Unsupervised OCR Model Evaluation Using GAN

Abhash Sinha
TU Kaiserslautern
Kaiserslautern, Germany
abhashsinha09@gmail.com

Martin Jenckel
TU Kaiserslautern, DFKI
Kaiserslautern, Germany
martin.jenckel@dfki.de

Syed Saqib Bukhari
DFKI
Kaiserslautern, Germany
saqib.bukhari@dfki.de

Andreas Dengel
TU Kaiserslautern, DFKI
Kaiserslautern, Germany
andreas.Dengel@dfki.de

***Abstract*—Optical Character Recognition (OCR) has achieved its state-of-the-art performance with the use of Deep Learning for character recognition. Deep Learning techniques need large amount of data along with ground truth. Out of the available data, small portion of it has to be used for validation purpose as well. Preparing ground truth for historical documents is expensive and hence availability of data is of utmost concern. Jenckel et al. [7] came up with an idea of using all the available data for training the OCR model and for the purpose of validation, they generated the input image from Softmax layer of the OCR model; using the decoder setup which can be used to compare with the original input image to validate the OCR model. In this paper, we have explored the possibilities of using Generative Adversial Networks (GANs) [6] for generating the image directly from the text obtained from OCR model instead of using the Softmax layer which is not always accessible for all the Deep Learning based OCR models. Using text directly to generate the input image back gives us the advantage to use this pipeline for any OCR models even whose Softmax layer is not accessible. In the results section, we have shown that the current state of using GANs for unsupervised OCR model evaluation.**

Keywords*—*OCR, Historical Document Image Processing, GAN

I. Introduction

There has been a significant progress in the field of Optical Character Recognition (OCR) in recent years. Traditionally, OCR applications were pipelined with hand-built and highly tuned modules. For example, one module might find lines of text, then the next module would find words and segment letters, then another module might apply different techniques to each piece of a character's recognition.

Inclusion of Deep Learning in OCR further improved the accuracy of OCR. They have surpassed the performance of traditional Computer Vision techniques for OCR by a huge margin in terms of accuracy. But training such an OCR model needs large amount of data. Also, to validate the trained models, small set of data is needed which has not been used in training the model. The task becomes more daunting as the training is mostly Supervised Learning in which labeled data is needed. Usually, this labelling is carried out by humans and is often expensive.

To overcome this problem, Jenckel et al. [7] proposed an approach in which we can use all the available data only for training and for validation, we can generate the input image back from the OCR output text. The idea is that if the OCR model has any errors, those errors will be propagated to the output text and to the fake image generated out of that text. Hence, when we compare the input image with the fake image, we can evaluate the performance of OCR models.

On top of Jenckel et al's approach, I have used Generative Adversial Networks (GANs) [6], first introduced by Goodfellow et al. in 2014, to generate the fake image. The GAN architecture combines two networks, Generator and Discriminator, which competes against each other which is also known as min-max game. Generator generates data and Discriminator tries to predict whether the data is fake or real. There have been various research in the field of GANs like [9], [11], [8].

In Section II, we will be discussing about the related works. In Section III of this paper, the evaluation of OCR trained models will be discussed. In Section IV, we will discuss about the data sets used in the experiments. Section V follows the discussion about the approach based on GANs to generate the image and Section VI follows the result of the experiments.

II. Related Works

Jenckel et al. [7] proposed a novel method for evaluation of Deep Learning based OCR models without transcription. The approach uses a secondary Long Short-Term Memory (LSTM) as an encoder-decoder setup. The goal is to generate the image containing the text line from the output of the OCR model. The trained OCR model can then be evaluated by comparing the generated text line image and the original input image. The paper has exploited the idea that if there are errors in the OCR model, it will be propagated to the generated image as well. The proposed method uses LSTM variation of auto-encoders in which the encoder and decoder are trained independently. The encoder is the OCR model and the output of its Softmax layer is used as input to the decoder setup.

The advantage of using Softmax layer as input to decoder is that it contains information like alignment between the characters and the corresponding character in the input image. Also, it contains the confidence score of each character which can be useful in regenerating that character with some confidence. With all these advantages, there is a disadvantage as well. Not every OCR model's Softmax layer can be accessed. OCR engines like Tesseract [4] and ABBY [1] does not provide access to their Softmax layer. This problem can be solved if we can generate the text line image from the output text itself rather than the Softmax layer.

