EPFL

Improving handwritten text recognition

Curriculum Learning & Semi-Supervision

Optional Project in Data Science - COM-508
Msc in Data Science

Karthigan Sinnathamby
Student ID: 273790

IC section - EPFL - Fall 2019

Supervisor:	Sofia Ares Oliveira
Professor:	Frédéric Kaplan
Lab:	DHLAB

Contents

1 Introduction **1**

2 Related Work **1**

3 Handwritten Text Recognition **2**

4 Transfer Learning **3**

5 Curriculum Learning **4**
5.1 Description 4
5.2 Differences with the original paper 4
5.3 Results 5
5.4 Interpretation 9

6 Semi-Supervised Learning **9**
6.1 Description 9
6.2 Methods 9
6.3 Results & Interpretation 10
6.4 Future Work 11

7 Conclusion **11**

Abstract

Handwritten Text Recognition is a challenging task when dealing with few samples only. In this study, we tackle this difficult problem from the learning point of view. Transfer Learning has come to be very handy in such cases. In order to improve the accuracy of this method, we explore two methods orthogonal to it: Curriculum Learning and Semi-Supervised Learning.

1 Introduction

In recent years, handwritten text recognition (HTR) systems have seen important improvements in accuracy allowing to transcribe scripts from different sources in time and in space. However, current models (whether using Deep Learning) perform usually well on large datasets but poorly when applied to other small collections. While many datasets with different styles of writing and language have been made available to the HTR community, it is very difficult to obtain a good accuracy when dealing with a new collection. More importantly, for a given new collection, we have access to only a few samples.

Transfer Learning (TL) has come to be very handy in those situations. In the case of HTR, it aims to extract the knowledge from one (or more) large known datasets and apply this knowledge to new collection containing a few samples. Many benchmarks have been done using TL. Building on top of this method, we explore two types of learning orthogonal to TL: Curriculum Learning (CL) and Semi-Supervised Learning (SSL). The former reshape how samples are presented to the learning algorithm while the latter tries to use unlabelled data along with labelled data.

In this report, we will first describe the task of HTR and study it through the prism of TL. However the substantial part of the work will focus first on the CL study - finally on the SSL. *One should notice that we do not explore CL combined with SSL as the title may suggest it.* We eventually propose future works to continue our work.

This project aims to explore new ways of learning in order to improve the HTR accuracy when having small datasets. As such, we use transfer learning in both cases but more importantly we only focus in these learning methods and do not use necessarily the best architecture or best techniques possible.

2 Related Work

The task of HTR has been addressed many times through the prism of Transfer Learning for different datasets in different languages (Aradillas et al. (2018), Chatterjee et al. (2019)).

Here, we focused on the task of Curriculum Learning theorized by Bengio et al. (2009). CL has been explored several times, in particular by Hacohen and Weinshall (2019) on the task of image classification. This type of learning has not been used in HTR to our knowledge.

In the same context, semi-supervision learning is a widely known method (Bagherzadeh and Asil (2019)) but never used in the case of HTR to our knowledge.

3 Handwritten Text Recognition

Handwritten Text Recognition is the task of extracting text from images of cursive text written by humans. For many years, HTR systems have used the Hidden Markov Models (HMM). However recently through Deep Learning, the Convolutional Recurrent Neural Networks (CRNN) approach has been used to overcome some limitations of HMM. It has proven to be far more powerful.

We describe here the entire pipeline. The input image is fed into the CNN layers to extract features giving a feature map as output. We then perform a column-wise concatenation of the channels before giving the matrix to a RNN which is able to propagate information over longer distances and provide more robust features. The output matrix of the RNN is fed to a Connectionist Temporal Classification (CTC) layer (Graves et al. (2006)) which calculates the loss value and also outputs a prediction text.

The main problem with HTR models is to guess the letter for a given horizontal position. The RNN considers different "timesteps" and gives a prediction for each of the timesteps. From that, we need to guess the final text string as we can see on fig. 1. However we do not want to annotate the images at each horizontal position. With the CTC layer, we only feed the output matrix of the RNN and the ground-truth label and the CTC layer will try all possible alignments of the true text to compute a loss.

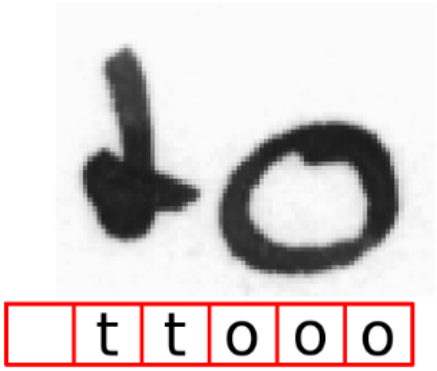

Figure 1: Example of RNN output

The exact model used in this study is the same as the one by Puigcerver (2017) which has proven to be close to state-of-the-art to this date. This paper has led to a well-known code for HTR written in Pytorch: Pylaia *PyLaia* (2017). In this study, the code is implemented on top of this library.

