UNIVERSITY OF CALIFORNIA SAN DIEGO

Unsupervised pretraining for semi-supervised OCR

A thesis submitted in partial satisfaction of the
requirements for the degree of Master of Science

in

Computer Science

by

Kishore Pacharamakalahalli Venkatswammy Reddy

Committee in charge:

Professor Taylor Berg-Kirkpatrick, Co-Chair
Professor Nuno Vasconcelos, Co-Chair
Professor Julian McAuley

2020

The thesis of Kishore Pacharamakalahalli Venkatswammy Reddy is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Co-Chair

Co-Chair

University of California San Diego

2020

DEDICATION

This work is dedicated to the tens of thousands of *Falun Dafa practitioners* killed (**at least 4363 confirmed** as of 09/02/2020 and **tens of thousands more to be confirmed**) by the Chinese Communist Party (CCP) in its persecution of *Falun Gong* [17].

EPIGRAPH

A wicked person is born of jealousy. Out of selfishness and anger he complains about supposed "unfairness."

A good person always has compassion in his heart. Free of discontentment and hatred, he sees hardship as joy.

An enlightened person has not any attachment. He quietly observes the people of the world lost in illusion.

Li Hongzhi, *Zhuan Falun Volume II*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to acknowledge Professor Taylor Berg-Kirkpatrick for his support as the co-chair of my committee. I am grateful for his guidance in this thesis and my research projects which helped me gain valuable experience in general research and NLP.

I would like to acknowledge Professor Nuno Vasconcelos for his support as the co-chair of my committee. I am thankful for his guidance in my research projects which helped me gain valuable research experience in the field of Computer Vision.

This whole work is co-authored with Berg-Kirkpatrick, Taylor and is currently being prepared for submission for publication of the material. PV Reddy, Kishore; Berg-Kirkpatrick, Taylor. The thesis author was the primary investigator and author of this material.

ABSTRACT OF THE THESIS

Unsupervised pretraining for semi-supervised OCR

by

Kishore Pacharamakalahalli Venkatswammy Reddy

Master of Science in Computer Science

University of California San Diego, 2020

Professor Taylor Berg-Kirkpatrick, Co-Chair
Professor Nuno Vasconcelos, Co-Chair

Recent works like BERT, GPT, ELMO, ULMFiT have successfully demonstrated the effectiveness of pretraining for a variety of downstream NLP tasks. We propose to use a similar approach for a different learning scenario - semi-supervised OCR/text recognition. We hypothesize that for the supervised character image classifier of an OCR system, it is more effective to classify pre-learned OCR representations of character images rather than learn to do OCR from scratch in the traditional sense. For this purpose, components of the classifier are pretrained in an unsupervised fashion to consume a sequence of character images obtained by segmenting an image of a line of text, and reconstruct the same at the output. The pretraining is

optimized for the masked language modeling objective, without access to the OCR labels for the character images. In our results, we show that a classifier trained on top of the pretrained representations achieves almost 100% higher accuracy than a classifier trained from scratch, when supervised with just 5 line images. Our pretraining procedure can leverage the large amount of unsupervised data to learn useful OCR representations and enhances the performance of supervised OCR systems, especially when supervised data is scarce like in historical documents.

Chapter 1

Introduction

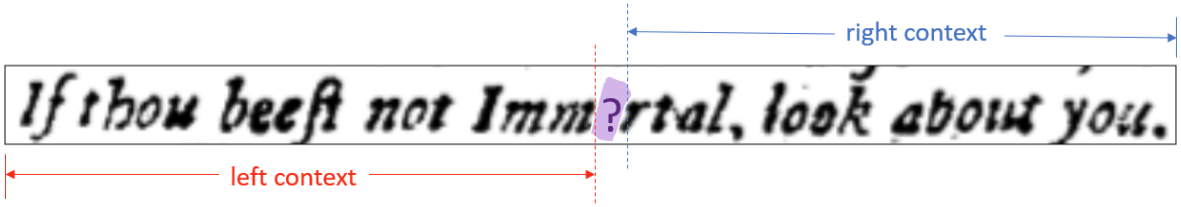

Figure 1.1. Context tells

The earliest ideas for Optical character recognition (OCR) date back to the early 1870s. OCR has come a long way since then to the modern day where we have free, open source OCR systems like Tesseract [25], OCRopus [4], Kraken [22] and Calamari [29]. Recently, multi-engine OCR systems like Okralact [1] are working towards the standardization and interoperability of the above popular OCR engines. These and most state-of-the-art OCR systems are based on deep neural networks. These supervised systems require labeled data ranging from character image annotation for a representative set of character images to annotations for whole line images. On the other hand, OCR systems like Ocular [2, 3] and anyOCR[1] [5] are unsupervised. Supervised OCR systems tend to be data hungry and labeling data for them can be time consuming and expensive. In contrast, while unsupervised systems with the promise of only using the abundant unlabeled data seem attractive, they are designed with specific applications in mind and hence do not work out of the box for unintended applications. For instance, the generative

[1] requires minimal language expert supervision

probabilistic model of Ocular is formulated to mimic the primary sources of variation and noise in the underlying printing processes for historical documents [2]. Techniques to improve the performance of supervised OCR systems in a general sense, without relying on labeled data is a practical solution that is useful and even more so especially when labeled data is scarce. Drawing inspiration from the success of unsupervised pretraining for diverse downstream NLP tasks in works like BERT [8], GPT [20], ELMO [18], ULMFit [13], we design an unsupervised pretraining procedure to improve the performance of supervised OCR systems. Specifically, we leverage unsupervised data to improve the performance of the character image classifier (in segmentation-based OCR) or line image transcriber (in segmentation-free OCR) component of a supervised OCR system. This work offers an effective solution for the former and builds the foundations for the latter.

