Application of Deep Convolutional Generative Adversarial Network for Russian Handwritten Text Recognition

Togzhan Kudaibergen[1], and Mohamed A. Hamada[1]

[1] *International Information Technology University, Manas St. 34/1, Almaty, 050000, Kazakhstan*

Abstract

Deep Convolutional Neural Networks (Deep CNNs) are widely used for text recognition. Currently, there are many automated frameworks for recognizing handwritten text and converting it into digital print. However, most of the research was conducted to provide text recognition primarily in English. Therefore, the aim of this paper is to provide handwritten text recognition model for Russian language. Furthermore, improving the model accuracy through deployment of Generative Adversarial Networks (GANs). Finally, the consecutive objective of this paper is to organize the process of handwritten text recognition. In this article, several types of GAN architectures have been manipulated. also, an experiment was conducted with the generation of synthetic data, using one of the architectures – ScrabbleGAN. Finally, this article concludes with the recognition model performance.

Keywords

AI, ML, DL, CNN, GAN, Russian handwritten text, text recognition

1. Introduction

The complexity of the text recognition task is largely determined by the form in which it is presented for recognition. It should be noted that printed text is characterized by the fact that it is always located on the sheet in even lines, the characters of the text most often have the same height and width within the document in question. In addition, the spacing that exists between letters is clearly distinguishable and most often has the same width within the same text. Thus, the presence of these parameters reduces the complexity of printed text recognition.

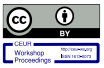
Handwriting recognition is a more complex task, which has not yet been fully resolved. In this regard, the consideration of this topic is considered relevant.

The Russian language (Cyrillic) has a large user base, as it is used throughout the CIS and former post-Soviet countries. The Russian language is written in Cyrillic, there are 33 letters in the language, and these characters are combined when writing, which changes their shape, that is, in a sense, there are ligatures. In Russian, diacritical marks are still considered separately. But there are letters that are ligatures in origin. Thus, the inscription of the letter "щ" reflects the pronunciation "шт" (preserved in modern Bulgarian): this is "ш" placed on the crossbar "т". Subsequently, in both Russian and Bulgarian, the tail of the letter moved to the right. The letter "у" (the old name for "epa") is composed of the sign "ъ" (the so-called hard er), which denoted a reduced labialized sound of the non-front row of the middle rise, ascending to the short Indo-European u, and "i".

Ligatures make it a little harder for optical recognition systems to recognize characters from images. In addition, there are other difficulties in Russian, such as changing the form depending on the context and a set of different styles.

Proceedings of the 7th International Conference on Digital Technologies in Education, Science and Industry (DTESI 2022), October 20–21, 2022, Almaty, Kazakhstan

EMAIL: 33104@iitu.edu.kz (Togzhan Kudaibergen); mohamed.hamada@iitu.edu.kz (Mohamed A. Hamada)
ORCID: 0000-0002-0442-3663 (Mohamed A. Hamada)



Any text written by hand is the characters of a predetermined alphabet of the language and characters that separate these characters. Periods, commas, hyphens, colons, etc. can act as separators. Note that an important property that is characteristic of text in any language is that the differences between the characters of the language are more significant than the differences between different spellings of the same character. Thus, any character of the language can be uniquely identified during recognition.

The handwriting of any person, regardless of the language in which the text is written, consists of strokes arranged in a certain sequence to obtain a single character or letter, and later on a whole word, sentence and text. It is worth noting that any new character, as a rule, begins only after the previous character has been completed. This rule has an exception when writing the characters "й" and "ё", for which the basis of the symbol "й" and "е" is written first, respectively, and only then a stroke for the symbol "и" and two dots for the symbol "е" are added to this basis.

All letters are characterized by the presence of dynamic and static properties. Among the first are the fact that when writing a letter, a person can write strokes in a different sequence, and letters can also consist of a different number of such strokes. However, when writing letters, there are also static properties. The meaning of these properties is that the shape and size of letters do not change within the same language.

Due to the fact that in Russian letters often have connecting lines, as well as random intersections, the recognition task is further complicated by the need to select each character separately in the original image. At the same time, we emphasize that if the image of the handwritten text was obtained by repeated scanning, then defects and inaccuracies may appear, which may adversely affect the recognition process.