HTR models fed the predicted text to a decoder, the basic one being the Greedy CTC Decoder (Hwang and Sung (2016)). More powerful methods do exist such as Word Beam Search (Scheidl et al. (2018)) but as the goal of the study is to explore new ways of learning, we only use the vanilla search.

We then compute the Levenshtein distance in order to eventually have the Character Error Rate (CER) which scores the accuracy of the models.

We must bear in mind that no semantic information is used in this pipeline, it is purely based on the image and not on any language model or whatsoever.

Information about the Experiences

We have modified the well written PyLaia Code (*PyLaia* (2017)).

We monitor the validation CER in all the trainings and we stop the training if the validation CER is not improving after 20 epochs (Early Stopping). This is important to bear in mind when looking at the results curves.

The training and validation split in this study are respectively 60% and 10% of the entire dataset unless explicitly said.

Concerning the datasets, a few collections are used recurrently when it comes to HTR. In this study, we have used the IAM (Marti and Bunke (2002)), Washington (Fischer et al. (2012)) and two collections of the ICFHR competition (*ICFHR* (2018)) - Bentham and Goethe.

A GPU NVIDIA P100 (10 teraFLOPS - FP32) was used on Gcloud for all trainings.

4 Transfer Learning

Transfer Learning (TL) is a method where a model developed for a task is reused as the starting point for a model on a second task. In the case of HTR, we train a model on a dataset A and starting from the weights obtained at the end of this first training, we train again the model on a dataset B. Usually, TL is used when we want to obtain good accuracy on a small dataset, in which case the dataset A is a large known one and we fine-tune the weight on the small dataset which interests us.

Aradillas et al. (2018) explores deeply TL in the case of HTR. In particular, when dealing with TL, one can freeze some layers and train the remaining layers on the dataset B. Aradillas et al. (2018) shows by experiment that the most effective way in the case of HTR is **to leave the RNN layers and the FC layer trainable while freezing (not training the weights) the CNN layers**. This goes along with the idea that CNN layers extract features from the images and given those features we can decode the text with the RNN and FC layers. When changing the datasets, one may agree that we only need to change how the features of the images are connected to each other (so the RNN and FC layers) with respect for instance to the style of writing.

If we use a different alphabet, we will need to train the dataset A on a new FC layer adapted to the new alphabet - most likely the network will put the new weights to zero.

In this study, the dataset A is the IAM dataset ($\sim 13K$ samples). A full training on IAM dataset gives an approximate CER of 5%.

A transfer learning is performed on the Washington dataset ($\sim$ 400 samples) giving a CER of $\sim$ 4.3%, Bentham $\sim$ 35% and Goethe $\sim$ 20% ($\sim$ 500 samples both).

Having recall the reader the method used to deal with Handwritten Text Recognition and how Transfer Learning is performed, we will now talk about the core subject: Curriculum Learning and Semi-Supervised Learning. Both methods relies on Transfer Learning as our main goal is to improve the accuracy on small datasets.

5 Curriculum Learning

5.1 Description

Theorized by Bengio et al. (2009), Curriculum Learning (CL) focuses on which specific sample should be presented to the model training at which step. The main idea is inspired by the fact that *"Humans and animals learn much better when the examples are not randomly presented but organized in a meaningful order which illustrates gradually more concepts, and gradually more complex ones."*.

In other words, one should present the samples of the dataset in a precise order and maybe not in a random order as it is usually done. We first need to rank the samples by "difficulty of learning" and then present these samples gradually to the model while training. In some sense, this smooths the learned function.

In this study, we implement the algorithm of Hacohen and Weinshall (2019) given in fig. 2.

Algorithm 1 Curriculum learning method

Input: *pacing function* g_ϑ, *scoring function* f, data $\mathbb{X}$.
Output: sequence of mini-batches $\left[\mathbb{B}'_1, ..., \mathbb{B}'_M\right]$.
sort $\mathbb{X}$ according to f, in ascending order
$result \leftarrow []$
for all $i = 1, ..., M$ **do**
  $size \leftarrow g_\vartheta(i)$
  $\mathbb{X}'_i \leftarrow \mathbb{X}\,[1, ..., size]$
  uniformly sample $\mathbb{B}'_i$ from $\mathbb{X}'$
  append $\mathbb{B}'_i$ to $result$
end for
return $result$

Figure 2: Curriculum Learning Algorithm from Hacohen and Weinshall (2019)

This algorithm supposes a ranking function f named "scoring function" that ranks the samples and a "pacing function" g that present them in a specific order to the model during the training. The difficulty is to find such function f and g as to have an efficient training.

For the pacing function g, one can refer to the fig. 3. Basically, it is a function that outputs the fraction of the dataset **sorted** according to the rank, used in the training for each iteration. Here an iteration is a batch iteration and not an epoch.

