will use the GRU as our temporal pooling strategy.

Table 4.2: Study on the different Temporal Pooling approaches of the discriminator, evaluated on the IAM validation set.

	CMV	**SPP**	**TPP**	**GRU**
CER	14.83	15.76	14.55	**13.58**
WER	36.83	38.86	36.44	**33.99**

Second, we analyze three different approaches to set the hyper-parameter λ, which controls the trade-off between the recognition loss L_r and the discriminator loss L_d. We choose to either set it as a constant $\lambda = 1$, increase its value linearly from 0 to 1 at each epoch, or increase its value from 0 to 1 in an exponential way. Although a gradual increase of the weight of the discriminator loss could potentially benefit the overall performance, in Table 4.3 we appreciate that simply setting λ as a constant value provides the best results.

Table 4.3: Study on the different λ strategies, evaluated on the IAM validation set.

λ	**Constant**	**Linear**	**Exponential**
CER	**13.58**	13.79	14.43
WER	**33.99**	35.42	36.65

Finally, we explore the effect of providing different amounts of unlabelled target data to the system during writer adaptation. In this experiment we use the GW dataset, since it contains almost $5,000$ words from the same writer. We observe in Fig. 4.3 that the higher the amount of unlabelled target data, the lower the error rate. Thus, for the subsequent experiments, we will use all the available target data at hand during the adaptation, no matter if the scenario concerns a single writer or several of them. For multi-writer collections, we could thus choose among two options: (i) the system is adapted to a particular writer, using just a subset of the collection; (ii) the system is adapted to the whole collection style (rather than to the individual writing characteristics) by providing the whole dataset during the adaptation.

4.4.3 From synthetic to real writer adaptation

For the unsupervised writer adaptation experiments, we will use all the available images from each dataset during the unsupervised writer adaptation process, in order to have as much real word instances as possible. According to the experiments in the previous section, this should yield the best performance. It should be noted that these datasets are always used in an unsupervised manner, i.e. the

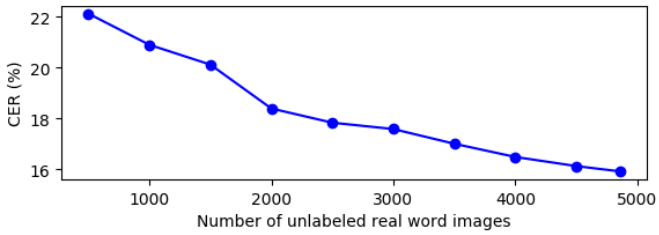


Figure 4.3: Influence of the amount of unlabeled real word images over the performance, evaluated on the GW dataset.

Table 4.4: Unsupervised writer adaptation results for handwritten word recognition. The gap reduction shows the improvement when the HTR, trained on synthetic data, is adapted to real data.

	GW		**IAM**		**Rimes**		**CVL**		**Esposalles**	
	CER	WER	CER	WER	CER	WER	CER	WER	CER	WER
Real target only	4.56	13.49	6.88	17.45	2.80	8.51	3.64	7.77	0.47	1.68
Synth. source only	26.05	56.79	26.44	54.56	21.46	52.48	26.30	55.64	30.78	66.33
Uns. adaptation	16.28	39.95	14.05	34.86	14.39	39.21	19.19	44.29	20.96	50.00
Gap reduction (%)	45.46	38.89	63.34	53.09	37.89	30.18	31.38	23.71	32.40	25.26

system has access to the word images, but never to their transcriptions (labels). However, the CER and WER results are computed on the official test set partitions in all datasets, so that those results are comparable with the literature. Qualitative results are shown in Figure 4.4.

In Table 4.4 we present our writer adaptation results on the five different datasets. For each dataset we also provide two baseline results. Training using target labels and training just using the synthetic samples provide baselines for the best and worst case scenarios respectively, either using ground-truth labels or ignoring any labelled information. The gap reduction is an measurement used to measure the effectiveness of the adaptation method which is defined as:

$$\text{gap reduction} = \frac{\text{error(synth.)} - \text{error(adapted)}}{\text{error(synth.)} - \text{error(real)}} \tag{4.5}$$

We appreciate that, in general, the difference in CER between these two baselines, lower bound $error(synth.)$ and upper bound $error(real)$, is about 20 points, with the exception of the Esposalles dataset, which presents a much higher gap. This difference is most likely justified because it is the dataset in which the handwriting style differs more from a visual point of view from the synthetically generated samples.

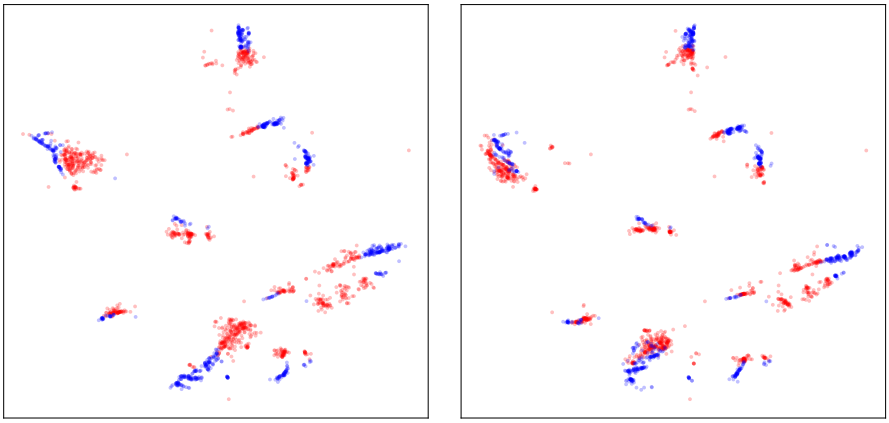
Concerning the unsupervised writer adaptation results (in Table 4.4, *Uns. adaptation*), we appreciate a significant improvement when compared with the sole use of synthetic training samples. The gap reduction ranges from 20% in the

Figure 4.4: Handwritten word recognition results with our model trained only using synthetically generated word samples. We show the transcription before and after (in boldface) the unsupervised writer adaptation, for the GW, IAM, RIMES, Esposalles and CVL datasets respectively.

worse case (CVL), up to 60% in the best case (IAM). It is true that these results are worse than the ones obtained by a recognizer trained on labelled target data. However, the loss in accuracy is compensated by the fact that our approach is more generic and flexible: it is trained with synthetically generated data and it does not require any manually annotated target data for writer adaptation. In Fig. 4.5 we provide a tSNE visualization of the sample distribution before and after the unsupervised writer adaptation in the single-writer GW dataset.

4.4.4 Writer adaptation with few samples

This experiment is devised to evaluate whether the adaptation ability of our approach decreases when there are few samples in the target domain. Indeed, in the experiments presented in Table 4.4, the system is adapting to a particular individual handwriting style for the GW and Esposalles datasets, because they are single writer. Given that the IAM, Rimes and CVL datasets contain samples from multiple writers, the system is adapting from synthetic samples to the overall collection style. Since in the IAM dataset we do have groundtruth information about



a) Before adaptation b) After adaptation

Figure 4.5: The distribution of source (blue) and target (red) domain samples before (a) and after (b) the adaption to the GW dataset for the ten most common words.

which specific writer produced each word, we choosed it for this writer specific adaptation experiment, taking into account that the volume of words per writer that we can use as target domain is very reduced. Within the IAM validation set, each writer has written between 13 and 602 words. As source domain we randomly selected 600 synthetic words (images and labels) for every single writer specific adaptation experiment.

From the results shown in Table 4.5, we appreciate that our model boosts the recognition performance on every writer even though when there is a very reduced amount of both source and unlabelled target samples. Due to the limited space, we only show the top five best and worse cases ranked by the improvement percentage between the CER measure obtained with a system trained with just synthetic data or after writer adaptation using this low amount of samples. We observe that for all the writers in the IAM validation set, the CER measure is enhanced after the proposed unsupervised adaptation. By inspecting the qualitative results, we observe that the writers that present the lowest improvement corresponded to specimens with writing styles that are visually very dissimilar to our synthetically generated source material.

In general, this experiment depicts a realistic scenario in which our generic handwritten word recognizer, fully trained with synthetic data, is adapted to a new incoming writer by just providing a very reduced set of his handwriting. From the results, we can conclude that the recognition performance for this new writer is significantly boosted in most cases, in an unsupervised and efficient manner.

Table 4.5: Writer adaptation results, in terms of the CER, ranked by the improvement percentage with respect to the synthetic training.

Writer ID	**Words**	**Synth.**	**Adapt.**	**Improv.**(%)
ID202	396	13.65	3.96	71.0
ID521	48	21.68	7.39	65.9
ID278	129	7.71	3.66	52.5
ID625	80	23.18	11.76	49.3
ID210	136	9.41	5.29	43.8
...	...	...	...	...
ID533	52	37.50	32.50	13.3
ID182	69	29.89	26.05	12.8
ID515	74	38.29	34.20	10.7
ID527	127	24.86	22.20	10.7
ID612	55	29.83	28.57	4.2
Mean	135	24.32	18.34	27.4

4.4.5 Comparison with the state of the art

Supervised fine-tuning. In order to put into context our reached results, we compare in Table 4.6 our model with the state-of-the-art approaches that propose to pre-train a handwriting recognizer with a large dataset, e.g. IAM, and then fine-tune the network to transfer the learned parameters to a different collection, e.g. GW, with a disparate style. We compare against the recent works proposed by Nair *et al.* [115] and Arandillas *et al.* [4]. They achieve CER values of 59.3% and 82%, respectively with their models trained on IAM and tested over the GW test set. Our baseline model, pre-trained just using a synthetically produced data, already achieves a 26.05% CER on the GW dataset. This backs up the intuition that the use of a synthetic dataset, which can contain as many training samples as desired, provides better generalization than training with a much shorter amount of real data.

Our unsupervised writer adaptation reaches a 16.28% CER while Nair *et al.* and Arandillas *et al.* reach a 8.26% and 5.3% CER respectively when fine-tuning, at the expense of requiring a fair amount of manually labeled data. Obviously, our unsupervised approach does not reach the same performance as these supervised approaches, because they use labelled GW words. Although it is not the main scope of our work, if we do use labels for the target domain (last row in Table 4.6), i.e. we adapt to the new incoming writer in a supervised manner, our approach outperforms the above methods, reaching a 2.99% CER.

Unsupervised domain adaptation. To the best of our knowledge, only the work of Zhang *et al.* [160] report results for unsupervised writer adaptation at word level. However, for the case of handwriting words, they propose to use

Table 4.6: Comparison with supervised fine-tuning.

Method	Train	Fine-tuning adaptation	CER
Nair [115]	IAM	None	59.30
	IAM	Sup. GW	8.26
Arandillas [4]	IAM	None	82.00
	IAM	Sup. GW	5.30
Proposed	Synth.	None	26.05
	Synth.	Uns. GW	16.28
	Synth.	Sup. GW	2.99

Table 4.7: Comparison with sequence-to-sequence domain adaptation on IAM dataset.

Method	CER	WER	Average
Zhang *et al.* [160]	8.50	22.20	15.35
Proposed	**6.75**	**17.26**	**12.01**

labelled IAM training data as source and unlabelled IAM test data as target domains. In our opinion, such experiment does not present any significant domain shift. When using their same experimental setting, shown in Table 4.7, our approach achieves a significant better performance.

4.5 Conclusion

We have proposed a novel unsupervised writer adaptation application for handwritten text recognition. Our method is able to adapt a generic HTR model, trained only with synthetic data, towards real handwritten data in a completely unsupervised way. The system mutually makes the high-level feature distribution of synthetic and real handwritten words align towards each other, while training the recognizer with this common feature distribution.

Our approach has shown very good performance on different datasets, including modern, historical, single and multi-writer document collections. Even when compared to supervised approaches, our approach demonstrates competitive results. Moreover, since our unsupervised approach only requires to have access to a few amount of word images from the target domain, but not their labels, we believe that it is a promising direction towards a universal HTR for unconstrained scenarios, e.g. industrial applications.

However, the writer adaptation method can only diminish the gap between

source and target data, but never eliminate it. If the synthetic data could mimic the exact visual appearance of target data, obtaining a good performance trained with *that* synthetic data would guarantee the same performance on the target one. We do believe that this will be a more effective method.

Chapter 5

Handwriting Synthesizer using GANs and the Boost on HTR

5.1 Introduction

As discussed in the previous chapter, the writer adaptation method can diminish but not eliminate the domain gap between source and target data. In this chapter, we want to alleviate this problem by making the source data to mimic the same visual appearance as the target one. The intuition is to generate target-like synthetic data based on unlabeled target images.

In the field of image generation, we have witnessed a remarkable success in generating natural scene images based on Generative Adversarial Networks (GANs) [54], which are even indiscernible from real ones by humans [80]. Conditional Generative Adversarial Networks (cGANs) [111] were proposed to condition the generation process with a class label. Thus, controllable samples can be generated from different given types [26]. However, these conditioned class labels have to be predefined and hard-coded in the model before the training process, so that it lacks the flexibility to generate images from unseen classes at inference time.

Concerning the specific case of generating samples of handwritten text, there are two different approaches to the problem. Since handwritten text is a sequential signal in nature, the same as natural language strings [158], sketch drawings [63, 161], audio signals [36] or video streams [144], it is natural that the first attempts at generating handwritten data [56] were based on Recurrent Neural Networks (RNNs) [96]. Such approaches generate a sequence of strokes in vectorial format that are used to render images. On the contrary, some more recent approaches propose to directly generate images instead of sequences of strokes. By producing images directly, long-range dependency and gradient vanishing problems of recurrencies are avoided, while achieving a better efficiency. Furthermore,

such approaches are able to produce richer results in the sense that they go beyond producing just nib locations, but also provide visual appearance such as the calligraphic styles, such as slant, glyph shapes, stroke width, darkness, character roundness, ligatures, etc., and background paper features like texture, opacity, show-through effects, etc.

Current state-of-the-art methods that directly generate handwriting images work at different levels. First, some approaches are focused on producing isolated characters or ideograms [60, 23]. Such approaches often work over a set of predefined classes, so that they can only generate a reduced set of contents. Second, some approaches are able to generate handwritten words [3, 77], allowing not to be restricted to a closed vocabulary. Finally, some works like [45, 30] go beyond isolated words and produce full text-lines. The generation on text-line level is difficult because not only the handwritten text should be readable and realistic, but also the writing flow should be natural and smooth.

In this chapter, we would propose two approaches that are able to artificially render realistic handwritten word and text-line images that match a certain textual content and that mimic some style features (text skew, slant, roundness, stroke width, ligatures, etc.) from an exemplar writer. To this end, we guide the learning process by three different learning objectives [117]. First, an adversarial discriminator ensures that the images are realistic and that its visual appearance is as closest as possible to real handwritten word images. Second, a style classifier guarantees that the provided calligraphic attributes, characterizing a particular handwriting style, are properly transferred to the generated word instances. Finally, a state-of-the-art sequence-to-sequence handwritten word recognizer 2 or a transformer-based handwritten text-line recognizer 3 controls that the textual contents have been properly conveyed during the image generation. The extension from word to text-line generation requires for several modifications with Periodic Padding and curriculum learning strategy. In addition, we propose a novel version of the Fréchet Inception Distance (FID) metric to guide the method to choose the best hyper-parameters specifically for variable-length samples like handwritten text images. The proposed method is particularly focused on improving the HTR performance, demonstrating that the use of realistic synthetic generated text at training time is indeed useful for improving HTR. Moreover, we have tried a disentanglement experiment by modifying the method into an image-to-image translation setting as an auxiliary module for the recognizer, which could distill the textual content information from handwriting styles so as to boost HTR performance.