Consider what problems you may encounter when recognizing handwritten text:

- the element is not written exactly, has deviations;
- there are no expected intersections within one letter;
- there are intersections with other letters or extra intersections within one letter;
- the presence of a decorative element near the letter;
- the size and position of the elements are different from the expected ones.

Since there are already solutions for recognizing printed Russian, but there are no specific recognizers for handwritten Russian yet. Also, since the Kazakh language is also written in Cyrillic, we will try in the future to create a recognizer for the Kazakh language as well. Written Russian is widely used in the post-Soviet space, since in some regions of Russia itself there is still a huge gap in digital distribution. Many documents and many historical documents are presented in written form along with various business processes. Because of all this, the overall process from public processes to private processes is very slow, as manual verification is required. The digitization of this problem will have a huge impact on the lives of many and can help organizations deliver efficient and fast services. The Russian language has different handwriting styles, so the biggest problem is the different handwriting of different people.

Online handwriting is a form of writing on a numeric keypad using a stylus, while offline handwriting is written on a sheet of paper. Online handwriting recognition has become easier as handwriting information at the pixel level is stored in the computer, which can help identify text as well as eliminate noise. But if you look at Offline Handwriting, then it is written on paper, which does not contain any other information other than the image itself, which makes it much more difficult for a computer to understand it. Moreover, the images contain background noise, which complicates the problem. In this study, we worked with Cyrillic handwritten text written offline, as not everything has been extensively researched. Our ultimate goal is to digitize documents, forms, receipts, etc. in Russian that fall under the umbrella of offline Cyrillic handwriting. Finally, the most difficult aspect of Russian handwriting is the change in form with a change in position or context. Figure 1 shows an example of offline handwriting:

Figure 1: Offline input example [32]

The variety of shapes and types of handwritten characters has meant that algorithms designed to recognize handwritten characters have had less success than algorithms designed to recognize printed characters. Cyrillic character recognition is an important task and an important step in the much more difficult task of recognizing Russian words.

In the modern world, there are already various architectures for recognizing other languages. For example, the algorithm of Graves et al. A Connectionist Temporal Classification (CTC) is used to train a custom network architecture that consists of Convolutional Neural Networks (CNNs) and LSTM layers. OCR4all achieved a mean character error rate (CER) of 0.47 tested on 17 historical English and German manuscripts. While the Calamari architecture is robust, it only supports English and German.

Ahmad et al. experimented with the APTI dataset using an adaptive window to extract features of both characters and words (achieving an error rate of 0.57% for character level and 2.12% for words) for Arabic.

In the field of handwriting synthesis, deep learning and neural networks have helped research a lot. In the research, the scientists used traditional machine learning algorithms to model handwriting, but none of the methods gave good results [17]. Almost all discoveries were made before the advent of neural networks. Professor Graves (2013) showed that by predicting one data point at a time, long short-term memory (LSTM) recurrent neural networks (RNNs) can generate complex sequences with a long structure. This method was first tested for plain text and then extended to handwriting, where the model could predict based on character sequences, resulting in a system capable of producing very realistic handwriting in a variety of styles.

In 2013, scholars Saabni and El-Sana created a system that was efficient and where prototypes for every word in Arabic appeared, using multiple occurrences of each character. Also in 2015, Elarian and others created a system for synthesizing Arabic handwriting where glyphs were combined into words, they used an Extended-Glyph join and a Synthetic-Extension join. Next, we implemented the created synthetic data to improve the results of handwritten Arabic text recognition [42].

In 2017, scientists Wigington and others demonstrated two data augmentation and normalization methods for Latin and French, which, when combined with CNN-LSTM, showed good results. They used random perturbations to complement existing text images, and used a new image normalization method [14].

In 2019, Alonso, Moysset & Messina created an image synthesis system for handwritten Arabic and French words based on generative adversarial networks (GANs). They created a word embedding that was passed to the generator using recurrent LSTM layers. They improved the standard GAN network and added a recognition network to it. Combining all GAN extensions, the network obtained realistic images of handwritten text in Arabic and French [2].

In 2020, Jha & Cecotti showed an approach for handwritten digit recognition based on generative adversarial networks. It has been tested on handwritten Latin, Bangla, Devanagari and Oriya numerals [26].

