Synthetic handwritten text generation

Adrià Rico Blanes

Abstract– Handwritten text recognition requires a large quantity of labelled samples, which are costly to produce. In this project, we explore the possibility of synthetically generating human-like handwritten text, allowing for an effectively infinite amount of labelled samples. We perform text generation through font rendering and apply image processing techniques to give the text more variability and realism. We also perform an evaluation of our system by generating a fully synthetic clone of a real dataset and comparing the effect of using our samples combined with a portion of the real samples to train a handwritten text recognizer. We conclude that our system allows to obtain an accuracy close to the baseline using only a small portion of real samples.

Keywords– Document analysis, Optical character recognition, Handwritten text recognition, Handwritten text generation, Machine learning, Deep learning, Data augmentation

1 Introduction

CURRENTLY, OCR (Optical Character Recognition) for printed text could be considered a solved problem. There are commercial and open-source programs which can achieve very good results, such as Google's *Tesseract OCR* [23]. However, HTR (Handwritten Text Recognition) is still an open problem, because the variability of the styles and the deformations the characters undergo present some challenging problems.

In the last years, deep learning based HTRs have managed to achieve very good results, but they need an extensive dataset to be trained with [25] [20]. The production of these datasets is costly, because a human must manually label the samples. Some methods have been developed to mitigate this cost by distributing the work amongst a large amount of people, such as *crowdsourcing* [14], but it is still not an ideal solution.

The most cost-effective method would be to artificially generate already labelled samples. This would allow us to have an effectively infinite dataset, without any human labour required. Thus, the idea of developing a synthetic handwritten text generator is very attractive.

The main objective of this project is to develop a program capable of automatically generating synthetic handwritten text, with the purpose of training and improving the performance of HTR systems. This objective can be divided into three sub-objectives:

- Researching the state of the art for synthetic handwritten text generation and HTR.
- Implementing a generator using the most convenient techniques.
- Choosing an existing HTR system and training it with real and synthetic samples, then analyzing the performance of our model.

In this article, the process for achieving these objectives will be detailed, followed by the results obtained and some final conclusions.

2 State of the art

2.1 Handwritten text recognition

A lot of work has been done in the field of HTR, from adapting classic OCR techniques to work with printed-like human handwritten text, to applying modern deep learning techniques to directly recognize cursive script.

In general, there are two main types of HTR: online and offline. Offline HTR consists of recognizing text directly from images, having only pixel information, while online HTR consists of recognizing text written digitally with the support of a tablet or touch-screen, having information about the dynamics of the pen, the order in which each stroke has been written, etc. As online handwriting provides more information about the text, the recognition task is easier. However, its applications are more limited, since the writing has to be recorded digitally, but in many cases the text of interest would have been written on paper.

There are different levels at which HTRs can work, depending on how abstractly they aim to understand the text: character, word, line, paragraph and page.

The advantages and disadvantages of each are a trade-off between the difficulty of recognition and the difficulty of

• E-mail de contacte: adria.rico@e-campus.uab.cat
• Menció realitzada: Computació
• Treball tutoritzat per: Alicia Fornes Bisquerra i Marçal Rossinyol Sanabra
• Curs 2017/18

segmentation. Character level HTRs present a very difficult segmentation problem, since splitting each character individually in cursive script can be hard even for humans, but can classify each symbol using straightforward techniques, such as classic Convolutional Neural Networks [13]. On the other hand, page level recognition needs no segmentation, as it automatically detects the structure of the text, but needs a much more complicated network architecture and training process.

Currently, the methods which produce the best results are based on deep learning with Recurrent Neural Networks (RNNs) and their variants [7] [25]. These techniques will be explained in the next sections.

For a more complete review on HTR methods, see [3].

2.1.1 Recurrent Neural Networks

Recurrent Neural Networks (RNN) are a special type of Neural Network which have proven to be very effective for many kinds of sequence recognition, such as speech and text [8] [9].

The main feature they have, in contrast with classical networks, is the ability to accept an input of variable size and analyse it part by part, taking into account all previously processed parts.

To accomplish this, they use a hidden state, which receives the last output and is used for the next input, forming a feedback loop and allowing for data to persist between iterations.

RNNs can also be used to produce a sequence as an output, which is something that was considered for this project.

2.1.2 Long Short-Term Memory Networks

Long Short-Term Memory Networks (LSTM) are one of the most widely used types of RNN. They were first introduced by Hochreiter and Schmidhuber [22] [17]. The main problem with plain RNNs is that it's difficult for them to relate inputs too far away from each other. LSTMs solve this problem by introducing a new mechanism to update the hidden state. The main idea is to control which information is kept from the previous state and which information is added from the current output through a forget gate and an input gate, respectively.

This allows the network to learn the proper parameters to make the most relevant information persist indefinitely and to remove the information which is no longer needed.

There is a variation of this type of network which was used in this project, called Bidirectional Long Short-Term Memory Network (BLSTM) [5]. The key difference is that they scan the sequence in both directions, rather than only one, thus being able to take advantage of future elements apart from past elements.

2.2 Handwritten text generation

A lot of work has been done in the field of synthetic text generation, but only some is specifically aimed at handwritten text.

Firstly, there are data augmentation and texture synthesis methods, which take real, labelled text documents, then generate new, automatically labelled documents, combining the layouts and text of the supplied samples and applying some distortions to increase the realism and variability [10] [11].

Secondly, there are font-based methods, which are similar to data augmentation methods, but they do not need any real documents. Instead, the desired text is rendered in a specified layout using handwritten typefaces, then some distortions are applied to make the text more human-like and variable [12]. Some work had already been done using this method at the Computer Vision Centre (CVC), where this project has been developed.

Finally, there are generational methods, which aim to synthetically generate the strokes of the handwriting, usually employing RNNs, GANs, or other types of neural networks [4] [2] [26]. Similarly to data augmentation methods, they need real, already labelled samples. They work much better with online data, since it gives more information about how letters and words are formed.

3 Methodology

After examining the state of the art, it was decided that, firstly, a font-based synthesizer would be developed, it would then be evaluated, and depending on its performance, the relevant parts would be improved, be it the generation (explore combining multiple fonts or using networks to generate text) or the augmentation (explore adding more distortions or improving already existing ones).

The language of choice for the development of the synthesizer has been Python, due to the vast amount of libraries it has available and the extensive usage it has thorough many projects in the CVC, which allows benefiting from already developed code. Furthermore, many neural networks have a Python implementation, and thus the generator could be directly piped to them to allow for real-time sample generation while training.

To evaluate the system, a BLSTM network implemented in TensorFlow [1] was used, together with the IAM dataset [15], which is one of the most widely used, public handwritten text datasets, providing about 100,000 words written by more than 600 different authors.

The work has been autonomous, with a meeting about once a week to check the progress made and to talk about any corrections which had to be made on the direction of the project.

In the following sections, the process to create the synthesizer and to train the HTR model will be explained in detail.

4 Synthesizer

The main objective of this project is creating a synthesizer capable of taking plain text as the input and producing human-like handwritten text images as the output, together with its corresponding ground truth at any desired level.

To accomplish this task, we need to consider three separate sub-tasks: rendering the text, applying distortions, and generating the groundtruth. A high level scheme of the generation process is shown in figure 1.

We have focused on generating samples for offline recognition, since the ultimate objective would be to improve the recognition rate of old handwritten documents.