The central objective of our work is to improve the performance of a character image classifier, the component of an OCR system that comes into play after the segmentation of character images from a line image. For this, it will be useful if we have an encoder that has learned to understand character images and subsequently generate good OCR representations in a language contextual setting.To achieve that, we pretrain our encoder (*Line-image Encoder* in Figure 3.2) in an unsupervised fashion on a Masked Language Model (MLM) objective, which consumes and produces sequences of character images. The hope is that, by matching at the pixel level and using an LSTM backbone to model language, we induce linguistic hidden representations which are necessary to predict missing characters to be reconstructed in running text. The training objective for this model pressures it to do unsupervised OCR inside the model. Once we have successfully trained the *Line-image Encoder*, then the *Classifier* head just has to extract the learned OCR representations from the *Line-image Encoder* rather than learn to do OCR. Our experiments demonstrate that the *Classifier* head paired with our pretrained *Line-image Encoder* learns to classify character images with very little supervised data. This proves that the *Line-image Encoder* learned useful OCR representation in the unsupervised pretraining.

This whole chapter is co-authored with Berg-Kirkpatrick, Taylor and is currently being prepared for submission for publication of the material. PV Reddy, Kishore; Berg-Kirkpatrick, Taylor. The thesis author was the primary investigator and author of this material.

Chapter 2

Related work

Unsupervised pretraining has been widely shown to be effective for a variety of Computer Vision tasks like image classification [6], image retrieval [19], segmentation, object tracking etc. In the field of NLP models like BERT [8], ULMFit [13] are pretrained on unlabeled text and achieve state-of-the-art results on downstream tasks like text classification, sentiment analysis, Question answering etc. All these works indicate the effectiveness of finetuning on useful representations learned in an unsupervised/self-supervised manner.

Unfortunately, in the domain of OCR, there is very little work that uses unsupervised learning to improve OCR systems. An approach for this is to train an unsupervised feature extractor and use it with a recurrent network like LSTM. Adopting this approach, in [24] a Restricted Boltzmann Machine is used as the unsupervised feature extractor. As another approach, in [15] they pretrain their OCR system in an unsupervised manner on line images by jointly optimizing for reconstruction and Connectionist Temporal Classification (CTC) [9] objectives.

There are a few works like Ocular [2, 3] and anyOCR [5] that do OCR in an unsupervised manner for historical texts. In [10], OCR is done in an unsupervised manner using line images and a sample of lexically valid strings similar to the target corpora.

Some works like [21] perform finetuning on a model pretrained on similar labeled data to get better overall OCR performance.

This whole chapter is co-authored with Berg-Kirkpatrick, Taylor and is currently being prepared for submission for publication of the material. PV Reddy, Kishore; Berg-Kirkpatrick, Taylor. The thesis author was the primary investigator and author of this material.

Chapter 3

Method and datasets

3.1 Method

For a character image classifier, it is easy to learn to classify embeddings of characters rather than character images directly. To aid our *Classifier* model in this, we pretrain its *Line-image Encoder* component in an unsupervised fashion with the expectation that it learns useful OCR representations from sequences of character images, without ever looking at the character classes. If the *Linguistic-Context* model (i.e *Line-image Encoder* coupled with a *Character-image Decoder*) did learn that, then the classifier with pretrained *Line-image Encoder* should be able to easily classify character images with very little supervision. Our results show exactly that. As enumerated in the results, with as little as training on 5 sequences of character images, the pretrained *Classifier* model can achieve accuracy close to 70%.

In the following subsections we cover notation, the components of the semi-supervised *Classifier* model, its training objective and implicit masking, and the architecture of the unsupervised *Linguistic-Context* model.

3.1.1 Notation

Following is the notation that will be followed throughout.

L - the image of a whole line of text, obtained by segmenting image of a page of text.

C - sequence of segmented character images, obtained by segmenting L.

C_i - ith character image in the C.

This notation is depicted in Figure 3.1.

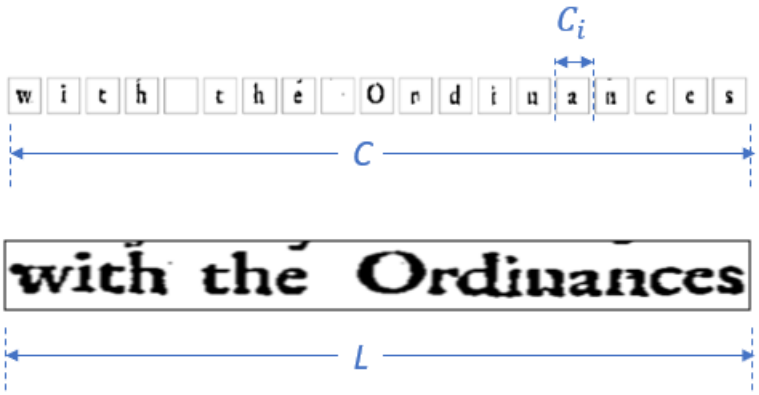

Figure 3.1. Notation used to refer to a Line image (below) and its corresponding segmented sequence of character images (above)

3.1.2 The semi-supervised *Classifier* Model

The *Classifier* model takes a sequence of character images, C as input and identifies the character classes they belong to. Its architecture is depicted in Figure 3.2.

The model has two components - the *Line-image Encoder* and *Classification Head*.

1. ***Line-image Encoder*** - it consists of ***Character-image Encoder*** and ***LSTM Block***. The *Character-image Encoder* encodes each individual character image, C_i into a fixed-size vector that is fed sequentially to the *LSTM Block*. The *Character-image Encoder* serves similar function as an embedding layer in a Language Model - to convert tokens to respective continuous vector representations. The *LSTM Block* processes the sequence, C in both forward and backward direction and produces the OCR representations to be consumed by *Classification Head*.

2. *Classification Head* - It is a linear layer that classifies each individual OCR representations from the *Line-image Encoder* to the respective character classes.