2. Methodology

Having done research in the field of handwriting synthesis and interested in the idea of generating data using GAN, in this article was made to review several models of GAN networks, and make an experiment by connecting the GAN architecture with an already trained OCR model. As a result of this experiment, we will find out how much the OCR model has been improved and make a comparison with other work.

The first GAN model we'll look at is the TextStyleBrush from the Facebook article (not open source code). The TextStyleBrush model is designed to change the text on the picture to a new one. Since almost all architectures require style markup, you can supply data to TextStyleBrush without markup, that is, an image with a style and new text for replacement are fed to the input. Markup is a sign that determines the ownership of a handwritten text to a particular person [29].

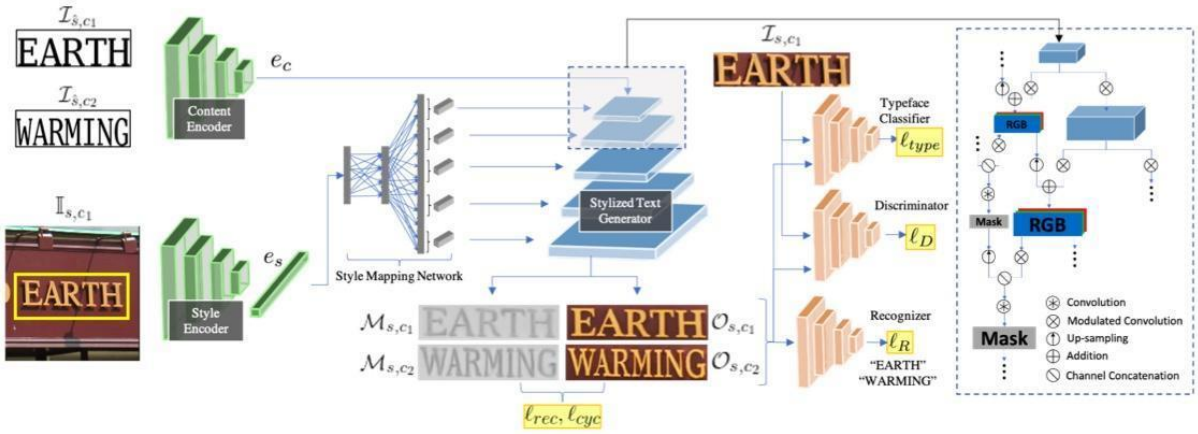


Figure 2: Architecture of TextStyleBrush [29]

The TextStyleBrush model can generate one given style, since it does not accept random noise as input, like other GAN architectures, and you can also get the text style we need as an output. The TextStyleBrush architecture consists of 7 grids (of which 5 are losses).

To train a TextStyleBrush, you need to review the TextStyleBrush model, you can assume that for training you need:

- pretrained model;
- pretrained font classifier;
- according to the authors, to improve the quality, bounding box markup is needed (the input image is cut out + context space around the framing triangle);
- you also need markup for the text itself for images.

The authors of the article have finalized StyleGAN, they have finalized the generator in such a way that it accepts the results of the work of two encoders (text and style).

The generator predicts a text mask on the output. Then it is used in losses, there is no markup for them, the grid itself learns to predict them. They improve the architecture by better separating text from style.

In terms of TextStyleBrush properties, it handles text replacement, usually on billboards, road signs, etc., and handles handwritten text very well (Figure 3).

Figure 3: Example of results (TextStyleBrush) [29]

But it has some drawbacks in the form:

- Due to the complicated architecture of the network, it becomes difficult to change, since it is necessary to change a large number of components in the network later.
- The TextStyleBrush architecture does not know how to create new examples, it only generates the styles, the examples of which are passed to it, that is, one image with the style and the output is a new text with the same style.
- One of the losses is a typed text classifier, that is, this architecture does not work well with handwritten text.

Since there is no open source code in the review article, there is no way to test this model.

The next GANwriting model is the GAN model for generating handwritten pictures. GANwriting works a little differently than TextStyleBrush. The text to be printed is also fed to the GANwriting input, but we submit several images at once, since here we do not replace the text, but create a new font from the image model data [30].