It is worth mentioning that two versions of the synthesizer have been done. Firstly, a generic synthesizer was developed, which generates any number of pages, with random parameters for each page, using text from the supplied corpus. Then, in order to more fairly evaluate our system, a variant version was developed, which generates a full clone of the IAM dataset, with random parameters for each author.

In this section, the generic synthesizer will be explained in detail, then the modifications made to generate the IAM clone will be described.

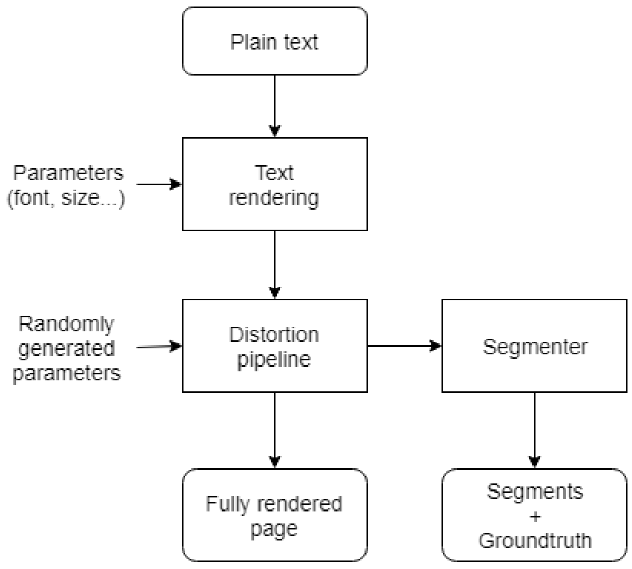


Fig. 1: Pipeline of the generic synthesizer. For each new rendered page, the process is repeated, pulling new plain text from the corpus and generating new random parameters.

4.1 Text rendering

Text rendering has been one of the main issues of this project. We tried to attack the problem using many different technologies, such as the *Python Imaging Library (PIL)*, *OpenCV*, *pyvips*, and *imagemagick*, but all of them had some problems or drawbacks which would make the synthesizer either unreliable or unable to produce some of the desired results.

In the end, we settled with a python binding for Pango [24], which offers high quality text rendering and allows to get precise bounding boxes for each character.

To be able to produce an image with the rendered text, Pango requires an image rendering back-end. For this, we used Cairo [19], as it has a python binding which specifically integrates with Pango.

The development of the rendering process presented mainly two challenges: achieving a consistent font size and guaranteeing that the typeface is being rendered correctly.

The first problem arises due to how font metrics work. Each character of the same font has a constant, set height, in which the character is guaranteed to lie in, then a variable width which varies with each letter. The problem is that some fonts do not fully use all the vertical space available, and produce text which looks much smaller than the text produced by other fonts, even when setting the same font size. To solve this problem, first, a single line is rendered containing every letter and number. Then, the real size of the line is calculated, and the font size is adjusted so that it will generate text with the desired size. There is still a significant amount of width variation between fonts, but the height remains consistent.

The second problem arises because Pango cannot directly load a typeface file, they have to be referred to by an alias which the system recognizes. In Linux, the system used for font location is *fontconfig* [18]. When *fontconfig* cannot locate the specified font, it is replaced by a default font, without giving any feedback. To make sure that all fonts were being rendered correctly, some separate scripts had to be written to acquire a proper alias for each of them and to check that the alias was being properly recognized by the system. The utilities *fc-scan* and *fc-match* were key to accomplish this task.

The final output of this module is a grayscale image with the rendered text, together with the bounding box of each character.

4.2 Distortions

After the text is rendered, some operations are applied to it to produce more variable, human-like handwritten text. These operations can be categorised into two different types: pixel and geometric.

Pixel operations are operations which directly map each pixel of the image one to one and do not modify the bounding boxes of the characters, whilst geometric operations change the geometry of the image, possibly modifying the character bounding boxes. For this reason, all geometric operations are also applied to the groundtruth bounding boxes.

There are three steps of distortion application:

Firstly, all operations related to the ink are performed. These operations aim to distort the text in ways which would be caused by the author's writing and other factors which could modify the ink in the document. They are performed first, since the raw rendered text is effectively the ink written by the author.

Secondly, the distorted ink is blended with a background. This aims to imitate the surface in which the text would have been written, such as a paper.

Lastly, all operations related to the surface are applied. These operations aim to imitate distortions caused by the surface after the text has already been written, and also include possible noise caused by the method of acquiring the image (for example, scanning).

All distortions make use of random parameters to simulate the variability found in real documents. The framework allows to save the seed used for random generation, so that all distortions will happen in exactly the same way when loading a custom seed. Furthermore, all parameters are calculated separately before the operations are applied. In this way, it is possible to disable any given distortion and still generate the same parameters for the rest. This is specially useful for performing tests evaluating the performance improvement each distortion provides.

The full distortion pipeline is shown in figure 2. In the following sections, each operation will be explained in more detail.

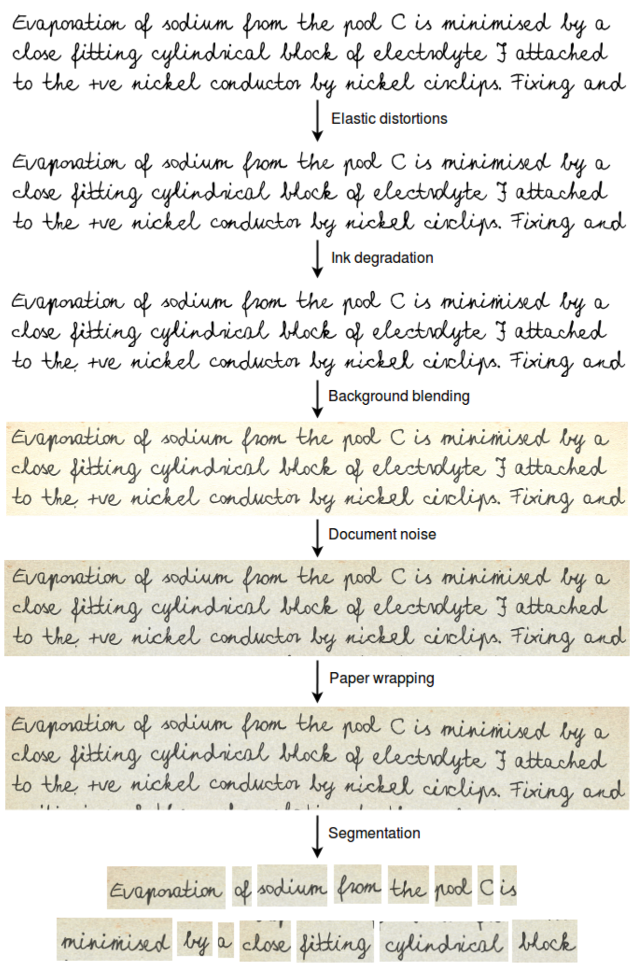


Fig. 2: The distortion pipeline. Starting from the top, each operation described in section 4.2 is applied in order. In the bottom, some example segments from the segmentation step are shown. In this case, the segments were chosen to be made at word level.

4.2.1 Elastic distortion

The first operation applied is an elastic distortion. In real handwriting, it is very rare to find two letters or words written in exactly the same way. This operation aims to imitate this variability.

To accomplish this, a grid is created for each individual word, and random vectors are assigned to each tile. These vectors distort the grid in a non-linear way, dilating and contracting the tiles depending on their magnitude and direction.