In this paper, we have used Generative Adversial Network (GANs) [6] for generating the text line image instead of using the encoder-decoder setup of Jenckel et al. Also, we have used the output text of the OCR model to generate the image instead of using the Softmax layer of the model. In order to evaluate the model, the generated text line image is compared with the original input image. This idea is similar to the idea proposed by Alex Graves in his paper [2] in which he generated handwritten characters and sentences from the text.

III. Evaluation of OCR models

Once the Deep Learning based OCR models are trained or while the training is still going on, they must be evaluated on the validation set to analyze the use of hyper parameters in the training and the result which is being produced. Monitoring the training process helps in determining whether the training is progressing in the right direction or has it stopped.

Jenckel et al. [7] used the approach of encoder-decoder setup to generate the input image and then compared this generated fake image with original input image to find the error produced by the OCR model. We propose to use GANs [6] to generate the image instead of using the encoder-decoder setup.

IV. Data sets used

For the experiments, we have used two kinds of data sets. The first one is a historical data set in Latin [12], discussed in Section IV-A and the second one is a contemporary data set UWIII [10] which is discussed in Section IV-B.

Latin data set is relatively smaller in number (3418 sample text lines) than UWIII data set but contains historical characters and fonts whereas UWIII data set was used because of its large size of 10520 samples.

A. Historical data set: Latin

Latin data set [12] contains 3418 text lines taken from a Latin version of the novel "Narrenschiff". Some samples has been shown in the Figure 1.

The alphabet of the Latin document consists of 86 different characters. The whole data set was split into three sub sets: training set, validation set and test set. The training set had 2673 text lines, validation set had 396 text lines and the test set had 349 text lines.

B. Contemporary data set: UWIII

UWIII stands for University of Washington 3 [10]. UWIII data set contains 105206 text lines which was divided into three parts as done with Latin data set. Training set had 78905 text lines, validation set had 15781 samples and the test set had 10520 samples. A random sample from the UWIII data set has been shown below in the Figure 2. In the figure, the top line is the text line from the .txt file and the bottom line is the corresponding image of the text.

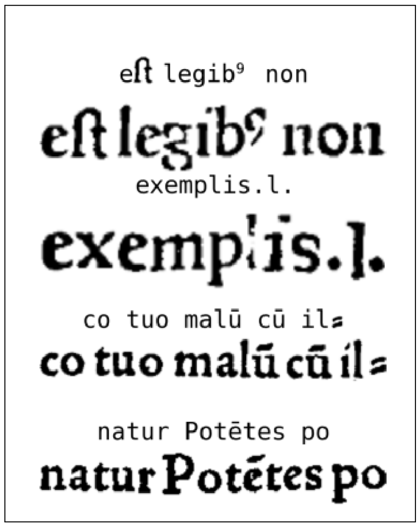
eſt legib⁹ non
eſt legib⁹ non
exemplis.l.
exemplis.l.
co tuo malū cū il⸗
co tuo malū cū il⸗
natur Potētes po
natur Potētes po

Fig. 1. Four samples from Latin data set with text on top and its corresponding image below. Credit: [7].

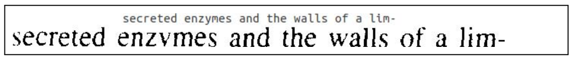
secreted enzymes and the walls of a lim-
secreted enzymes and the walls of a lim-

Fig. 2. A random sample from UWIII data set. Top line is the text and the bottom line is the image corresponding to the text.

C. Preprocessing

Since, we used TensorFlow for training our models, we had to keep the dimension of our tensors of fixed size. So, we did some preprocessing on the data which are:

- Before feeding the data for training, the height and width of the images were normalized to same height and same width maintaining the aspect ratio. The height was kept at 48 pixels and the corresponding width was calculated using the below formula:

$$w_{new} = w_{orig} \times \frac{h_{new}}{h_{orig}} \tag{1}$$

- Once the new widths are calculated, all the images were resized to same width by padding.

V. Training setup

For training the evaluation model, three individual experiments were done. In the first experiment, Latin data set was used as it consists of historical document lines. In the second experiment, UWIII data set was used, as it has more training samples compared to Latin data set but contains contemporary English sentences. In the last experiment, pix3pixHD [13] was used. This experiment is also referred as "normalization of OCRed text and input image to same font in image"

The focus of all the three experiments was to use the text output of the OCR models to generate the fake image which resembles the original input image. Then, we can use a simple Sum of Absolute Difference (SAD) for comparing the fake image and input image to evaluate the OCR models.