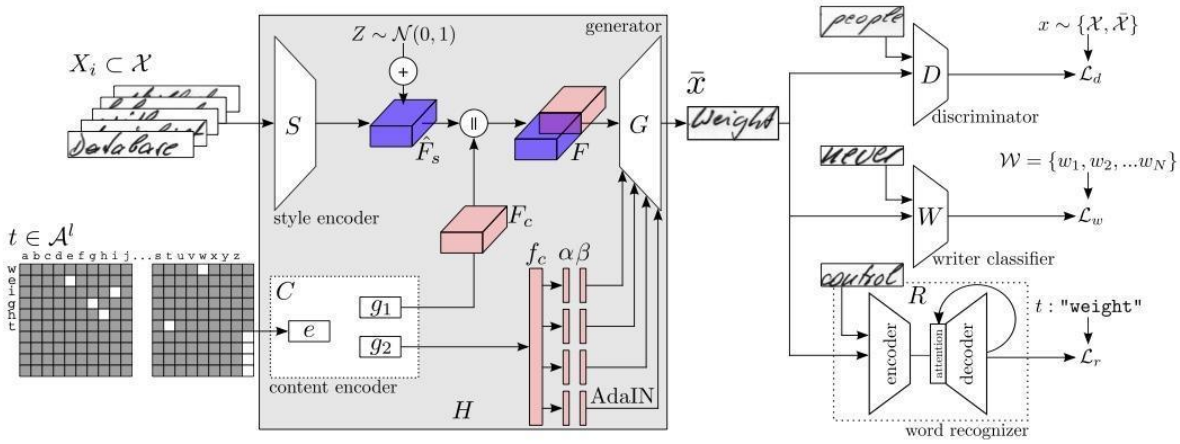


Figure 4: Architecture of GANwriting [30]

The architecture is a bit similar to that of TextStyleBrush, there are encoders for style pictures and text. The Style encoder from the input images produces a tensor that encodes the style.

After the Content encoder receives the text as a one-hot matrix, the encoder is divided into two heads g1 and g2. The output of g1 is connected to the output of the style encoder and such a combined tensor is already fed into the input of the generator, which then upscales it into the resulting image. The output

content vectors of the g2 head are fed to the generator at its four levels in the AdaIN layers (whereas in the TextStyleBrush the style was passed into the generator).

The discriminator, style classifier and OCR model act as losses. writer classifier gives the generator additional information about handwriting features. The GANwriting architecture requires an author markup dataset for each image. Pictures should be marked in advance during data collection, since in the process it will no longer be possible to understand who the author of the images is.

The developers adopted a learning-from-scratch approach, as it gave better results than using pre-trained meshes of architecture parts.

Examples of generation were given, where the input parameters were text or images that were not used during model training. This can be called an advantage of the GANwriting architecture, since it can be used to create synthetic datasets, while having several examples of styles (without the need for additional training).

Also, the GANwriting architecture can mix the styles of different people, that is, using different handwriting for entry, we can diversify the synthetically obtained dataset. Below in the figure you can see how the handwriting changes (Figure 5).