To summarize, the main contributions of this chapter are the following:

- We propose a group of methods for handwritten word and text-line image generation conditioning on textual content and visual appearance information, which is capable of generating open vocabulary text and visual appearance.

- We introduce an improved version of the FID measure, namely vFID, as a novel metric to evaluate the quality of the generated handwritten image. It is more robust to variable-length images and particularly suited for the handwriting case.
- We conduct extensive experiments to demonstrate, on the one hand, the realism of the generated handwritten text images and, on the other hand, the boost in HTR performance avoiding the manual labeling effort.
- We modify the method as an auxiliary disentanglement module, which is equipped with a handwriting recognizer in an end-to-end fashion to boost HTR performance.

5.2 Related Work

Traditional methods [149, 95, 84, 139] approached the generation of word samples by manually segmenting individual characters or glyphs and then tune a deformation to match the target writing style. Recently, based on these rendering methods, Haines *et al.* [64] succeeded in generating indistinguishable historical manuscripts of Sir Arthur Conan Doyle, Abraham Lincoln and Frida Kahlo with new textual contents, but these impressive results are obtained at the cost of a high manual intervention.

The generation of sequential handwritten data consists of producing stroke sequences in vector form with nib locations and sometimes velocity records. With the coming of deep learning era, Graves [56] utilized Long Short-Term Memory (LSTM) to predict point by point at each time step to generate stroke sequences conditioned on a given writing style and a certain text string. Following this sequential-based idea, some recent works [63, 48, 161, 106] have reached an impressive performance on text or sketch generation. However, sequential handwritten data is expensive to obtain and loose some richer visual appearance such as text thickness, darkness of strokes or paper textures.

Thus, the direct generation of images of handwritten data draws more attention, since it contains all the details of strokes and background paper. Different levels of handwritten data can be processed: characters/glyphs, words and textlines. Based on the ideas of variational auto-encoders [82] or GANs [54], some works achieve impressive performance on synthesizing Chinese ideograms [100, 140, 23, 73, 154] and glyphs [6]. However, these methods are restricted to a predefined set of content classes and the input images have fixed size. To overcome the limitation of incapability of generating out of vocabulary (OOV) texts, Alonso *et al.* [3] proposed a cGAN-based method to generate handwritten word samples, which is conditioned on RNN-embedded text information. However, this proposed approach suffers from the mode collapse problem so that it learns the general writing style of the training set and does not offer variability of the generated samples.

Fogel *et al.* [45] equip a style-promoting discriminator to be able to generate diverse styles for handwritten image samples. However, the generated characters have the same receptive field width, which can make the generated samples look unrealistic. Davis *et al.* [30] takes advantage of CTC activations [57] to produce spaced text, which helps the generator to achieve horizontal alignment with the input style image. The style information is the concatenation of both global style feature and character-wise style feature. However, the character-wise style feature highly depends on the performance of CTC, so mode collapse problem may happen when tackling the unseen style images from the target dataset.

Summarizing, state-of-the-art generative methods are still unable to produce plausible yet diverse images of whatever handwritten word or text-line automatically. In this chapter, we propose to condition a generative model for handwritten words or text-lines with unconstrained text sequences and stylistic typographic attributes, so that we are able to generate any text with a great diversity over the produced results.

5.3 Conditioned Handwritten Text Generation

5.3.1 Problem Formulation

Let $\{\mathcal{X}, \mathcal{Y}, \mathcal{W}\} = \{(x_i, y_i, w_i)\}_{i=1}^{N}$ be a multi-writer handwritten text dataset, containing gray-scale text images $\mathcal{X}$, their corresponding transcription strings $\mathcal{Y}$ and their writer identifiers $\mathcal{W}$. In this work, the handwriting calligraphic style is considered as an inherent feature for each of the different writers, and we also hypothesize that the background paper features are consistent within each writer. Thus, the visual appearance is identified with $w_i \in \mathcal{W}$. Therefore, let $X_i = \{x_{w_i,j}\}_{j=1}^{K} \subset \mathcal{X}$ be a subset of K real text images with the same style defined by writer $w_i \in \mathcal{W}$. Besides, $\mathcal{A}$ denotes the alphabet containing all the supported characters such as lower and upper case letters, digits and punctuation signs that the generator will be able to produce.

In this setting, the realistic handwritten text generation problem is formulated in terms of few-shot learning. Two inputs are given to the model: 1) a set of images X_i as a support example of the visual appearance attributes of a particular writer w_i; and 2) a textual content provided by any text string t where $t_i \in \mathcal{A}$. The proposed conditioned handwritten text generation model is able to combine both sources of information in order to yield realistic handwritten text-line images, which share the visual appearance attributes of writer w_i and the textual content provided by the string t. Finally, our objective model H, able to generate handwritten text, is formally defined as

$$\bar{x} = H\left(t, X_i\right) = H\left(t, \{x_1, \ldots, x_K\}\right), \tag{5.1}$$

where $\bar{x}$ is the artificially generated handwritten text image with the desired prop-

erties. From now on, we denote $\bar{\mathcal{X}}$ as the output distribution of the generative network H.

5.3.2 Handwritten Word Generation

The proposed architecture is divided in two main components. The generative network produces human-readable images conditioned to the combination of calligraphic style and textual content information. The second component are the learning objectives which guide the generative process towards producing images that look realistic; exhibiting a particular calligraphic style attributes; and having a specific textual content. Fig. 5.1 gives an overview of our model.

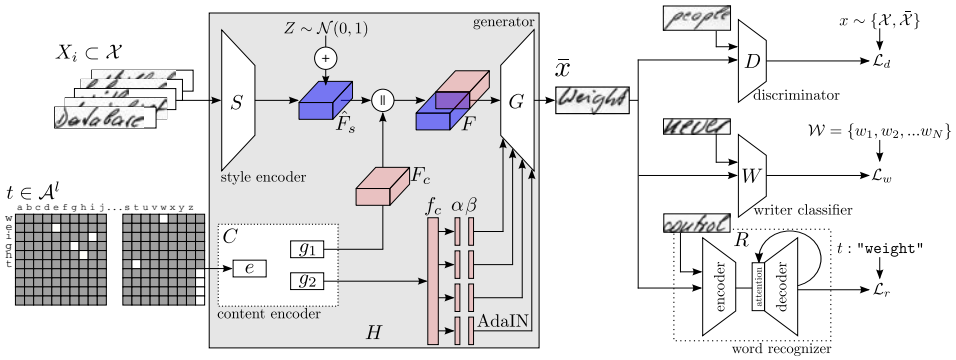


Figure 5.1: Architecture of the proposed handwriting generation model.

Generative Network

The proposed generative architecture H consists of a calligraphic style encoder S, a textual content encoder C and a conditioned image generator G. The overall calligraphic style of input images X_i is disentangled from their individual textual contents, whereas the string t provides the desired content.

Calligraphic style encoding. Given the set $X_i \subset \mathcal{X}$ of $K = 15$ word images from the same writer w_i, the style encoder aims at extracting the calligraphic style attributes, *i.e.* slant, glyph shapes, stroke width, character roundness, ligatures etc. from the provided input samples. Specifically, our proposed network S learns a style latent space mapping, in which the obtained style representations $F_s = S(X_i)$ are disentangled from the actual textual contents of the images X_i. The VGG-19-BN [132] architecture is used as the backbone of S. In order to process the input image set X_i, all the images are resized to have the same height h, padded to meet a maximum width w and concatenated channel-wise to end up with a single tensor $h \times w \times K$. If we ask a human to write the same word several times, slight

involuntary variations appear. In order to imitate this phenomenon, randomly choosing permutations of the subset X_i will already produce such characteristic fluctuations. In addition, an additive noise $Z \sim \mathcal{N}(0,1)$ is applied to the output latent space to obtain a subtly distorted feature representation $\hat{F}_s = F_s + Z$.

Textual content encoding. The textual content network C is devoted to produce an encoding of the given text string t that we want to artificially write. The proposed architecture outputs content features at two different levels. Low-level features encode the different characters that form a word and their spatial position within the string. A subsequent broader representation aims at guiding the whole word consistency. Formally, let $t \in \mathcal{A}^l$ be the input text string, character sequences shorter than l are padded with the empty symbol ε. Let us define a character-wise embedding function $e\colon \mathcal{A} \to \mathbb{R}^n$. The first step of the content encoding stage embeds with a linear layer each character $c \in t$, represented by a one-hot vector, into a character-wise latent space. Then, the architecture is divided into two branches.

Character-wise encoding: Let $g_1\colon \mathbb{R}^n \to \mathbb{R}^m$ be a Multi-Layer Perceptron (MLP). Each embedded character $e(c)$ is processed individually by g_1 and their results are later stacked together. In order to combine such representation with style features, we have to ensure that the content feature map meets the shape of $\hat{F}_s$. Each character embedding is repeated multiple times horizontally to coarsely align the content features with the visual ones extracted from the style network, and the tensor is finally vertically expanded. The two feature representations are concatenated to be fed to the generator $F = [\hat{F}_s \parallel F_c]$. Such a character-wise encoding enables the network to produce OOV words, *i.e.* words that have never been seen during training.

Global string encoding: Let $g_2\colon \mathbb{R}^{l \cdot n} \to \mathbb{R}^{2p \cdot q}$ be another MLP aimed at obtaining a much broader and global string representation. The character embeddings $e(c)$ are concatenated into a large one-dimensional vector of size $l \cdot n$ that is then processed by g_2. Such global representation vector f_c will be then injected into the generator splitted into p pairs of parameters.

Both functions $g_1(\cdot)$ and $g_2(\cdot)$ make use of three fully-connected layers with ReLU activation functions and batch normalization [70].

Generator. Let F be the combination of the calligraphic style attributes and the textual content information character-wise; and f_c the global textual encoding. The generator G is composed of two residual blocks [69] using the AdaIN as the normalization layer. Then, four convolutional modules with nearest neighbor upsampling and a final tanh activation layer generates the output image $\bar{x}$. AdaIN is formally defined as

$$\mathrm{AdaIN}\left(z, \alpha, \beta\right) = \alpha \left(\frac{z - \mu\left(z\right)}{\sigma\left(z\right)} \right) + \beta, \tag{5.2}$$

where $z \in F$, μ and σ are the channel-wise mean and standard deviations. The global content information is injected four times ($p = 4$) during the generative

process by the AdaIN layers. Their parameters α and β are obtained by splitting f_c in four pairs. Hence, the generative network is defined as

$$\bar{x} = H\left(t, X_i\right) = G\left(C\left(t\right), S\left(X_i\right)\right) = G\left(g_1\left(\hat{t}\right), g_2\left(\hat{t}\right), S\left(X_i\right)\right), \tag{5.3}$$

where $\hat{t} = [e(c); \forall c \in t]$ is the encoding of the string t character by character.

Learning Objectives

We propose to combine three complementary learning objectives: a discriminative loss, a style classification loss and a textual content loss. Each one of these losses aim at enforcing different properties of the desired generated image $\bar{x}$.

Discriminative Loss. Following the paradigm of GANs [54], we make use of a discriminative model D to estimate the probability that samples come from a real source, *i.e.* training data $\mathcal{X}$, or belong to the artificially generated distribution $\bar{\mathcal{X}}$. Taking the generative network H and the discriminator D, this setting corresponds to a min max optimization problem. The proposed discriminator D starts with a convolutional layer, followed by six residual blocks with LeakyReLU activations and average poolings. A final binary classification layer is used to discern between fake and real images. Thus, the discriminative loss only controls that the general visual appearance of the generated image looks realistic. However, it does not take into consideration neither the calligraphic styles nor the textual contents. This loss is formally defined as

$$\mathcal{L}_d\left(H, D\right) = \mathbb{E}_{x\sim\mathcal{X}}\left[\log\left(D\left(x\right)\right)\right] + \mathbb{E}_{\bar{x}\sim\bar{\mathcal{X}}}\left[\log\left(1 - D\left(\bar{x}\right)\right)\right]. \tag{5.4}$$

Style Loss. When generating realistic handwritten word images, encoding information related to calligraphic styles not only provides diversity on the generated samples, but also prevents the mode collapse problem. Calligraphy is a strong identifier of different writers. In that sense, the proposed style loss guides the generative network H to generate samples conditioned to a particular writing style by means of a writer classifier W. Given a handwritten word image, W tries to identify the writer $w_i \in \mathcal{W}$ who produced it. The writer classifier W follows the same architecture of the discriminator D with a final classification MLP with the amount of writers in our training dataset. The classifier W is only optimized with real samples drawn from $\mathcal{X}$, but it is used to guide the generation of the synthetic ones. We use the cross entropy loss, formally defined as

$$\mathcal{L}_w\left(H, W\right) = -\mathbb{E}_{x\sim\left\{\mathcal{X},\bar{\mathcal{X}}\right\}}\left[\sum_{i=1}^{|\mathcal{W}|} w_i \log\left(\hat{w}_i\right)\right], \tag{5.5}$$

where $\hat{w} = W(x)$ is the predicted probability distribution over writers in $\mathcal{W}$ and w_i the real writer distribution. Generated samples should be classified as the writer w_i used to construct the input style conditioning image set X_i.

Content Loss. A final handwritten word recognizer network R is used to guide our generator towards producing synthetic word images with a specific textual content. We implemented a state-of-the-art sequence-to-sequence model as detailed in Chapter 2 for handwritten word recognition to examine whether the produced images $\bar{x}$ are actually decoded as the string t. The Kullback-Leibler divergence loss is used as the recognition loss at each time step. This is formally defined as

$$\mathcal{L}_r\left(H,R\right) = -\mathbb{E}_{x\sim\{\mathcal{X},\bar{\mathcal{X}}\}}\left[\sum_{i=0}^{l}\sum_{j=0}^{|\mathcal{A}|} t_{i,j}\log\left(\frac{t_{i,j}}{\hat{t}_{i,j}}\right)\right], \tag{5.6}$$

where $\hat{t} = R(x)$; $\hat{t}_i$ being the i-th decoded character probability distribution by the word recognizer, $\hat{t}_{i,j}$being the probability of j-th symbol in $\mathcal{A}$ for $\hat{t}_i$, and $t_{i,j}$ being the real probability corresponding to $\hat{t}_{i,j}$. The empty symbol ε is ignored in the loss computation; t_i denotes the i-th character on the input text t.

5.3.3 End-to-end Training

Overall, the whole architecture is trained end to end with the combination of the three proposed loss functions

$$\mathcal{L}(H,D,W,R) = \mathcal{L}_d(H,D) + \mathcal{L}_w(H,W) + \mathcal{L}_r(H,R), \tag{5.7}$$

$$\min_{H,W,R}\max_{D}\mathcal{L}(H,D,W,R). \tag{5.8}$$

Algorithm 1 presents the training strategy that has been followed in this work. $\Gamma(\cdot)$ denotes the optimizer function. Note that the parameter optimization is performed in two steps. First, the discriminative loss is computed using both real and generated samples (line 3). The style and content losses are computed by just providing real data (line 4). Even though W and D are optimized using only real data and, therefore, they could be pre-trained independently from the generative network H, we obtained better results by initializing all the networks from scratch and jointly training them altogether. The network parameters Θ_D are optimized by gradient ascent following the GAN paradigm whereas the parameters Θ_W and Θ_R are optimized by gradient descent. Finally, the overall generator loss is computed following Equation 5.7 where only the generator parameters Θ_H are optimized (line 8).

