Manifold Mixup improves text recognition with CTC loss

Bastien Moysset
A2iA SA, Paris, France

Ronaldo Messina
A2iA SA, Paris, France

***Abstract* — Modern handwritten text recognition techniques employ deep recurrent neural networks. The use of these techniques is especially efficient when a large amount of annotated data is available for parameter estimation. Data augmentation can be used to enhance the performance of the systems when data is scarce. Manifold Mixup is a modern method of data augmentation with a regularization effect that meld two images, or the feature maps corresponding to these images, and the targets are fused accordingly. We propose to apply the Manifold Mixup to text recognition while adapting it to work with a Connectionist Temporal Classification cost. We show that Manifold Mixup consistently improves text recognition results on various languages and datasets.**

I. Introduction

Text recognition is an important step in most document image analysis applications. It enables to automatically access the information contained in the pages.

Huge improvement of handwritten text recognition systems has been obtained during the last decade. On the one hand, this amelioration is due to the recognition of text lines using the Connectionist Temporal Classification (CTC) [1] in order to implicitly align the image and the target sequence. On the other hand, this is enabled by modern recurrent neural network techniques, whether based on interleaved convolutional and 2D Long Short-Term Memory (LSTM) layers [2], [3] or on convolutions followed by 1D-LSTM layers [4], [5].

A. State of the Art

The use of neural networks helped to create systems that can cope with high style heterogeneity within a character class. However, these powerful algorithms with a high number of trained parameters do need a large amount of annotated images to reach an optimal performance.

Several methods have been proposed to reduce the need of annotated data when training the text recognition systems.

First, in the line of what has been proposed in image classification [6], data augmentation can be performed to enlarge the number of training samples. Real images can be slanted and stretched to form new samples [3] or they can be warped with a random grid-based distortion [7].

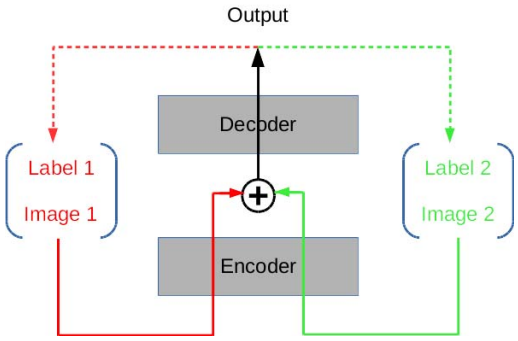


Figure 1: Illustration of the Manifold Mixup principle. Two images, respectively shown in red and green, are mixed after being encoded. The fused features are passed through a decoder and the common output is compared separately to both image labels.

Secondly, the training set can be extended by adding to it artificial images. This can be done by using handwritten-like fonts [8] or by creating text line images from a re-composition of individually extracted real letter images [9], [10]. Artificial text images can also be synthetized from a recurrent model trained to estimate the ink paths [11]. More recently, Alonso et al. [12] proposed a generative adversarial network (GAN) based technique to synthetize handwritten word images.

On the other hand, the lack of training data should be fought to prevent the network from overfitting. For this, regularization techniques like weight decay or dropout can be applied during the network training [13].

Recently, Mixup strategies have been proposed in the Machine Learning community, and mainly for object classification tasks, with the aim to cope with reduced amount of available data. The common idea of these strategies is to fuse several (usually two) images or their transformations and use the interpolated resulting image as an input for the training.

Inspired by Chawla et al. [14] that interpolate input features of objects of the same class, DeVries et al. [15] proposes to mix samples from the same class after being encoded by some of the neural network layers. Input images of different classes can also effectively be fused if the cost function is adapted so that the network learns to estimate

2379-2140/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDAR.2019.00133

an interpolation of the labels [16] and the mixing strategy itself can be learned to avoid manifold intrusions [17].

Verma et al. [18] merges both these ideas and propose to mix the labels and images from different classes, or their encoding, at various layers in the network. They show improved results on unseen data and resistance to adversarial examples.

We based our proposed technique on this idea from Verma et al. and will discuss it more in details in Section II.