The distortion is applied with fairly non-aggressive parameters, because otherwise it would make the text seem very unrealistic, causing deformations not usually found in real handwriting.

4.2.2 Ink degradation

The last operation from the ink block is ink degradation. Often, in real documents, ink has gotten degraded over time, and there are some small regions in which it is missing. There could also be ink stains, caused by either the author or other external factors. This operation aims to imitate both of those scenarios.

To accomplish this, two masks consisting of random blobs are generated. One of the masks creates smaller and rougher blobs, whilst the other creates bigger and smoother blobs. The first mask is used to remove pixels from the ink, imitating degradation over time, and the second one is used to add ink pixels, imitating stains.

4.2.3 Background blending

Once the final ink image is obtained, it is blended with the background. This aims to imitate the surface in which the text has been written.

To accomplish this, a random background image is chosen from a given list. Then, the image is either tiled or resized so that it matches the size of the generated ink image. After that, the ink image is interpreted as a semi-transparent layer and it is multiplicatively blended with the background image.

4.2.4 Document noise

After the ink has been placed on the surface, some noise is added to the image. This aims to imitate possible artifacts when acquiring the image with a camera or scanner, and also adds variability to the background.

To accomplish this, a random grayscale image is generated, a gaussian filter is applied to it, then it is blended with the current generated image.

4.2.5 Spline transformation

Finally, the whole page undergoes a distortion which gives it some curvature. This aims to imitate paper wrapping, found for example when acquiring samples from books.

To accomplish this, a spline transformation is applied. First, several random points are chosen along an horizontal line through the center of the image. It is guaranteed that the points are properly distributed, because otherwise the transformation could yield an overly distorted, unrealistic image. The vertical position of the points is then randomly changed given a maximum delta. Once the point coordinates have been chosen, they are interpolated using the *B-spline* technique, which creates a curve composed of third-degree polynomials using the points provided as control points. The whole image is then transformed following this curve.

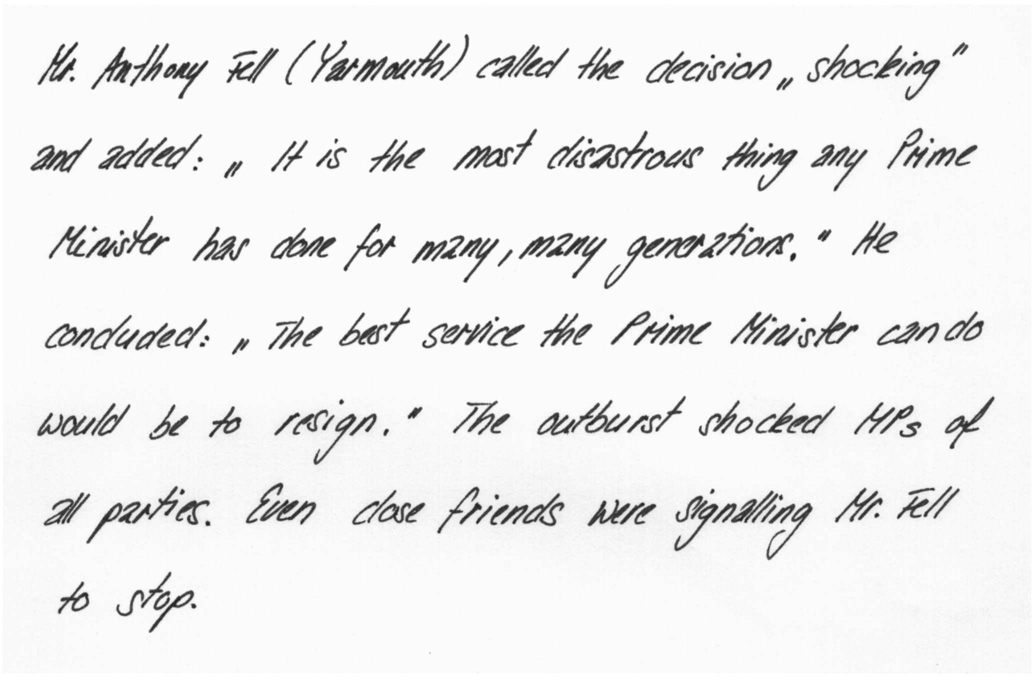
This transformation is also applied to the bounding boxes, as it changes the geometry of the image.

4.3 Groundtruth generation

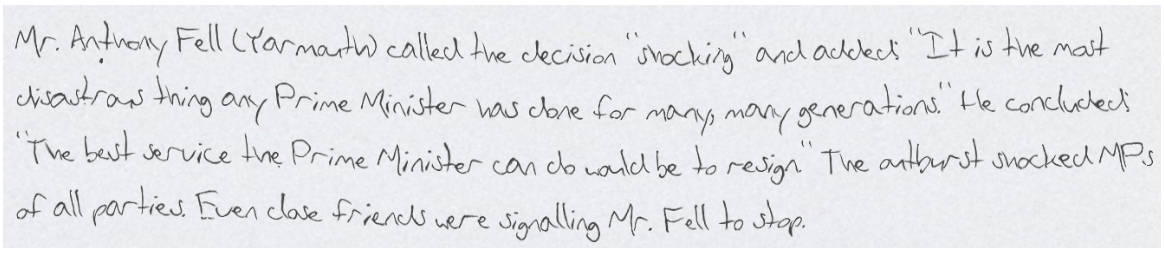
Once the final image has been synthesized, the character bounding boxes are stitched together to form words or lines, depending on the selected option.

Words can directly be split in the plain text, and the starting and ending index of their character's bounding boxes can be obtained to create the whole word bounding box.

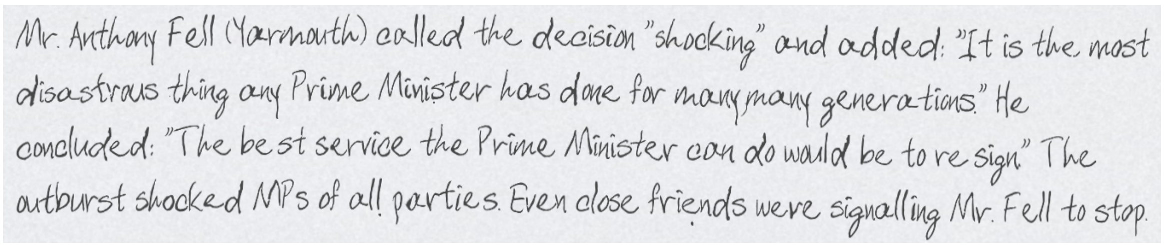
For lines, however, it is harder, because word wrapping happens at unknown locations in the text. To circumvent this problem, an algorithm is run, which uses a simple rule to determine when the next character is on a new line: if the right edge of the next character is to the left of the left edge of the current character, then the next character has been wrapped to a new line.


Mr. Anthony Fell (Yarmouth) called the decision „shocking"
and added: „ It is the most disastrous thing any Prime
Minister has done for many, many generations." He
concluded: „ The best service the Prime Minister can do
would be to resign." The outburst shocked MPs of
all parties. Even close friends were signalling Mr. Fell
to stop.