For the ranking function, we can simply score the samples with the CER given by a classical Transfer Learning.

5.2 Differences with the original paper

In the paper, they use a metric computed on the training set as the rank of the sample which introduce bias. With this in mind, we use the validation CER as the rank of the

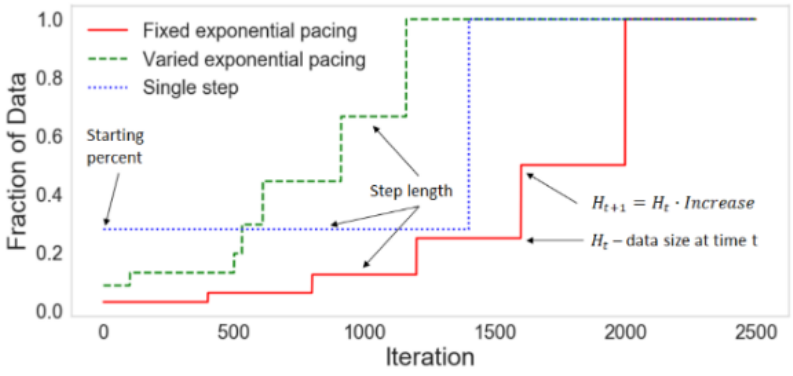

Figure 3: Pacing function from Hacohen and Weinshall (2019)

samples. In order to have one rank per sample, we must then perform a cross-validation (with the same training/validation split of 6-to-1). To summarize, we apply the following algorithm:

- Step 1: Cross-validation to extract the validation CER per sample using transfer learning
- Step 2: Use curriculum learning algorithm fig. 2 along with transfer learning on a new split

5.3 Results

We present here the results obtained on the three datasets. For all the experiments, we have computed the Validation CER across iteration for the following methods:

- Baseline - 100%: classical Transfer Learning on the entire dataset
- Baseline - 30%: classical Transfer Learning on 30% of the entire dataset
- CL: we use the Single Step pacing function of fig. 3, starting with a fraction of 30% of the dataset and using the entire dataset after 1500 iterations - for ICFHR collections, we use 1000 iterations instead of 1500.
- CL - Reverse - Single Step - (-, 1500, 0.3): the same as the previous method but in a reverse order.

Washington

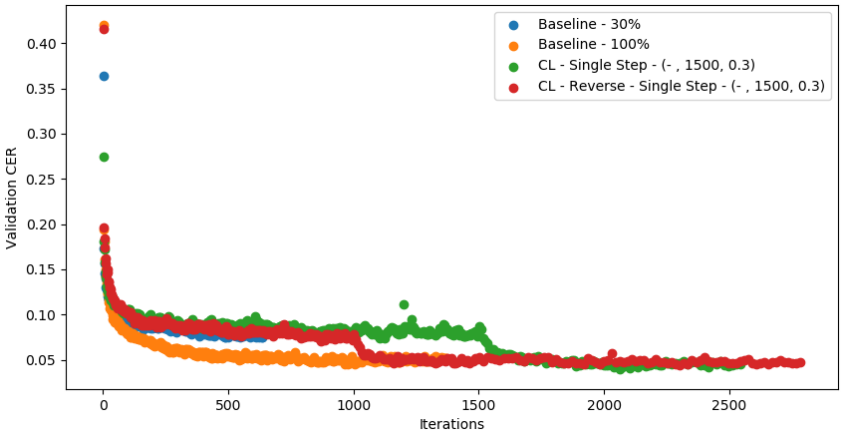

Figure 4: Validation CER across iterations for Washington Dataset

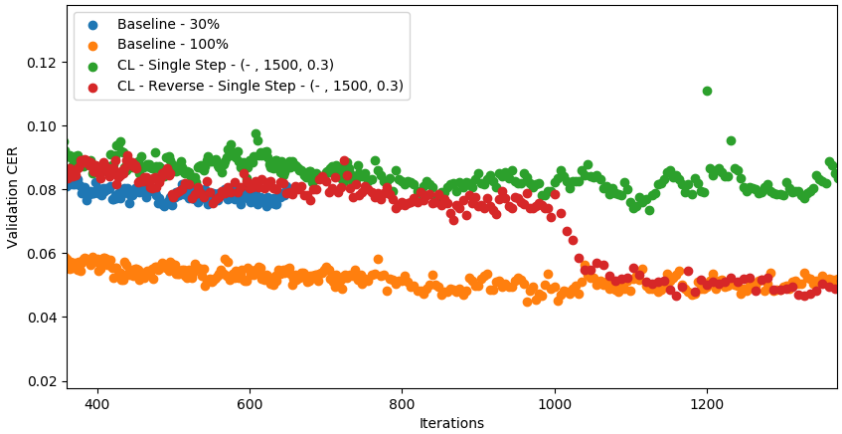

Figure 5: Validation CER across iterations for Washington Dataset - Focus on last iterations

We can see from fig. 5 that in the case of Washington dataset, we have no improvement of CL related methods. We have tried other pacing functions and other parameters for the "Single Step" function without any more improvement. The distribution of the ranks is quite spread as shown in fig. 6 which is important: the CL needs to see samples of different difficulty of learning. Maybe it is not spread enough.

