The Academic College of Tel Aviv-Yaffo

THE SCHOOL OF COMPUTER SCIENCE

Improved Handwritten Text Recognition (HTR) Using Classic Data-Augmentation and Domain-Adaptation Methodologies

July 2023

Thesis submitted in partial fulfilment of the requirements for the M.Sc. degree in the School of Computer Science of the Academic College of Tel Aviv University

By

Ofer Spivak

The research work for the thesis has been carried out under the supervision of

Dr Sarel Cohen

Contents

1 Background

This chapter provides the theoretical background necessary for understanding the research.

First, define machine learning, neural networks and computer vision. We will then define Convolutional Neural Networks (CNN) and Optical Character Recognition (OCR).

1.1 Machine learning

Machine learning focuses on constructing algorithms for enabling predictions from data. A machine learning task aims to identify ("learn") a function f: X $\rightarrow$ Y that maps the input domain X (of data) onto output domain Y (of possible predictions). The function f is selected from a certain function class, which is different for each family of learning algorithms [3].

It is common to classify machine-learning learning methods into four categories. This categorisation depends on the type of inputs and outputs of the learning system. We will review supervised and unsupervised learning in this section. The other types of learning methods are semi-supervised and reinforcement learning.

Supervised learning Supervised learning algorithms' goal is to construct an accurate prediction function f for new input data based on what it has learnt from previous data (Training dataset). To gain this ability, this type of learning requires labelled examples ($(x, y) \in X \times Y$) with the correct answer. The objective of the learning process is to minimise the difference between the prediction function output $f(x)$ and the labelled true output y.

The supervised learning tasks can be divided into two main areas -

Classification Classification tasks goal is to predict an output of a finite set of classes (categories), $y = \{c_1, ..., c_n\}$. The simplest classification scenario is binary, in which there are two classes [3]. For example, predicting whether an email message is a spam or not.

Regression Regression tasks goal is to predict a simple pattern in the data and output a real number, $y = \mathbb{R}$. For example to find a linear function that best predicts a baby's birth weight on the basis of ultrasound measures [14].

Unsupervised learning In unsupervised learning the training data consists of a set of input vectors X without any corresponding target values [4].

The most striking example of unsupervised learning is when it is used to discover groups of similar examples within the data (clustering). The goal is to construct a function f that partitions an unlabeled dataset into $k = |Y|$ clusters, with Y being the set of cluster indices. Data instances assigned to the same

cluster should presumably be more similar to each other than to data instances assigned to any other cluster [3].

Other examples determine the distribution of data within the input space, known as density estimation, or project the data from a high-dimensional space down to two or three dimensions for the purpose of visualization [4].

1.1.1 Artificial Neural Network

Artificial neural networks are machine learning models inspired by the structure of neural networks in the brain.

An artificial neural network can be described as a directed graph whose nodes correspond to neurons and whose edges correspond to links between them. Each neuron receives as input a weighted sum of the outputs of the neurons connected to its incoming edges [14].

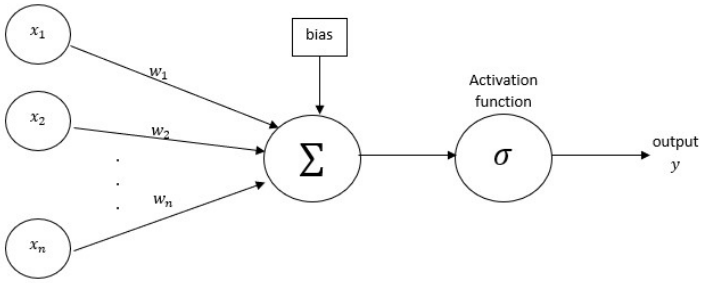


Figure 1: Single neuron schema

The function of a single neuron can be demonstrated as depicted in Fig. 1. First, it receives *inputs* $\{x_1, ..., x_n\}$ from the previous layer that it is connected to. Weights $\{w_1, ..., w_n\}$ are set for each connection of two neurons respectively. Then the weighted sum is calculated and the *bias* is added to the total sum value. (see Eq. 1). The bias plays a role similar to the constant in linear functions, whereby the line is shifted according to the constant's value.

$$y = f\left(b + \sum_{i=1}^{n} x_i w_i\right) \tag{1}$$

After calculating the weighted sum with the bias the neuron applies the *activation funcation* σ on the sum value (see Fig. 1). The activation function goal is to set the output values, for example, for classification, the activation function returns a value between 0 and 1 which we can infer to be how confident the model is that an example instance belongs to the specific predicted class. Or for regression, the activation function can turn the output values into a continuous numerical value. This is achieved by implementing the activation function as Linear, Binary step, Sigmoid or Hyperbolic tangent functions.

The artificial neural network is built by connecting its neurons (see Fig 2). The First layer is called *input layer*, the last layer is called *output layer* and in case there are additional layers, in which data must pass, between the input and output layers they are called *hidden layers*.