In this paper, we have tried various versions of GANs. Also, we tried using the combination of two GANs like Conditional GAN [9] (CGAN) and Least Squared GAN [8] (LSGAN). The text is taken as input and corresponding text line image as ground truth for all the experiments performed.

A. Experiments with Latin data set

We combined the Conditional GAN with Least Squared GAN to generate the fake image. The architecture of the network was based on CGAN and the loss function was the combination of loss from LSGAN and Sigmoid Cross Entropy by taking the sum of losses. An abstract architecture of the GAN is shown in Figure 3.

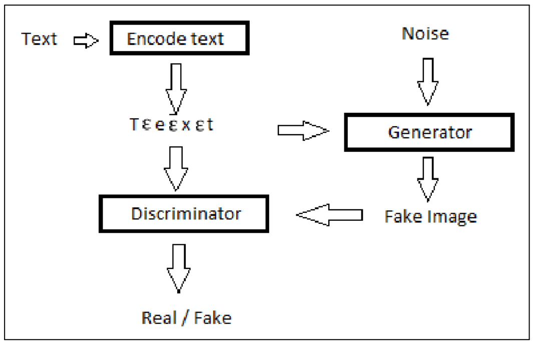

Fig. 3. The network architecture. Text has been encoded to include ϵ after every character and then it is passed to the Generator which takes noise as another input and generates a fake image. The encoded text is also passed to the Discriminator which takes the fake image from Generator or real image from ground truth as another input and tries to predict whether the input image to Discriminator is real or fake. It predicts 1 for real image and 0 for fake image.

The characters in the text line were separated from each other using a separator (ϵ). The characters were separated in order to make the repetition of characters in a word look more obvious. The text lines were padded with zeros on either side before feeding it to the Generator part of the model. This padding was done to keep the width of all the text in the data set to be same. Each character was encoded with its corresponding index from the vocabulary of characters created over the whole data set.

In the Generator, a Dense layer takes the input of encoded text. The output of the Dense layer was concatenated with the noise and then fed to BLSTM (Bi-directional LSTM) layer. On top of the BLSTM layer, a Dense layer was used to generate fake image with same width and height as the corresponding width and height of ground truth. The abstract architecture of Generator is shown in the Figure 4.

In the Discriminator, the input text was first passed to a Dense layer so that the output is equal to the width of the image to be generated and then it is concatenated with the image either coming from Generator or ground truth. The concatenated output is then provided to a BLSTM layer. A final Dense layer was placed on top of the BLSTM layer with *sigmoid* activation function to predict whether the input image

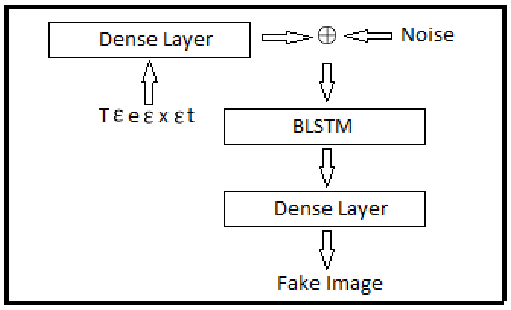

Fig. 4. Generator with new encoded text and Dense layer. The encoded text is passed to a Dense layer and then concatenated with a noise which is another input to the Generator. The concatenated tensor is passed to BLSTM network and a final Dense layer is stacked on top of it with *tanh* as the activation function which generates the image with pixel values between -1 and 1.

to Discriminaor is fake or real. The abstract architecture of Discriminator is shown in the Figure 5.

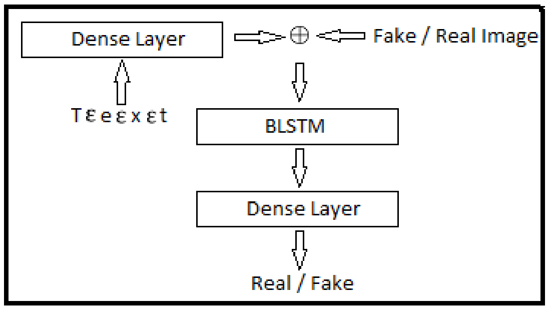

Fig. 5. Discriminator with new encoded text. The encoded text is passed to the Dense layer and then concatenated with the fake or real image which is another input to the Discriminator. The concatenated tensor is passed to BLSTM network and then to final Dense layer which has *sigmoid* as activation function. It predicts 1 for real image and 0 for fake image.