We can clearly see that once the CL algorithm uses the entire dataset we go down to the convergence value of the baseline. Also, we do not improve compared to when using 30% of the dataset. Even the reverse CL does good on this dataset.

Even though, we have presented the results only for one seed, the analysis is the same with respect to how close the curves are to each other.

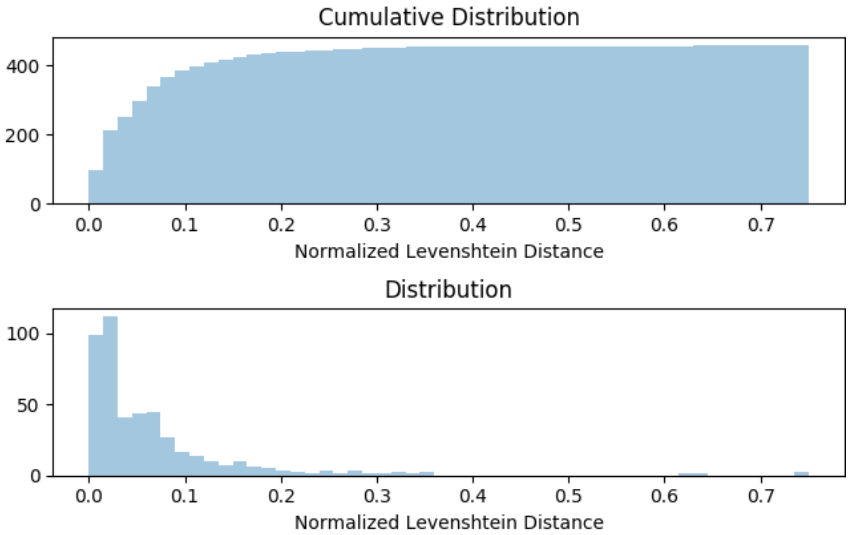

Figure 6: Normalized Levenshtein Distance Distribution for Washington Dataset

Bentham

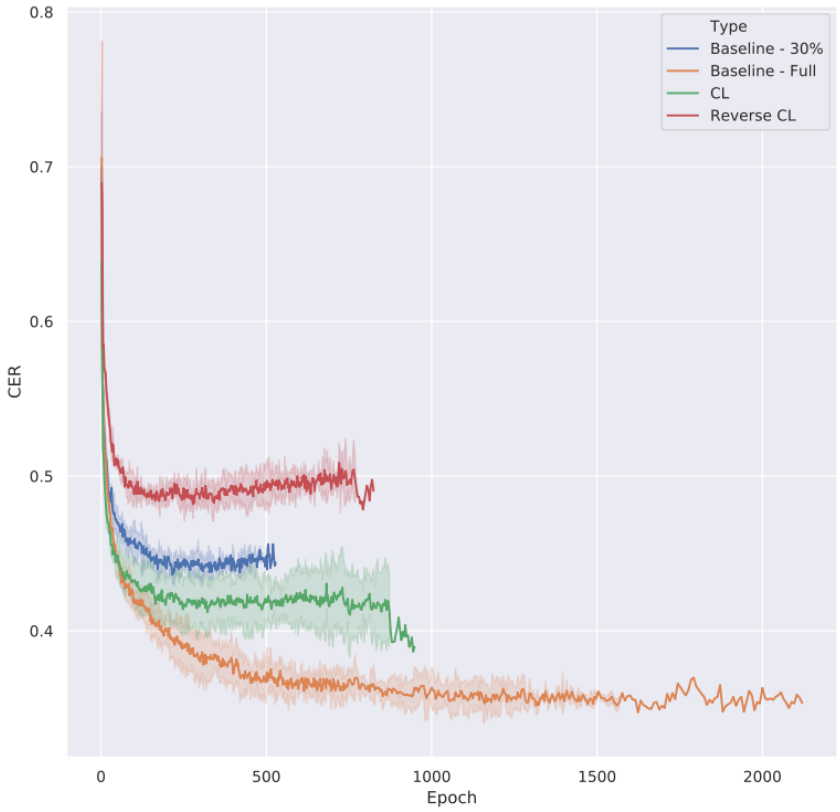

Figure 7: Validation CER across iterations for Bentham collection of ICFHR 2018 Dataset using 3 seeds

