Exploring multiple feature combination strategies with a recurrent neural network architecture for off-line handwriting recognition

Mioulet L.[a,b], Bideault G.[a], Chatelain C.[c], Paquet T.[a] and Brunessaux S.[b]

[a]Laboratoire LITIS - EA 4108, Universite de Rouen, FRANCE 76800;
[b]Airbus Defence and Space, Parc d'affaires des portes, Val de Reuil, FRANCE 27106;
[c]Laboratoire LITIS - EA 4108, INSA Rouen, FRANCE 76800

ABSTRACT

The BLSTM-CTC is a novel recurrent neural network architecture that has outperformed previous state of the art algorithms in tasks such as speech recognition or handwriting recognition. It has the ability to process long term dependencies in temporal signals in order to label unsegmented data. This paper describes different ways of combining features using a BLSTM-CTC architecture. Not only do we explore the low level combination (feature space combination) but we also explore high level combination (decoding combination) and mid-level (internal system representation combination). The results are compared on the RIMES word database. Our results show that the low level combination works best, thanks to the powerful data modeling of the LSTM neurons.

Keywords: Neural networks, Recurrent networks, Feature combination, System Combination, Handwriting recognition

1. INTRODUCTION

Handwriting recognition deals with the extraction (localization and recognition) of words from images containing a graphical representation of these words. The recognition of offline handwriting is difficult due to the variability between writers for a same word, presence of overlapping letters, and complex long term context dependencies. However, handwriting systems have been successfully applied to different tasks, such as bank cheque recognition[1] and postal address recognition.[2] Indeed bank cheques and addresses follow a strict format and have a very limited vocabulary : the prior knowledge on these task is extensively used to boost the recognition performances. However, the recognition of unconstrained handwriting that can occur in various documents, such as letters, books, notes, is a very complex task. Exploring recognition in context free systems is now a main interest of researchers.

A simple handwriting system is generally divided in four steps : preprocessing, feature extraction, recognition and post-processing. In this paper we focus on the recognition stage, for which the Hidden Markov Models[3] (HMM) have been massively used.[4–6] HMMs are states machines that combine two stochastic processes: a stochastic event that cannot be observed directly (the hidden states) is observed via a second stochastic process (the observations).

These models have several advantages. First they provide a way of breaking out of Sayre's paradox.[7] Indeed recognizing a letter in a word image requires its prior detection. But the reverse is also true, thus leading to an ill posed problem known as Sayre's paradox in the literature. Methods using HMMs break up word images in atomic parts, either as graphemes or frames extracted using a sliding window. This will cause a difference between the input sequence length and the output sequence length. However, HMMs are able to cope with this difference in length, they can label unsegmented sequences. They do so by recognizing every window as a character or part of a character and modeling character length. A second advantage of HMMs is their robustness to noise since they are statistical models. Third the algorithms used for HMMs decoding integrate language modeling, lexicon check or N-gram models, which makes them very powerful tools for handwriting recognition.

Further author information:
Mioulet, L.: E-mail: luc.mioulet@cassidian.com

However, they also have some shortcomings. First of all, they are generative systems, which means that when compared to other classifiers, they have a lesser ability to discriminate between data since they provide a likelihood measure to decide . Secondly, within the HMMs framework the current hidden state only depends on the current observation and the previous state (markovian condition) which prohibits the modeling of long term dependencies.

Recently a new system based on recurrent neural networks[8] has overcome these shortcomings. The Bi-directional Long Short Term Memory neural network[9,10] (BLSTM) consists in combining two recurrent neural networks with special neural network units. The output of these networks are then processed by a special Softmax layer, the Connectionist Temporal Classification[11] (CTC), that enables the labelling of unsegmented data. The composed system, referred as the BLSTM-CTC, has shown very impressive results on challenging databases.[12–15]

In this paper, we present two baseline systems using the same architecture, built around the BLSTM-CTC network. These systems only differ by the features that are extracted using a sliding window method. These baseline systems enable a direct comparison between the two sets of features, which enables us to prefer one set of features over the other. However, it is well known that combining systems or features may improve the overall results.[15,16]

In this paper we explore different ways of combining an ensemble of F BLSTM-CTC. We explore different levels of information combination, from a low level combination (feature space combination), to mid-level combinations (internal system representation combinations), and high level combinations (decoding combinations). The experiments are carried out on the handwriting word recognition task WR2 of the ICDAR 2009 competition.[14] We first present the baseline system used for handwriting recognition. We then detail the different level of combinations available throughout the BLSTM architecture, finally we present the results and analyse them.