B. Problem statement

In this work, we tackle with a manifold mixup based approach illustrated in Figure 1 the following issues of training neural networks for handwritten text recognition:

- The heterogeneity of the handwritten text images to recognize due to varying styles between writers and to different document backgrounds.
- A rather small amount of annotated data available making it harder to generalize on unseen images.
- A trend to overfitting due to both the upper mentioned items and to the potentially high number of parameters in the network.

We make several contributions in order to create a manifold mixup system that cope with the specificities of the text recognition task (compared to image classification).

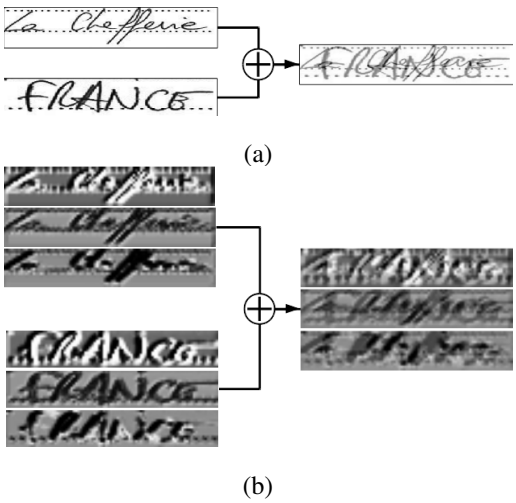
- We introduce a padding and a width-based grouping strategy within mini-batches in order to handle the varying size of the input images.
- We propose a fusion of gradients from two CTC losses in order to mix the two image targets which are label sequences of unequal lengths.
- We study the impact of the position where the mixup is done and of other implementation choices on a standard recurrent text recognition network.
- We prove the consistent effectiveness of the proposed method on a set of handwritten databases in several languages and with varying sizes.

We will describe in details the manifold mixup strategy in Section II. The adaptation to a CTC recognition training will be explained in Section IV. Finally, experimental results will be shown in Section V.

II. Manifold Mixup

The Manifold Mixup strategy for training classifiers was introduced by Verma et al. [18]. It is related to the input image Mixup strategy of Zhang et al. [16]. The main idea of these methods is to, during training, do a randomly weighted interpolation of two images (or of the transformation of these images) and of their respective labels. The Figure 2 illustrates this interpolation, both within the input image space and in a feature map space obtained after forwarding the images through the first layers of the network.

The idea behind Mixup training is the following: in the image to labels space, we want to model the manifold that transforms any image into its label. For this, we use (input, label) data that correspond to points in this space. Because the number of trainable parameters is usually higher than the number of training data available, this setup is prone to overfitting. Using Mixup means that we do not only learn from the data points, but from all the segments linking any two data couples.



(a)

(b)

Figure 2: Illustration of the Mixup process. Two feature map sets (left) and their interpolation (right) are shown, both for the input image space (a) and for the space encoded by the 4 first layers of the network (b).

More formally, consider a neural network made of N stacked layers. For a layer k of the neural network, let h_k be the function calculated by the layers of the neural network up to this layer k. Similarly, let g_k be the function made by the layers from this layer k. If f is the function corresponding to the full network, it means that for an input image $\mathbf{x}$, we have:

$$\forall k \in [1, N], f(\mathbf{x}) = g_k(h_k(\mathbf{x})) \tag{1}$$

The k value is chosen randomly between several possible values. The Mixup function f_{mixup}, for a random weighting value λ, is applied to two training data input-label couples $(\mathbf{x_i}, \mathbf{y_i})$ and $(\mathbf{x_j}, \mathbf{y_j})$, so that:

$$f_{mixup}(\mathbf{x_i}, \mathbf{x_j}, \lambda) = g_k(\lambda h_k(\mathbf{x_i}) + (1 - \lambda)h_k(\mathbf{x_j})) \tag{2}$$

The loss $l(\hat{\mathbf{y}}, \mathbf{y})$ we optimize for is a function of an input and a label where the estimated output is $\hat{\mathbf{y}} = f(\mathbf{x})$. For a classification problem with M classes, this loss is a cross entropy:

$$l(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{c=1}^{M} y_c log(\hat{y_c}) \tag{3}$$