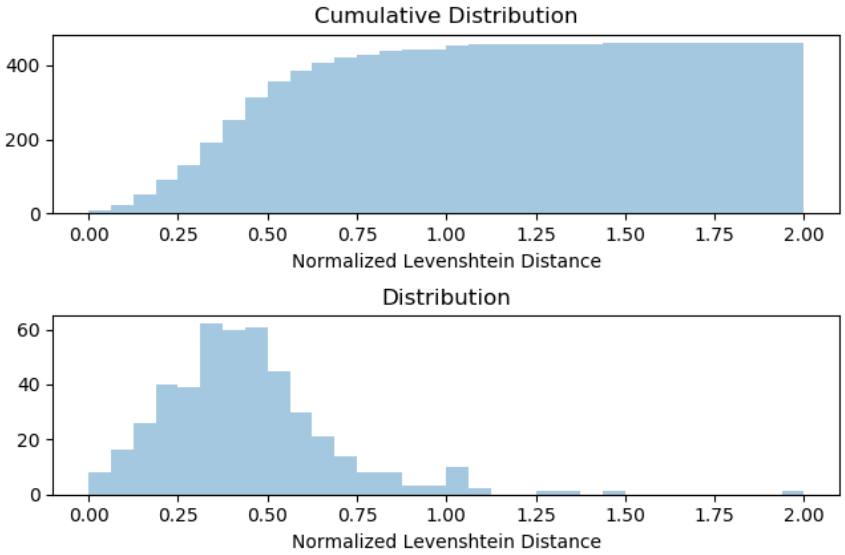

Figure 8: Normalized Levenshtein Distance Distribution for Bentham collection of ICFHR 2018 Dataset

Goethe

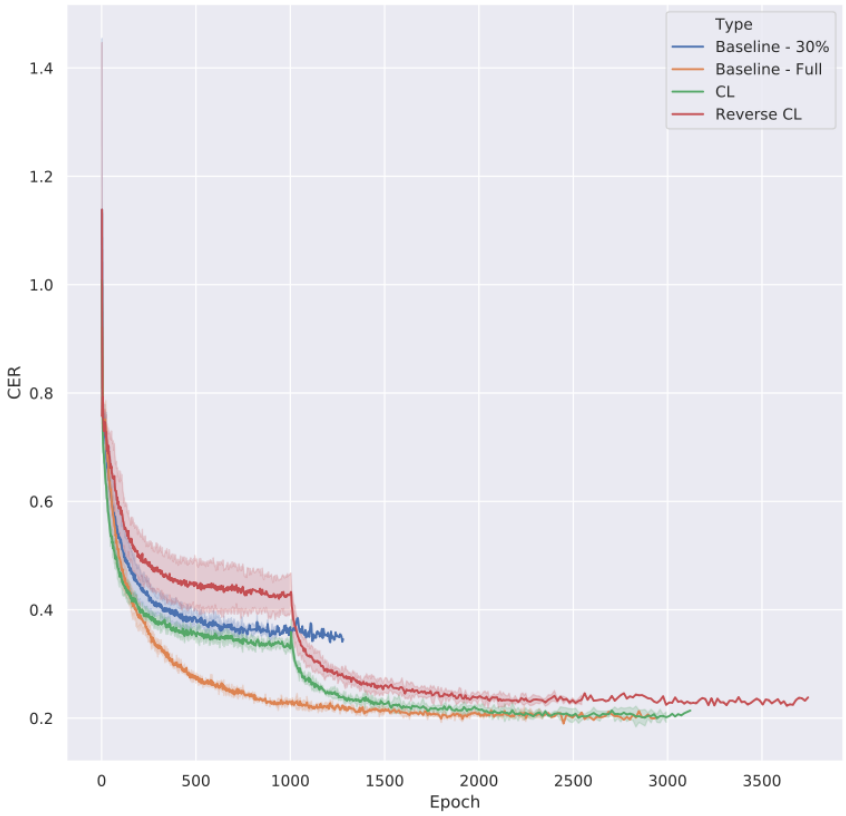

Figure 9: Validation CER across iterations for Goethe collection of ICFHR 2018 Dataset using 3 seeds

From the previous figures Bentham (fig. 7) and Goethe (fig. 9), we can see that CL does not improve the baseline on the full dataset. However, it does better than using 30% of the dataset. The Reverse CL is far worse as expected.

5.4 Interpretation

From a general perspective, Curriculum Learning seems not to improve the results in our case.

We can explain this by pointing out that we use Curriculum Learning with Transfer Learning. As such, the action potential / leeway of CL is lower. More precisely, we can see that CL is not better than baseline-30% on Washington dataset; but it is more accurate than the baseline-30% on Bentham than it is on Goethe. This suggest that whenever TL leaves place for improvement CL gets advantage of it. Therefore, on complex datasets like Bentham, CL can be advantageous but we are talking of 40% of validation CER which is quite bad already,

Another explanation could be the speed of the convergence. We can see on the curves that the baselines fastly converge to the final value which does not let enough time for CL to learn enough.