5.3.4 Handwritten Text-line Generation

Since the generation problem extends from word to text-line level, the method proposed above cannot deal with the increasing difficulty. Thus, we introduce a Periodic Padding module and replace the Seq2Seq recognizer with Transformer-based recognizer. In addition, curriculum learning strategy is applied to help the

Algorithm 1 Training algorithm for the proposed model.

Input: Input data $\{\mathcal{X}, \mathcal{Y}, \mathcal{W}\}$; alphabet $\mathcal{A}$; max training iterations T
Output: Networks parameters $\{\Theta_H, \Theta_D, \Theta_W, \Theta_R\}$.

repeat
  Get style and content mini-batches $\{X_i, w_i\}_{i=1}^{N_B}$ and $\{t^i\}_{i=1}^{N_B}$
  $\mathcal{L}_d \leftarrow$ Eq. 5.4 ▷ Real and generated samples $x \sim \{\mathcal{X}, \bar{\mathcal{X}}\}$
  $\mathcal{L}_{w,r} \leftarrow$ Eq. 5.5 + Eq. 5.6 ▷ Real samples $x \sim \mathcal{X}$
  $\Theta_D \leftarrow \Theta_D + \Gamma(\nabla_{\Theta_D} \mathcal{L}_d)$
  $\Theta_{W,R} \leftarrow \Theta_{W,R} - \Gamma(\nabla_{\Theta_{W,R}} \mathcal{L}_{w,d})$
  $\mathcal{L} \leftarrow$ Eq. 5.7 ▷ Generated samples $x \sim \bar{\mathcal{X}}$
  $\Theta_H \leftarrow \Theta_H - \Gamma(\nabla_{\Theta_H} \mathcal{L})$
until Max training iterations T

proposed method to generalize from short words to longer text-lines. Thus, the method is updated in Figure 5.2. X_i becomes the handwritten text-line images from writer w_i and t is a text-line string sampled from an external language corpus.

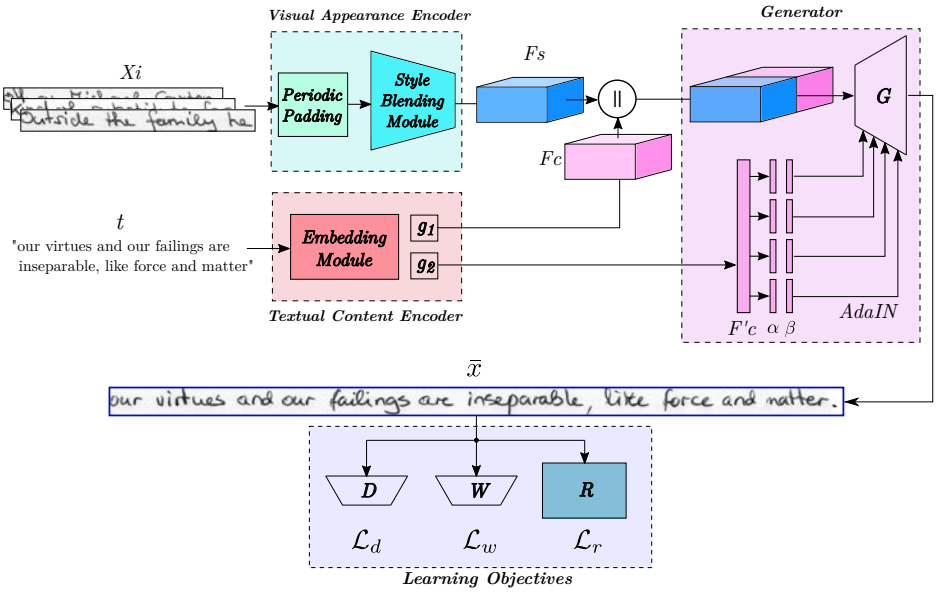


Figure 5.2: Architecture of the proposed handwriting synthesis model. It consists of a Visual Appearance Encoder (green box), a Textual Content Encoder (red box), a Generator (magenta box) and learning objectives (blue box). X_i and t are the images and text string input, respectively. The $\bar{x}$ is the generated sample that shares the visual appearance with X_i and contains the textual information with t.

Periodic Padding Module

As the style image samples X_i have varied shapes, they are firstly resized to the same 64 pixels height while keeping the aspect ratio. Let L be the maximum length of both input and output images. In the HTR literature, the usual technique to align all the images to have the same length in a mini-batch is to add 0-padding to the right of each image until reaching the maximum length L. We have experimentally observed that 0-padding has a severe impact on the handwritten text-line image generation process, which can easily collapse in terms of style in the padded regions. This is especially important when there is a huge difference in the length of input images. The style representations F_s contain not only the visual appearance attributes, but also the spatial information. Thus, the padding would make longer texts to loose the handwritten style consistency in the final generated output. To overcome this problem, we introduce a simple periodic padding module, which consists in repeating the input image several times to the right until the length fits the maximum width L. An example is shown in Figure 5.3. Bear in mind that the style images X_i are only used to extract style features, which are completely independent from the textual content in the image.

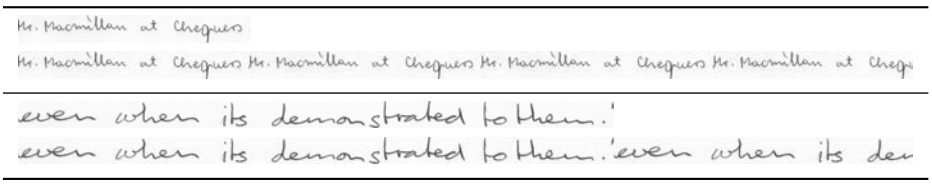


Figure 5.3: Periodic padding example. Given a real image, periodic padding to the right is applied several times until the maximum image width L is reached.

5.4 Variable-length Fréchet Inception Distance

The Fréchet Inception Distance (FID) [39] is the metric that calculates the distance between two feature vectors, which are obtained from two image sets. FID has been widely used in evaluating the performance of GANs at image generation. This metric follows two steps: first, it extracts features from an InceptionV3 network [67] while keeping activations of the last pooling layer, which is pretrained on the ImageNet dataset [32]; then, it calculates the distance between the feature vectors. Even though FID has been widely used for the evaluation of the generated natural scene images, it is not well suited for handwritten image data. The main drawbacks in such case are (i) the ImageNet dataset consists of natural scene image samples that have very few common features with handwritten text images; (ii) the InceptionV3 model used by the FID requires a fixed size input, which could not handle the variable-length scenario of handwritten text images. Thus, we introduce a novel version of FID, namely vFID (Variable-length Fréchet Inception

Distance), specially suited for such variable-length images such as handwritten text images. Similarly to the original FID, the proposed metric vFID share the same InceptionV3 network as the convolutional backbone. However, instead of the average pooling used by the FID, we first reshape the convolutional feature into a 2-dimensional feature map which is then fed into a Temporal Pyramid Pooling (TPP) layer [150] as shown in Figure 5.4. TPP is especially useful when the input is a variable-length sequence of features, which is the case for handwritten text-line images. Based on the pretrained InceptionV3, we fine-tune the vFID model with the IAM dataset by fitting a writer classifier.

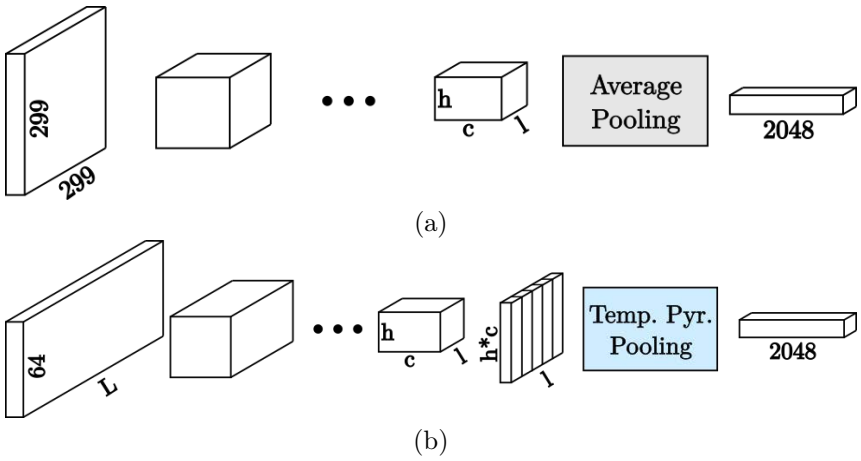
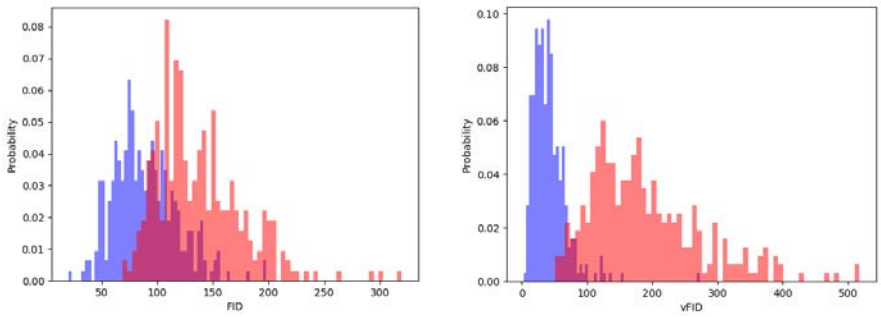


Figure 5.4: (a) Inception module of FID with Average Pooling, (b) Updated Inception module of vFID with Temporal Pyramid Pooling.

The performance comparison of FID and vFID for the IAM dataset is shown in Figure 5.5. The blue distribution indicates the performance for the same writer, while the red one indicates the performance for a different writer pair. The lower value of the FID/vFID, the better similarity is obtained. Therefore, we aim to achieve a robust metric that produces a lower value for the same writer (blue) and a higher value for the different writers (red). In Figure 5.5(a), we observe that the performance of FID do not have a good behaviour since it has a big overlapping area, so that it cannot provide a reasonable judgement on the performance of the generated text. Contrary, our proposed vFID in Figure 5.5(b) could provide a more trustful measure.

5.5 Experiments

In this section, we present the extensive evaluation of our proposed approach. First, we perform several ablation studies on the key modules to find the best balance between performance and efficiency. Then, we demonstrate qualitative



(a) (b)

Figure 5.5: Histogram of FID (a) and vFID (b). The x-axis indicates the FID/vFID values, and the y-axis indicates the counts. The FID/vFID between subsets of samples in the same writer is shown in blue, and between different writers in red. The distribution of blue and red should be apart as far as possible. Both histograms are normalized to sum up to one.

and quantitative results on synthetically generated images. Finally, we make use of the generated samples to boost the HTR performance in different experimental settings. Moreover, we modify the proposed method into an image-to-image translation setting to perform the disentanglement between textual content and handwriting style, which is then equipped into a normal recognizer to boost HTR performance.

5.5.1 Datasets and Metrics

The IAM offline dataset [103], the Rimes dataset [5] and the Spanish Numbers dataset [142] are utilized in our experiments as shown in Table 5.1. However, our proposed generative method is only trained with IAM dataset. All the three datasets are utilized in the HTR experiments. Note that we only use IAM dataset for the word-level experiments. The word and text-line examples are shown in Figure 5.6 and Figure 5.7, respectively.

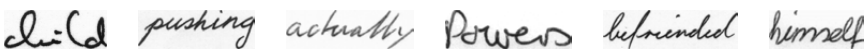


Figure 5.6: Real word samples in IAM dataset. Each example has different characteristics such as shear, stroke width, language, etc.

In the word level setting, a subset of 22,500 unique English words from the Brown [13] corpus is utilized as the source of strings for the content input. While in the text-line level setting, WikiText-103 [107] is chosen to be our external text corpus when selecting random text strings as textual input. We select texts in WikiText-103 from one word to N_t words to create sentences. We end up with 3.6

Table 5.1: Overview of the datasets used in our HTR experiments: Number of images used for training, validation and test sets, and number of writers.

Type	Dataset	Train	Val.	Test	Writers	Language
Word	IAM [103]	55081	8895	25920	657	English
Line	IAM [103]	6482	976	2914	657	English
	Rimes [5]	11333	–	778	1300	French
	Spanish Num. [142]	298	–	187	30	Spanish

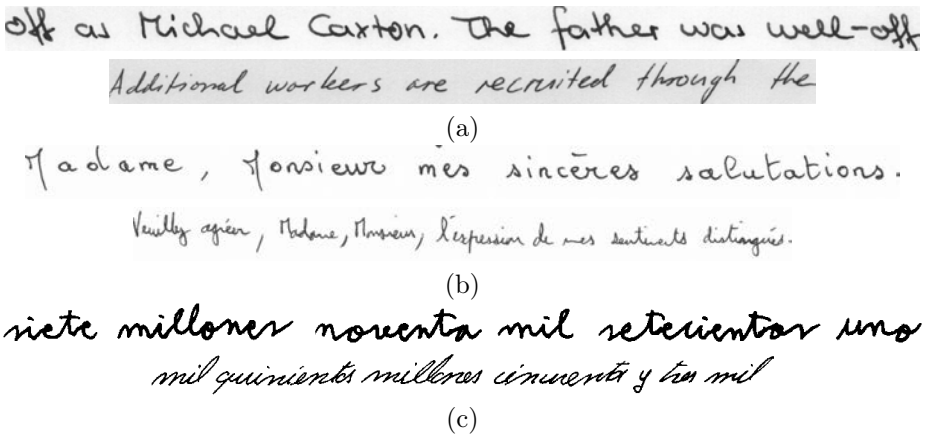


Figure 5.7: Text-line examples of the IAM, Rimes and Spanish Numbers datasets are shown in (a), (b) and (c), respectively.

million text-lines with number of characters varying from 1 to 88.

The *Character Error Rate* (CER) and the *Word Error Rate* (WER) [47] are the performance measures. The CER is computed as the Levenshtein distance, which is the sum of the character substitutions (S_c), insertions (I_c) and deletions (D_c) that are needed to transform one string into the other, divided by the total number of characters in the groundtruth (N_c). Formally,

$$CER = \frac{S_c + I_c + D_c}{N_c} \tag{5.9}$$

Similarly, the WER is computed as the sum of the word substitutions (S_w), insertions (I_w) and deletions (D_w) that are required to transform one string into the other, divided by the total number of words in the groundtruth (N_w). Formally,

$$WER = \frac{S_w + I_w + D_w}{N_w} \tag{5.10}$$

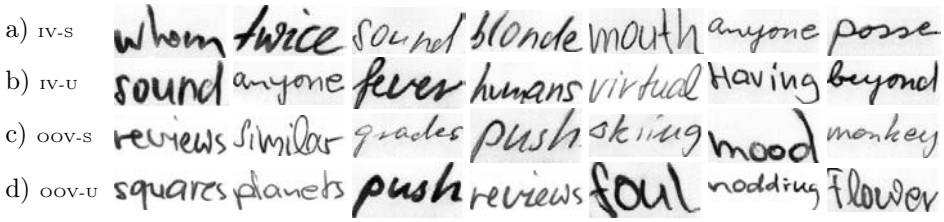


Figure 5.8: Word image generation. a) In-Vocabulary (IV) words and seen (S) styles; b) In-Vocabulary (IV) words and unseen (U) styles; c) Out-of-Vocabulary (OOV) words and seen (S) styles and d) Out-of-Vocabulary (OOV) words and unseen (U) styles.