2. THE BASELINE PROCESSING SYSTEM

In this section we present the baseline system we use to recognize handwritten word images. We first give an overview of our system, we then describe each step with further detail.

2.1 Overview

Our system is dedicated to recognizing images of isolated handwritten words. In order to do so we implemented a very common processing workflow that is represented in Fig. 1.

First the image is preprocessed in order to remove noise and normalize character appearance between different writing styles. Given that our main interest is the combination of systems we did not focus on the preprocessing, hence we used well known algorithms that have been widely used in the literature. After cleaning the image we use a sliding window to extract the features : a window of length P is used to extract a subframe from the image, this subframe is then analyzed and transformed by mathematical processes into a M dimensional vector representation. For instance a 2D image of length N is transformed into a sequence of N feature vectors of dimension M.

This sequence is then processed by a BLSTM-CTC network, in order to transform this M dimensional signal into a sequence of labels (characters). This output is finally processed by a HMM in order to apply a directed decoding. It has to be stressed that it is possible to add a lexicon to the BLSTM-CTC[13] decoding stage. However, we did not opt for this solution since the HMMs enable us to be more modular, e.g. offering the possibility to integrate language models with various decoding strategies available on standard decoding platforms such as HTK,[17] Julius[18] or Kaldi,[19] without deteriorating the performances.

We now go into further details for each part.

2.2 Preprocessing

The preprocessing of images consists in a very simple three step image correction. First a binarization by thresholding is applied. Secondly a deslanting process[20] is applied. The image is rotated between $[-\Theta; \Theta]$ and for each angle θ the histogram of vertical continuous traits H_θ is measured. The deslanting angle is determined by the H_θ with the maximum number of continuous traits. Thirdly a height normalization technique is applied to center the baseline and normalize the heights of the ascenders and descenders.

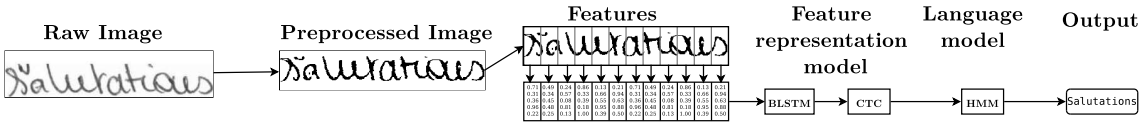


Figure 1. The baseline system

2.3 Feature extraction

Among numerous feature representations, we selected two efficiency proven features : the features presented in[13] and the Histogram of Oriented Gradients[21,22] (HOG). The two feature sets are extracted using a sliding window method on the images.

The features presented in[13] are referred as Simple Pixel Features (SPF). It is a very simple set of features, based on a spatial representation of a single colon of pixels: the mean grey value of the pixels, the position of the center of gravity, the second order vertical moment of black pixels, the positions of the uppermost and lowermost black pixels, the rate of change of these positions (with respect to the neighbouring windows), the number of black-white transitions between the uppermost and lowermost pixels, the proportion of black pixels between the uppermost and lowermost pixels. SPF are a low level feature set, very basic information is extracted from the pixels. It mainly compresses the information for faster processing. We decline this descriptor over three colons of pixels, hence composing a vector of $3 \times 9 = 27$ dimensions.

HOG features are extracted from windows of 8×64 pixels. HOG describes the window by dividing it into sub-windows of $n \times m$ pixels. Within each sub-window a histogram is computed, it describes the distribution of the local intensity gradients (edge direction). The histograms from all sub-windows are then concatenated to form the final representation. We used 8 non-overlapping sub-windows of 8×8 using 8 directions, hence composing a feature vector of 64 dimensions. HOG features are medium level features, they are more complex features than SPF, they describe the local context of the window.

2.4 BLSTM-CTC

After extracting the features from a database (RIMES database) they are then independently used to train a BLSTM-CTC network. The network transforms a sequence of unsegmented data into a one dimensional output vector, e.g. it transforms a sequence of SPF features of length N into a word of length L. We first present the BLSTM network and then the CTC.