Finally, the use of TL can introduce some bias on the CL as we use TL to rank the samples as well as during the CL part.

All in all, Curriculum Learning does not seem to benefit in our challenging problem. Reverse Curriculum Learning is performing worse or does not change the performance as expected: we present the most difficult samples first during the training.

6 Semi-Supervised Learning

6.1 Description

Semi-Supervised Learning takes advantage of both Supervised Learning and Unsupervised Learning. The goal is to use labelled data as well as unlabelled data. Indeed, we have tons on unlabelled data but of course not exploitable. The power of Deep Learning methods comes from the size of the dataset. The bigger the dataset, the more accurate the model is. If we find a way, to give a kind of label to the unlabelled samples and treat those as if they were really labelled, we will have a bigger dataset to work with. In this context, given a way to label unlabelled data, Semi-Supervised Learning comes at the border of both worlds.

As one can imagine, the difficulty is to label those unlabelled data. A way to deal with this is to take the output of the model as labels for those samples.

In this part, we focus on Bentham and Goethe as these are the most difficult datasets we have dealt with in this study.

6.2 Methods

We split the training dataset in two parts: one that is considered labelled for the model and another unlabelled. The split fraction is a parameter of the training. We name the labelled dataset "dataset A" and "dataset B" a subset of the unlabelled dataset. The dataset B is the one added to A when training. **The subset is not necessarily fixed during training.**

After a few steps of transfer learning on the labelled dataset, we can create B. We either start after achieving a certain validation CER threshold, or after no improvement in validation CER, or directly at the beginning. This is parametrized.

To counter the fact that some predictions are too bad to be put in B, we need a metric to rank the unlabelled data to be added to the labelled dataset. This metric quantifies how confident the model is about the output.

We consider two metrics: we call the first one "diff-proba" while the second one is entropy based. The goal is to have a high score whenever the model is confident about its prediction. For the "diff-proba" method, we consider the difference of the two best probabilities at each timestep and we take the median of all these differences to have one score per sample. For the entropy based method, we compute the entropy at each timestep defined according to Inkeaw et al. (2018) as in eq. (1). We recall that the entropy is a measure of disorder that is why we have the opposite of the entropy here to have a score (a measure of order and thus of confidence). We take the samples that have a metric above a threshold. This threshold can be defined for instance to be the median of the metric computed on the dataset A to have a number to refer to dynamically.

$$p_{max} * log(p_{max}) + \sum_{p \neq p_{max}} (1-p) * log(1-p) \tag{1}$$

Moreover, every X epochs, we update dataset B with the current state of the model. Whether we consider dataset A along with dataset B is also a parameter (a fraction of the dataset A). If we use a fraction of A, we simply randomly sample from it.

Furthermore, in order to improve the output of the model used as labels for the dataset B, we consider a Spell Checker (*SymSpell: Symmetric Delete spelling correction algorithm* (n.d.)) to try to correct the output and to have a better label. This Spell Checker is primative, it is corpus-based and computes the Levenshtein distance between the input and the words in its corpus to recommend better words. If we have a better word, we take it (as we have no mean to distinguish between two recommendations that are at the same Levenshtein distance, we take one at random). If not, we keep the prediction.

All in all we have multiple parameters:

- When we start considering a dataset B, in other words when se start SSL
- The frequency of update of the dataset B
- The fraction of dataset A used when using a dataset B
- The threshold on the rank if we do not use the median on dataset A
- The metric used to rank the samples
- The use of Spell Checker

6.3 Results & Interpretation

For these experiments, the dataset A contains 200 samples and the dataset B is a subset of the remaining samples. We precise in table 1 some of the results for Bentham. The results for Goethe lead to the same interpretation below.

	Validation CER
Baseline	39%
Start after 50% validation CER with entropy, full dataset A	41%
Start after 50% validation CER with diff-proba, full dataset A	40%
Start after 50% validation CER with diff-proba, 30% of dataset A	45%
Start after 50% validation CER with diff-proba, no dataset A	49%
Start after 50% validation CER with diff-proba, full dataset A use of the third quartile instead of the median of the metric on A	41%

Table 1: Validation CER for different methods and parameters on Bentham. Frequency of update (8 epochs), use of spell checker

The Semi-Supervised Learning seem not to improve the performance. We converge in the best case to the same value. The only difference is the speed of learning which is longer than when being in the baseline case.

The problem is that we tell the model that whenever it is confident, then it is good as we take the output of the model as true label of one of its next dataset sample. As such, it does not really improve on those samples and stabilizes itself. This is confirmed when we remove some samples of the dataset A and that we see the accuracy is worse. The model needs good data with true label as in dataset A to make progress.

In the same context, one sanity check that has been done is not to use dataset A but only a dataset B. In this case, the model does not make progress because it cannot learn much from those fake labels which are actually its own predictions (cf line 5 of table 1). We come to the previous problem.