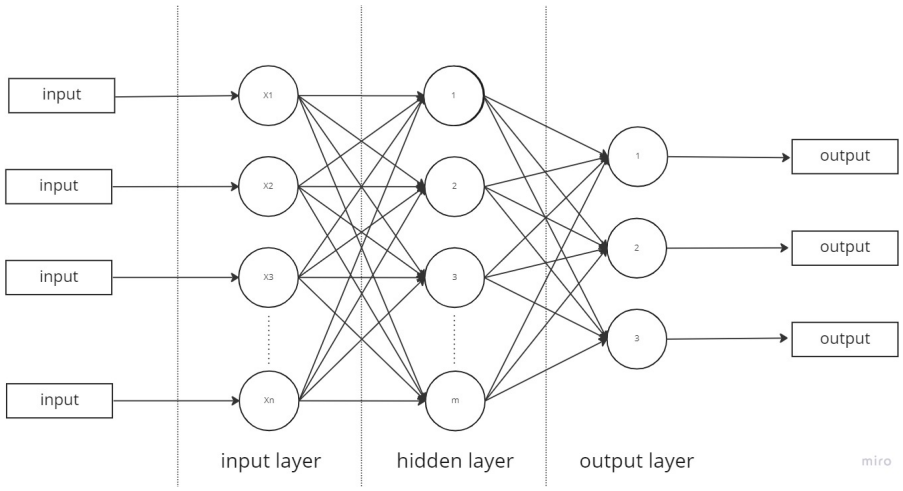


Figure 2: Artificial neural network structure

Training an artificial neural network To enable effective training we must measure the performance of the artificial neural network. This is done by calculating the *loss function* (also called a *cost function*). The basic loss function is to compute the difference between the output of each training example as it passes through the network and the actual expected value.

Training the artificial neural network is adapting the network to better handle its task by going over the input samples. In each sample pass, the adaption is done by adjusting the network weights to improve the overall result's accuracy (meaning reducing the loss function value). It's common to initialize the network weights randomly.

Deciding which weight and how much adjustment is required for it is determined by the *back-propagation* process. The process involves going back through the artificial neural network and examining for each connection how the output will alter based on the weight change. One common and simplest back-propagation method is the optimisation algorithm *stochastic gradient descent*. Gradient descent aims to find a function local minimum by taking repeated steps in the opposite direction of the function gradient.

The step size taken during back-propagation is determined by the *learning rate* parameter. This parameter determines how the network weights will adjust, by smaller or larger steps. In essence, the learning rate, as its name suggests, determines the "speed" the artificial neural network learns.

Deep neural network Deep neural networks are artificial neural networks with multiple, two or more, hidden layers.

Deep neural networks can represent functions of increasing complexity [8] and give the ability to approximate them much more efficiently (with fewer neurons) than shallower ones [12].

Each layer in the network recognises a distinct set of features based on the previous layer's output. As we advance further through the network, the features are aggregated and recombined thus each next layer can detect more complex features. Since simple and more complex features can be detected, deep neural networks can handle more complex tasks and large high-dimensional datasets.

Even though deep neural networks have existed in the past, several problems prevented the effective training of the networks [8]. The main problem was the lack of resources available to train deep neural networks. In the past decades, the availability of more powerful computers, especially in the field of graphics processing units (GPUs), and larger datasets resulted in a revival and flourishing of deep neuronal networks.

1.2 Computer vision

The objectives of computer vision are to gain a high-level understanding of the content and extract meaningful information from digital images or videos. Computer vision applications include image classification, object detection, image retrieval, augmented reality, and traffic automation [17]. It is worth mentioning that machine learning is an important pillar in advancing computer vision performance.

Object detection Object detection is one of the classic problems of computer vision. Object detection includes localization and classification of image regions and usually uses machine learning (or deep learning) to produce meaningful results.

Object localization refers to identifying the location of one or more objects on an image, as well as their boundaries. Object classification refers to assigning a label to the objects in an image. Object detection combines these two tasks - draws a bounding box around each object of interest and assigns it a class label.

1.2.1 Convolutional neural network (CNN)

The advancements in Computer Vision with Deep Learning have been perfected over time, primarily through one particular algorithm — a Convolutional Neural Network [13].

Even a small size image contains a large amount of information thus the number of parameters in the network quickly becomes extremely large as the image size increases. The role of the convolutional network is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction [13]. The price paid for this reduction in parameters is that our features are now translation invariant [18].

The basic idea of the convolutional neural network was inspired by a concept in biology called the *receptive field* [7]. Receptive fields act as detectors that are sensitive to certain types of stimulus in animals' visual cortex, for example, edges. A collection of such fields overlap to cover the entire visual area. The convolution operation is used in computer vision to mimic this ability. In image processing, images can be filtered using convolution to produce different visible effects. Fig. 3 shows how a convolutional filter detects edges from an image, functioning similarly to a receptive field.

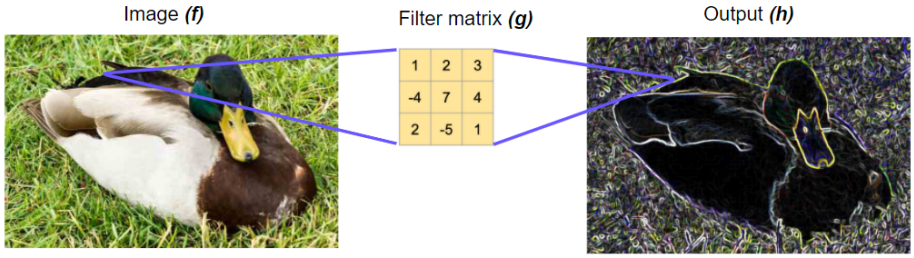


Figure 3: How convolution filtering is used to detect edges from an image.