B. Experiments with UWIII data set

Normal setup for this experiment is same as shown in Figure 3. The difference here is that the characters in the text line are no more separated by ϵ.

In the Generator, we used transposed Convolution layer which upsamples the encoded text. The output of transposed Convolution layer was concatenated with noise and then fed into BLSTM layer followed by final Dense layer with *tanh* as activation function. The architecture of the Generator is shown in the Figure 6.

In the Discriminator, as shown in the Figure 7, the encoded text was passed to a Dense layer with number of neurons equal to the width of the input image which was given to the Discriminator as another input. The output of Dense layer and the input image were concatenated vertically and then passed to series of Convolution layers which extract features and can be helpful in distinguishing real image and fake image. On top of Convolution layers, two Dense layers were stacked. The final Dense layer had only one output neuron with *sigmoid*

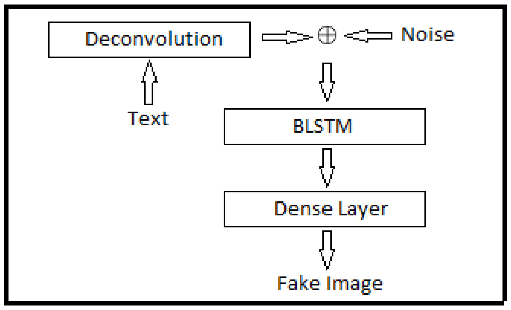

Fig. 6. Generator: In this architecture, an encoded text is passed to a transposed Convolution layer which translates the encoded text into a tensor which has width equal to the image to be generated. This output is then concatenated with the noise and passed to BLSTM layers. On top of the BLSTM layers, a Dense layer is placed with number of neurons equal to the number of pixels in the fake image to be generated. *tanh* function is used as activation to keep the value of generated image between -1 and 1.

activation function to predict whether the input image is real or fake.

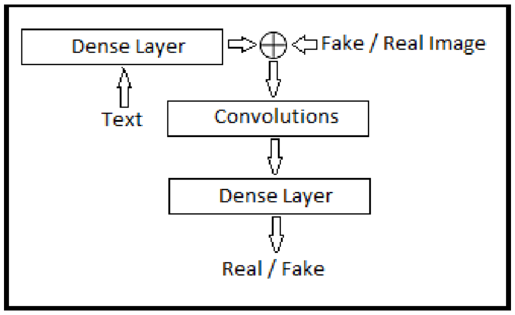

Fig. 7. Discriminator: In this architecture, an encoded text is passed to a Dense layer which translates the encoded text into a tensor which has width equal to the image which the Discriminator receives as another input. Both the Dense layer output and the input image are concatenated and passed to series of Convolution layers to extract the distinguishing features. Finally, a Dense layer is stacked on top of the Convolution layers which has only one neuron with *sigmoid* as activation function to predict whether the image is real or fake.

C. Experiments with pix2pixHD

In this experiment, we generated synthetic image out of OCRed text in one single font (e.g. Arial) using Pango library [5] in Python. Images present in UWIII data set were also rendered into the same font, in this case Arial, using pix2pixHD GAN [13]. After this step, the text was transformed into image with Arial font and the UWIII image was transformed into an image with Arial font as well. These two output images were compared to verify whether the OCRed text was correct or not as both of them were in the same font.

The whole idea behind using pix2pixHD approach was to capture the OCR error by comparing the synthetic image from the OCRed text and the transformed image from the input image. The synthetic image was generated by passing the OCRed text as input to Pango and the input image was transformed by pix2pixHD network.

For this experiment, we were passing the padded image of UWIII data set, with 1200 pixels as width and 48 pixels as height, to the pix2pixhd architecture as input image and the ground truth image was the synthetic image generated by using the corresponding text of the input image which also had the width of 1200 pixels and height as 48 pixels. Input image has been shown in the Figure 8 and corresponding ground truth has been shown in the Figure 9.

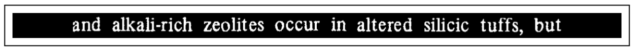
and alkali-rich zeolites occur in altered silicic tuffs, but

Fig. 8. A random sample from UWIII data set but not original. The foreground and background color have been inverted with no specific reason behind. The size of the image is 1200 x 48 pixels.