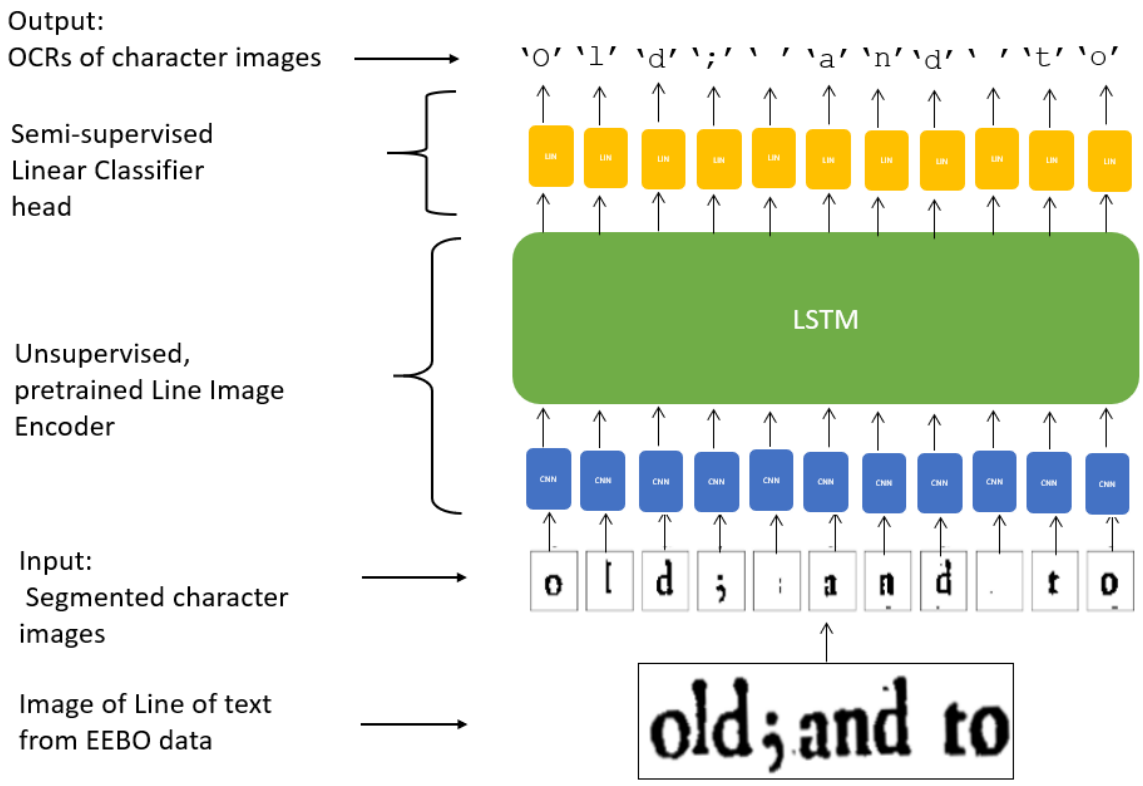

Figure 3.2. The *Classifier* model architecture

3.1.3 The unsupervised *Linguistic-Context* Model

The *Linguistic-Context* model takes a sequence of character images, C as input and reconstructs the same sequence of character images at the output. The model consists of two components - the *Line-image Encoder* and *Character-image Decoder*. Its architecture is depicted in Figure 3.3.

1. *Line-image Encoder* - It is the same as explained in section 3.1.2

2. *Character-image Decoder* - It is used to reconstruct each character image, C_i of the sequence, C independently, being fed the representations output by the *Line-image Encoder*. We experiment with a Transposed CNN and PixelCNN [26] as the decoder.

The Transposed CNN upsamples a vector representation to a character image, C_i.

The PixelCNN reconstructs each C_i pixel-by-pixel auto-regressively. It is conditioned on the output of the *Line-image Encoder*, which is a vector or a whole image (in case of our Hybrid architecture, which is generated by passing the outputs of *Line-image Encoder* through a Transposed CNN first).

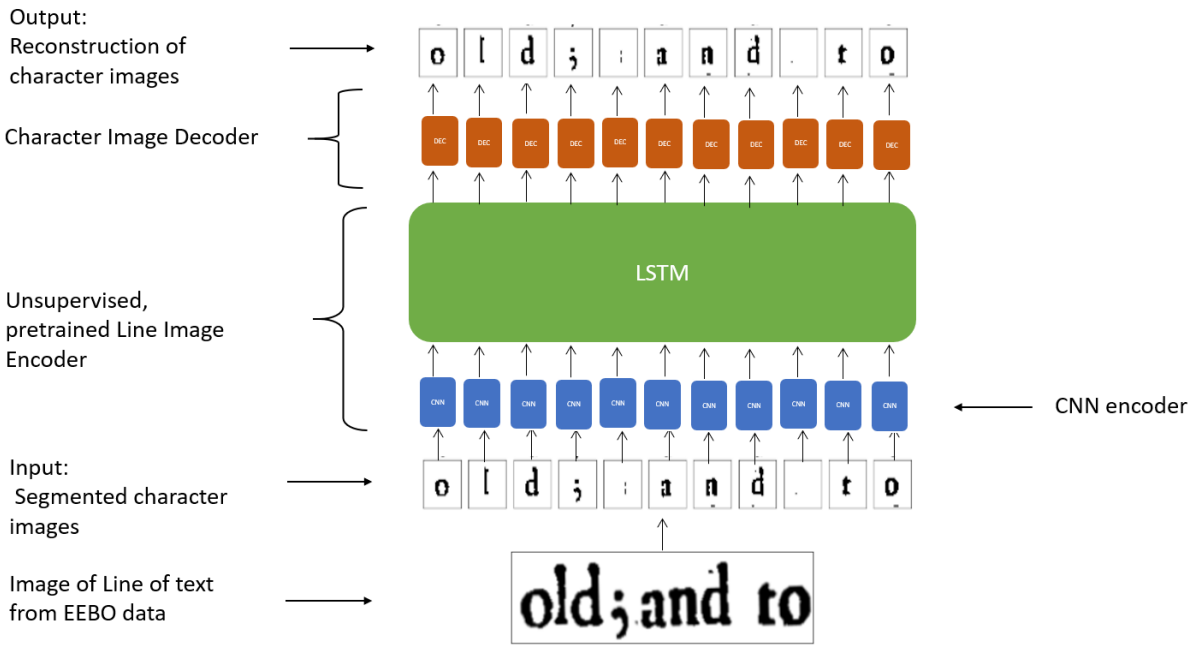

Figure 3.3. The *Linguistic-Context* model architecture

3.1.4 Training objective and Implicit Masking

We use Masked Language Model (MLM) training objective, same as that of BERT. Besides using a sequence of character images, C instead of words, our training procedure differs from BERT's training in two other ways. First, we do not explicitly mask any character image, C_i of C. Rather we have an implicit masking mechanism, that masks every C_i in C in parallel. Second, as a consequence of implicit masking, we calculate the loss on every image in the sequence at the output. The training objective is pixel-wise Cross Entropy.