The element involved in carrying out the convolution operation in the first part of a convolutional layer is called the Kernel/Filter [13]. The shape of the convolution window is given by the height and width of the kernel [18].

In the two-dimensional convolutional operation, we begin with the convolution window positioned at the upper-left corner of the input and slide it across it, both from left to right and top to bottom. When the convolution window slides to a certain position, the input sub-matrix contained in that window and the kernel are multiplied element-wise and summed up (see Eq. 2). The result is the value of the output at the corresponding location [18]. In neural networks, the output matrix is also called a *feature map* [8].

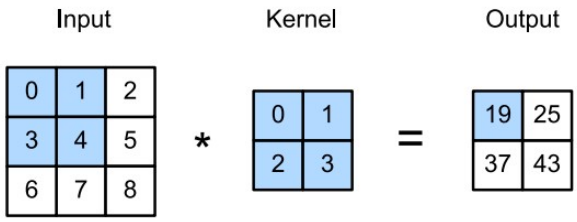


Figure 4: Two-dimensional convolutional operation [18]

To be exact, in mathematics, the discrete convolution between two two-

dimensional functions is defined as:

$$h[i,j] = (f * g)(i,j) = \sum_a \sum_b f(a,b)g(i-a, j-b) \tag{2}$$

where:

h: Output of the convolution operation

f: Input (image matrix)

g: Kernel/filter

i, j: row, column indexes of the result matrix

a, b: both positive and negative offsets from i and j, covering the entire image

For the values in the example Fig. 4 the output computations are:

$0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19$

$1 \times 0 + 2 \times 1 + 4 \times 2 + 5 \times 3 = 25$

$3 \times 0 + 4 \times 1 + 6 \times 2 + 7 \times 3 = 37$

$4 \times 0 + 5 \times 1 + 7 \times 2 + 8 \times 3 = 43$

Successive convolutional layers form a convolutional neural network. In machine learning training the convolutional neural network is done by treating the kernel matrix as the parameters of the neurons and replacing the multiplication operator with the convolution operation. Likewise, the back-propagation is also applicable to convolutional networks (see section 1.1.1).

1.2.2 Generative adversarial networks (GAN)

Generative adversarial networks (GANs) are machine learning frameworks introduced in a paper by Ian Goodfellow and his colleagues [9]. The GANs frameworks are used to generate new synthetic instances of data that holds the same statistics as the training set and are used broadly in image generation.

GANs use two neural networks, one, the generative model, which generates new synthetic data instances while the other, the discriminative model, evaluates whether the data is statistically similar to the actual training dataset, or in other words, whether they are real or fake.

In essence, the generative model attempt to predict the features given a label, in contrast, the discriminative model attempt to predict a label to which the data belongs.

The two models are trained together - the generative model generates instances that are provided to the discriminator along with actual cases from the training dataset. The discriminator classifies them as real or fake. The discriminator is then updated to improve the distinction, and importantly, the generator is updated based on how well the generated samples passed the discriminator.

1.3 Optical character recognition (OCR)

Optical character recognition (OCR) is a field of research in computer vision, pattern recognition and machine learning. This technology enables the identification and extraction of text or numerical data, whether typed, handwritten, or printed, from a digital image.

OCR applications have evolved into multiple domain-specific OCRs: License plate recognition, Automatic Passport recognition, Traffic sign recognition, Defeating CAPTCHAs, Textual version of scanned or printed documents and converting handwritten documents into machine-encoded text.

There are indeed good solutions for certain OCR tasks that do not require deep learning. However, to step forward towards better, more general solutions, deep learning will be mandatory [16].

In recent years most OCRs have been built by concatenating several neural networks (two or more), in which one network feeds the other. For example, the STN-OCR integrates and jointly learns with a spatial transformer network that detects text regions in an image and a text recognition network that takes the identified text regions and recognizes their textual content [2].

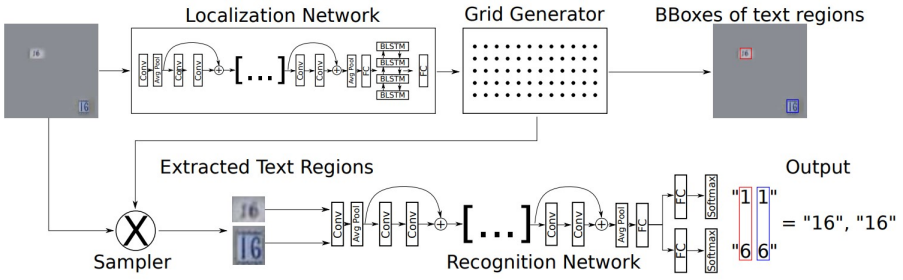


Figure 5: STN-OCR Structure

1.4 Data augmentation

Data augmentations are a set of techniques aimed at increasing the amount of data by adding realistic synthetic data based on existing real data. It became standard practice for training machine learning models for computer vision applications. One possible way to enable it is to use these newly generated images by adding them to the original training set, essentially augmenting the set in a bootstrap manner [15].