2.4.1 BLSTM network

The BLSTM is a complex memory management network, it is in charge of processing a N dimensional input signal to produce an output signal that takes into account long term dependencies. In order to do this the BLSTM is composed of two recurrent neural networks with Long Short Term Memory neural units.

One network processes the data chronologically while the other processes the data in reverse chronological order. Therefore at time t a decision can be taken by combining the outputs of the two networks, using past and future context. For handwriting recognition having a certain knowledge of previous and future possible characters is important since in most cases characters follow a logical order induced by the underlying lexicon of the training database. Instead of modeling this order at a high level using N-grams they are integrated at a low level of decision inside the BLSTM networks. Moreover, handwritten characters have various length which can be modeled efficiently by the recurrent network.

These networks integrate special neural network units : Long Short Time Memory[11] (LSTM). LSTM neurons consist in a memory cell an input and three control gates. The gates control the memory of the cell, namely : how an input will affect the memory (input gate), if a new input should reset the memory cell (forget gate) and if the memory of the network should be presented to the following neural network layer (output gate). The gates enable a very precise and long term control of the memory cell. Compared to traditional recurrent neural network layers, LSTM layers can model much longer and more complex dependencies. A LSTM layer is a fully

recurrent layer, the input and the three gates receive at each instant t the input at time t from the previous layer and the previous output $t-1$.

The combination of the bidirectional networks with LSTM units enables the BLSTM to provide a complex output taking into account past and future long term dependencies.

2.4.2 CTC layer

The CTC is a specialized neural network layer dedicated to transforming BLSTM outputs into class posterior probabilities. It is designed to be trained using unsegmented data, such as handwriting or speech. It is a Softmax layer[23] where each output represents a character, it transforms the BLSTM signal into a sequence of characters. This layer has as many outputs as characters in the alphabet plus one additional output, a "blank" or "no decision" output. Therefore it has the ability to avoid taking a decision in uncertain zones instead of continuously being forced to decide on a character signal in a low context area (e.g. uncertain).

The power of the CTC layer is in its training method. Indeed for handwriting recognition, images are labelled at a word level, therefore making it impossible to learn characters individually. The CTC layer provides a learning mechanism for such data. Given a word, e.g. "hand", first this word is transformed to integrate the blank label, in order to learn to model this label. Using an operator B the blank label is inserted between each character and at the beginning and end of a word w of length L. For instance the word "hand" is replaced by "_h_a_n_d_" where "_" represents the blank label.

Unsegmented data has a disparity between the input sequence length N and the replaced word length $2L+1$, with $N \geq 2L+1$. In order to train the CTC network an output has to be provided for the complete input sequence. Inspired by the HMM algorithms, the CTC uses an objective function integrating a forward and backward variable to determine the best path through a lattice of possibilities. This lattice describes all possible transitions in a word, hence representing all paths $\pi_{1:T}$ possible over time. The authorized transitions are from a character to a blank, from a character to another character, staying on the same character or on the blank. The forward and backward variable are calculated using the authorized transitions. The forward variable is defined as the summed probability of all paths ending in position s of the modified word at time t in the sequence:

$$\alpha_t(s) = \sum_{B(\pi_{1:t})} \prod_{t'=1}^{t} y_{\pi_{t'}}^{t'} \quad , \tag{1}$$

with $y_{\pi_{t'}}^{t'}$ the output of the CTC layer at time t'. The backward variable is defined as the summed probability of all paths beginning in position s of the modified word at time t in the sequence:

$$\beta_t(s) = \sum_{B(\pi_{t:T})} \prod_{t'=t+1}^{T} y_{\pi_{t'}}^{t'} \quad . \tag{2}$$

These variables enable the calculation of all possible outputs at all times, hence covering the disparity of length between the input and output signal. Given a sequence $\mathbf{x}$, a label output $\mathbf{l}$, the label sequence probability at time t is defined as:

$$p(\mathbf{l}|\mathbf{x}) = \sum_{s=1}^{|B(\mathbf{l})|} \alpha_t(s)\beta_t(s) \tag{3}$$

An objective function can then be calculated to backpropagate the error through the CTC and the BLSTM network using the Back Propagation Through Time[24] algorithm.