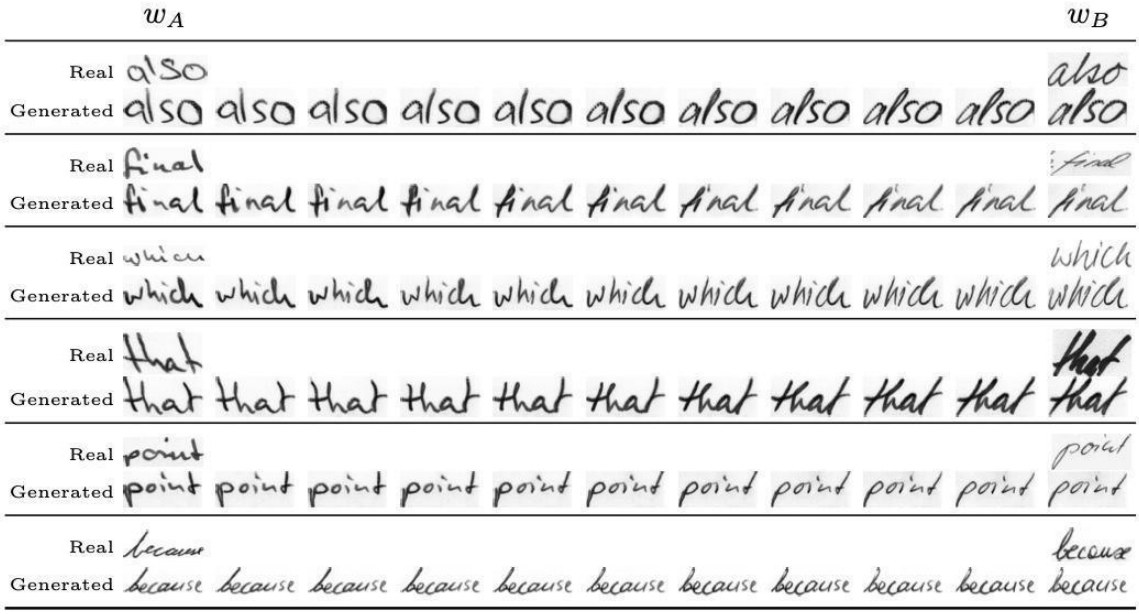


Figure 5: Example GANwriting model Results [30]

The third architecture is ScrabbleGAN. In 2020, Amazon released an article about ScrabbleGAN. The ScrabbleGAN architecture itself is simpler, which we discussed above. The architecture consists only of a generator, a decriminator and OCR, that is, text is fed into the input, and the output is an image. ScrabbleGAN only needs images with text and annotations to it (does not require a font classifier) [19].

A little about the architecture: one-hot text is multiplied by a random noise vector, goes through linear layers and then upscaled by a generator (BigGAN model). The noise vector is responsible for the style of the letters: handwriting / thickness / italics and so on. On inference, if you do not change the noise, then the pictures, respectively, will all be in the same style. And the main disadvantage of ScrabbleGAN: the generation style can only be influenced by the random noise vector.

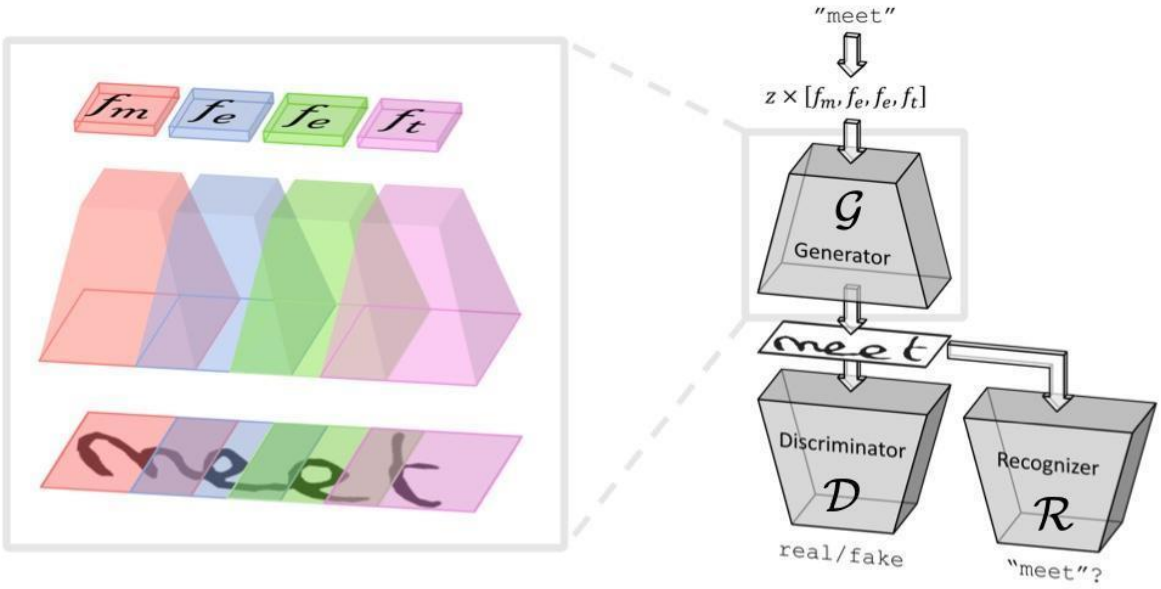


Figure 6: Architecture of ScrabbleGAN [19]

Next, the discriminator takes the generated image as input, which improves the image quality, then passes it to the OCR recognizer.