Classic and advanced augmentation methods can create new synthetic data. Classic augmentation refers to rotation, transformation, stretching, noise injection, colour modification, cropping, etc. to enlarge the visual variability of the data [15]. Advanced augmentation refers to using machine learning tools to transform an image from one domain to another, mainly by using Generative adversarial networks (GANs).

1.5 Terms

Training dataset The training dataset is used to train the machine learning model. The model goes over this dataset and learns from it (i.e. adjust the weights in an artificial neural network).

Test dataset The test dataset is used to evaluate the model. It is used only after the model is trained. Its content is chosen so it contains data belonging to various classes that the model has to deal with.

Loss function The loss function computes the difference between the output of each training example as it passes through the network and the actual expected value. Thus measures how well the neural network models the training data. Our goal is to reduce the loss function value between training iterations.

Receptive field In animals' visual cortex the receptive field act as a detector that is sensitive to certain types of stimulus. In computer vision, the convolution operation is used to mimic this ability. For example, a convolutional filter detects image edges functioning similarly to a receptive field (see section 1.2.1).

Feature Map The feature map is the output matrix that is generated by applying the convolutional operation on an input image (see section 1.2.1).

2 Introduction

One of the greatest achievements of humankind is the ability to write and thus document and share knowledge between people. The invention of writing is a significant step in the transition from pre-history to history, and indeed most of the evidence for historical events can be found in surviving manuscripts.

Technological progress and the extensive use of technological means led to the documentation of events using print. From the days of the invention of the printing press to the present day when the information is digital and most likely the text created will not meet a page, the printed text is dominant.

And despite all this, handwritten text, even today, constitutes a significant part of documenting and sharing text. In addition, it should be noted that there are many texts and books that were written by hand in the past that we would like to process.

While OCR systems performance has improved significantly in the deep learning era [15] and in recent years are capable to handle printed text, the handwritten text recognition doesn't seem to be up to par. We attribute this gap to the lack of versatile, annotated handwritten text datasets and the difficulty and cost of obtaining it [15].

One possible approach to reduce the difficulty in collecting and labelling is by using automation to create additional synthetic training data, whether it be using generative adversarial networks (GANs) or classic augmentation methods (see 1.4). In most cases, the approach uses deep learning methods for generating synthetic training data. But we need to take into account that deep learning methods are limited by their tendency to error even when introducing small (in some cases even invisible) modifications to the input.

This thesis aims to check if OCRs' handwritten text recognition performance can be improved via domain adaptation and classic augmentation. Our approach is to use classic augmentation methods to create synthetic training datasets based on one of our obtained databases' styling and lexicon.

Firstly, we obtained the IAM and CVL handwriting databases and augmented based on the CVL database. The new augmented, statistically similar, database is marked as $\overline{CVL}$.

Afterwards, we trained the TPS-ResNet-BiLSTM-Attn OCR (see 3.2) on the IAM Handwriting database. Then we fine-tuned our trained OCR with the newly created synthetic augmented database ($\overline{CVL}$).

Now we have 2 trained OCRs, the IAM Handwriting database trained TPS-ResNet-BiLSTM-Attn OCR and the ($\overline{CVL}$) fine-tuned TPS-ResNet-BiLSTM-Attn OCR.

We verified our approach by running the ($\overline{CVL}$) fine-tuned OCR on the original CVL database test set. We also compared it with the result of the IAM-trained OCR running against the original CVL database test set. Note, this is without exposing the OCRs to the real CVL database training set.
(see sections 3 and 4)