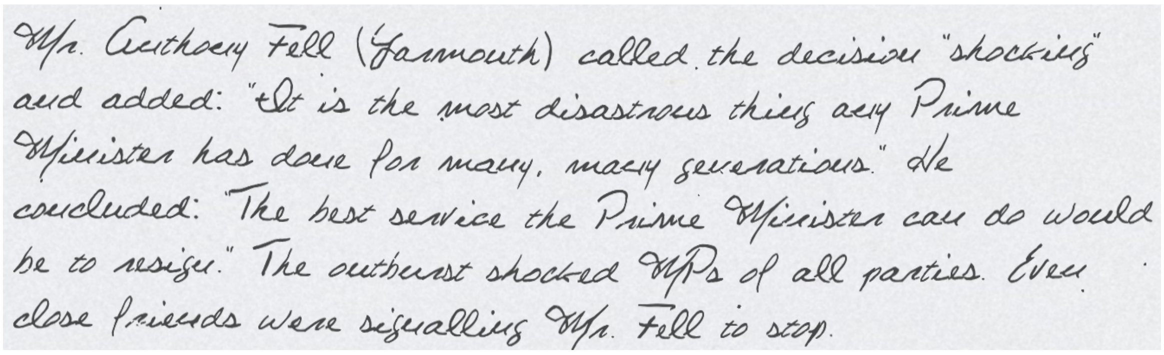
(a)


Mr. Anthony Fell (Yarmouth) called the decision "shocking" and added: "It is the most
disastrous thing any Prime Minister has done for many, many generations." He concluded:
"The best service the Prime Minister can do would be to resign." The outburst shocked MPs
of all parties. Even close friends were signalling Mr. Fell to stop.

(b)


Mr. Anthony Fell (Yarmouth) called the decision "shocking" and added: "It is the most
disastrous thing any Prime Minister has done for many many generations." He
concluded: "The best service the Prime Minister can do would be to resign." The
outburst shocked MPs of all parties. Even close friends were signalling Mr. Fell to stop.

(c)


Mr. Anthony Fell (Yarmouth) called the decision "shocking"
and added: "It is the most disastrous thing any Prime
Minister has done for many, many generations." He
concluded: "The best service the Prime Minister can do would
be to resign." The outburst shocked MPs of all parties. Even
close friends were signalling Mr. Fell to stop.

(d)

Fig. 3: Form a06-044 of the IAM dataset. Figure (a) shows the original form, written by a real human. Figures (b), (c) and (d) show the same form synthesized with several randomly generated author parameters.

After the desired bounding boxes are obtained, they are dilated by a small factor, and they are randomly distorted. This aims to imitate the imprecision sometimes found in the segmentation of real handwritten documents.

Finally, the segments corresponding to the bounding boxes are saved, together with their associated ground truth.

4.4 IAM clone

The objective of the IAM clone is to create a database of synthetic samples containing the same text as the original dataset and imitating its variability of styles. The idea is to generate random parameters for each distinct writer who participated in the IAM, then use them to render all their texts. To accomplish this, the synthesizer underwent some changes, and some preliminary steps were performed.

4.4.1 Generating the corpus

Firstly, to be able to synthesize the IAM, we need to retrieve all the texts and their associated author. The dataset provides XML files which contain the author index and the full text for each form, allowing us to accomplish this quite easily.

Once all the XML files are parsed, the full IAM corpus is available to be read by the synthesizer in the following file structure: `author_index/from_index.txt`

4.4.2 Generating author parameters

For each distinct author, some random parameters are generated related to their writing style. These parameters are saved together with their rendered texts so that they could be reproduced if needed. In principle, a font, slant and skew would be assigned to each of them, but due to time constraints, only fonts were implemented.

Since there are 657 writers who contributed to the IAM, we tried to obtain as many different handwriting typefaces as possible. This proved to be a costly task, since each of them had to be checked manually to guarantee its validity. In the end, a pack of 404 typefaces which had already been validated was used. However, some of them failed the validation process explained in section 4.1, leaving a total of 389 distinct fonts able to be used for the synthesizer.

In figure 3, a form synthesized with several different author parameters is shown, together with the original one found in the IAM dataset.

5 HTR MODEL

The HTR model used to evaluate our system consists of a BLSTM, as explained in section 2.1.2. In this section, the main characteristics of the network will be explained.

After considering many publicly available networks for handwriting recognition, we settled on one made by a student at University of Seville [21]. This network was chosen because it is well-documented, crafted specifically for training with the IAM, and it is implemented in *TensorFlow*, which is very convenient and allows us to take advantage of GPUs in order to perform the training more quickly.

5.1 Preprocessing

Some preprocessing steps are done to the input before feeding it into the network.

First, all images are resized vertically, then padded horizontally, so that their size is 1024x64. This is due to the way in which the network has been constructed, requiring constant-sized images.

After that, all pixel values are normalized so that they are between 0 and 1, where 0 represents no ink and 1 represents the maximum amount of ink.

No other preprocessing techniques are applied.

5.2 Architecture

The network consists of the following modules:

- 5 convolutional layers, with max pooling and dropout, to extract meaningful features from the input.
- A BLSTM layer, which processes the features as described in section 2.1.2.
- A fully connected layer, which allows to reduce the dimensionality of the BLSTM output so that it matches the number of possible labels.
- A CTC (Connectionist Temporal Classification) layer [6], which gives the network a derivable loss function which can be used to perform backpropagation.

For more detailed information about the network, see the documentation provided in the cited repository [21].

6 TESTS AND RESULTS

In order to evaluate our system, a series of tests have been performed, training the model described in section 5 with different samples.

Although samples and groundtruth can be generated at any level up to paragraph, the evaluation has only been performed at word level for simplicity's sake.

In this section, it will be detailed how each test was performed, the results obtained, and why it was considered relevant for this project.

In all tests, the training was done using a batch size of 100 images, and was run for about 50,000 epochs, then the model with the least validation rate was used for testing.

To evaluate the results of our model against the test samples, the Character Error Rate (CER) between its output and the ground truth is calculated.

The CER is a measure which indicates how different a given word is from another. It is calculated in the following way:

$$CER = \frac{i+s+d}{n}$$

Where n is the length of the objective word, and $i+s+d$ is the *edit distance* between the objective word and the word output by the network, calculated by adding the number of insertions, substitutions and deletions necessary to transform one word into the other.

To calculate the total CER for more than one word pair, the edit distance of each pair is added and it is divided by the total number of characters. This yields a more accurate and realistic result than just computing the arithmetic mean of the individual CER of each pair.

6.1 IAM dataset

The first test has been training the network with the full IAM dataset. This sets the baseline for all future tests. With this baseline, it is possible to meaningfully extract conclusions from the results obtained by training the network with synthetic samples.

To perform this test, a subset of 67,207 words was selected out of all the images found in the IAM dataset. Some of the images were excluded because they were either faulty or contained symbols which were not considered for the experiments (only letters and digits were considered).

The samples were divided into the following sets:

- Training: 42,719
- Validation: 11,899
- Test: 12,589

It is important to use disjoint sets of samples for training, validating and testing the network, otherwise the results would be biased towards a higher accuracy.

For fairness sake, all subsequent tests which use samples from the IAM divide them in the same way. As such, they will be referred to, from now, on as the IAM training set, IAM validation set and IAM test set.

The CER obtained with this model is 19.83%. It is an acceptable result, given that we use a normal BLSTM, without any of the more advanced techniques used in other works, like multidimensional RNNs [5] or attention models. We also do not use any dictionary nor language model, which would help correct words which only have a few wrong characters.