After being trained the BLSTM-CTC is able to output for an unknown sequence $\mathbf{x}$ a label output $\mathbf{l}'$.

2.5 HMM

In order to perform lexicon directed recognition we use a modeling HMM. This stage is usually performed by the CTC using a Token Passing Algorithm.[13] However, we substitute this correction step by a modeling HMM, enabling us to have more flexibility in regards of the decoding strategy applied without deteriorating the results.

The BLSTM-CTC character posteriors substitute the Gaussian Mixture Models[25] (GMM), prior to this step we simplify the outputs of the CTC. As said previously the blank output covers most of the response of a CTC output signal, whereas character labels only represent spikes in the signal. We therefore remove all blank outputs above a certain threshold, this enables us to keep some uncertain blank labels that may be used for further correction by the HMM lexicon directed stage. This new CTC output supersedes the GMMs used to represent the underlying observable stochastic process. Subsequently the HMM use a Viterbi lexicon directed decoding algorithm (implementing the token passing decoding algorithm) and outputs the most likely word among a proposed dictionary.

3. SYSTEM COMBINATION

In the previous section we described the BLSTM-CTC neural network. Our main interest in this paper is to combine F feature representations in a BLSTM-CTC system to improve the recognition rate of the overall system. The BLSTM-CTC exhibits three different levels at which we can combine the features. 1-low level combination can be introduced through the combination of the input features. 2- mid level combination can be introduced by combining the BLSTM outputs into one single decision stage. This combination strategy assumes implicitly that each BLSTM provides specific features on which an optimized CTC based decision can take place. 3- finally a high level combination at the CTC decoding stage using the direct output of this level. Fig. 2 represents all the combinations we explored. This section describes the different combinations we propose using F different feature sets.

3.1 Low combination

First we consider the feature combination. The feature vectors of the different representations are concatenated into one unique vector, this method is also known as early integration. Combining at this level has the advantage of being the most straightforward method.

3.2 Mid level combination

The mid level combination concerns the BLSTM level combination. This method is inspired by previous work on deep neural networks, especially auto-encoders[26] and denoising auto-encoders.[27] An ensemble of F BLSTM-CTC is first trained on the F different feature representations. Then the CTCs are removed, hence removing the individual layers that transform each feature signal into a label sequence. The remaining BLSTMs output are then combined by training a new architecture containing a CTC. The BLSTMs weights stay unchanged during the training of the new architecture, we consider they are high level feature extractors that are independently trained to extract the information from each feature set.

We consider the addition of two different architectures for this level of combination. First a single CTC layer is added, this CTC level is a simple feature combination layer. Second a complete BLSTM-CTC architecture is added, hence building a new feature extractor using the previously acquired context knowledge as an input.

3.3 High level combination

Lastly we consider the combination of the CTC outputs, a late fusion stage. An ensemble of F BLSTM-CTC is first trained on the F different feature representations. The outputs of each BLSTM-CTC are then combined to form a new output signal. We propose five different combinations for the outputs : averaging outputs, averaging outputs and training a new BLSTM-CTC, combination of outputs and training a new BLSTM-CTC, selecting the maximum output at all times, selecting the maximum output and training a new BLSTM-CTC. The first combination is a simple average of outputs at every instant. The second combination adds a BLSTM-CTC in order to measure the ability of the BLSTM-CTC to learn and correct long term dependencies errors at a character level. The third combination is a simple concatenation of the F CTC outputs at every instant. Therefore creating a new output representation, if we consider an alphabet A of size $|A|$, the new representation is of dimension $(|A|+1) \times F$. This new output representation is largely superior to the HMM input dimensions capacity. Hence we use a BLSTM-CTC to learn and correct errors as well as reduce the signal dimensionality. The fourth combination selects at every instant the maximum output between the F output labels, the results