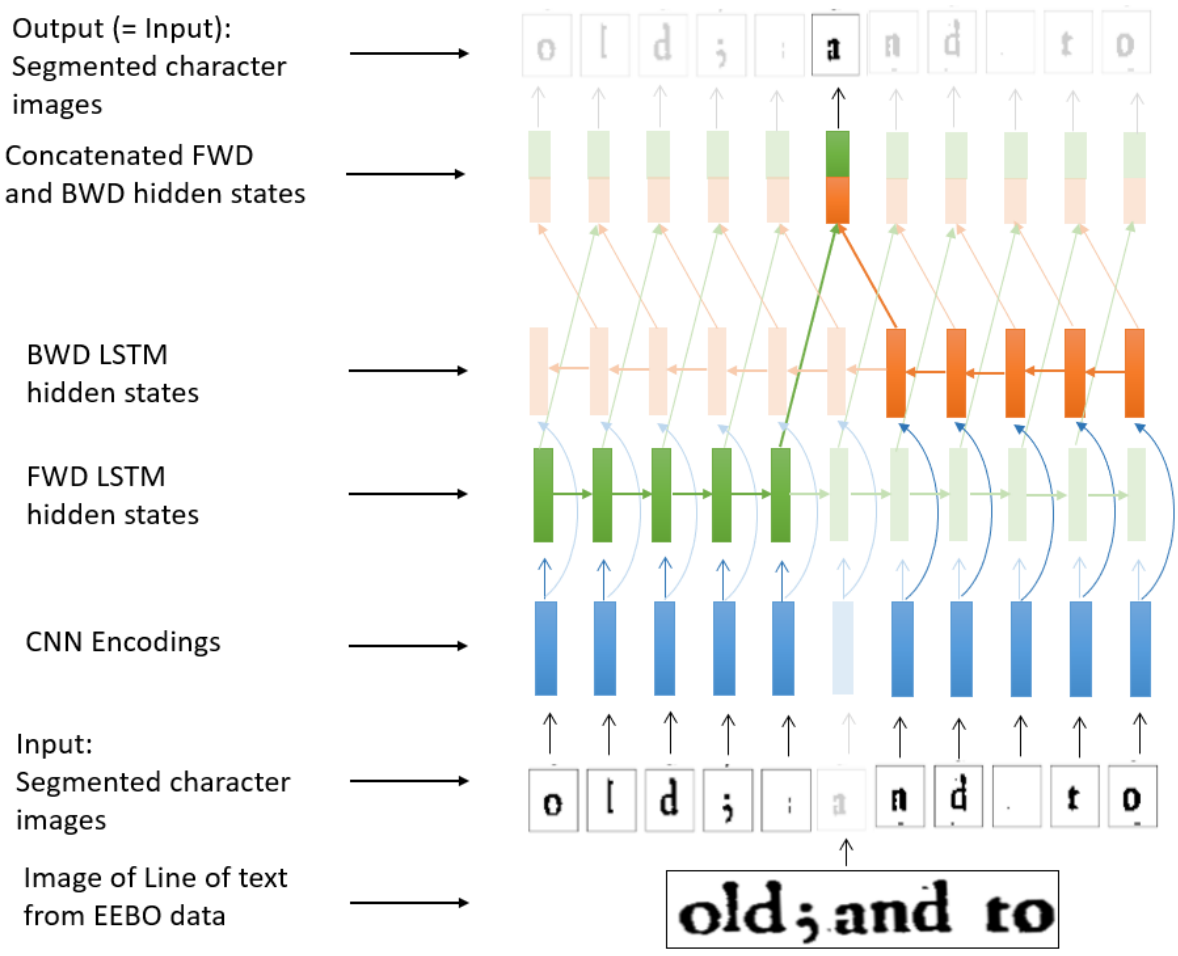

Figure 3.4. Implicit masking highlighted for the character image *a* in the input sequence

Figure 3.4 demonstrates our implicit masking procedure. We use a bidirectional LSTM in our *LSTM Block*. For each C_i in the sequence C, the forward LSTM encodes all the images before it ($C_{<i}$) and the backward LSTM encodes all the images after it ($C_{>i}$). These forward and backward hidden states are concatenated and fed to the decoder which reconstructs C_i. In the figure, the implicit masking for C_5 is highlighted (i.e. char image *a*). Note that the model does not get to see image C_5, but only its complete left and right context, while learning to produce its representation. This same implicit masking is done for all the C_i simultaneously. This parallel implicit masking can be efficiently achieved by processing C in an regular fashion by the forward and backward LSTMs, and shifting the forward and backward hidden states (with padding at the ends) before the concatenation step.

3.2 Dataset

We train and evaluate the *Linguistic-Context* model two datasets: *EEBO-79 dataset* and a synthetic dataset.

3.2.1 *EEBO-79* dataset

EEBO-79 dataset is a subset of the books in Early English Books Online (EEBO) database, an online database of around 146,000 printed English works in between 1473 and 1700.

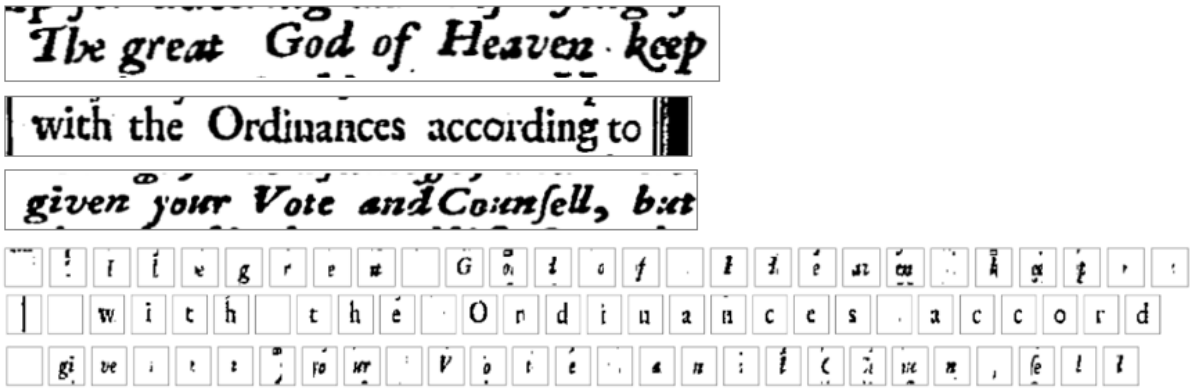

Figure 3.5. Sample line images from the *EEBO-79* dataset and their corresponding segmentations obtained from Ocular