5.5.2 Word Level Experiments

To carry out the different experiments, we have used a subset of the IAM corpus [103] as our multi-writer handwritten dataset $\{\mathcal{X}, \mathcal{Y}, \mathcal{W}\}$. It consists of $62,857$ handwritten word snippets (55,081, 8,895 and 25,920 word images for training, validation and test sets respectively according to RWTH Aachen partition), written by 500 different individuals (283, 56, 161 writers for training, validation and test sets respectively). Each word image has its associated writer and transcription metadata. A test subset of 160 writers has been kept apart during training to check whether the generative model is able to cope with unseen calligraphic styles. We have also used a subset of $22,500$ unique English words from the Brown [13] corpus as the source of strings for the content input. A test set of 400 unique words, disjoint from the IAM transcriptions has been used to test the performance when producing OOV words. To quantitatively measure the image quality, diversity and the ability to transfer style attributes of the proposed approach we will use the Fréchet Inception Distance (FID) [67, 21], measuring the distance between the Inception-v3 activation distributions for generated $\bar{\mathcal{X}}$ and real samples $\mathcal{X}$ for each writer w_i separately, and finally averaging them. Inception features, trained over ImageNet data, have not been designed to discern between different handwriting images. Although this measure might not be ideal to evaluate our specific case, it will still serve as an indication of the similarity between generated and real images.

In order to evaluate the effectiveness on the recognition performance, we use the standard error measures at character and word level. The Character Error Rate (CER) and the Word Error Rate (WER) [47], are computed as the Levenshtein distance at either character or word level. Since we focus our experiments on individual words, the WER measure is the inverse of the overall word accuracy.

Table 5.2: FID between generated images and real images of corresponding set.

	Real images	IV-S	IV-U	OOV-S	OOV-U
FID	90.43	120.07	124.30	125.87	130.68

Generating Handwritten Word Images

We present in Fig. 5.8 an illustrative selection of generated handwritten words. We appreciate the realistic and diverse aspect of the produced images. Qualitatively, we observe that the proposed approach is able to yield satisfactory results even when dealing with both words and calligraphic styles never seen during training. But, when analyzing the different experimental settings in Table 5.2, we appreciate that the FID measure slightly degrades when either we input an OOV word or a style never seen during training. Nevertheless, the reached FID measures in all four settings satisfactorily compare with the baseline achieved by real data.

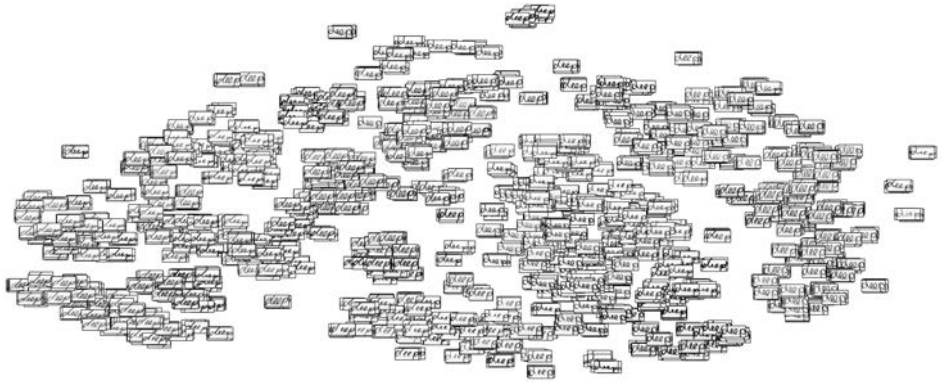


Figure 5.9: t-SNE embedding visualization of 2.500 generated instances of the word `"deep"`.

In order to show the ability of the proposed method to produce a diverse set of generated images, we present in Fig. 5.9 a t-SNE [101] visualization of different instances produced with a fixed textual content while varying the calligraphic style inputs. Different clusters corresponding to particular slants, stroke widths, character roundnesses, ligatures and cursive writings are observed.

To further demonstrate the ability of the proposed approach to coalesce content and style information into the generated handwritten word images, we compare in Fig. 5.10 our produced results with the outcomes of the state-of-the-art approach FUNIT [97]. Being an image-to-image translation method, FUNIT starts with a content image and then injects the style attributes derived from a second sample

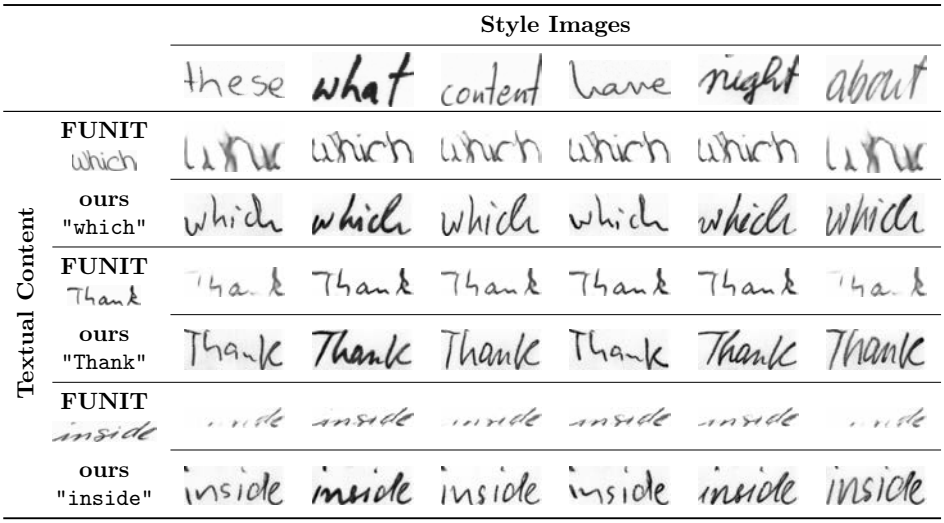


Figure 5.10: Comparison of handwritten word generation with FUNIT [97].

image. Although FUNIT performs well for natural scene images, it is clear that such kind of approaches do not apply well for the specific case of handwritten words. Starting with a content image instead of a text string confines the generative process to the shapes of the initial drawing. When infusing the style features, the FUNIT method is only able to deform the stroke textures, often resulting in extremely distorted words. Conversely, our proposed generative process is able to produce realistic and diverse word samples given a content text string and a calligraphic style example. We observe how for the different produced versions of the same word, the proposed approach is able to change style attributes as stroke width or slant, to produce both cursive words, where all characters are connected through ligatures as well as disconnected writings, and even render the same characters differently, *e.g.* note the characters `n` or `s` in `"Thank"` or `"inside"` respectively.

Latent Space Interpolations

The generator network G learns to map feature points F in the latent space to synthetic handwritten word images. Such latent space presents a structure worth exploring. We first interpolate in Fig. 5.11 between two different points F_s^A and F_s^B corresponding to two different calligraphic styles w_A and w_B while keeping the textual contents t fixed. We observe how the generated images smoothly adjust from one style to another. Again note how individual characters evolve from one typography to another, *e.g.* the `l` from `"also"`, or the `f` from `"final"`.

Contrary to the continuous nature of the style latent space, the original content

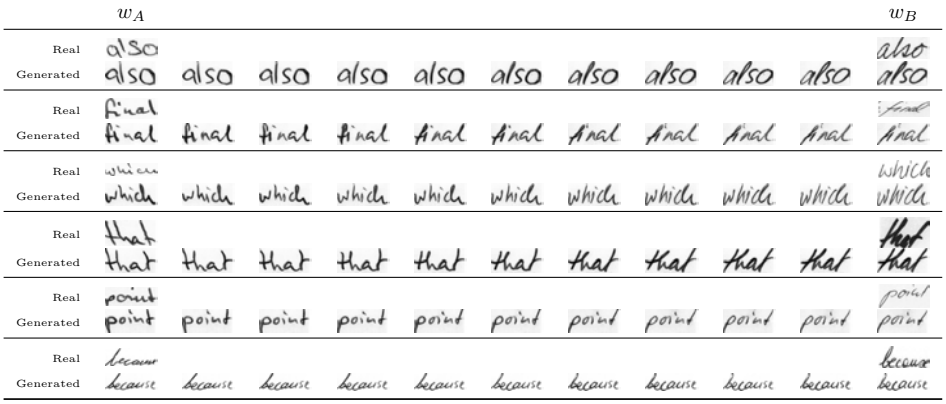


Figure 5.11: Latent space interpolation between two calligraphic styles for different words while keeping contents fixed.

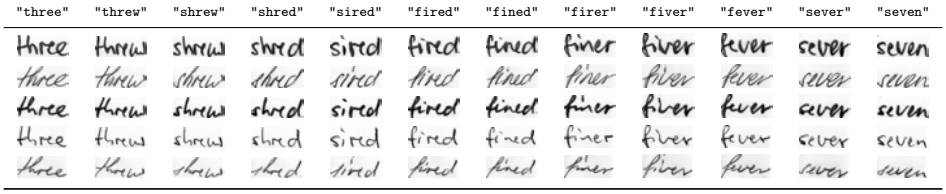


Figure 5.12: Word ladder. From `"three"` to `"seven"` changing one character at a time, generated for five different calligraphic styles.

space is discrete in nature. Instead of computing point-wise interpolations, we present in Fig. 5.12 the obtained word images for different styles when following a "word ladder" puzzle game, *i.e.* going from one word to another, one character difference at a time. Here we observe how different contents influence stylistic aspects. Usually `s` and `i` are disconnected when rendering the word `"sired"` but often appear with a ligature when jumping to the word `"fired"`.

Impact of the Learning Objectives

Along this chapter, we have proposed to guide the generation process by three complementary goals. The discriminator loss $\mathcal{L}_d$ controlling the genuine appearance of the generated images $\bar{x}$. The writer classification loss $\mathcal{L}_w$ forcing $\bar{x}$ to mimic the calligraphic style of input images X_i. The recognition loss $\mathcal{L}_r$ guaranteeing that $\bar{x}$ is readable and conveys the exact text information t. We analyze in Table 5.3 the effect of each learning objective.

The sole use of the $\mathcal{L}_d$ leads to constantly generating an image that is able to fool the discriminator. Although the generated image looks like handwritten

Table 5.3: Effect of each different learning objectives when generating the content $t =$ `"vision"` for different styles.

$\mathcal{L}_d$	$\mathcal{L}_w$	$\mathcal{L}_r$	FID	Style Images			
				these	what	have	about
✓	-	-	364.10				
✓	✓	-	207.47				
✓	-	✓	138.80				
✓	✓	✓	**130.68**				

strokes, the content and style inputs are ignored. When combining the discriminator with the auxiliary task of writer classification $\mathcal{L}_w$, the produced results are more encouraging, but the input text is still ignored, always tending to generate the word `"the"`, since it is the most common word seen during training. When combining the discriminator with the word recognizer loss $\mathcal{L}_r$, the desired word is rendered. However, as in [3], we suffer from the mode collapse problem, always producing unvarying word instances. When combining the three learning objectives we appreciate that we are able to correctly render the appropriate textual content while mimicking the input styles, producing diverse results. We appreciate that the FID measure also decreases for each successive combination.

Comparison with SOTA

we have shown the comparison results with the state-of-the-art methods on handwritten text generation in Table 5.4.

Human Evaluation

Finally, we also tested whether the generated images were actually indistinguishable from real ones by human judgments. We have conducted a human evaluation study as follows: we have asked 200 human examiners to assess whether a set of images were written by a human or artificially generated. Appraisers were presented a total of sixty images, one at a time, and they had to choose if each of them was real of fake. We chose thirty real words from the IAM test partition from ten different writers. We then generated thirty artificial samples by using OOV textual contents and by randomly taking the previous writers as the sources for the calligraphic styles. Such sets were not curated, so the only filter was that the generated samples had to be correctly transcribed by the word recognizer network R. In total we collected $12,000$ responses. In Table 5.5 we present the confusion matrix of the human assessments, with Accuracy (ACC), Precision (P), Recall (R),

Table 5.4: Qualitative comparison with Alonso *et al.* [3], Fogel *et al.* [45], and Davis *et al.* [30], whose images are cropped from their papers.

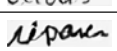
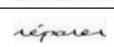
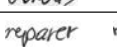
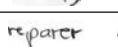
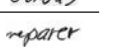
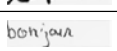
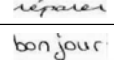
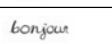
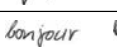
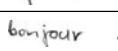
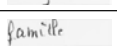
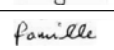
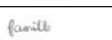
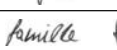
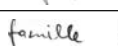
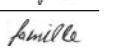
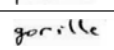
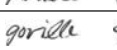
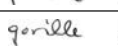
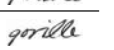
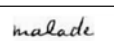
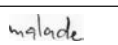
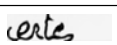
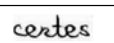
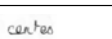
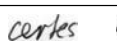
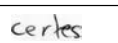
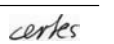
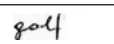
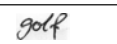
Content	[3]	[45]	[30]	Ours		
				Style *A*	Style *B*	Style *C*
"olibus"						
"reparer"						
"bonjour"						
"famille"						
"gorille"						
"malade"						
"certes"						
"golf"						
"des"						
"ski"						
"le"						

Table 5.5: Human evaluation plausibility experiment.

Actual	Predicted		
	Real	Fake	
Genuine	27.01	22.99	R: 54.1
Generated	27.69	22.31	FPR: 55.4
	P: 49.4	FOR: 50.8	ACC: 49.3

a) Confusion matrix (%)

b) Accuracy distribution

False Positive Rate (FPR) and False Omission Rate (FOR) values. The study revealed that our generative model was clearly perceived as plausible, since a great portion of the generated samples were deemed genuine. Only a 49.3% of the images were properly identified, which shows a similar performance than a random binary classifier. Accuracies over different examiners were normally distributed. We also observe that the synthetically generated word images were judged more often as being real than correctly identified as fraudulent, with a final FPR of 55.4%.

5.5.3 Text-line Level Experiments

Curriculum Learning Strategy

The line-level IAM dataset is used to train our generative method. It is a multi-writer dataset in English, which consists of $1,539$ scanned pages written by 657 writers, as detailed in Table 5.1. Since we can access to the groundtruth of the training data, including the bounding-boxes at word level, we could enlarge the training set using the N-gram cropping strategy. For example, given a sequence of words, we can iteratively crop out one-word, two-words, and so on until N-word sub-lines, where N is the maximum number of words in the given text-line. Thus, given the normalized height of 64 pixels, we end up with $598,489$ images with variable lengths from 64 to 2160 pixels and the number of characters from 1 to 88, where 2160 and 88 are the maximum image length and text length for IAM dataset, respectively.

Table 5.6: Three categories of the IAM offline dataset, from short to long text-lines.

Category	Num. of chars.	Image length
1	$1-24$	$64-600$
2	$24-48$	$600-1200$
3	$48-88$	$1200-2160$

To achieve a better handwriting generation with fine-grained details, we make use of curriculum learning strategy by splitting training data into 3 categories as shown in Table 5.6, from shorter to longer sentences. We start the training with data of Category 1 from scratch, then we fine-tune with data of Category 2, and finally we fine-tune with data of Category 3. The training is done step by step with increasing difficulty in the sense of image and text length. Note that the data used in the previous training step does not appear in the next fine-tuning step considering the training speed. In practice, the second and third steps just need to be fine-tuned for few epochs.