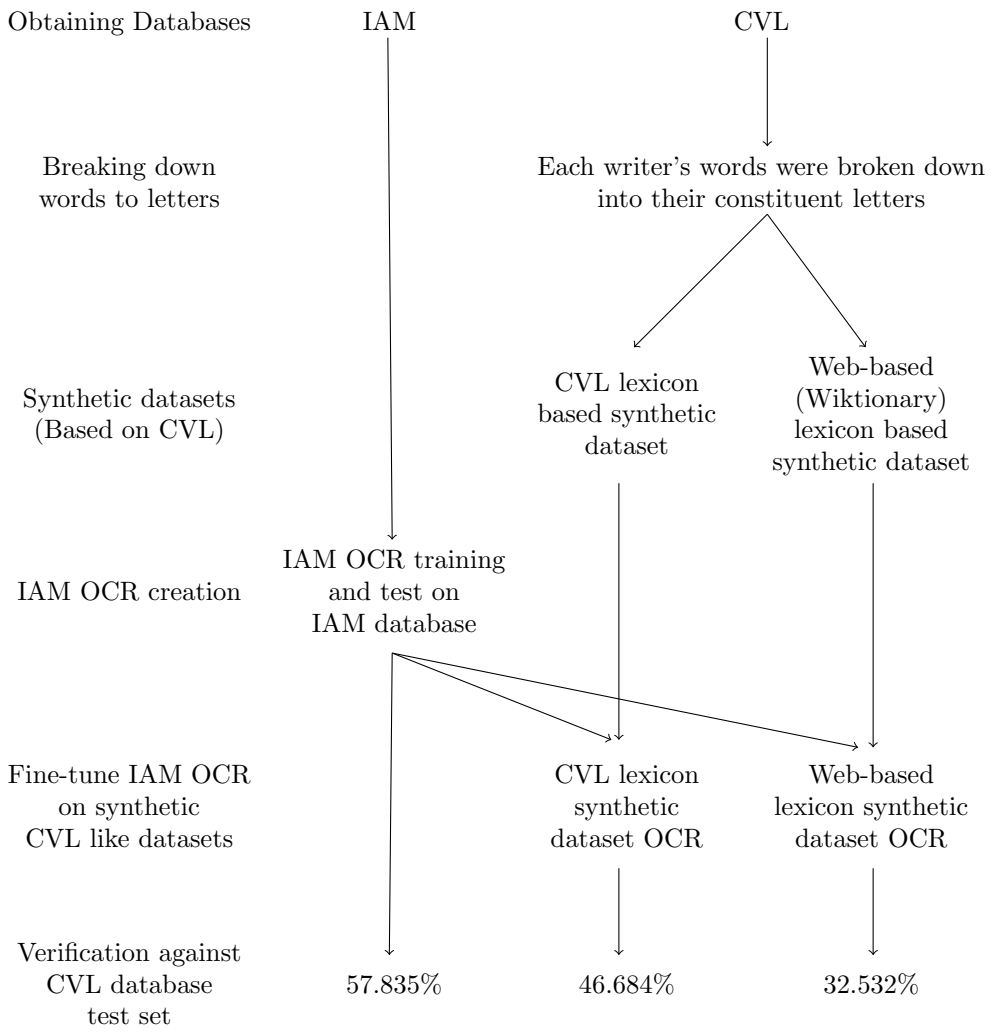


Table 1: Scheme describing the stages of the study

3 Method

This chapter provides the different tools used and stages of the research.

First, a review of the obtained handwriting databases, the PS-ResNet-BiLSTM-Attn OCR, and then, the stages of the data augmentation and domain adaptation.

3.1 Datasets

This study requires labelled handwritten word text image examples, therefore we obtained the following, well researched, datasets:

> IAM Handwriting Database - The IAM handwriting database contains forms of handwritten English text which can be used to train and test handwritten text recognizers and to perform writer identification and verification experiments [10]. It contains around 100K images of words from 657 writers.
>
> CVL - The CVL Database is a public database for writer retrieval, writer identification and word spotting. The database consists of 7 different handwritten texts (1 German and 6 English Texts). [5][6]

One of our goals is to be able to compare the results of this research and the *ScrabbleGAN: Semi-Supervised Varying Length Handwritten Text Generation* [15] paper. Hence, we used the same datasets' partition as described in it.

The IAM database partition into directories, training and test sets, was as described in the *Laia: A deep learning toolkit for HTR(handwritten text recognition)* paper [11][1].

The CVL database is already divided into a training set and a test set and is used as is.

OCRs use *Lightning Memory-Mapped Database (LMDB)* file format as their input. Therefore, To enable the use of the IAM and CVL datasets in OCRs the datasets were converted to the LMDB format. The conversion was done as described in the *ScrabbleGAN: Semi-Supervised Varying Length Handwritten Text Generation* [15] GitHub project[2]

3.2 TPS-ResNet-BiLSTM-Attn OCR

The TPS-ResNet-BiLSTM-Attn OCR was selected for its high prediction rate [1].

As its name suggests, the TPS-ResNet-BiLSTM-Attn OCR consists of 4 steps (4 concatenated artificial neural networks) -

[1] Direct URL for the IAM Laia paper partition -`https://github.com/jpuigcerver/Laia/tree/master/egs/iam/data/part/lines/original`

[2] "ScrabbleGAN: GitHub project - https://github.com/amzn/convolutional-handwriting-gan

Transformation (Trans.), Feature extraction (Feat.), Sequence modelling (Seq.), Prediction (Pred.).

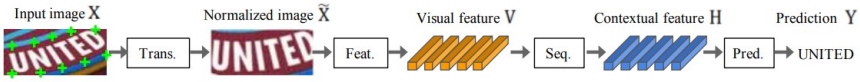


Figure 6: Visualisation of TPS-ResNet-BiLSTM-Attn flow [1]

This OCR model is derived from commonalities among independently proposed scene text recognition models [1]. Due to its task resemblance to computer vision tasks such as object detection and sequence prediction, it has benefited from convolutional neural networks (CNNs) and recurrent neural networks (RNNs).

Transformation This step is responsible for transforming and normalizing the input text image.

The real-world input image comes in various shapes and angles and the goal of the transformation step is to normalize it into a predefined rectangle. In case the input images are not normalized the successive feature extraction step will have to lift the burden of learning such geometries.

The transformation is done using *Thin-Plate Spline (TPS)*. The TPS finds several points at the upper and lower wrapping points (the green '+' signs in Fig. 6) and normalizes the text image's area to a predefined rectangle. The TPS is made up of several processes:

- Localisation network - finding a text boundary by calculating x, y coordinates of the wrapping points on the input image.
- Grid generator - linking the location of the pixels in the boundary to those of the normalized image by providing a mapping function from the localisation network identified region to the normalized image.
- Image sampler - generating a normalized image by using the input image pixels values and the grid generator linking information.