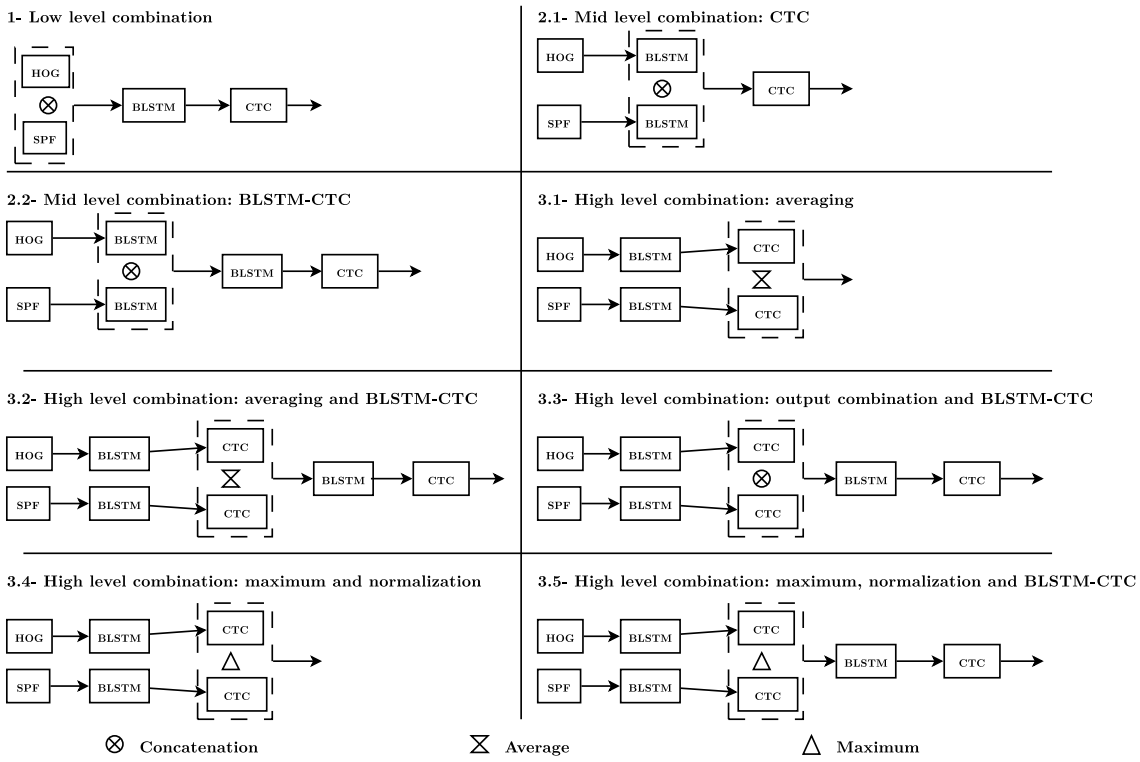


Figure 2. The different BSTM-CTC combinations

are then normalized to appear in the range $[0; 1]$. The fifth combination adds a BLSTM-CTC network on the fourth combination, for identical reasons than the second high level combination.

The different combination levels previously presented all have theoretical advantages and drawbacks. We now present the experimental results and we explain why certain combinations outperform the others.

4. EXPERIENCES

In this section we describe the database and recognition the task we used to compare the various combination performed on the BLSTM-CTC.

4.1 Database and task description

In order to compare our results we use the RIMES database. This database consists in handwritten letters addressed to various administrations and registration services. From these letters, single words were isolated and annotated. The database is divided in three different subsets : training, validation and test. Each set contains respectively 44197 images, 7542 images and 7464 images. The RIMES database was built to be a low noise database, i.e. the background and foreground pixels are easily identifiable. However, the intraword variation is very important since 1300 different writers participated for writing the letters. The word length distribution among the three sets is very similar. It has to be stressed that the words with three and less characters contribute to more than 40% of the database. This will affect the HMM lexicon correction since it is harder to correct short words.

We compare our results on the ICDAR 2009 WR2 task.[14] This task consists in finding the correct transcription of a word using a lexicon of 1600 words. The measure used is the Word Error Rate (WER) on the Top 1 outputs and Top 10 outputs.

In order to learn the system we use the whole training database to train each part of the system. This may cause some overfitting, however splitting the training base in order to train each system on separate bases causes some global loss on the learning performances.

4.2 Results

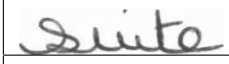
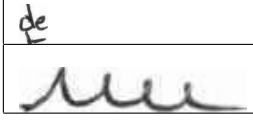
We now present the results of the various combination schemes on the RIMES database for BLSTM-CTC feature combination. Table 1 presents the results of the different combinations by displaying the top 1 and top 10 Word Error Rate (WER). We also provide the raw WER output of the last CTC layer prior to the HMM layer, i.e. the system without the lexicon decoding.