6.2 Synthetic IAM clone

We trained the network using only synthetic samples, and tested it against the IAM test set and against a synthetic test set. This was done to set a baseline for the performance of the network using exclusively our synthetic samples.

The samples were divided in sets following a proportion similar to the one used for dividing the IAM samples:

- Training: 67,497
- Validation: 17,387
- Test: 17,387

The total number of samples is larger than the samples used in the IAM dataset test. This is due to some discrepancies when segmenting the words. Some words which were discarded from the IAM dataset contained spaces. Our generator segments those into two or more words, thus producing more usable samples.

The CER obtained with this model is 5.64% when testing against synthetic samples and 93.65% when testing against the IAM test set.

The error is quite low when testing against the same type of samples the network has seen, specially compared to the result obtained in the IAM dataset test. This suggests that synthetic samples are easier to learn for the network and allow it to generalize better to recognize unseen samples. It is also likely that having a larger amount of training samples contributed to the improvement.

On the other hand, the error when testing against IAM samples is quite high. This suggests that our synthetic samples are different enough to the real ones for the network to not adequately recognize them. It must also be taken into account that the writing styles of the synthetic samples are not necessarily the ones found in the IAM dataset, so the network is trying to recognize writing styles it has never seen based on other different writing styles it has seen, which is something quite hard to accomplish.

6.3 Synthetic and real samples combination

p	CER (%)	Loss (%)
0	93.65	73.82
0.2	27.14	7.31
0.4	24.19	4.36
0.7	24.27	4.44
1	19.83	0

Table 1: Performance obtained by training the network with different amounts of real samples together with synthetic samples. The *Loss* column shows the difference between the obtained CER and the baseline CER (using all real samples).

After setting the baseline for fully real and fully synthetic training, series of tests consisting of combinations of both were performed. The objective is to explore how the model behaves when training it using all the synthetic samples but also feed it a portion of real samples, so that it can use them to generalize better and learn to recognize them.

The tests have been performed in the following way:

In each batch, a percentage p of real samples are fed to the network, while the rest are synthetic samples. Synthetic samples are pulled randomly from the full IAM clone dataset, whilst real samples are pulled randomly from a limited portion of the IAM train set, consisting of $p \cdot n$ samples, where n is the total number of IAM train set samples. In this way, the network is able to see all synthetic samples, but it can only see a partial amount of the real ones.

The $p \cdot n$ samples are chosen randomly from the IAM training set. Since a uniform distribution was used and the set is large enough, the random samples are assumed to be representative of the whole set.

The validation is done with the IAM validation set, since that is what we are aiming to train our network for. Finally, the test is made using the IAM test set.

This test was performed with a p value of 0.2, 0.4 and 0.7. The IAM dataset test could be considered to be this test with $p = 1$ and the Synthetic IAM clone test could be considered to be this test with $p = 0$.

The CER obtained for each value of p is shown in table 1.

We can observe that, by adding 20% of real samples, we obtain a CER much lower than the one obtained by using only synthetic samples. It is also considerably close to the

CER obtained by training with all the real samples, being only about a 7% higher. This difference further reduces when using 40% of real samples. This proves that it is possible to obtain a model with an accuracy close to the optimal one by using only a portion of the samples and our synthetic samples.

It must be noted that, when using 70% of the real samples, the CER does not decrease as would be expected. This could be explained by the fact that, since only 30% of images in each batch are synthetic, the network does not learn them as well as in the other cases, and leans more toward the IAM samples, which do not provide enough generalization, as seen in the next section.

It is also worth noting that the increase in accuracy when adding real samples is very high when having a small amount of samples, but gets lower as more samples are added, as seen in figure 4. Interestingly, this behaviour seems to be similar to that of a *knowledge transfer* system described in [16], in which a network is trained with a dataset, then it is fine-tuned and tested for a different one (which is analogous to training our network with synthetic samples and trying to recognize real samples).

6.4 Partial real samples

p	CER (%)	Loss (%)
0.2	42.13	22.3
0.4	30.65	10.82
0.7	27.92	8.09
1	19.83	0

Table 2: Performance obtained by training the network with different amounts of real samples, without including any synthetic one. The *Loss* column shows the difference between the obtained CER and the baseline CER (using all real samples).

Finally, we performed exactly the same tests as in section 6.3, but without including any synthetic samples in the batches. The objective of these tests is to explore how much of an improvement our synthetic samples provide when limiting the amount of real samples seen by the network.

The results are shown in table 2.

As expected, the CER decreases more as a larger portion of real samples is used to train the network.

A comparison between these results and the results in section 6.3 is shown in figure 4.

We can see that, in every case, the network performs better when we include our synthetic samples to complement the real samples, thus proving they allow the network to generalize better and recognize unseen real samples.

7 Conclusions and future work

The main objectives of the project have been fulfilled, although there are many possibilities which have been left unexplored.

Our system has proven to be able to partially replace real data for training HTR systems. By using only a small portion of real labelled samples and replacing the rest with synthetic samples, we were able to obtain a model which performed within 8% of the accuracy of a model trained with the full set of real samples. This is interesting, because it could be applied to real document recognition problems in which obtaining labelled samples is costly.

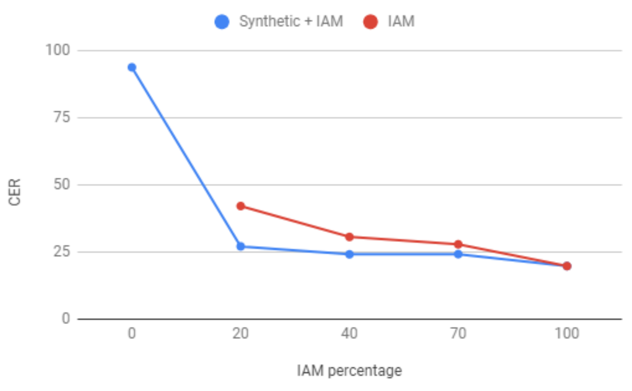


Fig. 4: A graph showing the results of the experiments described in sections 6.3 and 6.4. The blue plot shows the CER obtained when training the network using both synthetic and real samples, whilst the red plot shows the CER obtained when training the network only with real samples.

It has also been shown that adding synthetic samples to the training set always produces better results than using only real samples. However, the increase in performance produced by our system has not been compared to that of other techniques aimed at increasing the number of samples, such as data augmentation. A future work line would be to perform more tests and determine where our system lies within all other available techniques, or how much does the performance improve when combining our system with those techniques.

It would also be interesting to test if our system performs well when using it for transfer learning. The idea would be to train a model with a large amount of synthetic samples, so that it learns the more abstract aspects of handwriting recognition, and then fine-tune it using a small amount of labelled samples from the real documents which need to be recognized.

Finally, there is some room for improvement in the synthesis process. The font rendering step could be replaced by some of the generative methods mentioned in section 2.2. This would allow the text to be much more human-like, and different writing styles could be emulated by tuning the generator's parameters.

Furthermore, some tests could be performed to determine which distortions are the most useful for increasing the performance of the network, and new operations, such as text slant and skew, could be implemented and evaluated.

Acknowledgements

I would like to thank my tutors, Alicia Fornes and Marçal Rossinyol, for guiding me through the project and recommending me the best course of action whenever I was unsure about it.