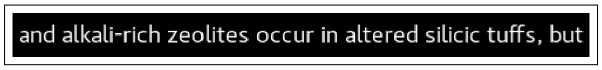
and alkali-rich zeolites occur in altered silicic tuffs, but

Fig. 9. Ground truth of the input image shown in Figure 8. This image has been synthetically generated by using Pango library. The library takes the text, font and size of the font to generate the image. The size of the image is same to the input image which is 1200 x 48.

The hyperparameters used for the experiment is shown in the Table I.

Parameters	Parameter's value
batch size	8
iterations	100
Generator's lr	2e-4
Discriminator's lr	2e-4

TABLE I
HYPERPARAMETERS USED IN THE EXPERIMENT WITH PIX2PIXHD

VI. RESULTS

The result section has been divided into three parts. In the first part, result for the experiment related to Latin data set is discussed. In the second part, result for the experiment related to UWIII data set is discussed followed by the discussion about the result of pix2pixHD experiment.

A. Result for Latin data set

In the Figure 10, result of the first epoch has been shown. In the figure, we can see that the model is just generalizing by distributing the black and white pixels all over the image. Also, there is a thick black line in the center of the image which denotes that the model has figured out that most of the black pixels are located near the center of the image.

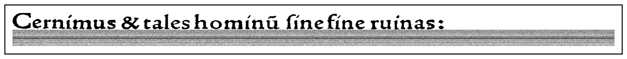
Cernímus & tales homínū ſíne ſíne ruínas:

Fig. 10. Fake image generated in first epoch after using Sigmoid Cross Entropy and Least Squares Error Losses. The top line is from the ground truth and the bottom line is the image generated by the Generator.

The fake image generated after 150 epochs is shown in Figure 11. We can see that the Generator is learning the width of the text but it is not learning the characters. Also, there are black pixels around the center of the image.

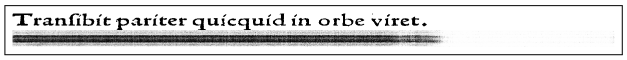
Tranſibít pariter quícquíd ín orbe víret.

Fig. 11. Fake image generated in 150th epoch after using Sigmoid Cross Entropy and Least Squares Error Losses. The top line is from the ground truth and the bottom line is the image generated by the Generator.

B. Result for UWIII data set

In the Figure 12, we can see that the Generator has learned the width of the text present in the ground truth image and also it has learned that there are white pixels gap between the characters of the text in the image.

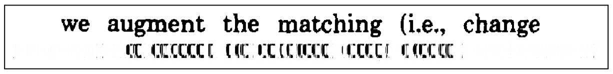
we augment the matching (i.e., change

Fig. 12. Fake image generated after first epoch. The top line is the ground truth image of the test sample and the bottom line is the image generated by the Generator of the GAN. The Generator has almost learned the width of the text present in the ground truth image and also that every character has gap filled by white pixels.

Figure 13 has been taken from the fiftieth epoch. In this iteration, we can see that the Generator has learned some uniform width of the characters present in the ground truth image. Also, it has figured out that most of the black pixels are present around the top region of the text and hence it has drawn a straight line of black pixels.

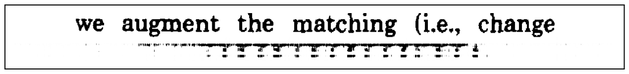
we augment the matching (i.e., change

Fig. 13. The top line is the ground truth image of the test sample and the bottom line is the image generated by the Generator of the GAN in the 50th epoch. In the iteration, we can see that the Generator has learned the characters width roughly and also that the upper line, in the fake image, shows that the Generator has figured out that there are more black pixels around the region.

C. Result of pix2pixHD normalization

Figure 14 shows the one random input image and Figure 15 shows the corresponding real image to input image shown in Figure 14.

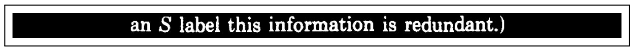
an *S* label this information is redundant.)

Fig. 14. One random input sample from UWIII data set.

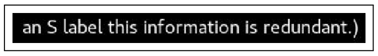
an S label this information is redundant.)

Fig. 15. Ground truth image of the input image shown in Figure 14. It has been generated using Pango.

Figure 16 shows the output of the pix2pixHD network on the input image. The initial epoch has shown promising result as we can see that the characters are taking shape and white pixels are around the original text as shown in the Figure 15.

The Figure 17 shows the input image to the model trained at 75th epoch and The Figure 18 shows the real image generated

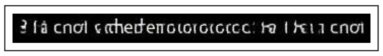
3 lâ cnol cchedenoιoιoιccc: ɦ 1 ʞι.ι cnoι

Fig. 16. Synthetic image generated by pix2pixHD network. Initial result is good. We can see few characters being formed across the width of the real image (Figure 15).