For Mixup, the loss L_{mixup} is computed from the output $\hat{\mathbf{y}}$ of the network and the weighted interpolation of the two labels.

$$L_{mixup} = l(f_{mixup}(\mathbf{x_i}, \mathbf{x_j}, \lambda), \lambda \mathbf{y_i} + (1-\lambda)\mathbf{y_j}) \quad (4)$$

$$L_{mixup}(\hat{\mathbf{y}}, (\mathbf{y_i}, \mathbf{y_j})) = -\sum_{c=1}^{M} (\lambda y_{i,c} + (1-\lambda) y_{j,c}) log(\hat{y_c}) \quad (5)$$

We can observe that this loss can be reformulated as the weighted sum of the individual losses:

$$L_{mixup}(\hat{\mathbf{y}}, (\mathbf{y_i}, \mathbf{y_j})) = \lambda l(\hat{\mathbf{y}}, \mathbf{y_i}) + (1-\lambda) l(\hat{\mathbf{y}}, \mathbf{y_j}) \quad (6)$$

And that the gradients are such that:

$$\frac{\partial L_{mixup}(\hat{\mathbf{y}}, (\mathbf{y_i}, \mathbf{y_j}))}{\partial \hat{y_c}} = \lambda \frac{\partial l(\hat{\mathbf{y}}, \mathbf{y_i})}{\partial \hat{y_c}} + (1-\lambda) \frac{\partial l(\hat{\mathbf{y}}, \mathbf{y_j})}{\partial \hat{y_c}} \quad (7)$$

III. Text Line Recognition

A. Gated Convolutional Network

For the text line recognition problem, we use a Gated Convolutional Network (GNN) inspired by Bluche et al. [5]. The grayscale input images are isotropically rescaled to a fixed height of 128 pixels, normalized and passed through a 2×2 tiling. The network is composed of 12 layers, as indicated in Table I and in Figure 3. The first 8 layers are convolutional and some of them are used as gates which means that their outputs are pointwise multiplied with their inputs. On top of this convolutional encoder, a vertical max-pooling is applied and a recurrent decoder, comprising two Bidirectional Long Short-Term Memory (LSTM) and two linear layers, is applied to the obtained one dimensional signal. The output has the depth of the labelset size (including the blank) and a width proportional to the width of the input image.

Layer	input Depth	output Depth	Filter Size	Stride
Tiling	1	4	2×2	2×2
Convolution	4	8	3×3	1×1
Convolution	8	16	4×2	4×2
Gated Conv.	16	16	3×3	1×1
Convolution	16	32	3×3	1×1
Gated Conv.	32	32	3×3	1×1
Convolution	32	64	4×2	4×2
Gated Conv.	64	64	3×3	1×1
Convolution	64	128	3×3	1×1
Max-Pooling	128	128	4×1	1×1
B-LSTM	128	128	1×1	1×1
Linear	128	128	1×1	1×1
B-LSTM	128	128	1×1	1×1
Linear	128	128	1×1	1×1

Table I: The architecture of the neural network used for text recognition.

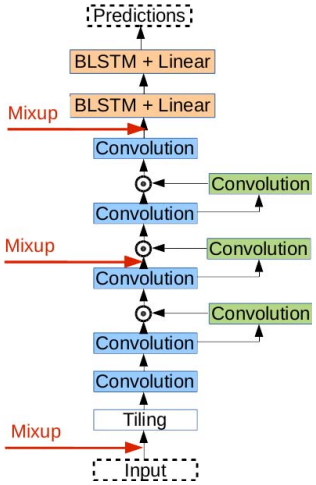


Figure 3: Illustration of the architecture of our text recognition neural network. Standard convolutional layers are shown in blue, while gated convolutional layers are in green and recurrent layers in orange. The red arrows indicate the places in the network where mixup is performed.