I would also like to thank all my colleagues in CVC, specially Anguelos Nicolaou, for helping me out with the synthesizer and sharing with me his thoughts about this project, and Joan Mas, for allowing me to use his GPU and helping me train the neural network.

Finally, I would like to thank my friends and family, which have helped me keep a more positive attitude when I was feeling anxious or thought I would not make it through.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Ujjwal Bhattacharya, Réjean Plamondon, Souvik Dutta Chowdhury, Pankaj Goyal, and Swapan K. Parui. A sigma-lognormal model-based approach to generating large synthetic online handwriting sample databases. *International Journal on Document Analysis and Recognition (IJDAR)*, 20(3):155–171, Sep 2017.

[3] Volkmar Frinken and Horst Bunke. Continuous handwritten script recognition. pages 391–425, 01 2014.

[4] Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

[5] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Multi-dimensional recurrent neural networks. *CoRR*, abs/0705.2011, 2007.

[6] 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.

[7] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):855–868, 2009.

[8] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. Speech recognition with deep recurrent neural networks. *CoRR*, abs/1303.5778, 2013.

[9] Alex Graves and Jürgen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 545–552. Curran Associates, Inc., 2009.

[10] Tom SF Haines, Oisin Mac Aodha, and Gabriel J Brostow. My text in your handwriting. *ACM Transactions on Graphics (TOG)*, 35(3):26, 2016.

[11] Nicholas Journet, Muriel Visani, Boris Mansencal, Kieu Van-Cuong, and Antoine Billy. Doccreator: A new software for creating synthetic ground-truthed document images. *Journal of imaging*, 3(4):62, 2017.

[12] Praveen Krishnan and CV Jawahar. Generating synthetic data for text recognition. *arXiv preprint arXiv:1608.04224*, 2016.

[13] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[14] I. King M.-C. Yuen and K.-S. Leung. A survey of crowdsourcing systems. *IEEE third International Conference on Privacy, security, risk and trust (PASSAT), and IEEE third International Conference on Social Computing (Socialcom)*, pages 766–773, 2011.

[15] U.-V. Marti and H. Bunke. The iam-database: an english sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5(1):39–46, Nov 2002.

[16] Rathin Radhakrishnan Nair, Nishant Sankaran, Bhargava Urala Kota, Sergey Tulyakov, Srirangaraj Setlur, and Venu Govindaraju. Knowledge transfer using neural network based approach for handwritten text recognition. In *13th IAPR International Workshop on Document Analysis Systems, DAS 2018, Vienna, Austria, April 24-27, 2018*, pages 441–446, 2018.

[17] Christopher Olah. Understanding lstm networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs. Online, last revision on August 27, 2015.

[18] Keith Packard. Fontconfig: a library for configuring and customizing font access. `https://www.fontconfig.org/`, 2018.

[19] Keith Packard and Carl Worth. Cairo: a 2d graphics library with support for multiple output devices. `https://www.cairographics.org/`, 2014.

[20] A. Poznanski and L. Wolf. Cnn-n-gram for handwriting word recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2305–2314, June 2016.

[21] Hugo Rubio. Offline handwriting recognition with deep learning implemented in tensorflow. `https://github.com/hugrubsan/Offline-Handwriting-Recognition-with-TensorFlow`, 2017.

[22] J. Schmidhuber S. Hochreiter. Long short-term memory. *Neural computation, 9(8):1735–1780*, 1997.

[23] Ray Smith. An overview of the tesseract ocr engine. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 2, pages 629–633. IEEE, 2007.

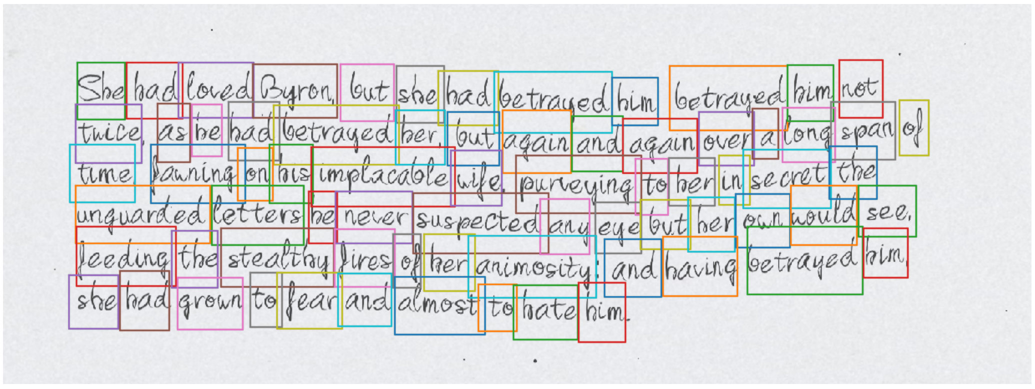
[24] Owen Taylor and Raph Levien. Pango: a library for laying out and rendering of text, with an emphasis on internationalization. `https://www.pango.org/`, 2012.

[25] P. Voigtlaender, P. Doetsch, and H. Ney. Handwriting recognition with large multidimensional long short-term memory recurrent neural networks. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 228–233, Oct 2016.

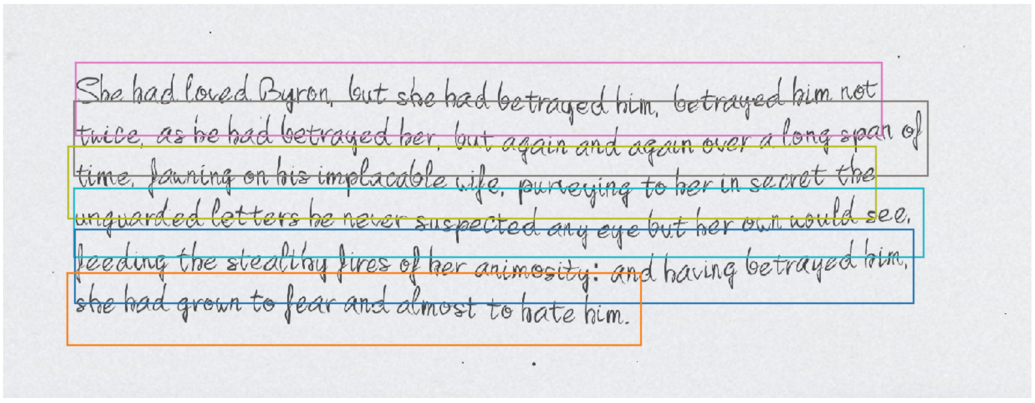
[26] Taylor Tian Yuchen. zi2zi: Master chinese calligraphy with conditional adversarial networks. https://kaonashi-tyc.github.io/2017/04/06/zi2zi.html. Online, last revision on April 6, 2017.

Appendix

In this section, some extra figures related to the synthesized texts are shown.

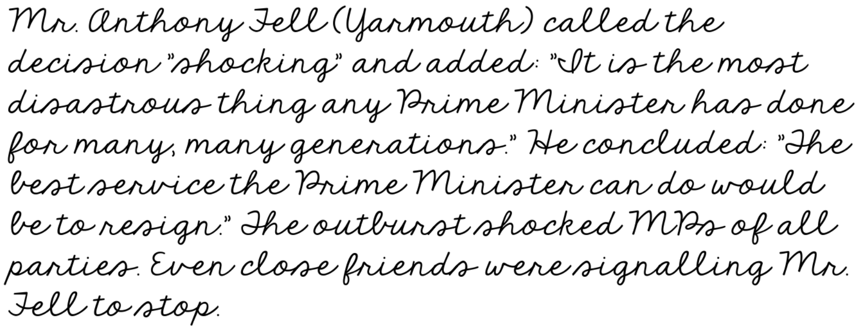

She had loved Byron, but she had betrayed him, betrayed him not
twice, as he had betrayed her, but again and again over a long span of
time, fawning on his implacable wife, purveying to her in secret the
unguarded letters he never suspected any eye but her own would see,
feeding the stealthy fires of her animosity; and having betrayed him,
she had grown to fear and almost to hate him.