Table 1. Results of the different BLSTM-CTC combination experiments on the RIMES database using Word Error Rate

System	Raw WER	Top 1	Top 10
Single features :			
HOG	39.91	14.31	4.95
SPF	45.44	17.95	7.01
1-Low Level combinations	33.93	12.19	3.6
2-Mid level combinations :			
2.1-Output combination and CTC	37.75	14.27	6.16
2.2-Output combination and BLSTM-CTC	38.62	14.06	5.48
3-High Level combinations :			
3.1-Averaging	94.2	48.76	17.72
3.2-Averaging and BLSTM-CTC	40.4	23.86	6.9
3.3-Output combination and BLSTM-CTC	43.94	26.05	9.28
3.4-Maximum and normalization	60.0	13.96	4.1
3.5-Maximum, normalization and BLSTM-CTC	38.2	15.2	6.2

The best combination is the low level feature combination. The reason is that the BLSTM-CTC is able to model very efficiently the data. In a very similar fashion to deep neural networks, it models very well the spatial and time domain dependencies of features. In order to combine several features using a BLSTM-CTC it is therefore preferable to directly combine the features. Table 2 demonstrates the differences between the input and output of the two baseline systems and this combination. These examples show that this combination learns the best and worst of both features, improving the overall results but some errors are kept or added.

Table 2. Examples of different outputs between the baseline systems, the low level combination system and the maximum system

Image	Ground truth	HOG output	SPF output	Low level combination output	Maximum and normalization output
	suite	autre	vente	suite	attente
	prie	prie	plume	prie	prie
	je	je	Je	Je	Je
	de	je	de	de	de
	un	rue	un	une	un

An interesting result concerns the mid levels combinations. The raw WER points out to the fact that the best mid level combination is by simply adding a new CTC. However, after applying the HMM the combination using a BLSTM-CTC to combine outputs bests the combination by CTC. Therefore, the combination by CTC corrects some words but adds noise on other, hence it hinders the HMM lexicon directed decoding. Relearning some long term dependencies (using a BLSTM-CTC) improves by 0.2 the recognition after the HMM lexicon directed decoding.

The outcome of the high level combinations is more contrasted. On the one hand the first three and the last high level combination deteriorate the results, they inject noise in the outputs hindering the directed decoding.

The averaging in particular is notably bad, the HMM lexicon directed decoding is unable to correct this combination of features. This can be explained by the CTC output signal : most of the time the blank label output is on a plateau of value 1, while all other outputs (labels) are close to null. As soon as the BLSTM-CTC has collected enough information to be certain that a character is present, then the CTC output of this character spikes to 1 and all other outputs become 0. This behavior can be explained by the modification of the training target. Inserting blanks between each character of the real target produces this spiked signal. However, the two systems may not output the labels at the same timestep, therefore a good output label may be lost after the averaging stage.

Relearning long term dependencies from the averaging slightly improves the results, but not enough to be of interest. The BLSTM-CTC is able to learn some information post averaging, but not enough to improve the overall performance.

On the other hand maximum selection and normalization improves the results. In this case the signal produced by this stage includes much weaker spikes on uncertain zones, compared to the spiked output of the baseline systems. As can be seen on Fig. 3, the output signals of this combination seems much more balanced among classes on a given timestep. Thus, producing more uncertainty between characters at any time, enabling the HMM lexicon decoding to explore more paths.

As a final note it is interesting to see that system 3.3 is not able to outperform a maximization and normalization step, we can therefore deduce that the BLSTM-CTC cannot build an internal representation of such a function.

5. CONCLUSION

In this paper we presented different combinations of feature representations using a BLSTM-CTC. The best result is achieved using a low level feature combination(early integration), indeed the internal spatial and time modeling ability of the BLSTM network is very efficient. A second system outperforming the baseline systems, only when using a lexicon directed decoding, is achieved by a maximum selection and a normalization step. However, a mid level combination is also quite interesting, as we mentioned previously we did not relearn the internal weights of the BLSTM after adding the new output layer, future work will explore this possibility as well as adding more features. Indeed adding too many features may lead to a saturation of a single BLSTM-CTC, hence training multiple BLSTM layers with low dimensional inputs may prove a better solution than working with high dimensional input. Future work may also investigate the combination of combinations. As we can see from Fig. 3 and Table 2 different combinations produce different outputs, especially after a lexicon directed decoding, therefore a combination of these two steps may improve further the results.