We use line images from 79 books from EEBO for our work, referred to as *EEBO-79* here. Ocular [2, 3], an unsupervised historical OCR system, is used to segment page scans from the books into line images and subsequently each line image to character images. Line images having less than 3 characters are ignored from all the books. Our dataset consists of 224,038 line images in total. Henceforth, we refer to this subset of 79 books as *EEBO-79 dataset*.
Sample line images along with their corresponding segmentations are shown in Figure 3.5. As shown in the figure, the data from this dataset are very noisy, in terms of the overall image quality as well as the character image segmentation.

3.2.2 *Synthetic* Dataset

This dataset is synthesized from a subset of text in the Penn Tree Bank [16]. We randomly selected $211,056$ sentences from PTB. We ignore sentences which are shorter than 3 characters, and long sentences are truncated to 172 characters. The sentences have an average length of 55.2 characters. The selected sentences are converted to images by rendering the text of a sentence into a blank image [1]. During the rendering process, a small amount of Gaussian noise is added to the final line image. As it is synthetically generated, it is easy to automatically obtain the character segmentation from the line images. We split the dataset to train, dev and test sets in 60-20-20 proportion.

The primary purpose of this synthetic dataset is to perform control experiments to determine if the failure of an experiment was because of the poor quality of the *EEBO-79* data or due to some other reason.

3.2.3 Data Preprocessing

Every character image, C_i is binarized and padded to a fixed size of $32x32$ pixels.

The sequences in a batch are padded to match the length of the longest sequence and the padded entries are ignored during the learning and evaluation.

This whole chapter is co-authored with Berg-Kirkpatrick, Taylor and is currently being prepared for submission for publication of the material. PV Reddy, Kishore; Berg-Kirkpatrick, Taylor. The thesis author was the primary investigator and author of this material.

[1] Performed using *PIL library* [7]

Chapter 4

Experiments and results

4.1 Pretraining and downstream classification

4.1.1 Unsupervised Pretraining

For all the unsupervised pretraining experiments, we use the same *Line-image Encoder* with different types of *Character-image Decoder*.

Line-image Encoder - consists of a *Character-image Encoder* and an *LSTM Block*. The *Character-image Encoder* is a 4-layered CNN. Each layer has 3 x 3 convolutional filters followed by Batch Normalization [14] and a ReLU [11] non-linearity. The *LSTM Block* consists of 3 bidirectional LSTM layers. Note that the outputs of the forward and backward LSTMs are concatenated only at the final layer to ensure correct implicit masking. It produces a sequence of 256 length vectors, with the vector at time-step t corresponding to the implicitly masked t-th character image in the input sequence.

The *Character-image Decoder* is either a Transposed Convolution Network or a Pixel CNN [27]. The *Character-image Decoder* takes each 256-dimensional vector from the *Line-image Encoder* and reconstructs the corresponding (implicitly masked) 32 x 32 character image in the input sequence. The two variants of the model are discussed below.

1. TransposeConv Model - It consists of *Line-image Encoder* with Transposed Convolution decoder. The decoder consists of 3 layers of Transpose CNN. Each layer is followed by

Batch Normalization and a ReLU non-linearity. The architecture is the same as in Figure 3.3, where the decoder is Transposed Convolution decoder.

2. Hybrid Model - This model consists of a *Line-image Encoder*, above mentioned Transposed Convolution decoder followed by a Pixel CNN decoder i.e the TransposeConv model with a Pixel CNN as the decoder on the top. The Pixel CNN in this case is conditioned on the spatial output of the Transposed Convolution Block and reconstructs the 32 x 32 pixel output character image. We can think of the Pixel CNN in this case as refining the 32 x 32 dimensional output of the Transposed Convolution Block, in an auto-regressive fashion. The architecture is shown in Figure 4.1