B. Training with the CTC

The text line recognition problem has some specificities in comparison to classification. The output of the network is not a vector of probabilities, but a sequence $\hat{\mathbf{y}} = \{\hat{\mathbf{y^1}} ... \hat{\mathbf{y^T}}\}$ of vectors of probabilities where the sequence size T is varying and proportional to the width of the text line image. Similarly, the target is not a label $\mathbf{y}$ but a sequence of labels $\mathbf{l} = \{\mathbf{l_1} ... \mathbf{l_S}\}$ where the label sequence size S corresponds to the number of characters in the text line.

Because S and T are both different and varying independently ($T > S$), we need to perform some sort of alignment between the predictions and the targets. This alignment is implicitly made by the Connectionist Temporal Classification (CTC) [1].

If we define $\mathcal{B}(\mathbf{l})$ as the function that transform a given label sequence $\mathbf{l}$ in all the possible sequences, including blanks, of size T, the CTC loss L_{ctc} is defined as:

$$L_{ctc}(\hat{\mathbf{y}}, \mathbf{l}) = -log \left(\sum_{\boldsymbol{\pi} \in \mathcal{B}(\mathbf{l})} \prod_{t=1}^{T} y_{\pi_t}^t \right) \quad (8)$$

This cost can be computed efficiently using dynamic programming alongside the gradients that will be backpropagated in the network.

$$\frac{\partial L_{ctc}(\hat{\mathbf{y}}, \mathbf{l})}{\partial \hat{y_c^t}} = -\frac{1}{\hat{y_c^t} \sum_{\boldsymbol{\pi} \in \mathcal{B}(\mathbf{l})} \prod_{z=1}^{T} y_{\pi_z}^z} \sum_{\boldsymbol{\pi} \in \mathcal{B}(\mathbf{l}) | \pi_t = k} \prod_{z=1}^{T} y_{\pi_z}^z \quad (9)$$

IV. Applying Mixup to text recognition

In this work, we aim at applying a mixup strategy to the sequence recognition problem. It means that we want, at a given level k, to fuse the feature maps $h_k(\mathbf{x_i})$ and $h_k(\mathbf{x_j})$ from two text line input images $\mathbf{x_i}$ and $\mathbf{x_j}$. In practice, we choose k randomly within $\{0,4,8\}$ as illustrated in Figure 3.

Because $\mathbf{x_i}$ and $\mathbf{x_j}$ may have different horizontal sizes, we pad (with white) all the images from a mini-batch to the max horizontal size within the mini-batch. In order to save computing time and to get more coherent mixups, we group within a same mini-batch images of similar width in the line of what is done in bucketing techniques [19]. Images are weighted with a random valued λ ratio. In practice, λ is drawn from a $Beta(0.5)$ distribution. After going through the end of the network, we obtain a common sequence of prediction probabilities for both lines.

Nevertheless, interpolating the targets like it is done in equation 4 is not possible due to the fact that the two label sequences have different lengths and because the forward-backward algorithm used to compute the CTC works well only for one-hot encodings of targets.

For this reason, we choose to directly use as a loss L_{ctc_mixup} the weighted sum of two CTC losses, one with each label sequence $\mathbf{l_i}$ and $\mathbf{l_j}$. We note that this weighting of losses is analogous to what was observed for the classification in Equation 6

$$L_{ctc\_mixup}(\hat{\mathbf{y}}, (\mathbf{l_i}, \mathbf{l_j})) = \lambda L_{ctc}(\hat{\mathbf{y}}, \mathbf{l_i}) + (1-\lambda) L_{ctc}(\hat{\mathbf{y}}, \mathbf{l_j}) \tag{10}$$

Like in Equation 7, we get gradients relatively to the weighted gradients from the two targets.

$$\frac{\partial L_{ctc\_mixup}(\hat{\mathbf{y}}, (\mathbf{y_i}, \mathbf{y_j}))}{\partial \hat{y}_c^t} = \lambda \frac{\partial L_{ctc}(\hat{\mathbf{y}}, \mathbf{y_i})}{\partial \hat{y}_c^t} + (1-\lambda) \frac{\partial L_{ctc}(\hat{\mathbf{y}}, \mathbf{y_j})}{\partial \hat{y}_c^t} \tag{11}$$

Therefore, we can compute in parallel the two CTC losses and their gradients as detailed in equation 9. Namely, the network will learn to predict both the sequences of characters of the two label sequences. The two CTC losses will be in charge to independently align a single a single prediction sequence with two different label sequences as illustrated in Figure 4. This will lead to less sharp predictions which explain why the Mixup helps to regularize the training.