Therefore, we have the same performance as if we had only the dataset A but with a longer training.

As such, we need a better way of giving labels to unlabelled data and this is where the Spell Checking comes forward. However, the performance is not better, As we can see in fig. 10 and fig. 11, the spell checker is not good enough to help the model. We would need a better language model which comes to a future work.

6.4 Future Work

Even though th Spell Checking is not helping much in our case, we can see from the examples in fig. 10 and fig. 11 that there is room for improvements. Indeed, with a language model that takes into account semantic similarity, entity recognition, etc... we would be able to transform the prediction of the model to a better label for the semi-supervision. Most of the work is done by the transfer-learning; if we can now help the labelling part a bit, it will be a win-win. For instance, on sample 5 of Goethe examples, a language model will most likely find that the right word is not "wat" but "was" or in sample 8 instead of "meinne" having "meinem" which is most likely if we know German grammar. With some better "fake" labels, we believe the model will make considerable improvement on the performance.

7 Conclusion

In this project, we have explored two different types of learning on top of Transfer Learning. Both Curriculum Learning and Semi-Supervised Learning have not proven to work

better in the challenging task of HTR on small datasets. The Transfer Learning seems to do much of the work that can be done which does not leave Curriculum Learning enough room for improvement. However, Semi-Supervised Learning along with a language model seems to be a good track to explore. In HTR, people usually distinguish the character recognition part (which we have mainly tackled in this study) and the string correction part (which deals with the language itself). We are quite confident that mixing both here can improve the accuracy.

References

Aradillas, J. C., Murillo-Fuentes, J. J. and Olmos, P. M. (2018). Boosting handwriting text recognition in small databases with transfer learning, *CoRR* **abs/1804.01527**.
URL: *http://arxiv.org/abs/1804.01527*

Bagherzadeh, J. and Asil, H. (2019). A review of various semi-supervised learning models with a deep learning and memory approach, *Iran Journal of Computer Science* **2**(2): 65–80.
URL: *https://doi.org/10.1007/s42044-018-00027-6*

Bengio, Y., Louradour, J., Collobert, R. and Weston, J. (2009). Curriculum learning., *in* A. P. Danyluk, L. Bottou and M. L. Littman (eds), *ICML*, Vol. 382 of *ACM International Conference Proceeding Series*, ACM, pp. 41–48.
URL: *http://dblp.uni-trier.de/db/conf/icml/icml2009.htmlBengioLCW09*

Chatterjee, S., Dutta, R. K., Ganguly, D., Chatterjee, K. and Roy, S. (2019). Bengali handwritten character classification using transfer learning on deep convolutional neural network, *CoRR* **abs/1902.11133**.
URL: *http://arxiv.org/abs/1902.11133*

Fischer, A., Keller, A., Frinken, V. and Bunke, H. (2012). Lexicon-free handwritten word spotting using character hmms, *Pattern Recognition Letters - PRL* **33**: 934–942.

Graves, A., Fernández, S., Gomez, F. and Schmidhuber, J. (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural 'networks, Vol. 2006, pp. 369–376.

Hacohen, G. and Weinshall, D. (2019). On the power of curriculum learning in training deep networks, *CoRR* **abs/1904.03626**.
URL: *http://arxiv.org/abs/1904.03626*

Hwang, K. and Sung, W. (2016). Character-level incremental speech recognition with recurrent neural networks, *CoRR* **abs/1601.06581**.
URL: *http://arxiv.org/abs/1601.06581*

ICFHR (2018). `https://scriptnet.iit.demokritos.gr/competitions/10/1/`.

Inkeaw, P., Bootkrajang, J., Gonçalves, T. and Chaijaruwanich, J. (2018). Handwritten character recognition using active semi-supervised learning, *in* H. Yin, D. Camacho, P. Novais and A. J. Tallón-Ballesteros (eds), *Intelligent Data Engineering and Automated Learning – IDEAL 2018*, Springer International Publishing, Cham, pp. 69–78.

Marti, U.-V. and Bunke, H. (2002). The iam-database: An english sentence database for offline handwriting recognition, *International Journal on Document Analysis and Recognition* **5**: 39–46.

Puigcerver, J. (2017). Are multidimensional recurrent layers really necessary for handwritten text recognition?, *14th IAPR International Conference on Document Analysis and Recognition, ICDAR 2017, Kyoto, Japan, November 9-15, 2017*, pp. 67–72.
URL: *https://doi.org/10.1109/ICDAR.2017.20*

PyLaia (2017). `https://github.com/jpuigcerver/PyLaia`.

Scheidl, H., Fiel, S. and Sablatnig, R. (2018). Word beam search: A connectionist temporal classification decoding algorithm, *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp. 253–258.