Updated Modules with Ablation Study

First, we compare the generated samples with and without Periodic Padding Module as shown in Figure 5.13. The style input is randomly selected from a specific writer, and it may be a shorter image than what we expect to generate. In the upper part of Figure 5.13, the style input is padded with 0 to the maximum width, so that the generated image suffers from the style collapse problem in the corresponding padded area. Contrary, in the lower part of Figure 5.13, with the periodic padding process, the style image has been extended to the maximum width that is sure of covering all the possibly generated area. Thus, the generated sample keeps

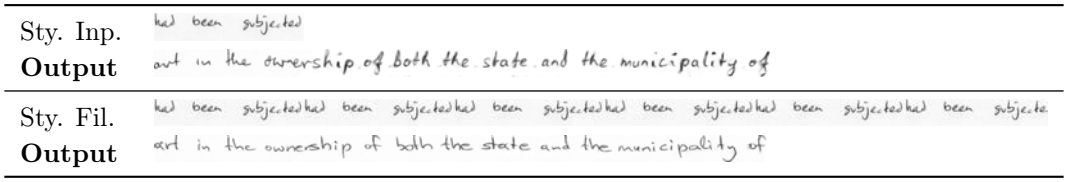


Figure 5.13: Comparison of the generated results for the same text string "art in the ownership of both the state and the municipality of" with and without the periodic padding process.

the consistency in the visual appearance from the first character until the end.

Second, we modify the convolutional layers from VGG19 to ResNet34, and study the performance and training speed. The performance is evaluated on the generated samples based on the style information from IAM test set and content information from a subset of WikiText-103. The models are trained until 500 epochs. Speed is the total time for a forward and backward pass. From Table 5.7, we observe that ResNet34 achieves a higher training speed while obtaining a slightly better performance.

Table 5.7: Ablation study for Convolutional layers on the IAM test set.

Conv.	vFID	Speed (ms)
VGG19	115.46	144.13
ResNet34	**114.39**	**136.80**

Table 5.8: vFID performance on generating different length of images for the sequence-to-sequence and Transformer-based HTR methods. The lower the value, the better the performance.

Method	**Num. of chars to be generated**	
	1-10 (words)	1-90 (lines)
Seq2Seq	**136.51**	249.94
Transfo.	146.74	**114.39**

Finally, we analyze the effect of replacing the sequence-to-sequence recognizer with the Transformer-based recognizer. Based on the number of characters to be rendered in the generated samples, we have two categories: words with 1 up to 10 characters and text-lines with 1 up to 90 characters, as shown in Table 5.8. From the Table we observe that sequence-to-sequence-based method performs well at word level but it significantly degrades when extending to text-lines. Contrary, the Transformer-based HTR method achieves a better performance when dealing with longer text sequences. Since the transformer network has the ability of dealing with long-term dependencies, it becomes more powerful to control the textual

content of the generated samples.

Latent Space Interpolation

Once the system is trained, the generator G has learned a map in the handwriting style latent space. Each writer corresponds to a point in this latent space and different writers are connected in a continuous way. Thus, we can explore it by randomly choosing two writers and try to traverse between the corresponded two points in the style latent space as shown in Figures 5.14 and 5.15. The first and last rows show the real samples from writer A and B, respectively. The samples in between are synthetically generated ones that try to traverse from writer A to B. The rendered text is the quote of "our virtues and our failings are inseparable, like force and matter" from Nikola Tesla, which has not been seen during training.

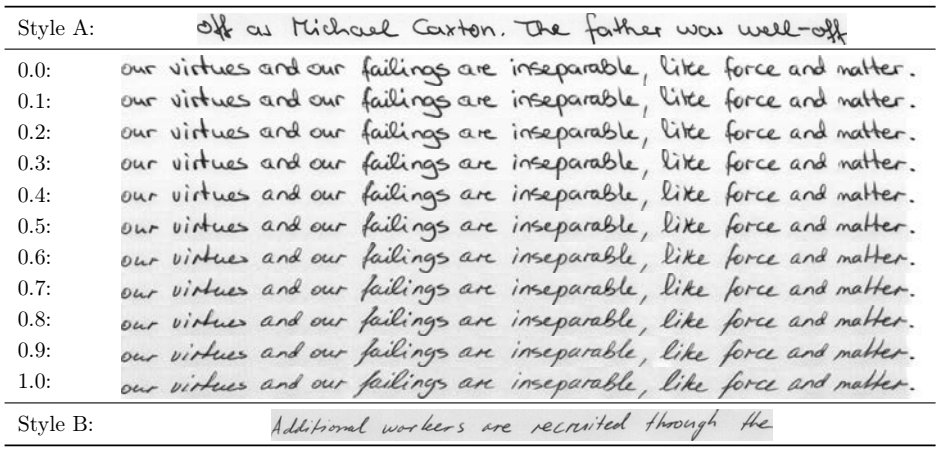


Figure 5.14: Example of interpolations in the style latent space.

Conditioned Handwritten Text-line Generation

For the qualitative experiments, we show the results in two cases. First, given a same writing style, we try to generate samples with different text strings. Second, given a specific text string, we try to generate samples in different writing styles. The first case is shown in Figure 5.16. The text string is the quote of "the progressive development of man is vitally dependent on invention." from Nikola Tesla. We translate it into German, French and Spanish while replacing special characters with the corresponding letters (e.g. "é" to "e"). In Figure 5.16, the first row is a sample of the style input, and the following rows are (text string, synthetically generated sample) pairs. By showing the generation on different languages, our method proves to be not restricted to a training corpus nor a language model.

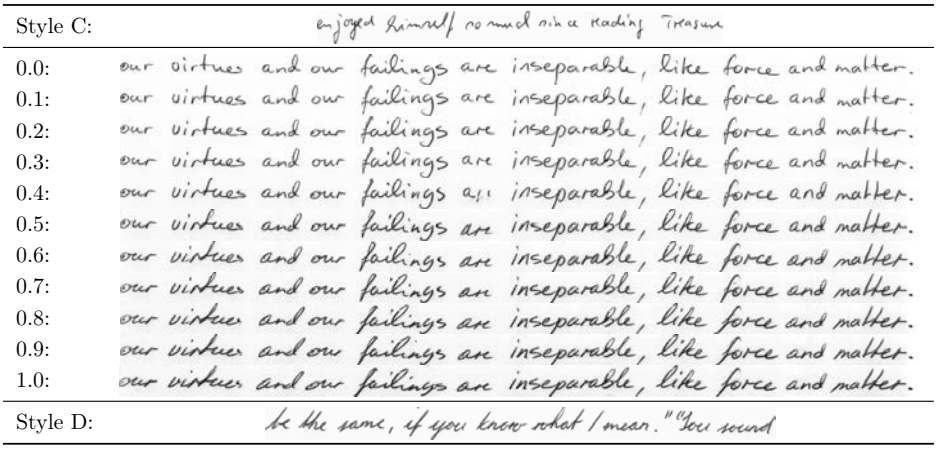


Figure 5.15: Example of interpolations in the style latent space.

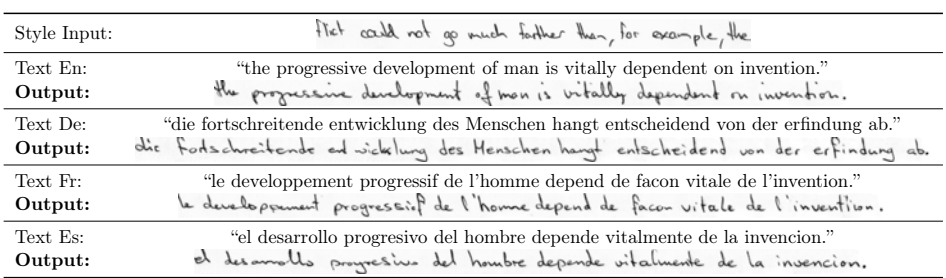


Figure 5.16: Generation on varied texts of multi-lingual cases.

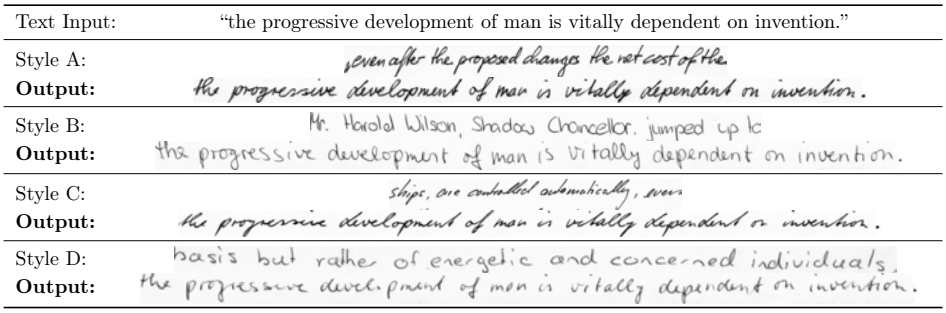


Figure 5.17: Generation on varied styles.

Thus, this method can be applied to generate any OOV words and sentences. The second case is shown in Figure 5.17. The first row is the text string input, and the following rows are (handwriting style sample, synthetically generated sample)

pairs. From the results we observe that our method has the ability to generate text-line samples with diverse writing styles.

Furthermore, we show a comparative with the state-of-the-art methods on handwritten text generation in Table 5.4.

HTR Performance Improvement

The main objective of this test is to prove that the generated text-lines can indeed be used as training data in order to the HTR performance at text-line level. For this purpose, we define three settings: first, a conventional supervised learning on the IAM dataset; second, transfer learning from the IAM to the Rimes dataset; and third, a realistic few-shot setting on the Spanish Numbers dataset. To be fairly comparable, we do not use any data augmentation techniques nor pretrained modules.

Enhance the training set

The most straightforward way to improve the HTR performance is to incorporate extra synthetically generated data to the training set. In Table 5.9, we evaluate the improvements in different cases. The first row shows the results when using the IAM training set only. To keep the training data balanced between real and generated samples, we generate $8,000$ text-line images based on the style of the IAM images and a lexicon. Concerning the lexicon, we have two choices: WikiText-103 or the groundtruth of IAM training set, shown in the second and third row, respectively. Note that the HTR performance is boosted when adding the $8,000$ synthetically generated samples. Furthermore, the choice of lexicon also matters because the performance is further boosted when using a similar lexicon to the target dataset. Finally, we even apply data augmentation techniques on both the real and generated training samples so that we end up with the best result as shown in the fourth row. Furthermore, our method shows a significant improvement over the performance achieved by ScrabbleGAN [45] in comparable settings. Thus, we can conclude that our proposed generative method generates useful samples that are useful to train HTR networks.

Transfer learning on a new dataset

Another useful setting is transfer learning, which consists of transferring a trained recognizer to an unknown dataset. In this case, the source data is the IAM dataset and the target is the Rimes dataset. Both datasets are at text-line level and share some characters in vocabularies such as English letters, space, punctuation marks and numbers. However, the IAM dataset is in English while the Rimes dataset is in French, so some special letters like “é” or “â” are exclusive from the Rimes dataset. This scenario may occur in real use cases in which there is a *general* recognizer, which has been properly trained, that is used to recognize a target dataset, containing some exclusive characters. Instead of manually labeling a subset of target data and training the recognizer again, we could provide a faster

Table 5.9: HTR experiments. Results are evaluated on the IAM test set at text-line level. The Error rate reduction is calculated taking the results of the first and last rows.

Aug.	GAN	Lexicon	ScrabbleGAN [45]		**Proposed**	
			CER	WER	CER	WER
–	–	IAM	13.82	25.10	10.46	33.40
–	✓	WikiText			9.66	31.87
–	✓	IAM			9.37	30.58
✓	✓	IAM	13.57	23.98	8.62	26.69
Error Rate Reduction (%)			1.8	4.5	**17.6**	**20.1**

solution: to generate a set of synthetic samples mimicking the style of target dataset and then fine-tuning on it. In this way, the HTR performance on the target data is boosted to some extent with a manual-free effort, although it can not recognize those special characters.

Table 5.10 shows the transfer learning results. In the first row, considered as a lower bound, the recognizer is pretrained on the IAM dataset and directly evaluated on the Rimes test set. As an upper bound, we train the recognizer from scratch using the Rimes dataset. Below these two baselines, we show the performance when using the IAM training set and $8,000$ synthetically generated samples using IAM handwriting styles and random text strings from WikiText-103. Secondly, we assume that we have access to images of the Rimes dataset (but not their labels), and we generate $8,000$ synthetic samples that mimic the style of the Rimes dataset while sampling text strings from WikiText-103. We observe that by incorporating these extra synthetically generated samples, the HTR performance for the unlabeled Rimes target dataset is boosted in a transfer learning setting (the CER decreases from 27.3% down to 18.19%).

Table 5.10: Transfer learning setting from IAM to Rimes. Results are evaluated on Rimes test set at text-line level. Only the second row has access to labeled Rimes data, while the Adaptation indicates the usage of unlabeled images and external text strings.

	Train set	Adaptation	CER	WER
Baselines	IAM	–	27.30	74.57
	Rimes	–	6.45	19.56
Transfer	IAM	IAM + WikiText (8K)	20.55	63.20
	IAM	Rimes + WikiText (8K)	18.19	54.83

Few-shot setting on target dataset

We are also interested in investigating how to make use of the generated images

to improve the HTR performance in another realistic scenario: when the target dataset is very small, such as the Spanish Number dataset. Here, we take our baseline method, a recognizer pretrained on IAM dataset, and test it with the test set of Spanish Number data directly, so that we obtain the lower bound with CER 27.82% as shown as the dashed black line in Figure 5.18. We hypothesize that we have access to the whole labeled training set of Spanish Number data (298 images), thus we further fine-tune our pretrained recognizer and achieve a CER of 4.94%, which is the ideal case as shown as the dashed magenta line. Then, we select 5, 10, 20, 40, 80, 160 labeled real samples from the Spanish Number training set randomly to carry on the next experiments. Based on our baseline recognizer, we fine-tune on the selected subset of labeled real images to end up with the red curve. Ten individual experiments have been done for each subset of labeled real data, and the data sampling process is also randomly done for every experiment. From the red curve, we can see that the performance is significantly improved with few labeled real samples, while remaining steady when adding more data.

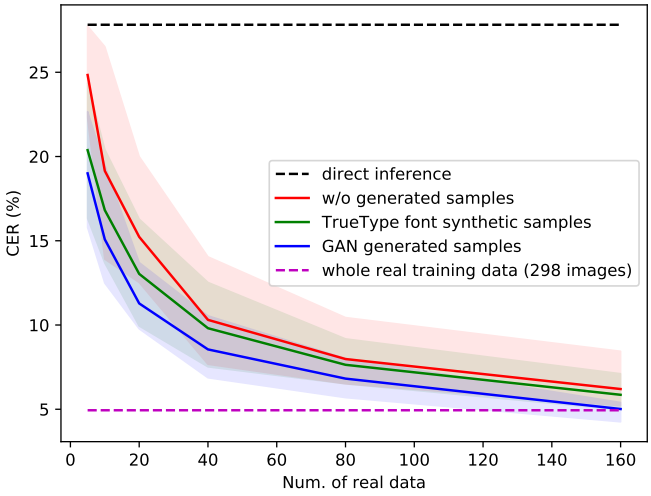


Figure 5.18: HTR improvement in a real use case on Spanish Number dataset.