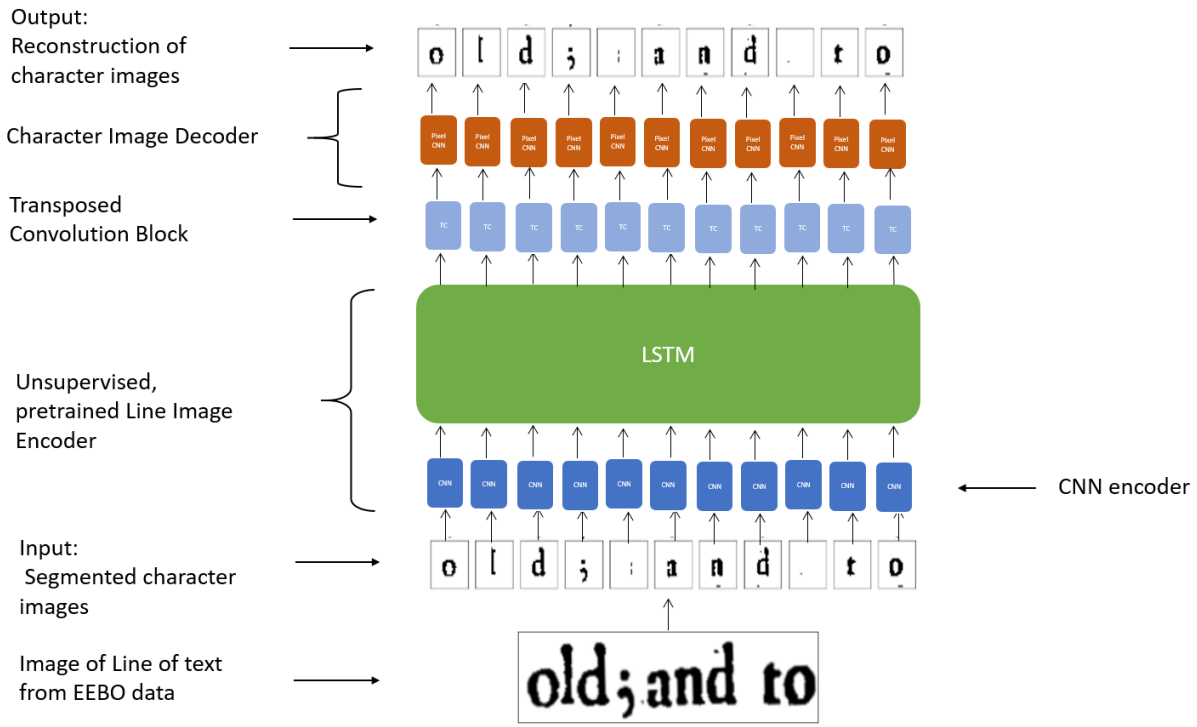

Figure 4.1. The Hybrid model architecture

Both the above experiments have two settings. In setting A, the model is trained from scratch, with random weight initializations while in setting B we pretrain the *Character-image Encoder* and the *Character-image Decoder* as an autoencoder [23] on individual character

images extracted from the line images.

We trained an oracle character-image classifier supervised on character-class labels generated by Ocular for *EEBO-79* dataset. To evaluate the quality of the reconstructions, we report the accuracy of this oracle classifier on each character image of the reconstructed sequence of images.

The results for *EEBO-79* and *Synthetic* datasets are shown in Table 4.1 and Table 4.2 respectively. For both the datasets, the Hybrid model learns better than the TransposeConv model. On *EEBO-79* dataset, the hybrid model finetuned on autoencoder weights gives the best performance. Surprisingly for the *Synthetic* dataset, the finetuning of the Hybrid model initialized with autoencoder weights decreases the performance sharply.

Table 4.1. Results for pretraining experiments on *EEBO-79* dataset

	Accuracy	
Experiment	**From Scratch**	**Pretrain as AE**
TransposeConv Model	19.0	21.0
Hybrid Model	34.4	**35.7**

Table 4.2. Results for pretraining experiments on *Synthetic* dataset

	Accuracy	
Experiment	From Scratch	Pretrain as AE
TransposeConv Model	51.0	73.0
Hybrid Model	**83.0**	73.0

4.1.2 Semi-supervised Downstream Classification

To test of effectiveness of the OCR representations learned by the *Line-image Encoder*, we train a character-class classifier on these representations. We experiment with different amounts of supervised data to demonstrate the effectiveness of our unsupervised pretraining over training the classifier from scratch.

Dataset

We use line images from the *EEBO-79* dataset and the OCR labels generated by Ocular to create the dataset. For this, we randomly pick 1500 line images, balanced across the 79 different books in the dataset. We split these sequences into train, dev and test data, ensuring that the books in each split do not overlap. The train set has 250 lines images and the dev and test have 1000 line images each.

We formulate the following experiments:

1. *Line-image Encoder* with *Classification Head* - This architecture is illustrated in Figure 3.2. We run this experiment in three settings - A,B and C. In setting A (LEC-Scratch) , we train the whole model from scratch. In setting B (LEC-PretrainLE(Hybrid)), we initialize the whole *Line-image Encoder* with the weights from pretrained *Hybrid* model and train just the *Classification Head*. Setting C (LEC-PretrainLE(Hybrid)-Concat) is the same as that of B, and we additionally concatenate the learned embeddings of the *Character-image Encoder* along with the *Line-image Encoder* before feeding it to the *Classification Head*. The *LSTM Block* in these experiments is a 3-layer bidirectional LSTM[12], with 256-dimensional state and hidden vectors. Here, we do not do implicit masking for the *Line-image Encoder*.

2. *Character-image Encoder* with *Classification Head* - This architecture is the same as in Figure 3.2, minus the *LSTM Block*. The outputs of the *Character-image Encoder* are directly classified by the head, without access to information about contextual character images provided by the *LSTM Block*. The experiment is run in three settings - A,B and C. In setting A (CEC-scratch), we train the whole model from scratch. In setting B (CEC-PretrainCE(AE)), we initialize the *Character-image Encoder* with the weights from a pretrained character image autoencoder model and train just the *Classification Head*. In setting C (CEC-PretrainCE(Hybrid)), we initialize the *Character-image Encoder* with the weights from a pretrained *Hybrid* model and train just *Classification Head*.

For the above, we train with 6 different train set sizes - 5, 10, 25, 50, 100 and 250 sequences, and test each of them with all the 1000 test sequences. The classifier is trained on the Cross Entropy objective. The test set accuracies are show in Table 4.3 and plotted in Figure 4.2.

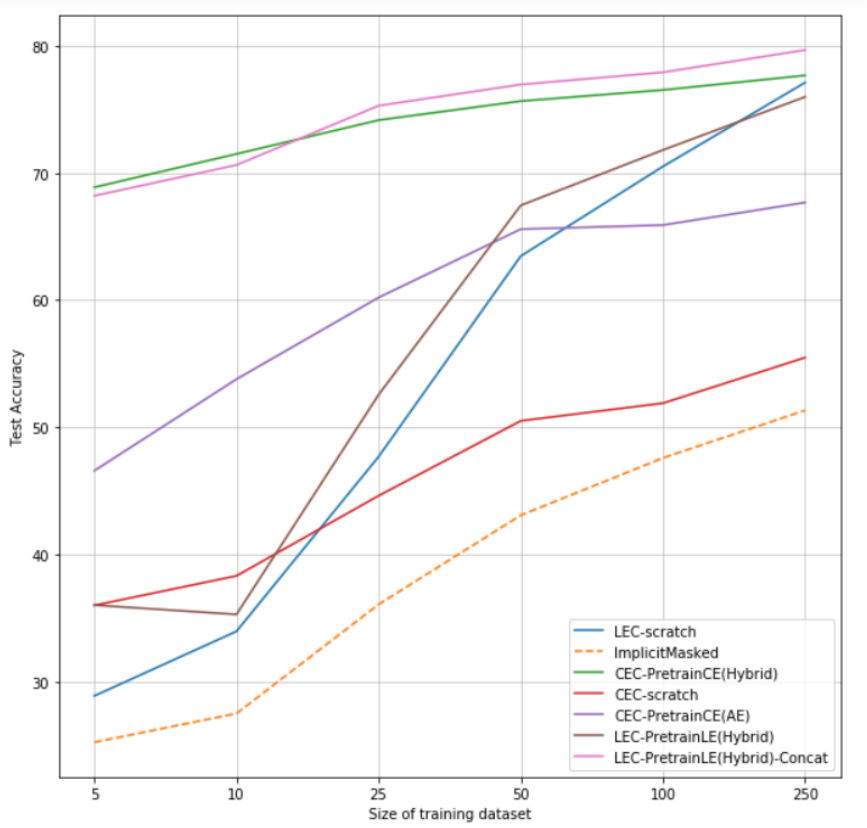

Figure 4.2. Results for downstream classification for varying amount of train data