(a) Word level bounding boxes

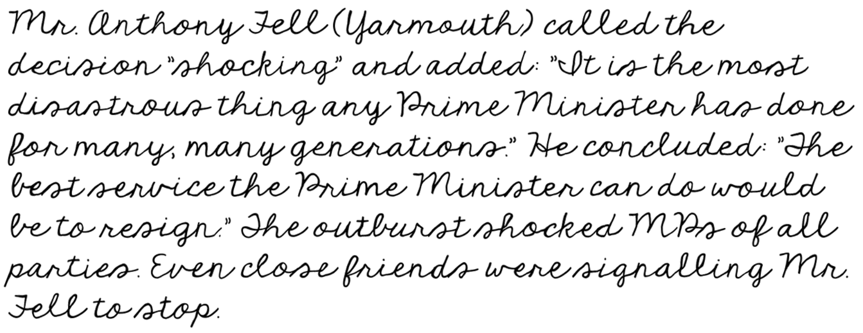

She had loved Byron, but she had betrayed him, betrayed him not
twice, as he had betrayed her, but again and again over a long span of
time, fawning on his implacable wife, purveying to her in secret the
unguarded letters he never suspected any eye but her own would see,
feeding the stealthy fires of her animosity; and having betrayed him,
she had grown to fear and almost to hate him.

(b) Line level bounding boxes

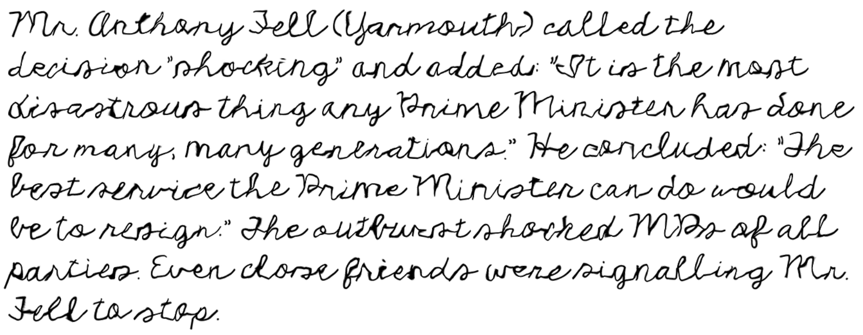
Fig. 5: Bounding boxes of the segments at different levels of segmentation for the same form. Character level is not shown due to the image becoming too cluttered.


Mr. Anthony Fell (Yarmouth) called the
decision "shocking" and added: "It is the most
disastrous thing any Prime Minister has done
for many, many generations." He concluded: "The
best service the Prime Minister can do would
be to resign." The outburst shocked MPs of all
parties. Even close friends were signalling Mr.
Fell to stop.

(a) Original text

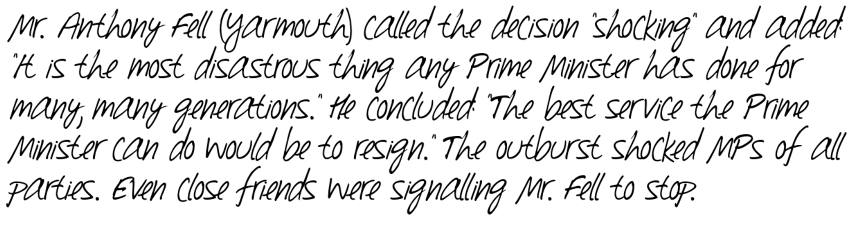

Mr. Anthony Fell (Yarmouth) called the
decision "shocking" and added: "It is the most
disastrous thing any Prime Minister has done
for many, many generations." He concluded: "The
best service the Prime Minister can do would
be to resign." The outburst shocked MPs of all
parties. Even close friends were signalling Mr.
Fell to stop.

(b) Low elastic distortion

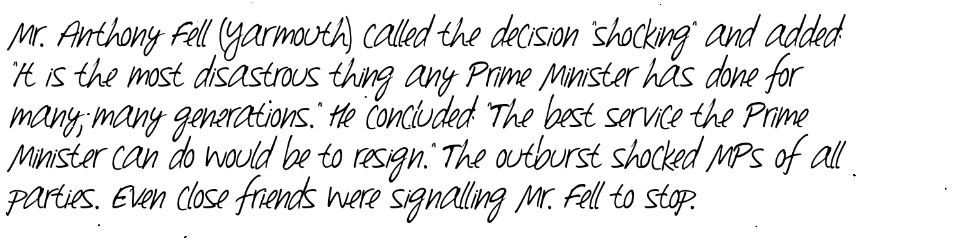

Mr. Anthony Fell (Yarmouth) called the
decision "shocking" and added: "It is the most
disastrous thing any Prime Minister has done
for many, many generations." He concluded: "The
best service the Prime Minister can do would
be to resign." The outburst shocked MPs of all
parties. Even close friends were signalling Mr.
Fell to stop.

(c) High elastic distortion

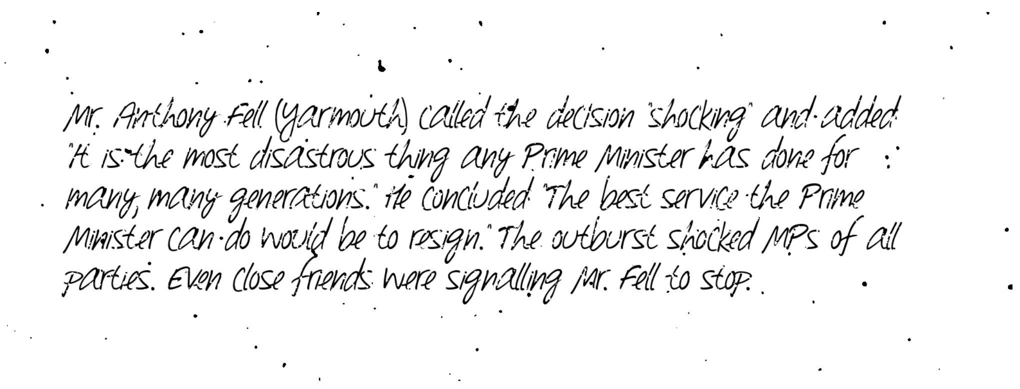
Fig. 6: A comparison between different levels at which the elastic distortion can be applied.


Mr. Anthony Fell (Yarmouth) called the decision "shocking" and added: "It is the most disastrous thing any Prime Minister has done for many, many generations." He concluded: "The best service the Prime Minister can do would be to resign." The outburst shocked MPs of all parties. Even close friends were signalling Mr. Fell to stop.