For comparisons, we carry on experiments with a sequence-to-sequence method that uses synthetic handwritten images based on TrueType fonts as detailed in Chapter 2. To avoid a large unbalance between synthetic and real data, we make use of 100, 300, 500, 700 and 900 synthetic data with a specific amount of real subset (indicated as x-axis) to fine-tune the recognizer. We carry on 10 individual experiments with randomly selected synthetic and real subsets, so that we obtain the green curve that behaves better than the red one. The results prove that using extra synthetic data along the training set boosts the HTR performance. However,

the handwriting style diversity that the synthetic data provides is very limited to the chosen fonts, so the improvement is also limited.

Since we already have the generative model pretrained on the IAM dataset, we produce synthetic samples based on the unlabeled Spanish Number images and random text strings of WikiText-103. We follow the same experimental setting with the TrueType font based experiments, and generate the blue curve. We observe that our generated samples significantly boost the HTR performance over both the red and green curves. Even more, when using our generated samples with 160 labeled real ones, the recognizer performs better than when using the whole real training set (298 images).

5.6 Disentanglement in Image-to-Image Setting

5.6.1 Disentanglement Method

As explained in Section 5.3.2, the method takes images and text strings as input. Following an image-to-image translation setting, we modify the input as two images. One image is in charge of providing textual content information, while the other one provides handwriting style information. This modification is based on our hypothesis that every handwritten text image can be disentangled into textual content and handwriting style features. Thus, by joining this disentanglement module into a Seq2Seq recognizer, we end up with an end-to-end method for HTR tasks as shown in Figure 5.19. To note that we have simplified the combination process between content feature F_c and style feature F_s with only AdaIN normalization layer. The whole disentanglement process is defined as

$$\bar{x}_{ij} = G(\bar{F}_c, F_s) = G(\bar{C}(x_i), S(x_j)), \tag{5.11}$$

where $x_i, x_j \in \mathcal{X}$. Thus, $\bar{x}_{ij}$ is expected to contain the same textual content of x_i and to share calligraphic style attributes with x_j.

5.6.2 Experiments

Dataset and Performance Evaluation

To carry our experiments, we have chosen the IAM offline handwriting dataset [103], being one of the most popular and widely used benchmarks in the field of handwriting recognition. We have used the RWTH Aachen partition for the dataset, composed of 55,081, 8,895 and 25,920 word images for training, validation and test sets respectively. Furthermore, the IAM dataset provides not only text images and their corresponding transcriptions, but also the writer identifier. Based on the assumption that each writer has one specific writing style, we have 500

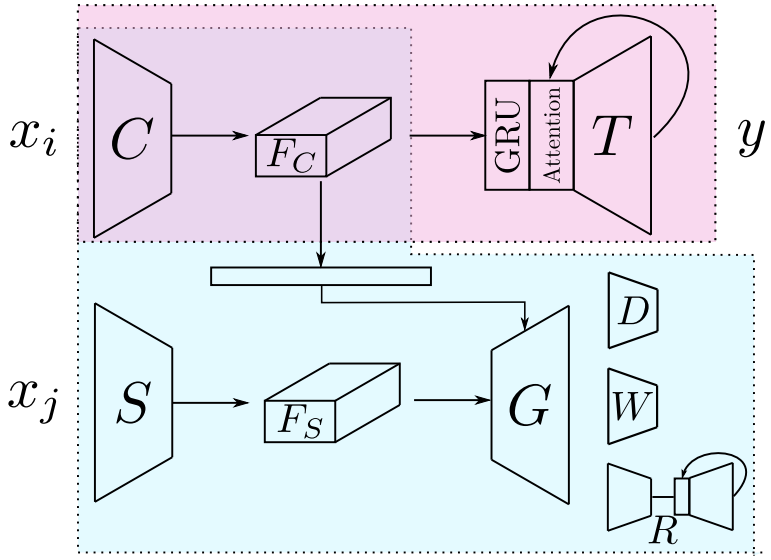


Figure 5.19: Architecture of the proposed model with HWR module in red box and disentanglement module in blue box. Note that the content encoder C is used by both the HWR module and the disentanglement module.

handwriting styles in the IAM dataset, with 283 writers in the training set, 56 writers for validation and 161 writers in the test set.

In order to evaluate the effectiveness on the recognition performance, we will use the standard error measures at character and word level. Since we focus our experiments on individual words, the WER measure is the inverse of the overall word accuracy.

Qualitative Evaluation of the Generative Process

First and foremost, in order to validate that the proposed method is able to really learn properly disentangled feature representations for both contents and writing styles, from the content and style encoders C and S, we will qualitatively analyze the generative part of our approach.

Given a pair of input word images x_i and x_j, four different permutations, style and content-wise, are possible to be generated. Formally, the generated images $\bar{x}_{ii}$, $\bar{x}_{ij}$, $\bar{x}_{ji}$ and $\bar{x}_{jj}$ will be the result of using either x_i or x_j as input for the content or style encoders (*c.f.* equation 5.11). We show in Table 5.11 some results of such generative process. Images $\bar{x}_{ii}$ and $\bar{x}_{jj}$ shall be the re-writings with the same content and style of x_i and x_j respectively, while $\bar{x}_{ij}$ and $\bar{x}_{ji}$ shall correspond to images conveying the content of x_i while appearing to be written with the style of x_j, and vice-versa. In order to properly transfer both style and content within

the generator, the feature tensors F_c and F_s should be completely disentangled, which means that the content representation F_c should just encode the textual features that conform a certain word completely disregarding the writing style. We observe in such sample results how effectively the learned content and style feature representations have been properly disentangled one from each other. The qualitative results from the generative part, although not being the ultimate goal of this work, are encouraging enough to think that the content features F_c shall be at a certain extent agnostic to the handwriting styles.

Table 5.11: Qualitative results of the content and style guided generative process. Given a pair of input images, we are able to generate the four different permutations of styles and textual contents.

	pair 1	pair 2	pair 3	pair 4	pair 5	pair 6	pair 7	pair 8
x_i								
x_j								
$\bar{x}_{ii}$								
$\bar{x}_{ij}$								
$\bar{x}_{ji}$								
$\bar{x}_{jj}$								

Handwriting Recognition Performance

In order to quantitatively evaluate the performance of the proposed style invariant content features within the recognition pipeline, we present in Table 5.12 some comparative results. In this experiment we have trained exactly the same sequence-to-sequence neural architecture in three different setups that are objectively comparable. On the one hand, we just use the IAM training set images with data augmentation, and we reach a CER value of 6.88%. To push that value forward, we make use of not only IAM training set, but also unlabelled IAM test set to do a domain adaptation between both sets by using the adversarial domain adaptation technique proposed in [49], so that the feature distribution of test set samples shall be properly adapted to that of training samples. We observe that the yielded CER is of 6.75% in that case. We finally jointly train the recognizer with the generative pipeline in an end-to-end fashion, and, we just use IAM training samples, without any other additional image. We observe that the obtained error rates are lower than the two previous approaches, 6.43% and 16.39% respectively, indicating that the learned features are actually better focused to the conveyed

textual contents, while being resilient to handwriting style changes.

Table 5.12: Recognition performance for sequence-to-sequence network with three different training strategies.

Training Procedure	CER (%)	WER (%)
IAM Training Set [79]	6.88	17.45
Domain Adaptation [78]	6.75	17.26
Content Distillation (Proposed)	**6.43**	**16.39**

Comparison with State of the Art

Finally, in order to put in context the previous performance evaluation measures, we provide in Table 5.13 a comparison with the the state-of-the-art methods for handwritten word recognition. To give a fair comparison, we have just selected works focused at word level, and from them, we report the error rates from those models that do not entail any language model nor closed lexicon. We observe that our proposed approach compares quite satisfactorily with the rest of the state-of-the-art methods. The exception is the approach of Dutta *et al.* [42], however this work provides their results on IAM by pre-training on 9M synthetic data of IIIT-HWS [88], de-slanting word images as pre-processing, and using test-time augmentation, which make such results not directly comparable with the rest of the reported error measures.

Table 5.13: Comparison with the state-of-the-art methods.

Approach	Method	CER (%)	WER (%)
RNN + CTC	Mor *et al.* [112]	–	20.49
	Pham *et al.* [123]	13.92	31.48
	Krishnan *et al.* [86]	6.34	16.19
	Wiginton *et al.* [152]	6.07	19.07
Seq2Seq + Attention	Bluche *et al.* [17]	12.60	–
	Sueiras *et al.* [134]	8.80	23.80
	Zhang *et al.* [160]	8.50	22.20
	Dutta *et al.* [42]	4.88	12.61
	Proposed	6.43	16.39

5.7 Conclusion

We have presented a group of image generation architectures that produce realistic and varied artificially rendered samples of handwritten words and text-lines.

Our pipeline can yield credible text images by conditioning the generative process with both calligraphic style and textual content. Furthermore, by jointly guiding our generator with three different cues: a discriminator, a style classifier and a content recognizer, our model is able to render any input text, not depending on any predefined vocabulary, while incorporating calligraphic styles in a few-shot setup. Experimental results demonstrate that the proposed method yields images with such a great realistic quality that are indistinguishable by humans and can effectively boost the HTR performance.

Furthermore, by modifying the proposed architecture into an image-to-image translation setting, we end up obtaining a novel training approach for handwriting recognition that is able to disentangle the content and style features of input images. The proposed method jointly optimizes a generative process and a handwritten word recognizer with the aim of yielding a style-independent content-centric feature representation that boosts the recognition performance in multi-writer scenarios. The presented results prove that by coupling a generative and a recognition process we are able to achieve separated content and calligraphic stylistic features that serve both to style and content transfer and for better recognition performance.

Chapter 6

Conclusions and Future Work

This thesis has addressed the general task of handwritten text recognition. We have proposed two main groups of methods in either exploring different neural network architectures or diminishing the bias between training and test sets. As far as we know, we are the first to propose transformer-based recognizer and disentanglement method for HTR, and we also propose pioneer works of Seq2Seq recognizer, writer adaptation method and handwriting synthesis method among the contemporary state-of-the-arts. With these proposed method, the recognition performance is largely improved for both scientific dataset and real industrial use cases.

This last Chapter is organized as follows. In Section 6.1, the contributions are summarized. Then, the pros and cons are discussed in Section 6.2. Finally, the future work is outlined in Section 6.3.

6.1 Summary of the Contributions

The main contributions of this thesis are two-folds: on the one hand, the proposal of novel neural network architectures for HTR; on the other hand, the methods for diminishing the bias between training and test sets.

6.1.1 Proposal of Novel Architectures

Seq2Seq Model with Attention

We have applied Seq2Seq model to HTR tasks, whose architecture and modules are carefully chosen so as to achieve state-of-the-art performance. In addition, attention mechanism is studied extensively from content- to location-based methods.

Further more, optimizing strategies such as attention smoothing, multi-nomial decoding and label smoothing are evaluated. To conclude, these explorations contribute to the final state-of-the-art recognizer.

Novel language modeling

We have proposed a novel method to inject a RNN language model into the Seq2Seq recognizer in an end-to-end fashion. It has the advantage of equipping an external language knowledge with the base recognizer effectively, which is prior to the state-of-the-art approaches such as Shallow Fusion and Deep Fusion. In addition, this language modeling is in character-level, which enables recognizing OOV words.

Transformer-based Recognizer

As far as we know, we are the first to propose a transformer-based recognizer for HTR tasks. We have made modifications of the architecture to fit HTR settings, such as a convolutional module that extracts high-level features of variable-length handwritten images and a one-dimensional positional encoding that provides the basic rule of writing from left to right. In addition, we have explored this method in few shot setting, which gives up a much better performance than that of Seq2Seq method. Thus, transformer-based recognizer is a good fit for industrial HTR problems that have limited training data.

6.1.2 Diminishing the Bias between Training and Test Sets

Unsupervised Writer Adaptation

We have proposed an unsupervised method for writer adaptation, which consists of a Seq2Seq recognizer and an auxiliary domain classifier. It does not require extra effort of labeling, so that it is beneficial to further boost the HTR performance in real use cases. This method can be applied to diminish the bias between two domains, which could be from synthetic to target data or from real training to test set.

Disentanglement of Content and Style

As far as we know, we are the first to propose the disentanglement method for HTR tasks. The method consists of both a handwriting recognizer and an image-to-image translation module, which follows the end-to-end fashion. Via our experiments, any handwritten text image can be disentangled into textual content and visual appearance style features using the generative method. By distilling

the content from varied styles, the handwriting recognizer can achieve a better generalization and performance.

Handwriting Synthesis

We have proposed a generative method for handwriting synthesis using GANs, which can produce zillions of synthetic data conditioned on text strings and handwritten images. Text strings can be selected from any text corpus in any language, while the handwritten images are from the target data. Once properly trained, the generative method can adapt to new writers so as to produce realistic handwritten samples mimicking the target style and containing the known texts. Thus, pre-trained on the zillions of synthetic data, the recognizer can adapt to the target data easily and achieve a better performance.

6.2 Discussion

The Seq2Seq recognizer achieves the state-of-the-art performance in word level, but it suffers in text-line level. As the usage of recurrent nets in the Seq2Seq model, the gradient vanishing problem is inevitable and the recognition performance would decrease with longer text input. In addition, the recurrent nets also prohibit the parallel computing during the training process, so that the training speed is slow. To overcome the limitations of Seq2Seq method, we have proposed the transformer-based approach for HTR. Without the usage of recurrent nets, transformer-based recognizer can achieve a faster training speed even with a larger architecture than that of Seq2Seq method. Even more, the transformer-based recognizer is good at dealing with longer text input because of the full parallel architecture. So transformer-based method can achieve a much better performance than that of Seq2Seq method, even in few shot settings. However, to achieve the state-of-the-art performance, the architecture has to be huge to equip enough computing capacity. Furthermore, the evaluation is done in a loop that is quite slow, because it still depends on previous prediction for the current one. To conclude, we can choose either Seq2Seq or transformer method depending on the specific HTR task. If the task consists only short words and sentences, Seq2Seq method is a good option with smaller architecture and faster evaluation. While if the task focuses on long sentences, transformer method is a good choice to achieve a better performance. The balance between efficiency and performance is important especially for industrial use cases.

Unsupervised writer adaptation method is proved to be effective in boosting performance on new target data without labeling. When applying this method to HTR tasks, the balance between source and target data should be considered. In industrial use cases, we always have our own data, which is large in quantity. There are two tips for the real applications: first, if the target data shares similar

visual appearance styles, we can merge all the target data together and do writer adaptation between our own data and the target one; second, if the styles of target data vary a lot from image to image, we need to cluster different mini-groups for the similar style images, then select a similar quantity subset for our own data, and finally do writer adaptation between our own subset and the target mini-group. The limitation of this method is that the writer adaptation process need to be done whenever a new target data comes. To overcome this limitation and propose a robust recognizer, we have proposed the disentanglement method for HTR tasks. Once trained properly, the disentanglement method is expected to distill textual content from any handwritten text images, so that a robust recognizer can be obtained with the textual content feature extractor. However, in practice, the distilled textual content features still contain some visual appearance style information. So it has some limitations to generalize well for any target data. In real use cases, we need to select the training data carefully to cover as much writing styles as we can, so that the disentanglement module is able to access a large distribution of style features. Thus, the textual content feature extractor could be more robust for new target data.