The results clearly show the benefits of pretraining. When trained with just 5 sequences, the models trained from scratch achieve an accuracy of 36.0%, while one of our pretrained variants achieves almost double accuracy of 68.9%. The pretrained variants (MLM as well as autoencoder) prove to be superior to the equivalent models trained from scratch as we increase the size of the training dataset, till 100 samples. While pretraining as an autoencoder (purple plot in figure) is better than training from scratch, it still lags behind the models pretrained with our method. Pretrained model utilizing the learned embeddings of both the *Character-image Encoder* along with the *Line-image Encoder* (LEC-PretrainLE(Hybrid)-Concat, pink plot in

figure) gives the best results on the classification task for the majority of train sizes. As expected, the model trained from scratch catches up to the pretrained models as the size of the train dataset increases. The fact that our best pretrained models (in pink and green) start off with almost 70% accuracy after fine tuning on just 5 sequences implies that our pretraining procedure was successful in learning OCR representations without any supervision. Additionally, the accuracy does not drastically improve as we exponentially increase the number of train samples. This further supports our claim.

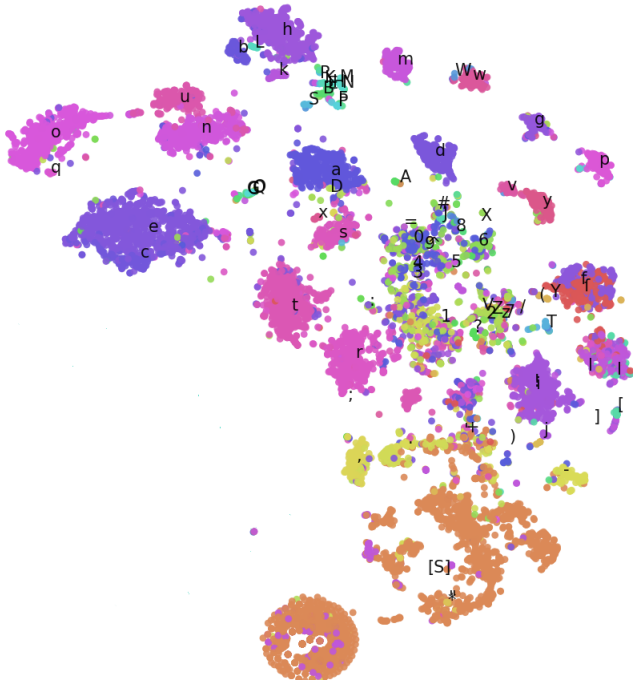

Figure 4.3. t-SNE plot of embeddings learned by the *Character-image Encoder* of *Linguistic-Context* model on *EEBO-79* dataset, where the clusters are encoded by different colors with the character annotation at the corresponding cluster center

Figure 4.3 shows the t-SNE [28] plot of the character embeddings (of *Character-image Encoder*) learned by the Hybrid model. The model learns to segregate different characters in the vocabulary, at the same time grouping semantic or visual symbols closer. Punctuation marks, space (labeled as *[s]* in the figure) and non-alphanumeric symbols are separated from the alphanumeric symbols. Similar looking pairs like *c* and *e*, *v* and *y*, *l* and *I* etc. are in close

vicinity. Clusters of all the digits appear close, though they vary visually. The clusters for some symbols are almost completely overlapping (ex: *S*, *B*, *K* and *R* clusters) and others which should be close together are not (ex: *(* and *)*). This behavior is partly due to the noisy nature of data and the error-prone segmentation of line image to character images.

Table 4.3. Test results for downstream classification experiments using a subset of *EEBO-79* data

Experiments	5 seqs	10 seqs	25 seqs	50 seqs	100 seqs	250 seqs
LEC-scratch	28.9	34.0	47.7	63.5	70.5	77.1
ImplicitMasked	25.2	27.5	36.1	43.1	47.6	51.3
CEC-PretrainCE(Hybrid)	**68.9**	**71.5**	74.1	75.6	76.5	77.6
CEC-scratch	36.0	38.3	44.6	50.5	51.9	55.5
CEC-PretrainCE(AE)	46.6	53.8	60.2	65.6	65.9	67.7
LEC-PretrainLE(Hybrid)	36.0	35.3	52.6	67.4	71.8	76.0
LEC-PretrainLE(Hybrid)-Concat	68.2	70.6	**75.3**	**76.9**	**77.9**	**79.6**

4.2 Ablation

4.2.1 Effectiveness of Implicit Masking

To evaluate the effectiveness of the implicit masking in the downstream classification task, we ran the LEC-PretrainLE(Hybrid) experiment with implicit masking for the *LSTM Block*. This ensured that for predicting the character-class for C_i, the model has to just use the contextual information (i.e. $C_{<i}$ and $C_{>i}$) without looking at C_i. The performance of this model (*ImplicitMasked*, shown in dashed-orange in Figure 4.2), quantifies the effectiveness of implicitly masking objective. This shows that contextual information can be confidently useful to predict masked/painted-out char images. The model manages to achieve 51.3% accuracy when trained on 250 line images.

4.2.2 Experiments with PixelCNN Decoder

We tested the effect of different forms of conditioning for the PixelCNN decoder of the *Linguistic-Context* model. We tried out three settings: conditioning on a vector (similar to TransposeConv Decoder), conditioning on a vector along with the information about the location of the top-left pixel in the char-image, and conditioning on a 2D matrix.