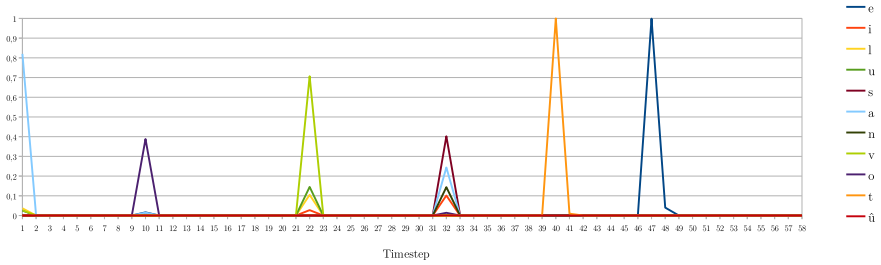
REFERENCES

[1] Knerr, S., Anisimov, V., Barret, O., Gorski, N., Price, D., and Simon, J., "The A2iA intercheque system: courtesy amount and legal amount recognition for French checks," *International journal of pattern recognition and artificial intelligence* **11**(4), 505–548 (1997).

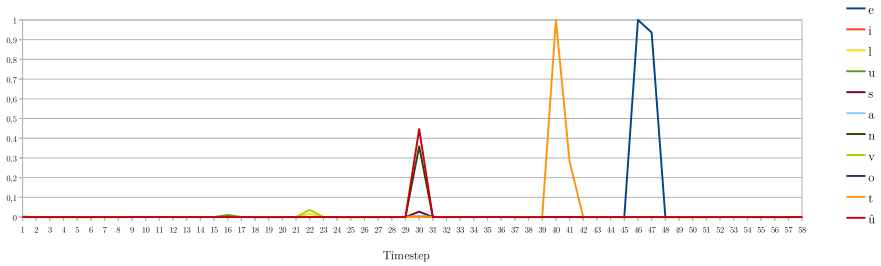
[2] El-Yacoubi, A., Bertille, J., and Gilloux, M., "Conjoined location and recognition of street names within a postal address delivery line," *Proceedings of the Third International Conference on Document Analysis and Recognition* **2**, 1024–1027 (1995).

[3] Rabiner, L., "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE* **77**(2), 257–286 (1989).