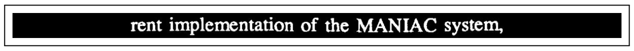
rent implementation of the MANIAC system,

Fig. 17. One random sample image from UWIII data set.

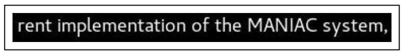
rent implementation of the MANIAC system,

Fig. 18. Ground truth image of the input image shown in Figure 17

by Pango using the text corresponding to the image shown in Figure 17.

The output image of the pix2pixHD network is shown in the Figure 19 for the images shown in Figure 17 and 18. More characters are taking shape and are getting more clear.

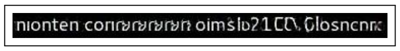
nionten conrrrrrrn oimslb?1CC\.Closncnr

Fig. 19. Output image to the image shown in Figure 17 generated by the pix2pixHD network. Characters are becoming more clear.

VII. Conclusion and Future Work

In this paper, we have tried to use GANs to generate text line image. Normal GANs did not show promising results but they were able to learn the width of the text and in one of the settings, the best result achieved was the black pixels aggregated around the character separated by white pixels. Once, these settings were not giving further good results, we tried to replicate Jenckel et al's [7] work in TensorFlow instead of PyTorch. That result was worse wherein only background pixels occupied almost all the available space of the image without showing any signs of forming characters.

Lastly, we have tried using the normalization approach to achieve the above mentioned motivation. We took the text of UWIII data set [10], generated synthetic image using Pango [5] library in Python and used this synthetic image as ground truth and original image of the data set as input image to train pix2pixHD network [13]. The idea was to transform the input image, which is given to OCR engine for text extraction, to one font image using pix2pixHD, e.g. Arial and also transform the OCRed text from the OCR engine to the same font image, Arial in this example by using Pango library. The output of pix2pixHD and Pango library can be compared, as both of them have the same font. Sum of Absolute Difference (SAD) can be used to find the difference between the generated text (OCRed text) and the text present in the input image. The initial iterations of training gave good results for normalization approach.

In the last setting, the information about the fonts is lost. For example, in the original image if a font is bold or italic is not captured when you get the text from the OCR engine.

When the above pipeline is used, the generated image will have characters in single font without corresponding characters being bold or italic. Such characters cannot be compared.

In future, in order to preserve the character level information, we can use one classifier which can give meta information (bold or italic) about every character in a sentence. While converting the OCRed text into image, these meta information can be used to generate bold characters for characters which were bold in original image and generate italic characters for the characters which were italic in the original image. With this method, we can generate an image from text which will be close to the input image and can help in evaluating the OCR model.

REFERENCES

[1] ABBYY FineReader (2018). ABBYY FineReader.*Wikipedia*

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

[3] Breuel, T. M. (2008). The OCRopus open source OCR system. *Proc.SPIE*, pages 6815 6815 15.

[4] Chandel, V. S. (2018). Deep Learning based Text Recognition (OCR) using Tesseractand OpenCV. *Learn OpenCV*.

[5] gnome (2018). Pango. https://www.pango.org/.

[6] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 26722680.

[7] Jenckel, M., Bukhari, S. S., and Dengel, A. (2018). Transcription free lstm ocr model evaluation. In *Proceedings of the 16th International Conference on Frontiers in Handwriting Recognition (ICFHR 2018). International Conference on Frontiers in Handwriting Recognition (ICFHR-2018), The 16th International Conference on Frontiers in Handwriting Recognition, August 5-8, Niagara Falls, New York, United States*. IAPR,IEEE.

[8] Mao, Xudong, et al. ”Least squares generative adversarial networks.” *Proceedings of the IEEE International Conference on Computer Vision*. 2017.

[9] Mirza, Mehdi, and Simon Osindero. ”Conditional generative adversarial nets.” *arXiv preprint arXiv:1411.1784* (2014).

[10] Phillips, I. (1996). Users reference manual for the uw english/technical document image database iii. *UW-III English/Technical Document Image Database Manual*.

[11] Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

[12] University of Wrzburg (2018). ”Narragonien digital”. http://kallimachos.de/kallimachos/index.php/Narragonien.

[13] Wang, T.-C., Liu, M.-Y., Zhu, J.-Y., Tao, A., Kautz, J., and Catanzaro, B. (2018). High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.