Due to the fact that recurrent layers can learn the implicit language model of the dataset and at the same time predict the correct option, while it is incorrect in the image, the authors of the article subtracted all other layers and left only the convolutional layers. Here, the OCR model should read exactly the text that the generator generated, without thinking it through. With recurrent layers in the experiments worsened the overall results.

Below in the picture (Figure 7) you can clearly see the training results for different coefficients alpha (□, is responsible for the weight of OCR-loss during training). GAN training without a discriminator case with □=□ (left side column), GAN training without a recognizer (right side column) with □=0 - all pictures look real, but there is no clear text everywhere [19].

Figure 7: Example of ScrabbleGAN+ OCR generation. All examples are generations of word "ScrabbleGAN" [19]

The authors of the article [2] conducted an experiment where they trained an OCR model on an IAM dataset and tested it on CVL (due to the difference in dataset domains, low recognition quality was obtained during testing).

For training, a synthetic dataset from ScrabbleGAN was added to the dataset, while the ScrabbleGAN discriminator predicting only two classes - real / fake - can be separately trained on unlabeled data. And you can also retrain on CVL styles, then in the generated images we will reproduce the style of the CVL dataset.

The picture below shows the learning outcomes:

Train data	Style	Lex.	WER[%]	NED[%]
IAM (naive)	N/A	IAM	39.95 ± 0.91	19.29 ± 0.95
IAM+100K	CVL	IAM	40.24 ± 0.51	19.49 ± 0.76
IAM+100K	IAM	CVL	35.98 ± 0.38	17.27 ± 0.23
IAM+100K	CVL	CVL	29.75 ± 0.67	$\mathbf{14.52 \pm 0.51}$
CVL (oracle)	N/A	CVL	$\mathbf{22.90 \pm 0.07}$	15.62 ± 0.15

Figure 8: OCR+ScrabbleGAN training results on IAM dataset and CVL dataset [19]

To begin with, the OCR model was trained only on the IAM dataset without synthetic data - then the results gave 40% errors on the CVL dataset. The addition of synthetic data from ScrabbleGAN reduces the error rate by up to 30%. ScrabbleGAN has been retrained on unlabeled CVL data. It is important to choose the right data vocabulary when teaching OCR, that is, in this situation, you can get good results. Since when training a model for medical purposes it is better to use vocabulary with medical terminology, in the end the model will give good results. Further, 23% WER gives an OCR that is trained directly on the CVL dataset - this experiment with CVL data was carried out in order to see the best result of the overall experiment.[19]

3. Experimental part

3.1. Dataset

In this article, for experimental purposes, Handwritten Kazakh and Russian (HKR) database for text recognition for the train and Forms dataset for tests will be used. These two datasets are different in style, handwriting and vocabulary. Forms - these are photographs of filling out various forms, and the texts there, respectively, are formal (names of cities, postal codes, names, dates, times, etc.). Whereas in HKR the lexicon is more “usual”, literary. The main goal of the experiments is to check that handwriting generation using GANs will increase the quality of OCR models on new domains. Also, has been added simple synthetic data to the train dataset (text based on ttf fonts. But such fonts are usually printed, and those that are handwritten are more likely to imitate italics. However, for comparison, it has been decided to train on such data. Below are examples from datasets, the first column is the HKR dataset, the second column is the Forms dataset, and the third column is the dataset based on ttf fonts (Figure 9).

3.2. Results of the experiment

An experiment was conducted where the OCR model was trained on synthetic data from ScrabbleGAN.

Below in the picture you can see the results of the generated data from ScrabbleGAN, which was trained on the Russian-language HKR dataset (Figure 10). As you can see, the handwriting has been slightly modified depending on the random noise vector (synthetic data is on the right side of Figure 10).