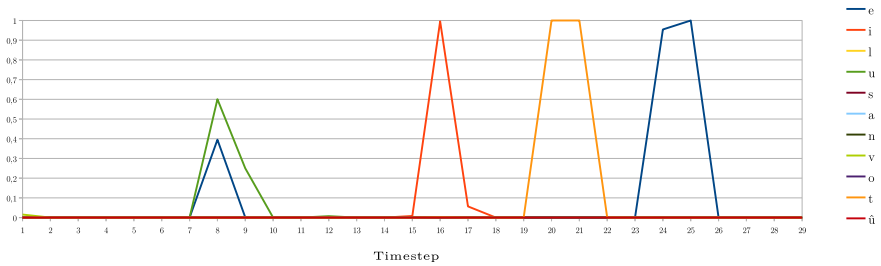
SPF



HOG



Low level combination



Maximum and normalization

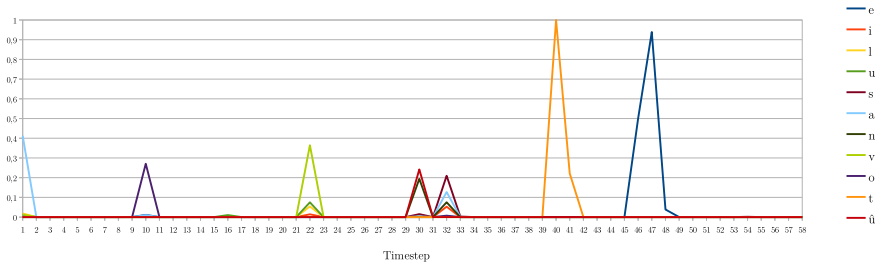


Figure 3. Output signal prior to the HMM on word “Suite“

[4] Kundu, A., He, Y., and Bahl, P., "Recognition of handwritten word: first and second order hidden Markov model based approach," *Computer Vision and Pattern Recognition* **22**(3), 457–462 (1988).
[5] El-Yacoubi, A., Gilloux, M., Sabourin, R., and Suen, C. Y., "An HMM-based approach for off-line unconstrained handwritten word modeling and recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **21**(8), 752–760 (1999).
[6] Bunke, H., Bengio, S., and Vinciarelli, A., "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).
[7] Vinciarelli, A., "A survey on off-line cursive word recognition," *Pattern recognition* **35**(7), 1433–1446 (2002).
[8] Graves, A., Liwicki, M., Bunke, H., Santiago, F., and Schmidhuber, J., "Unconstrained on-line handwriting recognition with recurrent neural networks," *Advances in Neural Information Processing Systems* **20**, 1–8 (2008).
[9] Hochreiter, S. and Schmidhuber, J., "Long short-term memory," *Neural Computation* **9**(8), 1735–1780 (1997).
[10] Gers, F. A. and Schraudolph, N. N., "Learning Precise Timing with LSTM Recurrent Networks," *Journal of Machine Learning Research* **3**, 115–143 (2002).
[11] Graves, A. and Gomez, F., "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in [*Proceedings of the 23rd International Conference on Machine Learning*], (2006).
[12] Graves, A., *Supervised sequence labelling with recurrent neural networks*, PhD thesis (2008).
[13] Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., and Schmidhuber, J., "A novel connectionist system for unconstrained handwriting recognition.," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **31**, 855–68 (May 2009).
[14] Grosicki, E. and Abed, H. E., "ICDAR 2009 Handwriting Recognition Competition," in [*10th Interational Conference on Document Analysis and Recognition*], 1398–1402 (2009).
[15] Menasri, F., Louradour, J., Bianne-Bernard, A., and Kermorvant, C., "The A2iA French handwriting recognition system at the Rimes-ICDAR2011 competition," *Society of Photo-Optical Instrumentation Engineers* **8297**, 51 (2012).
[16] Gehler, P. and Nowozin, S., "On feature combination for multiclass object classification," in [*IEEE International Conference on Computer Vision*], 221–228, IEEE (2009).
[17] Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Moore, G., Odell, J., Ollason, D., Povey, D., Valtchev, V., and Woodland, P., [*The HTK book*] (2006).
[18] Lee, A., Kawahara, T., and Shikano, K., "Julius an Open Source Real-Time Large Vocabulary Recognition Engine," in [*Eurospeech*], 1691–1694 (2001).
[19] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., and Vesely, K., "The kaldi speech recognition toolkit," in [*IEEE workshop on Automatic Speech Recognition and Understanding*], 1–4 (2011).
[20] Vinciarelli, A. and Luettin, J., "A new normalization technique for cursive handwritten words," *Pattern Recognition Letters* **22**, 1043–1050 (July 2001).
[21] Dalal, N. and Triggs, B., "Histograms of Oriented Gradients for Human Detection," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* **1**, 886–893 (2005).
[22] Ait-Mohand, K., Paquet, T., and Ragot, N., "Combining structure and parameter adaptation of HMMs for printed text recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence* (99) (2014).
[23] Bishop, C. M., [*Neural networks for pattern recognition*], Oxford University Press (1995).
[24] Werbos, P. J., "Backpropagation through time: What it does and how to do it," *Proceedings of the IEEE* **78**(10), 1550—-1560 (1990).
[25] Bengio, Y., De Mori, R., Flammia, G., and Kompe, R., "Global optimization of a neural network-hidden Markov model hybrid," *IEEE Transactions on Neural Networks* **3**, 252–259 (Jan. 1992).
[26] Bengio, Y., "Learning Deep Architectures for AI," *Foundations and Trends in Machine Learning* **2**(1), 1–127 (2009).

[27] Vincent, P., Larochelle, H., Yoshua, B., and Manzagol, P. A., "Extracting and composing robust features with denoising autoencoders," in [*Proceedings of the Twenty-fifth International Conference on Machine Learning*], (July), 1096–1103 (2008).