V. Experiments

A. Experimental setup

We performed most of the experiments on the handwritten lines in French from the Maurdor dataset [20]. This dataset has the particularity of being very challenging with heterogeneous images from different kind of documents (forms, letters, drawings, ...) and various scanning procedures. We also made control experiments on other handwritten sub-datasets from Maurdor (English and Arabic), on the easier tasks RIMES [21] and IAM [22] and on the Chinese CASIA [23] dataset (we do not use the isolated characters parts). The statistics regarding the number of lines in the used datasets are available in Table II.

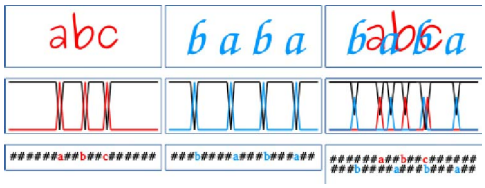


Figure 4: Schematic of the Mixup decoding. Input images, the resulting activations and the alignments (blank is in black and indicated as '#') are shown. Results are given for two images, respectively in red and blue, and for the mixup of these two images.

Dataset	Training lines	Validation lines
Maurdor French Handwritten	26870	2054
Maurdor English Handwritten	10825	1115
Maurdor Arabic Handwritten	11905	1125
RIMES	10532	801
IAM	6482	976
CASIA	35856	5914

Table II: Number of lines present in the datasets used in this work.

To assess the performances of the models, we compute a character error rate (CER) as the Levenshtein distance between the predicted and the ground truth sequences. Excepted for the Chinese, where a language model is used to recover the characters from an encoding as detailed in Bluche et al. [24], we do not use any language model and only the agglutinated best predictions are considered in all the experiments.

For all the models, we use a Glorot [25] initialization of the weights, RMSProp [26] based gradient descent, mini-batches of size 8, and a learning rate of 4.10^{-4}. Models are selected with an early stopping on the validation set after 200 epochs without improvement.

B. Ablation study

In this section, we validate and discuss the key design choices we made for our system.

1) Impact of the position of the Mixup: First, we compare, in Table III, the performance of our handwriting recognition system with respect to the position where the mixup of the feature maps is performed. We observe that, contrarily to what was observed by Zhang et al. [16] for several tasks, the performances with input mixup are worse than the baseline (no mixup) model. Accordingly to this same paper [16], our recognition performance decreases if the mixup is performed in higher latent spaces.

However, we observe that performing the mixup randomly at one of several positions in the network, like what was done

in Verma et al. [18], helps to improve the results compared to the no-mixup strategy. This is probably due to the fact that by adding some randomness, we force the network to learn with the interpolations and prevent it from disentangling the signals. No improvement was obtained by training both with and without mixup.

	CER (%)
No mixup	9.39
At the input	9.55
After the 4th convolutional layer	10.10
After the 8th convolutional layer	11.60
Randomly at one of the 3 positions	8.91
Randomly at one of the 3 positions or no fusion	9.15

Table III: Comparison of the CER with respect to the position(s) in the network where the feature maps from different images are fused. On the Maurdor HWR-FR validation set.

2) Number of images mixed-up: As shown in Table IV, we do not observe further improvement of the results when mixing 3 images instead of 2.

	CER (%)
Fusion of 2 images	8.91
Fusion of 3 images	9.02

Table IV: Comparison of CER when fusing 2 or 3 image feature maps. On the Maurdor HWR-FR validation set.