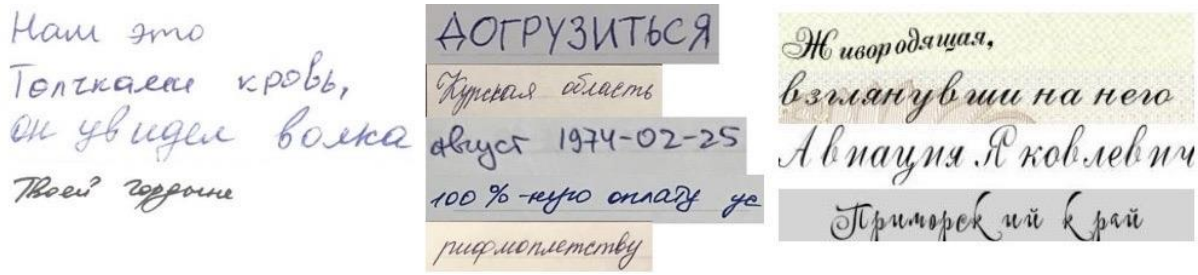


Figure 9: Examples of used datasets

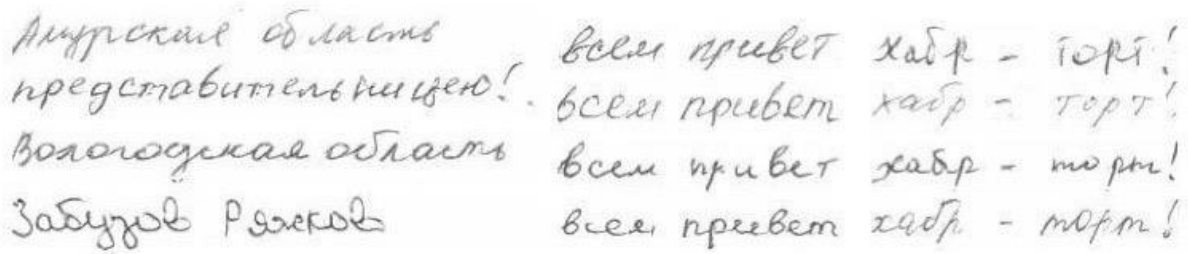


Figure 10: Results of the ScrabbleGAN model trained on the Russian language HKR dataset

Table 1
Results of experiments

Train dataset parameters	Accuracy, test Forms dataset
HKR (naive)	0,5%
HKR + TTF	13,9%
HKR + GAN	12,3%
HKR + TTF + GAN	24,1%
HKR + Forms (for max quality understanding)	66,5%

On the first line of Table 1, the results of the model, where it was trained only on the Russian-language HKR dataset and on the Forms test gives very low results - 0.5% (accuracy is when the predicted text for the line completely matches the target). Such low results are understandable, since the datasets are different both in style and in lexicon.

Further, on the second line of the results of the model, where synthetic data on ttf fonts was added, here the accuracy increased to 13.9%. But if you look at the fourth line, where images from ScrabbleGAN were added to the training sample, the accuracy increased to 24.1%. That is, the addition of synthetic data during training gives us an improvement in the quality of the OCR model when reading text. At the same time, the increase in accuracy when adding synth from GAN is comparable to the usual synth on ttf fonts: 9–13% accuracy on new domains. When combining synthetics from GAN and based on ttf fonts, the quality of the OCR model on new domains increases even more - up to 23-24%.

4. Conclusion

The use of GAN-based technologies is becoming more popular among AI scenarios, as they have more predictive and data generation capabilities. It is worth noting that the synthetic GAN data looks realistic, thus does not greatly affect the overall accuracy of the OCR model. However, it is not entirely true that the amount of data used is optimal for training the model, this requires even more detailed experiments. Therefore, you should carefully and with attention to the data for training the model. The models used have more value on the training data.

In addition, we considered three GAN architectures for generating handwritten images and made an experiment using one of them - ScrabbleGAN - on Russian-language datasets. For comparison, we also used a dataset based on ttf fonts. And we conducted an experiment to understand how synthetic data

from GANs affect recognition. As a result, the model works best when combining all three types of synthetic data.

5. References

[1] A. Abdallah, M. Hamada, and D. Nurseitov, Attention-based fully gated cnn-bgru for russian handwritten text, Journal of Imaging 6.12 (2020) 141.

[2] A. Graves et al., Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks, in: Proceedings of the 23rd international conference on Machine learning, 2006, pp. 369–376.