Feature extraction In this step, a convolutional neural network (CNN) is used to extract visual features map $V = \{v_i\}$[3] from the input image (received from the transformation step).

Each column in the feature extraction output features map is used to predict the character in it. These features map column is related to a distinguishable receptive field along the horizontal line of the input image.

ResNet was previously used as a feature extractor for OCR. It is a CNN with residual connections that eases the training of relatively deeper CNNs [1]. The output of the ResNet is 512 channels X 26 columns.

[3] i is the feature map columns number

Sequence modeling Each column in the feature extraction step feature map $v_i \in V$ is used as a frame of the sequence ($H = Seq.(V)$) [1].

Due to previous works in which the sequence suffered a lack of contextual information, the *Bidirectional LSTM (BiLSTM)* is used to add contextual information to the sequence.

Prediction In this stage, the *attention-based sequence prediction (Attn)* is used to predict the character sequence ($Y = y_1, y_2, ..., y_n$) from the Sequence modelling step output, H.

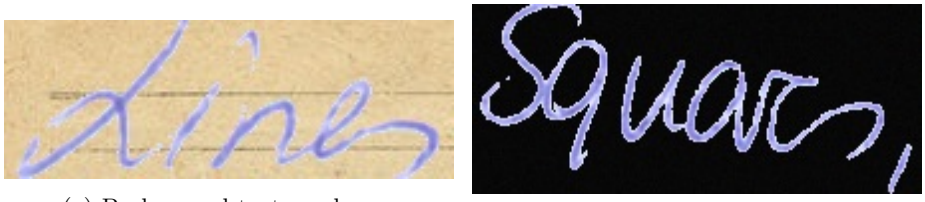
The Attn network automatically captures the information flow within the input sequence to predict the output sequence. It enables the model to learn a character-level language model representing output class dependencies [1].

3.3 Basic augmentations

Initially, we tried to use basic classic augmentation on the whole word image to create a synthetic training dataset (see 1.4).

The basic classic augmentation included, among other methods, the use of changing the background colour, changing the background texture, projections and rotation at different angles.

The augmentations were done on word images taken from the CVL database training dataset.



(a) Background texture change

(b) Background color change

Figure 7: Classic basic augmentation on CVL word images

These basic augmentations created a small variety and haven't enlarged significantly our synthetic training set permutations. Therefore, we decided not to proceed with this approach.

3.4 Letter-based augmentation

Our next approach was to create a letter-based synthetic dataset.

The CVL database training dataset was divided according to its writers, then, each writer's words-images were broken down into their constituent letters.

Thus we had for each letter, approximately, five types: a capital letter, a single letter, a letter at the beginning of a word, at the end of a word and in the middle of a word. Hence we had around 130 letters permutations per writer.

In some cases, we had more than one letter per type per writer. In other cases, not all letters had all five types.

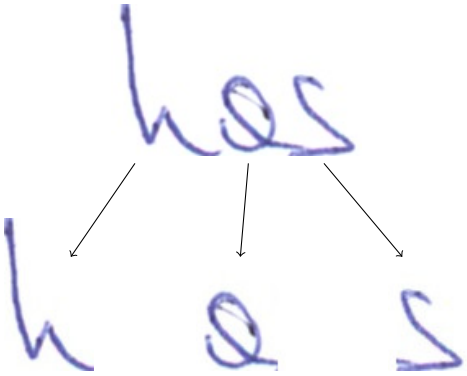


Figure 8: Breakdown of the word *has*

The dataset creation was done by going over a lexicon's words and concatenating a single writer's letters, consequently creating new synthetic word images for each of the CVL database training set writers.

Each newly created word was composed, only, of letters of the same writer due to significant differences in the letters' form between different writers.

Figure 9: The letter 'a' of writer 1 (left side) versus writer 3 (right side)

At this point, it was decided to use two lexicons for generating two synthetic datasets. The first one was by using the CVL database training set lexicon, which was revealed to be quite small. The second lexicon was a web-based words lexicon from Wiktionary (see appendix 8.1). The objective was to have a large set of words to process.

We created a different synthetic dataset for each lexicon, CVL and web-based. As mentioned in 3.1, to be able to process the newly created datasets by OCRs both datasets had been converted into the LMDB file format.

3.5 Domain adaptation

Domain adaptation is used to improve the performance of a machine learning model on a target domain by using a well-trained model on a related source domain (which has the same feature space) without re-training it on the target model. This technique saves computational resources and allows us to obtain trained models on different domains, even those that lack significant data or are difficult to train on.

in our case, the model is the TPS-ResNet-BiLSTM-Attn OCR, the source domain is the IAM database training dataset and the target model is the CVL database dataset.

IAM TPS-ResNet-BiLSTM-Attn OCR As we aimed to implement the domain adaptation between IAM and the CVL handwriting databases, firstly we need to obtain an IAM Handwriting database trained OCR.

To generate an *IAM Handwriting database trained OCR*, we obtained the pre-trained *TPS-ResNet-BiLSTM-Attn OCR* (see section 3.2) and based our training code according to the *What Is Wrong With Scene Text Recognition Model Comparisons? Dataset and Model Analysis* [1] GitHub project[4].