Finally, there is a main limitation for the generative method of handwriting synthesis that the training process asks for the labeled data. It means that we always need to label part of the target data so as to produce more synthetic data with similar visual appearance of target one. Even if we have demonstrated in the experiments that the generative method can be adapted to new target data without training, the performance is not satisfying when the types of training and target data are too different. Thus, in practice, when we try to synthesize modern handwritten images, we should collect the training data with similar modern ones. Similarly, if the task is historical document recognition, we have to collect the training data with similar visual appearance such as other contemporary historical documents.

To conclude, all these methods achieve state-of-the-art performance for scientific datasets. However, when we apply them to real industrial use cases, we need to focus more on the task itself and collect the proper training set to take advantage of the proposed methods.

6.3 Future Work

6.3.1 Tasks

Recently, we are doing and solving simplified tasks for handwritten text recognition. The simplicity is two-folds: first, the bounding boxes are obtained in word or text-line level; second, a hypothesis that the images contain only text inside the bounding boxes.

Full Paragraph Recognition

For the first case that the bounding boxes are in word or text-line level, collecting the training data is a huge effort. Especially in industrial use cases, the customers' documents are varied and huge in quantity. That is why the full paragraph recognition is in a great demand, which asks for much less manual segmentation and labeling efforts. Even more, when the recognition process is extended to full paragraph, a much better performance can be obtained with an external language model as more context information is involved.

Complex Layout Documents

Among the popular handwritten text datasets, the bounding boxes are carefully drawn for each word or text-line. This greatly simplifies the recognition process while on contrary introduces extra time-consuming labeling process. A potential future work would be the complicated document analysis, which consists of figures, tables and texts. In this sense, recognition is not a simple spelling process of character by character, instead, it will become the general understanding of a document with the judgement of text type and spelling.

Effective Combination with Language Model

In this thesis, we have explained that both Seq2Seq method and transformer-based method have an inherent language modeling, which learns from the training text corpus. In addition, we have proposed a novel method to inject an external language model for the Seq2Seq recognizer. This external language model is in character level, so that it can handle the OOV words. However, there are still some problems in the combination between a recognizer and a language model. First, how could we define that the external language model is pre-trained properly without nether over-fit nor under-fit? Due to the gap between the external language corpus and that of the training set, well pre-trained language model on the external text corpus does not guarantee the same performance on the target dataset. Second, character level language modeling delivers the capacity on OOV words, however, the potential power of language modeling might be used in a better way in word or word-piece level. Thus, how could we balance the language modeling between character and word level to achieve the best performance? These open questions worth to research in the future.

6.3.2 Architectures

With the development of neural networks, the architecture continues to evolve from recurrent nets to self-attentions. As we have discussed the limitations for each of the method in previous section, the improvements can be done for each method

accordingly. The general intuition would be obtaining a more robust recognizer with smaller architecture and faster training speed.

List of Publications

Journals

- L. Kang, P. Riba, M. Villegas, A. Fornés and M. Rusiñol. Candidate Fusion: Integrating Language Modelling into a Sequence-to-Sequence Handwritten Word Recognition Architecture. Submitted to *Pattern Recognition*, 2019.
- L. Kang, P. Riba, M. Rusiñol, A. Fornés and M. Villegas. Pay Attention to What You Read: Non-recurrent Handwritten Text-Line Recognition. Submitted to *Pattern Recognition*, 2020.
- L. Kang, P. Riba, M. Rusiñol, A. Fornés and M. Villegas. Content and Style Aware Generation of Text-line Images for Handwriting Recognition. Submitted to *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

International Conferences

- L. Kang, P. Riba, Y. Wang, M. Rusiñol, A. Fornés and M. Villegas. GANwriting: Content-Conditioned Generation of Styled Handwritten Word Images. In *European Conference on Computer Vision* (ECCV), 2020.
- L. Kang, P. Riba, M. Rusiñol, A. Fornés and M. Villegas. Distilling Content from Style for Handwritten Word Recognition. In *International Conference on Frontiers in Handwriting Recognition* (ICFHR), 2020.
- L. Kang, M. Rusiñol, A. Fornés, P. Riba and M. Villegas. Unsupervised Adaptation for Synthetic-to-Real Handwritten Word Recognition. In *IEEE Winter Conference on Applications of Computer Vision* (WACV), 2020.
- L. Kang, J.I. Toledo, P. Riba, M. Villegas, A. Fornés and M. Rusiñol. Convolve, attend and spell: An attention-based sequence-to-sequence model for handwritten word recognition. In *German Conference on Pattern Recognition* (GCPR), 2018.

Code Contribution

- https://github.com/omni-us/research-seq2seq-HTR
- https://github.com/omni-us/research-WriterAdaptation-HTR
- https://github.com/omni-us/research-ContentDistillation-HTR
- https://github.com/omni-us/research-GANwriting

Awards

- Best Paper Award of International Conference on Frontiers of Handwriting Recognition (ICFHR), 2020.

Bibliography

[1] Irfan Ahmad and Gernot A Fink. Training an arabic handwriting recognizer without a handwritten training data set. In *Proceedings of the International Conference on Document Analysis and Recognition*, 2015.

[2] Jon Almazán, Albert Gordo, Alicia Fornés, and Ernest Valveny. Word spotting and recognition with embedded attributes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(12):2552–2566, 2014.

[3] Eloi Alonso, Bastien Moysset, and Ronaldo Messina. Adversarial generation of handwritten text images conditioned on sequences. In *Proceedings of the International Conference on Document Analysis and Recognition*, 2019.

[4] José Carlos Aradillas, Juan José Murillo-Fuentes, and Pablo M Olmos. Boosting handwriting text recognition in small databases with transfer learning. In *Proceedings of the International Conference on Frontiers in Handwriting Recognition*, pages 429–434, 2018.

[5] Emmanuel Augustin, Matthieu Carré, Emmanuèle Grosicki, J-M Brodin, Edouard Geoffrois, and Françoise Prêteux. Rimes evaluation campaign for handwritten mail processing. In *Proceedings of the International Conference on Frontiers in Handwriting Recognition*, 2006.

[6] Samaneh Azadi, Matthew Fisher, Vladimir G Kim, Zhaowen Wang, Eli Shechtman, and Trevor Darrell. Multi-content gan for few-shot font style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7564–7573, 2018.

[7] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[8] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. End-to-end attention-based large vocabulary speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2016.

[9] Christian Bartz, Joseph Bethge, Haojin Yang, and Christoph Meinel. KISS: Keeping it simple for scene text recognition. *arXiv preprint arXiv:1911.08400*, 2019.

[10] Roman Bertolami and Horst Bunke. Hidden Markov model-based ensemble methods for offline handwritten text line recognition. *Pattern Recognition*, 41(11):3452–3460, 2008.

[11] Ayan Kumar Bhunia, Abhirup Das, Ankan Kumar Bhunia, Perla Sai Raj Kishore, and Partha Pratim Roy. Handwriting recognition in low-resource scripts using adversarial learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[12] Anne-Laure Bianne-Bernard, Fares Menasri, Rami Al-Hajj Mohamad, Chafic Mokbel, Christopher Kermorvant, and Laurence Likforman-Sulem. Dynamic and contextual information in HMM modeling for handwritten word recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(10):2066–2080, 2011.

[13] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc., 2009.

[14] Maurits Bleeker and Maarten de Rijke. Bidirectional scene text recognition with a single decoder. *arXiv preprint arXiv:1912.03656*, 2019.

[15] Théodore Bluche. Joint line segmentation and transcription for end-to-end handwritten paragraph recognition. In *Proceedings of the Neural Information Processing Systems Conference*, 2016.

[16] Théodore Bluche, Jérôme Louradour, Maxime Knibbe, Bastien Moysset, Mohamed Faouzi Benzeghiba, and Christopher Kermorvant. The a2ia arabic handwritten text recognition system at the open hart2013 evaluation. In *Proceedings of the IAPR International Workshop on Document Analysis Systems*, 2014.

[17] Théodore Bluche, Jérôome Louradour, and Ronaldo 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.

[18] Théodore Bluche and Ronaldo Messina. Gated convolutional recurrent neural networks for multilingual handwriting recognition. In *Proceedings of the International Conference on Document Analysis and Recognition*, 2017.

[19] Theodore Bluche, Hermann Ney, and Christopher Kermorvant. Tandem HMM with convolutional neural network for handwritten word recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013.

[20] Théodore Bluche, Hermann Ney, and Christopher Kermorvant. A comparison of sequence-trained deep neural networks and recurrent neural networks optical modeling for handwriting recognition. In *Proceedings of International Conference on Statistical Language and Speech Processing*, 2014.

[21] Ali Borji. Pros and cons of GAN evaluation measures. *Computer Vision and Image Understanding*, 179:41–65, 2019.

[22] Horst Bunke, Samy Bengio, and Alessandro Vinciarelli. Offline recognition of unconstrained handwritten texts using hmms and statistical language models. *IEEE transactions on Pattern analysis and Machine intelligence*, 26(6):709–720, 2004.

[23] Bo Chang, Qiong Zhang, Shenyi Pan, and Lili Meng. Generating handwritten Chinese characters using CycleGAN. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, 2018.

[24] Zhuo Chen, Yichao Wu, Fei Yin, and Cheng-Lin Liu. Simultaneous script identification and handwriting recognition via multi-task learning of recurrent neural networks. In *Proceedings of the International Conference on Document Analysis and Recognition*, 2017.

[25] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[26] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[27] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. In *Proceedings of the Neural Information Processing Systems Conference*, 2015.

[28] Arindam Chowdhury and Lovekesh Vig. An efficient end-to-end neural model for handwritten text recognition. In *Proceedings of the British Machine Vision Conference*, 2018.

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

[30] Brian Davis, Chris Tensmeyer, Brian Price, Curtis Wigington, Bryan Morse, and Rajiv Jain. Text and style conditioned gan for generation of offline handwriting lines. In *Proceedings of the British Machine Vision Conference*, 2020.

[31] Stephen A Della Pietra, Vincent J Della Pietra, Robert L Mercer, and Salim Roukos. Adaptive language modeling using minimum discriminant estimation. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York*, 1992.

[32] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[33] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[34] Patrick Doetsch, Michal Kozielski, and Hermann Ney. Fast and robust training of recurrent neural networks for offline handwriting recognition. In *Proceedings of the International Conference on Frontiers in Handwriting Recognition*, 2014.

[35] Patrick Doetsch, Hermann Ney, et al. Improvements in RWTH's system for off-line handwriting recognition. In *Proceedings of the International Conference on Document Analysis and Recognition*, 2013.

[36] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[37] Linhao Dong, Shuang Xu, and Bo Xu. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5884–5888, 2018.

[38] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, 2017.

[39] DC Dowson and BV Landau. The fréchet distance between multivariate normal distributions. *Journal of multivariate analysis*, 12(3):450–455, 1982.

[40] Philippe Dreuw, Patrick Doetsch, Christian Plahl, and Hermann Ney. Hierarchical hybrid MLP/HMM or rather MLP features for a discriminatively trained Gaussian HMM: a comparison for offline handwriting recognition. In *Proceedings of the IEEE International Conference on Image Processing*, pages 3541–3544, 2011.

[41] Philippe Dreuw, Georg Heigold, and Hermann Ney. Confidence-and margin-based mmi/mpe discriminative training for off-line handwriting recognition. *International Journal on Document Analysis and Recognition*, 14(3):273, 2011.

[42] Kartik Dutta, Praveen Krishnan, Minesh Mathew, and CV Jawahar. Improving CNN-RNN hybrid networks for handwriting recognition. In *Proceedings of the International Conference on Frontiers in Handwriting Recognition*, 2018.

[43] Shrey Dutta, Naveen Sankaran, K Pramod Sankar, and CV Jawahar. Robust recognition of degraded documents using character n-grams. In *Proceedings of the IAPR International Workshop on Document Analysis Systems*, 2012.

[44] Salvador España-Boquera, Maria Jose Castro-Bleda, Jorge Gorbe-Moya, and Francisco Zamora-Martínez. Improving offline handwritten text recognition with hybrid HMM/ANN models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(4):767–779, 2010.

[45] Sharon Fogel, Hadar Averbuch-Elor, Sarel Cohen, Shai Mazor, and Roee Litman. Scrabblegan: Semi-supervised varying length handwritten text generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4324–4333, 2020.

[46] Alicia Fornés, Verónica Romero, Arnau Baró, Juan Ignacio Toledo, Joan Andreu Sánchez, Enrique Vidal, and Josep Lladós. ICDAR2017 competition on information extraction in historical handwritten records. In *Proceedings of the International Conference on Document Analysis and Recognition*, 2017.

[47] Volkmar Frinken and Horst Bunke. Continuous handwritten script recognition. In *Handbook of Document Image Processing and Recognition*, pages 391–425. 2014.

[48] Yaroslav Ganin, Tejas Kulkarni, Igor Babuschkin, SM Eslami, and Oriol Vinyals. Synthesizing programs for images using reinforced adversarial learning. In *Proceedings of the International Conference on Machine Learning*, 2018.

[49] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*, 2014.

[50] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

[51] Michel Gilloux. Writer adaptation for handwritten word recognition using hidden Markov models. In *Proceedings of the International Conference on Pattern Recognition*, 1994.

[52] Adrià Giménez, Ihab Khoury, Jesús Andrés-Ferrer, and Alfons Juan. Handwriting word recognition using windowed Bernoulli HMMs. *Pattern Recognition Letters*, 35:149–156, 2014.

[53] Abel Gonzalez-Garcia, Joost Van De Weijer, and Yoshua Bengio. Image-to-image translation for cross-domain disentanglement. In *Proceedings of the Neural Information Processing Systems Conference*, 2018.

[54] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the Neural Information Processing Systems Conference*, 2014.

[55] Adeline Granet, Emmanuel Morin, Harold Mouchère, Solen Quiniou, and Christian Viard-Gaudin. Transfer learning for handwriting recognition on historical documents. In *Proceedings of International Conference on Pattern Recognition Applications and Methods*, 2018.

[56] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[57] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the International Conference on Machine Learning*, 2006.

[58] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):855–868, 2008.

[59] Alex Graves and Jürgen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Proceedings of the Neural Information Processing Systems Conference*, 2009.

[60] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. DRAW: A recurrent neural network for image generation. In *Proceedings of the International Conference on Machine Learning*, 2015.

[61] Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*, 2015.

[62] Neha Gurjar, Sebastian Sudholt, and Gernot A Fink. Learning deep representations for word spotting under weak supervision. In *Proceedings of the IAPR International Workshop on Document Analysis Systems*, 2018.

[63] David Ha and Douglas Eck. A neural representation of sketch drawings. In *Proceedings of the International Conference on Learning Representations*, 2018.

[64] Tom SF Haines, Oisin Mac Aodha, and Gabriel J Brostow. My text in your handwriting. *ACM Transactions on Graphics*, 35(3):1–18, 2016.

[65] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *Proceedings of the European Conference on Computer Vision*, 2014.

[66] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[67] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Proceedings of the Neural Information Processing Systems Conference*, 2017.

[68] Takaaki Hori, Shinji Watanabe, and John R Hershey. Multi-level language modeling and decoding for open vocabulary end-to-end speech recognition. In *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop*, 2017.

[69] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *Proceedings of the European Conference on Computer Vision*, 2018.

[70] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning*, 2015.

[71] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. In *NeurIPS Deep Learning Workshop*, 2014.