3) Multiplication of the gradients by the Mixup ratio: A key component of our system is the choice to multiply the gradients (or the loss, this is the same) by the same λ random value that is used to weight the two feature maps during the mixup. To confirm the validity of this choice, we train the network without this multiplication of the gradients by λ. It means that the network is trained to recognize both label sequences without taking into account the weighting ratio. Results, found in Table V, show that doing the multiplication by λ is indeed very important.

	CER (%)
With gradient multiplication	8.91
Without gradient multiplication	10.08

Table V: Influence on the CER of the multiplication of gradients by the mixup ratio. On the Maurdor HWR-FR validation set.

4) Impact of the mixup ratio distribution.: This λ parameter is drawn from a $Beta(0.5)$ distribution, as in Zhang et al. [16]. Verma et al. [18] use a $Beta(2)$ distribution. In Table VI, we compare the results with both these distributions alongside uniform distributions. No significative differences can be observed between the distributions.

C. Analysis of Mixup results

In order to prove the robustness of the presented manifold mixup approach, we compare the performances on the list of

Distribution	CER (%)
Uniform [0,1] (=$Beta(1)$)	8.92
Uniform [0.1 , 0.9]	8.95
$Beta(0.5)$	8.91
$Beta(2)$	9.12

Table VI: Impact of the chosen mixup ratio λ distribution on the CER, for the Maurdor HWR-FR validation set.

datasets introduced in Section V-A. Results can be found in Table VII. We observe a consistent decrease of the character error rates on the difficult datasets that are the three datasets from Maurdor and on the CASIA dataset. Results on the more simple RIMES and IAM datasets are similar with and without mixup.

Dataset	Without mixup	With mixup
Maurdor French Handwritten	9.39	8.91
Maurdor English Handwritten	16.0	14.8
Maurdor Arabic Handwritten	11.0	10.5
CASIA	27.5	23.9
RIMES	3.30	3.32
IAM	4.68	4.64

Table VII: Impact of the use of feature map mixup during the training on the CER (%), for various datasets.

The significativity of the improvement observed when adding the manifold mixup is addressed in Table VIII by training 10 different networks (meaning that 10 different random seeds are used for initialization), both with and without Mixup, on the Maurdor French Handwritten dataset.

	Min	Max	Mean	Median
Without mixup	9.08	9.46	9.30	9.32
With mixup	8.65	9.11	8.88	8.91

Table VIII: Study of the reproducibility over 10 trainings with and without mixup. CER (%) are shown on the Maurdor HWR-FR validation set.

We also compare, in Table IX, the impact of the amount of training data available on the performances. We observe that the relative improvement is progressively increasing from 5.1% to 12.3% when the dataset size is reduced from 26 000 to 5 000 examples. This illustrates how using manifold mixup for text recognition does increase the generalization ability of the network.

Number of images	With Mixup	Without Mixup
26870 (All)	8.91	9.39
20000	9.85	10.4
10000	13.3	14.4
5000	20.6	23.5

Table IX: Impact of the number of training examples on the CER (%), on the Maurdor HWR-FR validation set.

Similarly, we observe that using mixup strongly diminishes the overfitting. In particular, we observe that the

CTC loss on the validation set tends to go up at some point without manifold mixup, while it reaches some sort of horizontal asymptote when mixup is performed. This confirms that the mixup approach has some regularization effect on the network as mentioned in previous works. [16]–[18].

Following this line of reasoning, we study, in Table X the cross impact of an other common layer with a regularization effect, the Dropout [27], that is used in our training process. We can see that both of them, independently strongly improve the results of the text recognition. But their improvements are not completely co-linear as using both of them further improve the performances. We also compare, in this Table X the impact of Data augmentation on the results. As in [3], data augmentation is performed by stretching and shearing the input images. Again, we observe that results can be improved by using Manifold Mixup in combination with the standard data augmentation process.

Dropout	Data augmentation	Manifold Mixup	CER (%)
No	No	No	15.4
Yes	No	No	9.39
Yes	Yes	No	8.63
No	No	Yes	10.6
Yes	No	Yes	8.91
Yes	Yes	Yes	8.22

Table X: Cross study of two regularizer, dropout and manifold mixup alongside standard data augmentation. CER results on the Maurdor HWR-FR validation set.