(a) Original text

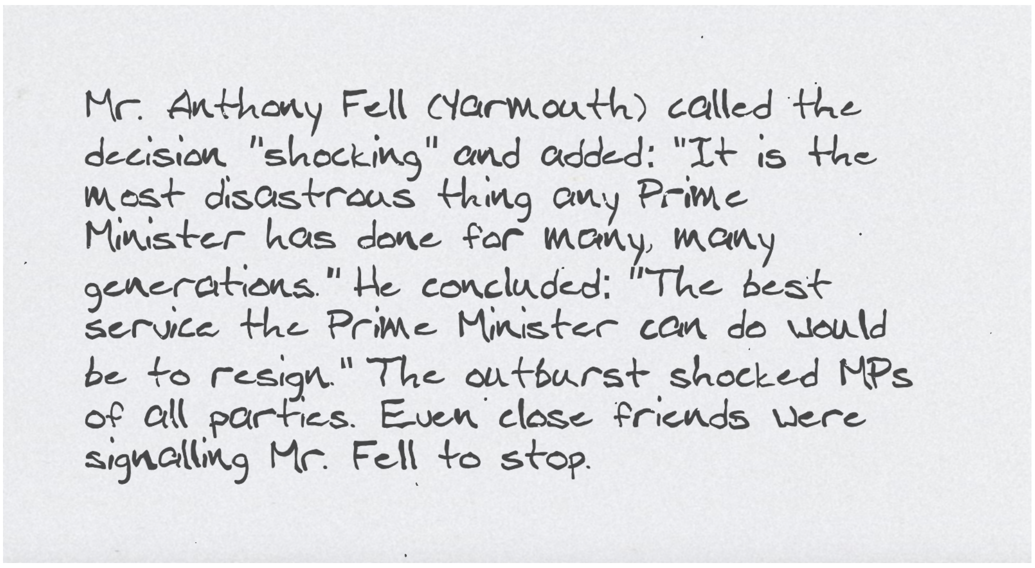

Mr. Anthony Fell (Yarmouth) called the decision "shocking" and added: "It is the most disastrous thing any Prime Minister has done for many, many generations." He concluded: "The best service the Prime Minister can do would be to resign." The outburst shocked MPs of all parties. Even close friends were signalling Mr. Fell to stop.

(b) Low ink degradation


Mr. Anthony Fell (Yarmouth) called the decision "shocking" and added: "It is the most disastrous thing any Prime Minister has done for many, many generations." He concluded: "The best service the Prime Minister can do would be to resign." The outburst shocked MPs of all parties. Even close friends were signalling Mr. Fell to stop.

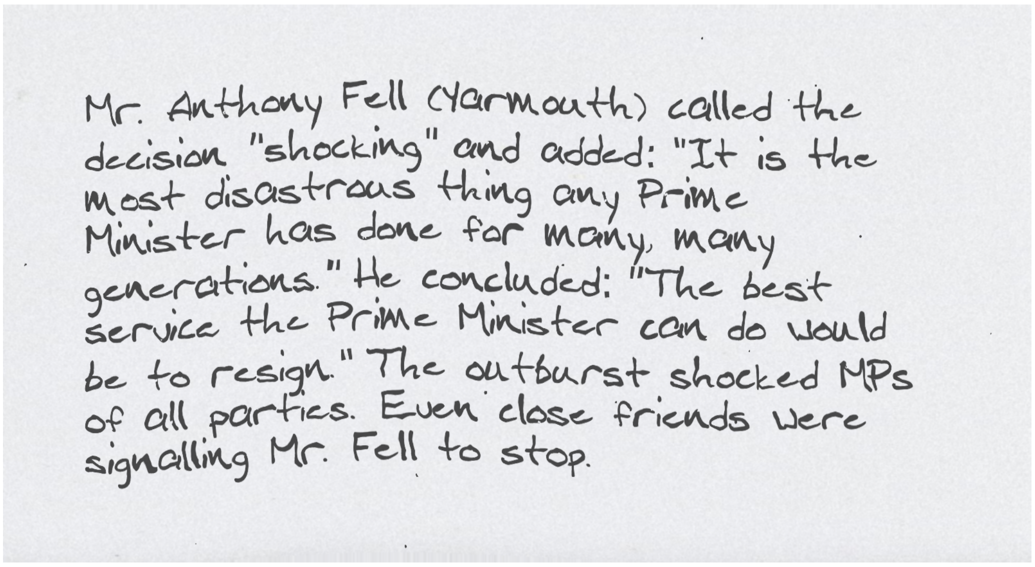
(c) High ink degradation

Fig. 7: A comparison between different levels at which the ink degradation can be applied.



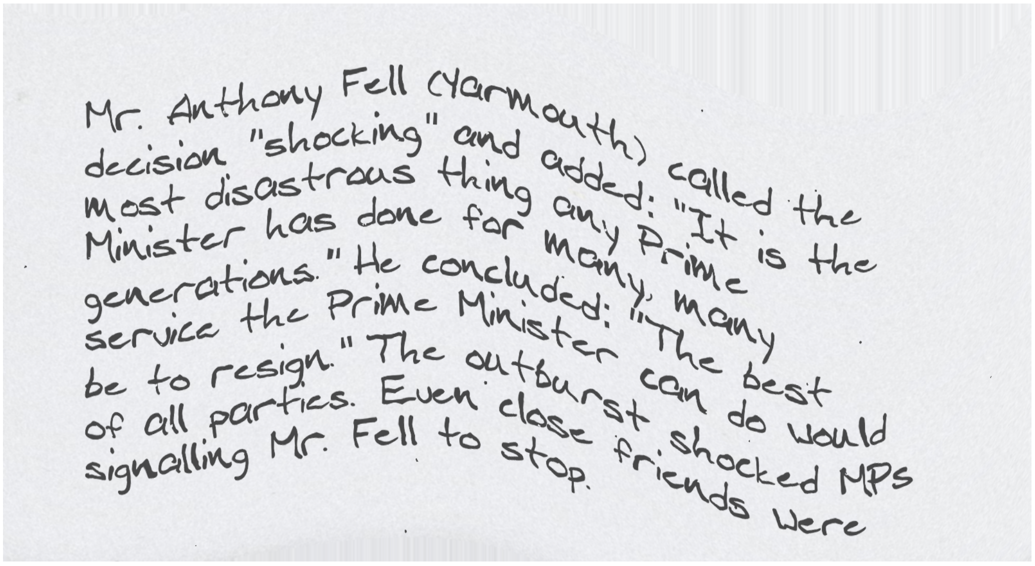
Mr. Anthony Fell (Yarmouth) called the
decision "shocking" and added: "It is the
most disastrous thing any Prime
Minister has done for many, many
generations." He concluded: "The best
service the Prime Minister can do would
be to resign." The outburst shocked MPs
of all parties. Even close friends were
signalling Mr. Fell to stop.

(a) Original page



Mr. Anthony Fell (Yarmouth) called the
decision "shocking" and added: "It is the
most disastrous thing any Prime
Minister has done for many, many
generations." He concluded: "The best
service the Prime Minister can do would
be to resign." The outburst shocked MPs
of all parties. Even close friends were
signalling Mr. Fell to stop.

(b) Low paper wrapping



Mr. Anthony Fell (Yarmouth) called the
decision "shocking" and added: "It is the
most disastrous thing any Prime
Minister has done for many, many
generations." He concluded: "The best
service the Prime Minister can do would
be to resign." The outburst shocked MPs
of all parties. Even close friends were
signalling Mr. Fell to stop.

(c) High paper wrapping

Fig. 8: A comparison between different levels at which paper wrapping can be applied.