1. *Condition on vector* - When the decoder was conditioned on a vector, the model performance was hardly better than the TransposeConv model, while the auto-regressive generation was time consuming. The problem was that though the PixelCNN did predict the masked character correctly, it was not aware where to position the reconstructed character image in the 32 x 32 pixel grid. This resulted in the reconstructed char images to be truncated due to bad positioning.

2. *Condition on vector along with a the top-left pixel information* - The location was that of the top-left pixel encoded as a pair of 32 dimensional 1-hot vectors specifying the x and y coordinates. The resultant model outperformed the TransposeConv model.We also experimented with the location vector as a learned vector, but it was not better than hardcoding.

3. *Condition on a 2D matrix* - The 2D matrix is produced by a Transpose Convolution block, as in the Hybrid model. This spatial conditioning provided rich information about position and results in the best performing model.

Table 4.4 shows the results of the three settings.

Table 4.4. Results of different conditionings for the PixelCNN decoder

PixelCNN conditioning	**Reconstruction accuracy**
Vector	21.0
Vector + location	34.0
2D matrix	**35.7**

This whole chapter is co-authored with Berg-Kirkpatrick, Taylor and is currently being prepared for submission for publication of the material. PV Reddy, Kishore; Berg-Kirkpatrick, Taylor. The thesis author was the primary investigator and author of this material.

Chapter 5

Conclusion

In this work we start with the hypothesize that for a supervised character image classifier in an OCR system, it is more effective to classify pre-learned OCR representations of character images rather than learning to do OCR from scratch. Our results show that is indeed the case. The downstream classifier is able to classify the learned OCR representations with higher accuracy than the traditional approach in settings where the train data is limited. We observe that naive pretraining, like an autoencoder, is better than no pretraining and our MLM-based pretraining is even better than the naive pretraining.

One drawback of our current approach is that it require the line image to be segmented into a sequence of character images (i.e. C from L). While it is a relatively easier task than OCR, it can be prone to errors in cases when the text is scanned from old books and other noisy sources. Our future work will address this problem by extending this approach to work with whole line images, where the segmentations are of some fixed-width, not necessarily at the character boundaries.

Given that there is very little prior work that tries to leverage unsupervised pretraining for improving OCR, our good results in the regime of limited train data makes us believe that this is a useful step towards building efficient OCR systems that can thrive with minimal supervision.

This whole chapter is co-authored with Berg-Kirkpatrick, Taylor and is currently being prepared for submission for publication of the material. PV Reddy, Kishore; Berg-Kirkpatrick, Taylor. The thesis author was the primary investigator and author of this material.

Bibliography

[1] Konstantin Baierer, Rui Dong, and Clemens Neudecker. Okralact - a multi-engine open source ocr training system. In *Proceedings of the 5th International Workshop on Historical Document Imaging and Processing*, HIP '19, page 25–30, New York, NY, USA, 2019. Association for Computing Machinery.

[2] Taylor Berg-Kirkpatrick, Greg Durrett, and Dan Klein. Unsupervised transcription of historical documents. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 207–217, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[3] Taylor Berg-Kirkpatrick and Dan Klein. Improved typesetting models for historical OCR. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 118–123, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[4] T. Breuel. The ocropus open source ocr system. In *Electronic Imaging*, 2008.

[5] S. S. Bukhari, A. Kadi, M. A. Jouneh, F. M. Mir, and A. Dengel. anyocr: An open-source ocr system for historical archives. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 305–310, 2017.

[6] M. Caron, P. Bojanowski, J. Mairal, and Armand Joulin. Leveraging large-scale uncurated data for unsupervised pre-training of visual features. *ArXiv*, abs/1905.01278, 2019.

[7] Alex Clark and Contributors. Python imaging library (fork). 2010.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805, 2019.

[9] Alex Graves, Santiago Fernández, and Faustino Gomez. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *In Proceedings of the International Conference on Machine Learning, ICML 2006*, pages 369–376, 2006.

[10] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Learning to read by spelling. *Proceedings of the 11th Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2018.

[11] Richard H. R. Hahnloser, Rahul Sarpeshkar, Misha A. Mahowald, Rodney J. Douglas, and H. Sebastian Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405:947–951, 2000.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[13] Jeremy Howard and Sebastian Ruder. Fine-tuned language models for text classification. *CoRR*, abs/1801.06146, 2018.

[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 448–456. JMLR.org, 2015.

[15] M. Jenckel, S. S. Bukhari, and A. Dengel. Analysis of unsupervised training approaches for lstm-based ocr. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1250–1255, 2019.

[16] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330, June 1993.

[17] minghui.org. *Falun Gong Practitioners Killed in Persecution by the Chinese Communist Party (CCP)*, 2020 (accessed September 2, 2020). https://en.minghui.org/emh/special_column/death_cases/.

[18] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *ArXiv*, abs/1802.05365, 2018.

[19] Di Qi, L. Su, Jia Song, E. Cui, Taroon Bharti, and Arun Sacheti. Imagebert: Cross-modal pre-training with large-scale weak-supervised image-text data. *ArXiv*, abs/2001.07966, 2020.

[20] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[21] Christian Reul, Uwe Springmann, C. Wick, and F. Puppe. Improving ocr accuracy on early printed books by combining pretraining, voting, and active learning. *J. Lang. Technol. Comput. Linguistics*, 33:3–24, 2018.

[22] Maxim Romanov, Matthew Thomas Miller, Sarah Bowen Savant, and Benjamin Kiessling. Important new developments in arabographic optical character recognition (ocr), 2017.

[23] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning Internal Representations by Error Propagation*, page 318–362. MIT Press, Cambridge, MA, USA, 1986.

[24] D. Sahu and C. Jawahar. Unsupervised feature learning for optical character recognition. *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1041–1045, 2015.

[25] R. Smith. An overview of the tesseract ocr engine. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition - Volume 02*, ICDAR '07, page 629–633, USA, 2007. IEEE Computer Society.

[26] Aäron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Koray Kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In *NIPS*, 2016.

[27] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4790–4798. Curran Associates, Inc., 2016.

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

[29] Christoph Wick, Christian Reul, and Frank Puppe. Calamari - a high-performance tensorflow-based deep learning package for optical character recognition. *ArXiv*, abs/1807.02004, 2018.