[72] Frederick Jelinek, Bernard Merialdo, Salim Roukos, and Martin Strauss. A dynamic language model for speech recognition. In *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19-22, 1991*, 1991.

[73] Haochuan Jiang, Guanyu Yang, Kaizhu Huang, and Rui Zhang. W-net: one-shot arbitrary-style Chinese character generation with deep neural networks. In *Proceedings of the International Conference on Neural Information Processing*, 2018.

[74] Lei Kang, Pau Riba, Marçal Rusiñol, Alicia Fornés, and Mauricio Villegas. Distilling content from style for handwritten word recognition. In *Proceedings of the International Conference on Frontiers in Handwriting Recognition*, 2020.

[75] Lei Kang, Pau Riba, Marçal Rusiñol, Alicia Fornés, and Mauricio Villegas. Pay attention to what you read: Non-recurrent handwritten text-line recognition. *arXiv preprint arXiv:2005.13044*, 2020.

[76] Lei Kang, Pau Riba, Mauricio Villegas, Alicia Fornés, and Marçal Rusiñol. Candidate fusion: Integrating language modelling into a sequence-to-sequence handwritten word recognition architecture. *arXiv preprint arXiv:1912.10308*, 2019.

[77] Lei Kang, Pau Riba, Yaxing Wang, Marçal Rusiñol, Alicia Fornés, and Mauricio Villegas. Ganwriting: Content-conditioned generation of styled handwritten word images. In *Proceedings of the European Conference on Computer Vision*, 2020.

[78] Lei Kang, Marçal Rusiñol, Alicia Fornés, Pau Riba, and Mauricio Villegas. Unsupervised adaptation for synthetic-to-real handwritten word recognition. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pages 3491–3500, 2020.

[79] Lei Kang, J Ignacio Toledo, Pau Riba, Mauricio Villegas, Alicia Fornés, and Marçal Rusiñol. 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.

[80] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[81] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[82] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. In *Proceedings of the International Conference on Learning Representations*, 2014.

[83] Florian Kleber, Stefan Fiel, Markus Diem, and Robert Sablatnig. CVL-database: An off-line database for writer retrieval, writer identification and word spotting. In *Proceedings of the International Conference on Document Analysis and Recognition*, 2013.

[84] Thomas Konidaris, Basilios Gatos, Kostas Ntzios, Ioannis Pratikakis, Sergios Theodoridis, and Stavros J Perantonis. Keyword-guided word spotting in historical printed documents using synthetic data and user feedback. *International Journal on Document Analysis and Recognition*, 9(2-4):167–177, 2007.

[85] Praveen Krishnan, Kartik Dutta, and CV Jawahar. Deep feature embedding for accurate recognition and retrieval of handwritten text. In *Proceedings of the International Conference on Frontiers in Handwriting Recognition*, 2016.

[86] Praveen Krishnan, Kartik Dutta, and CV Jawahar. Word spotting and recognition using deep embedding. In *Proceedings of the IAPR International Workshop on Document Analysis Systems*, 2018.

[87] Praveen Krishnan, Kartik Dutta, and CV Jawahar. Word spotting and recognition using deep embedding. In *Proceedings of the IAPR International Workshop on Document Analysis Systems*, 2018.

[88] Praveen Krishnan and CV Jawahar. Generating synthetic data for text recognition. *arXiv preprint arXiv:1608.04224*, 2016.

[89] Praveen Krishnan and CV Jawahar. Hwnet v2: An efficient word image representation for handwritten documents. *arXiv preprint arXiv:1802.06194*, 2018.

[90] Praveen Krishnan and CV Jawahar. HWNet v2: An efficient word image representation for handwritten documents. *International Journal on Document Analysis and Recognition*, 22(4):387–405, 2019.

[91] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, et al. The open images dataset v4. *International Journal of Computer Vision*, pages 1–26, 2020.

[92] Victor Lavrenko, Toni M Rath, and Raghavan Manmatha. Holistic word recognition for handwritten historical documents. In *International Workshop on Document Image Analysis for Libraries*, 2004.

[93] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[94] Junyeop Lee, Sungrae Park, Jeonghun Baek, Seong Joon Oh, Seonghyeon Kim, and Hwalsuk Lee. On recognizing texts of arbitrary shapes with 2D self-attention. *arXiv preprint arXiv:1910.04396*, 2019.

[95] Zhouchen Lin and Liang Wan. Style-preserving English handwriting synthesis. *Pattern Recognition*, 40(7):2097–2109, 2007.

[96] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[97] Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-shot unsupervised image-to-image translation. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[98] Qi Liu, Lijuan Wang, and Qiang Huo. A study on effects of implicit and explicit language model information for dblstm-ctc based handwriting recognition. In *Proceedings of the International Conference on Document Analysis and Recognition*, 2015.

[99] Ning Lu, Wenwen Yu, Xianbiao Qi, Yihao Chen, Ping Gong, and Rong Xiao. Master: Multi-aspect non-local network for scene text recognition. *arXiv preprint arXiv:1910.02562*, 2019.

[100] Pengyuan Lyu, Xiang Bai, Cong Yao, Zhen Zhu, Tengteng Huang, and Wenyu Liu. Auto-encoder guided GAN for Chinese calligraphy synthesis. In *Proceedings of the International Conference on Document Analysis and Recognition*, 2017.

[101] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[102] U-V Marti and Horst 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*, pages 65–90. World Scientific, 2001.

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

[104] Michael F Mathieu, Junbo Jake Zhao, Junbo Zhao, Aditya Ramesh, Pablo Sprechmann, and Yann LeCun. Disentangling factors of variation in deep representation using adversarial training. In *Proceedings of the Neural Information Processing Systems Conference*, 2016.

[105] Nada Matic, Isabelle Guyon, J Denker, and Vladimir Vapnik. Writer-adaptation for on-line handwritten character recognition. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 187–191, 1993.

[106] Martin Mayr, Martin Stumpf, Anguelos Nikolaou, Mathias Seuret, Andreas Maier, and Vincent Christlein. Spatio-temporal handwriting imitation. *arXiv preprint arXiv:2003.10593*, 2020.

[107] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[108] Paul Mermelstein and Murray Eyden. A system for automatic recognition of handwritten words. In *Proceedings of the Fall Joint Computer Conference*, 1964.

[109] Johannes Michael, Roger Labahn, Tobias Grüning, and Jochen Zöllner. Evaluating sequence-to-sequence models for handwritten text recognition. In *Proceedings of the International Conference on Document Analysis and Recognition*, 2019.

[110] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Proceedings of Eleventh annual conference of the international speech communication association*, 2010.

[111] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[112] Noam Mor and Lior Wolf. Confidence prediction for lexicon-free OCR. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, 2018.

[113] Bastien Moysset, Théodore Bluche, Maxime Knibbe, Mohamed Faouzi Benzeghiba, Ronaldo Messina, Jérôme Louradour, and Christopher 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.

[114] Bastien Moysset and Ronaldo Messina. Are 2d-lstm really dead for offline text recognition? *International Journal on Document Analysis and Recognition*, 22(3):193–208, 2019.

[115] Rathin Radhakrishnan Nair, Nishant Sankaran, Bharagava Urala Kota, Sergey Tulyakov, Srirangaraj Setlur, and Venu Govindaraju. Knowledge transfer using neural network based approach for handwritten text recognition. In *Proceedings of the IAPR International Workshop on Document Analysis Systems*, 2018.

[116] Ali Nosary, Laurent Heutte, and Thierry Paquet. Unsupervised writer adaptation applied to handwritten text recognition. *Pattern Recognition*, 37(2):385–388, 2004.

[117] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier GANs. In *Proceedings of the International Conference on Machine Learning*, 2017.

[118] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[119] Joan Pastor-Pellicer, Salvador Espana-Boquera, Marıa José Castro-Bleda, and Francisco Zamora-Martinez. A combined convolutional neural network and dynamic programming approach for text line normalization. In *Proceedings of the International Conference on Document Analysis and Recognition*, 2015.

[120] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS 2017 Autodiff Workshop: The Future of Gradient-based Machine Learning Software and Techniques*, 2017.

[121] Kuan-Chuan Peng, Ziyan Wu, and Jan Ernst. Zero-shot deep domain adaptation. In *Proceedings of the European Conference on Computer Vision*, 2018.

[122] Freddy Perraud, Christian Viard-Gaudin, Emmanuel Morin, and Pierre Michel Lallican. N-gram and n-class models for on line handwriting recognition. In *Proceedings of the International Conference on Document Analysis and Recognition*, 2003.

[123] Vu Pham, Théodore Bluche, Christopher Kermorvant, and Jérôme Louradour. Dropout improves recurrent neural networks for handwriting recognition. In *Proceedings of the International Conference on Frontiers in Handwriting Recognition*, 2014.

[124] John C Platt and Nada Matic. A constructive RBF network for writer adaptation. In *Proceedings of the Neural Information Processing Systems Conference*, 1997.

[125] Arik Poznanski and Lior Wolf. CNN-N-gram for handwriting word recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[126] Joan Puigcerver. Are multidimensional recurrent layers really necessary for handwritten text recognition? In *Proceedings of the International Conference on Document Analysis and Recognition*, 2017.

[127] José A Rodríguez-Serrano, Florent Perronnin, Gemma Sánchez, and Josep Lladós. Unsupervised writer adaptation of whole-word HMMs with application to word-spotting. *Pattern Recognition Letters*, 31(8):742–749, 2010.

[128] Verónica Romero, Alicia Fornés, Nicolás Serrano, Joan Andreu Sánchez, Alejandro H Toselli, Volkmar Frinken, Enrique Vidal, and Josep Lladós. The ESPOSALLES database: An ancient marriage license corpus for off-line handwriting recognition. *Pattern Recognition*, 46(6):1658–1669, 2013.

[129] Ekraam Sabir, Stephen Rawls, and Prem Natarajan. Implicit language model in lstm for ocr. In *Proceedings of the International Conference on Document Analysis and Recognition*, 2017.

[130] Fenfen Sheng, Zhineng Chen, and Bo Xu. NRTR: A no-recurrence sequence-to-sequence model for scene text recognition. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 781–786, 2019.

[131] Patrice Y Simard, David Steinkraus, John 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.

[132] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[133] Bruno Stuner, Clément Chatelain, and Thierry Paquet. Handwriting recognition using cohort of LSTM and lexicon verification with extremely large lexicon. *arXiv preprint arXiv:1612.07528*, 2016.

[134] Jorge Sueiras, Victoria Ruiz, Angel Sanchez, and Jose F Velez. Offline continuous handwriting recognition using sequence to sequence neural networks. *Neurocomputing*, 289:119–128, 2018.

[135] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Proceedings of the Neural Information Processing Systems Conference*, 2014.

[136] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[137] Martin Szummer and Christopher M Bishop. Discriminative writer adaptation. In *Proceedings of the International Conference on Frontiers in Handwriting Recognition*, 2006.

[138] Christopher Tensmeyer, Curtis Wigington, Brian Davis, Seth Stewart, Tony Martinez, and William Barrett. Language model supervision for handwriting recognition model adaptation. In *Proceedings of the International Conference on Frontiers in Handwriting Recognition*, 2018.

[139] Achint Oommen Thomas, Amalia Rusu, and Venu Govindaraju. Synthetic handwritten CAPTCHAs. *Pattern Recognition*, 42(12):3365–3373, 2009.

[140] Yuchen Tian. zi2zi: Master chinese calligraphy with conditional adversarial networks, 2017.

[141] J Ignacio Toledo, Sounak Dey, Alicia Fornés, and Josep Lladós. Handwriting recognition by attribute embedding and recurrent neural networks. In *Proceedings of the International Conference on Document Analysis and Recognition*, 2017.

[142] Alejandro H Toselli, Alfons Juan, Jorge González, Ismael Salvador, Enrique Vidal, Francisco Casacuberta, Daniel Keysers, and Hermann Ney. Integrated handwriting recognition and interpretation using finite-state models. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(04):519–539, 2004.

[143] Shubham Toshniwal, Anjuli Kannan, Chung-Cheng Chiu, Yonghui Wu, Tara N Sainath, and Karen Livescu. A comparison of techniques for language model integration in encoder-decoder speech recognition. In *Proceedings of IEEE Spoken Language Technology Workshop*. IEEE, 2018.

[144] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. MoCoGAN: Decomposing motion and content for video generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[145] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[146] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the Neural Information Processing Systems Conference*, 2017.

[147] Alessandro Vinciarelli, Samy Bengio, and Horst Bunke. Offline recognition of large vocabulary cursive handwritten text. In *Proceedings of the International Conference on Document Analysis and Recognition*, 2003.

[148] Paul Voigtlaender, Patrick Doetsch, and Hermann 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.

[149] Jue Wang, Chenyu Wu, Ying-Qing Xu, and Heung-Yeung Shum. Combining shape and physical models for online cursive handwriting synthesis. *International Journal on Document Analysis and Recognition*, 7(4):219–227, 2005.

[150] Peng Wang, Yuanzhouhan Cao, Chunhua Shen, Lingqiao Liu, and Heng Tao Shen. Temporal pyramid pooling-based convolutional neural network for action recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(12):2613–2622, 2016.

[151] Peng Wang, Yuanzhouhan Cao, Chunhua Shen, Lingqiao Liu, and Heng Tao Shen. Temporal pyramid pooling-based convolutional neural network for action recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(12):2613–2622, 2017.

[152] Curtis Wigington, Seth Stewart, Brian Davis, Bill Barrett, Brian Price, and Scott 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.

[153] Curtis Wigington, Chris Tensmeyer, Brian Davis, William Barrett, Brian Price, and Scott Cohen. Start, follow, read: End-to-end full-page handwriting recognition. In *Proceedings of the European Conference on Computer Vision*, 2018.

[154] Shan-Jean Wu, Chih-Yuan Yang, and Jane Yung-jen Hsu. Calligan: Style and structure-aware chinese calligraphy character generator. *arXiv preprint arXiv:2005.12500*, 2020.

[155] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the International Conference on Machine Learning*, 2015.

[156] Hong-Ming Yang, Xu-Yao Zhang, Fei Yin, Jun Sun, and Cheng-Lin Liu. Deep transfer mapping for unsupervised writer adaptation. In *Proceedings of the International Conference on Frontiers in Handwriting Recognition*, 2018.

[157] Mohamed Yousef, Khaled F Hussain, and Usama S Mohammed. Accurate, data-efficient, unconstrained text recognition with convolutional neural networks. *Pattern Recognition*, page 107482, 2020.

[158] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. SeqGAN: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

[159] Francisco Zamora-Martinez, Volkmar Frinken, Salvador España-Boquera, Maria Jose Castro-Bleda, Andreas Fischer, and Horst Bunke. Neural network language models for off-line handwriting recognition. *Pattern Recognition*, 47(4):1642–1652, 2014.

[160] Yaping Zhang, Shuai Nie, Wenju Liu, Xing Xu, Dongxiang Zhang, and Heng Tao Shen. Sequence-to-sequence domain adaptation network for robust text image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[161] Ningyuan Zheng, Yifan Jiang, and Dingjiang Huang. Strokenet: A neural painting environment. In *Proceedings of the International Conference on Learning Representations*, 2019.

[162] Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tie-Yan Liu. Incorporating bert into neural machine translation. *Proceedings of the International Conference on Learning Representations*, 2020.

This work was supported by EU H2020 SME Instrument project 849628, 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.