VI. Conclusion

In this paper, we presented a new training strategy for text recognition systems based on Manifold Mixup. We proved that this technique acts as a strong regularizer by interpolating the input images and the gradients. We proposed a technique to apply the mixup to varying size images and sequence prediction with Connectionist Temporal Classification alignments. We demonstrated a significative and consistent improvement of text recognition results on several handwritten datasets of varying sizes and languages.

References

[1] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber, "Connectionist Temporal Classification: Labelling unsegmented sequence data with recurrent neural nets," in *ICML*, 2006.

[2] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *NIPS*, 2009.

[3] B. Moysset, T. Bluche, M. Knibbe, M. F. Benzeghiba, R. Messina, J. Louradour, and C. Kermorvant, "The A2iA multi-lingual text recognition system at the second Maurdor evaluation," in *ICFHR*, 2014.

[4] J. Puigcerver, "Are multidimensional recurrent layers really necessary for handwritten text recognition?" in *ICDAR*, 2017.

[5] T. Bluche and R. Messina, "Gated convolutional recurrent neural networks for multilingual handwriting recognition," *ICDAR*, 2017.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[7] C. Wigington, S. Stewart, B. Davis, B. Barrett, B. Price, and S. Cohen, "Data augmentation for recognition of handwritten words and lines using a cnn-lstm network," in *ICDAR*, 2017.

[8] M. Helmers and H. Bunke, "Generation and use of synthetic training data in cursive handwriting recognition," in *Iberian Conference on Pattern Recognition and Image Analysis*, 2003.

[9] Y. Elarian, I. Ahmad, S. Awaida, W. G. Al-Khatib, and A. Zidouri, "An Arabic handwriting synthesis system," *Pattern Recognition*, 2015.

[10] X. Shen and R. Messina, "A method of synthesizing handwritten Chinese images for data augmentation," in *ICFHR*, 2016.

[11] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[12] E. Alonso, B. Moysset, and R. Messina, "Adversarial generation of handwritten text images conditioned on sequences," *ICDAR*, 2019.

[13] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour, "Dropout improves recurrent neural networks for handwriting recognition," in *ICFHR*, 2014.

[14] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, 2002.

[15] T. DeVries and G. W. Taylor, "Dataset augmentation in feature space," in *ICLR Workshop Track*, 2017.

[16] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *ICLR*, 2018.

[17] H. Guo, Y. Mao, and R. Zhang, "Mixup as locally linear out-of-manifold regularization," in *AAAI*, 2019.

[18] V. Verma, A. Lamb, C. Beckham, A. Courville, I. Mitliagkis, and Y. Bengio, "Manifold mixup: Encouraging meaningful on-manifold interpolation as a regularizer," *arXiv preprint arXiv:1806.05236*, 2018.

[19] V. Khomenko, O. Shyshkov, O. Radyvonenko, and K. Bokhan, "Accelerating recurrent neural network training using sequence bucketing and multi-gpu data parallelization," in *International Conference on Data Stream Mining & Processing*, 2016.

[20] S. Brunessaux, P. Giroux, B. Grilhères, M. Manta, M. Bodin, K. Choukri, O. Galibert, and J. Kahn, "The Maurdor project: Improving automatic processing of digital documents," in *DAS*, 2014.

[21] E. Grosicki and H. El-Abed, "ICDAR 2011: French handwriting recognition competition," in *ICDAR*, 2011.

[22] U.-V. Marti and H. Bunke, "The IAM-database: an English sentence database for offline handwriting recognition," *IJDAR*, 2002.

[23] C.-L. Liu, F. Yin, D.-H. Wang, and Q.-F. Wang, "CASIA online and offline Chinese handwriting databases," in *ICDAR*, 2011.

[24] T. Bluche and R. O. Messina, "Faster segmentation-free handwritten Chinese text recognition with character decompositions." in *ICFHR*, 2016.

[25] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *ICAIS*, 2010.

[26] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Networks for Machine Learning*, 2012.

[27] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, 2014.