SymSpell: Symmetric Delete spelling correction algorithm (n.d.). `https://github.com/wolfgarbe/SymSpell`.

- Sample 1
  - output: ['sale', 'of', 'the', 'care', 'the', 'right', 'has', 'for', 'its', 'subject', 'santa']
  - corrected: ['sale', 'of', 'the', 'care', 'the', 'right', 'has', 'for', 'its', 'subject', 'santa']
  - target: ['value.', 'In', 'this', 'case', 'the', 'right', 'has', 'for', 'its', 'subject', 'matter']
- Sample 2
  - output: ['shall', 'tate', 'on', 'hend', 'is', 'the', 'cred', 'bonnet', 'off', 'a', 'ord']
  - corrected: ['shall', 'tate', 'on', 'send', 'is', 'the', 'cred', 'bonnet', 'off', 'a', 'ord']
  - target: ['shall', 'take', 'in', 'hand', 'in', 'the', 'word', 'Command.', 'But', 'that', 'word']
- Sample 3
  - output: ['fro', 'shame', 'postal', 'an', 'the', 'comment', 'a']
  - corrected: ['fro', 'shame', 'postal', 'an', 'the', 'comment', 'a']
  - target: ['Two', 'volumes', 'published', 'since', 'the', 'commencement', 'of']
- Sample 4
  - output: ['to', 'the', 'hend', 'may', 'to', 'is', 'other', 'hend', 'con', 'the', 'salyet']
  - corrected: ['to', 'the', 'send', 'may', 'to', 'is', 'other', 'send', 'con', 'the', 'salyut']
  - target: ['To', 'this', 'head', 'may—to', 'no', 'other', 'head', 'can—the', 'subject']
- Sample 5
  - output: ['by', 'it', 'relation', 'to', 'that', 'of', 'the', 'second', 'rot', 'st', 'a', 'stare', 'is', 'all']
  - corrected: ['by', 'it', 'relation', 'to', 'that', 'of', 'the', 'second', 'rot', 'st', 'a', 'stare', 'is', 'all']
  - target: ['by', 'its', 'relation', 'to', 'that', 'of', 'the', 'second', '", 'set—a', 'stare', 'is', 'all']

Figure 10: Examples of Spell Checking in Bentham

- Sample 1
  - output: ['Sorh', 'atwesi', 'de', 'ich', 'wannshte', 'daß', 'da']
  - corrected: ['Sorh', 'atwesi', 'de', 'ich', 'wannshte', 'daß', 'da']
  - target: ['Noch', 'etwas:', 'da', 'ich', 'wunschte', 'daß', 'der']
- Sample 2
  - output: ['Gben', 'Sie', 'reht', 'wohed', 'und', 'mahshten', 'unich', 'den']
  - corrected: ['Gben', 'Sie', 'reht', 'wohed', 'und', 'mahshten', 'unich', 'den']
  - target: ['Leben', 'Sie', 'recht', 'wohl', 'und', 'empfehlen', 'mich', 'den']
- Sample 3
  - output: ['u', 'noh', 'etwes', 'geben', 'denm', 'ur', 'wirlhere']
  - corrected: ['u', 'noh', 'etwes', 'geben', 'denm', 'ur', 'wirlhere']
  - target: ['d', 'nach', 'etwas', 'geben,', 'denn', 'er', 'wird', 'nie']
- Sample 4
  - output: ['Wagen', 'des', 'Gobmmach', 'vnde', 'soffann', 'dan']
  - corrected: ['Wagen', 'des', 'Gobmmach', 'vnde', 'soffann', 'dan']
  - target: ['Wegen', 'des', 'Almanachs', 'werde', 'ich', 'Ihnen', 'den']
- Sample 5
  - output: ['nur', 'wat', 'die', 'minigen', 'lebricht', 'mit', 'ih', 'loften']
  - corrected: ['nur', 'wat', 'die', 'minigen', 'lebricht', 'mit', 'ih', 'loften']
  - target: ['nur', 'was', 'die', 'meinigen', 'betrifft', 'muß', 'ich', 'bitten']
- Sample 6
  - output: ['sitt', 'mir', 'den', 'Sligs', 'des', 'ersten', 'Bushes']
  - corrected: ['sitt', 'mir', 'den', 'Sligs', 'des', 'ersten', 'Bushes']
  - target: ['schickt', 'mir', 'den', 'Schluß', 'des', 'ersten', 'Buches']
- Sample 7
  - output: ['Sir', 'ih', 'sfh', 'ehreilich', 'dsß', 'Sie', 'mit', 'meinne']
  - corrected: ['Sir', 'ih', 'sfh', 'ehreilich', 'dsß', 'Sie', 'mit', 'meinne']
  - target: ['Mir', 'ist', 'sehr', 'erfreulich', 'daß', 'Sie', 'mit', 'meinem']

Figure 11: Examples of Spell Checking in Goethe