Afterwards, we fine-tuned, by running the OCR's train function iterations, on the obtained *TPS-ResNet-BiLSTM-Attn OCR* on the IAM database training set. The Accuracy was validated against the IAM database test set. We were able to reach an accuracy of 78.827% before we started to get overfitting.

At this stage, we have the ***IAM Handwriting database trained TPS-ResNet-BiLSTM-Attn OCR*** [5].

Fine-tuning using the classic by-letter augmentations We can now continue the domain adaptation process and fine-tune the *IAM Handwriting database trained OCR* using the letter-based augmentation datasets we created (see section 3.4).

For tracking ability, we perform the fine-tuning in batches, For every 500 training iterations, we performed the check against the CVL database test set. This was done on both CVL and web-based lexicon synthetic datasets.

The fine-tuned CVL and web-based lexicon synthetic datasets OCRs and as well as the *IAM Handwriting database trained OCR* results are discussed in section 4.

[4]"What Is Wrong With Scene Text Recognition Model Comparisons: GitHub project - https://github.com/clovaai/deep-text-recognition-benchmark

[5] *IAM Handwriting database trained OCR* for brevity

4 Results

As the first step and to establish a baseline, we ran the *IAM Handwriting database trained OCR* against the CVL database test set. Next, as mentioned in section 3.5, we ran the fine-tuned CVL and web-based lexicon synthetic datasets OCRs against the CVL database test set.

The *IAM Handwriting database trained OCR* resulting accuracy was 57.853%. In comparison, the *web-based lexicon synthetic dataset OCR*'s best-resulting accuracy was 33.225%. Significantly low compared to the established baseline. As can be seen in table 2, no improvements were made from one iteration batch to another. likewise, the *CVL lexicon synthetic dataset OCR*'s best-resulting accuracy was 46.684%. Significantly low compared to the baseline as well.

As can be seen in table 2 below, we stopped the *CVL lexicon synthetic dataset OCR* runs after 3 batch iterations. This is because no significant improvement was shown between the batch iterations, and we have already established in the *web-based lexicon synthetic dataset OCR* runs that there is no expectation for a leap in accuracy with additional iterations.

Furthermore, it can also be noticed that the accuracy results of the *CVL lexicon synthetic datasets OCR* are significantly higher than the web-based lexicon synthetic datasets OCR. This result might be explained by the fact that the *CVL lexicon synthetic datasets OCR* was fine-tuned using closer to or similar to the CVL database datasets samples.

OCR	after 500 iteration	after 1000 iteration	after 1500 iteration	after 2000 iteration	after 2500 iteration
IAM database trained OCR	57.835% [6]				
web-based lexicon synthetic datasets OCR	33.223%	32.060%	31.211%	32.532%	31.972%
CVL lexicon synthetic datasets OCR	43.661%	46.675%	46.684%	-	

Table 2: OCRs' performance on the CVL database test set per 500 iterations

[6] The IAM database trained OCR validated against the CVL database test set once it was trained on the IAM Handwriting database training set.

5 Future Work

We like to recommend several future steps and experiences that can be performed in order to expand the understanding in light of the above results.

Firstly, to fine-tune the *TPS-ResNet-BiLSTM-Attn OCR* on the CVL database training set and validate it against the IAM database test set.

This should give us an additional cross-domain benchmark. Assisting to understand whether the above results are correct and honest or are due to the route we chose - to produce synthetic data precisely from CVL and not from the IAM.

Additionally, to fine-tune the *IAM Handwriting database trained OCR* on the CVL database original training set and validate it against the CVL database test set. In the next step, compare this newly *CVL fine-tuned OCR* test set run result with the *IAM Handwriting database trained OCR* test set run result.

In case the *CVL fine-tuned OCR* doesn't show significantly better results, there is no point in continuing the experiments with the synthetic data. This indicates that there is a problem with the experimental infrastructure

Another suggested step is to check the relative contribution of synthetic data. This can be achieved by training the *TPS-ResNet-BiLSTM-Attn OCR* on the CVL database training set without involving the IAM database at all. (same as we did to generate the *IAM Handwriting database trained OCR* - see 3.5) Afterwards, to fine-tune the above newly created *CVL trained OCR* on the synthetic CVL datasets.

Now we can compare the results of the *CVL trained OCR* and the *synthetic CVL datasets OCR*s validations against the CVL database test set, and to see the impact of the synthetic datasets on the OCR's capabilities.

Also, to enhance the check for the contribution of synthetic data we suggest creating new training datasets based on the IAM training set. One consists of 40% of the words in the set, the second consists of 20% of words in the set, and another one consists of synthetic information created as explained next.

The synthetic dataset will be based on the remaining 80% of the set (in accordance with the second created dataset described above). Similar to what was done for the latter-based-augmentation (see section 3.4), we will break and reassemble words according to the IAM lexicon. From this constructed dataset, we will take a random amount equal to 20% of the words in the set and add it to the second dataset described above (the 20% dataset based on the original IAM).

Next, we will train a *TPS-ResNet-BiLSTM-Attn OCR* on the first constructed dataset (the 40% dataset based on the original IAM) and another *TPS-ResNet-BiLSTM-Attn OCR* on the synthetic dataset described above. Test both OCRs against the IAM test dataset and compare their results. This should give us an additional indication of the impact of the synthetic datasets on the OCRs' capabilities.

6 Conclusions

In this study, we have presented a new approach to the generation of handwritten text images using a classic augmentation method. Each word is constructed by concatenating characters' images to generate a new word image.

In essence, from the results (see section 4), we can conclude that using letter-based augmentation (see section 3.4) isn't the right approach to improve OCRs' performance.

However, as can be seen in the *ScrabbleGAN: Semi-Supervised Varying Length Handwritten Text Generation* ([15]), data augmentation indeed helps in enlarging the labelled training sets and should be continued used as a valid tool for improving machine learning models.

The above results can be reconciled by the fact that, certainly, the same letter, by the same writer, can be written differently in different words (due to contiguity with other letters, for example). Thus concatenating a writer's letters into a new word might not result in a word that is statistically similar to the actual training dataset words.

In addition, the concatenation might add noise at the pixel level, as each letter might be taken from a different word image, combined with the tendency of deep learning methods to error when introducing, even remote, modifications to the input resulting in the above results.

7 References

[1] Jeonghun Baek et al. "What is wrong with scene text recognition model comparisons? dataset and model analysis". In: *CoRR* abs/1904.01906 (2019). arXiv: 1904.01906. URL: http://arxiv.org/abs/1904.01906.

[2] Christian Bartz, Haojin Yang, and Christoph Meinel. *STN-OCR: A single Neural Network for Text Detection and Text Recognition*. 2017. arXiv: 1707.08831 [cs.CV].

[3] Ron Bekkerman, Mikhail Bilenko, and John Langford. *Scaling up machine learning: Parallel and distributed approaches*. Cambridge University Press, 2011.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning: All "just the Facts 101" Material*. Springer (India) Private Limited, 2016.

[5] *CVL-DATABASE*. https://cvl.tuwien.ac.at/research/cvl-databases/an-off-line-database-for-writer-retrieval-writer-identification-and-word-spotting/.

[6] Florian Kleber, Stefan Fiel, Markus Diem, Robert Sablatnig. "CVL-Database: An Off-line Database for Writer Retrieval, Writer Identification and Word Spotting, In Proc. of the 12th Int. Conference on Document Analysis and Recognition (ICDAR) 2013, pp. 560-564". In: (2013).

[7] Kunihiko Fukushima. "Neocognitron: A hierarchical neural network capable of visual pattern recognition". In: *Neural networks* 1.2 (1988), pp. 119–130.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[9] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[10] *IAM Handwriting Database*. https://fki.tic.heia-fr.ch/databases/iam-handwriting-database.

[11] Joan Puigcerver, Daniel Martin-Albo, and Mauricio Villegas. *Laia: A deep learning toolkit for HTR*. https://github.com/jpuigcerver/Laia. GitHub repository. 2016.

[12] David Rolnick and Max Tegmark. "The power of deeper networks for expressing natural functions". In: *CoRR* abs/1705.05502 (2017). arXiv: 1705.05502. URL: http://arxiv.org/abs/1705.05502.

[13] Sumit Saha. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[14] Shai Ben-David Shai Shalev-Shwartz. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[15] Sharon Fogel, Hadar Averbuch-Elor, Sarel Cohen, Shai Mazor, Roee Litman. "ScrabbleGAN: Semi-Supervised Varying Length Handwritten Text Generation". In: (2020).

[16] Gidi Shperber. *A gentle introduction to OCR*. `https://towardsdatascience.com/a-gentle-introduction-to-ocr-ee1469a201aa`.

[17] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[18] Aston Zhang et al. "Dive into deep learning". In: *arXiv preprint arXiv:2106.11342* (2021).

8 Appendix

8.1 Wiktionary-words-list

The list of the top 100,000 most frequently-used English words according to Wiktionary. It was compiled in August 2005 and coalesced into a handy list for use in John the Ripper.

Sources:

{http://en.wiktionary.org/wiki/Wiktionary:Frequency_lists#Top_English_words_lists

http://en.wiktionary.org/wiki/Wiktionary:Frequency_lists/PG/2005/08/1-10000

http://en.wiktionary.org/wiki/Wiktionary:Frequency_lists/PG/2005/08/10001-20000

http://en.wiktionary.org/wiki/Wiktionary:Frequency_lists/PG/2005/08/20001-30000

http://en.wiktionary.org/wiki/Wiktionary:Frequency_lists/PG/2005/08/30001-40000

http://en.wiktionary.org/wiki/Wiktionary:Frequency_lists/PG/2005/08/40001-50000

http://en.wiktionary.org/wiki/Wiktionary:Frequency_lists/PG/2005/08/50001-60000

http://en.wiktionary.org/wiki/Wiktionary:Frequency_lists/PG/2005/08/60001-70000

http://en.wiktionary.org/wiki/Wiktionary:Frequency_lists/PG/2005/08/70001-80000

http://en.wiktionary.org/wiki/Wiktionary:Frequency_lists/PG/2005/08/80001-90000

http://en.wiktionary.org/wiki/Wiktionary:Frequency_lists/PG/2005/08/90001-100000