[3] E. Alonso, B. Moysset, R. Messina, Adversarial generation of handwritten text images conditioned on sequences, in: Proceedings of the 2019 International conference on document analysis and recognition, ICDAR, IEEE, 2019, pp. 481-486.

[4] A. D. Le, H. T. Nguyen, and M. Nakagawa, Recognizing unconstrained vietnamese handwriting by attention based encoder decoder model, in: Proceedings of the 2018 International Conference on Advanced Computing and Applications, ACOMP, IEEE, 2018, pp. 83–87.

[5] R. Ahmad, et al., A deep learning based arabic script recognition system: benchmark on KHAT, Int. Arab J. Inf. Technol, T.17.3 (2020) 299-305.

[6] A. Vaswani et al., Attention is All You Need, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010. ISBN: 9781510860964.

[7] A. Graves, Generating sequences with recurrent neural networks, arXiv preprint arXiv:1308.0850, 2013.

[8] Amazon Inc., Amazon textract. URL: https://aws.amazon.com/textract, 2019-11-01.

[9] A. Brock, J. Donahue, and K. Simonyan, Large scale gan training for high fidelity natural image synthesis, arXiv preprint arXiv:1809.11096, 2018.

[10] A. Poznanski and L. Wolf. Cnn-n-gram for handwriting word recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, IEEE, 2016, pp. 2305–2314.

[11] A. Odena, C. Olah, and J. Shlens, Conditional image synthesis with auxiliary classifier gans, in: ICML, 2017.

[12] A. Brock, J. Donahue, and K. Simonyan, Large scale gan training for high fidelity natural image synthesis, arXiv preprint arXiv:1809.11096, 2018.

[13] B. Ji and T. Chen. Generative adversarial network for handwritten text, arXiv preprint arXiv:1907.11845, 2019.

[14] C. Wigington, S. Stewart, B. Davis, B. Barrett, B. Price, and S. Cohen, Data augmentation for recognition of handwritten words and lines using a CNN-LSTM network, in: ICDAR, 2017.

[15] D. Bahdanau, K. Cho, and Y. Ben- gio, Neural Machine Translation by Jointly Learning to Align and Translate, in: Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. Ed. by Y. Bengio and Y. LeCun, 2015. URL: http://arxiv.org/abs/1409.0473.

[16] E. Aksan, F. Pece, and O. Hilliges, Deepwriting: Making digital ink editable via deep generative modeling, in: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, 2018, pp. 1–14.

[17] M. Eltay et al., Improving Handwritten Arabic Text Recognition Using an Adaptive Data-Augmentation Algorithm, in: Proceedings of the International Conference on Document Analysis and Recognition, Springer, Cham, 2021, pp. 322-335.

[18] E. Alonso, B. Moysset, and R. Messina, Adversarial generation of handwritten text images conditioned on sequences, arXiv preprint arXiv:1903.00277, 2019.

[19] S. Fogel et al., Scrabblegan: Semi-supervised varying length handwritten text generation, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 4324-4333.

[20] G. Huang et al., Densely connected convolutional networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708.

[21] Google Inc., Detect text, URL: https://cloud.google.com/vision/docs/ocr, 11-01. 1 images, 2019-9.
[22] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, Sh. Ozair, A. Courville, and Y. Bengio, Generative adversarial nets, in Advances in neural information processing systems, 2014, pp. 2672–2680.
[23] J. Baek, G. Kim, J. Lee, S. Park, D. Han, S. Yun, S. J. Oh, and H. Lee, What is wrong with scene text recognition model comparisons? dataset and model analysis, 2019.
[24] J. Hyun Lim and J. Chul Ye, Geometric gan, arXiv preprint arXiv:1705.02894, 2017.
[25] J. Puigcerver, Are multidimensional recurrent layers really necessary for handwritten text recognition? In: Procedings of the 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), IEEE, vol. 1, 2017, pp. 67–72.
[26] G. Jha, H. Cecotti, Data augmentation for handwritten digit recognition using generative adversarial networks, Multimedia Tools and Applications 79.47 (2020) 35055-35068.
[27] L. M. Lorigo, V. Govindaraju, Offline Arabic handwriting recognition: a survey, IEEE transactions on pattern analysis and machine intelligence, 28.5 (2006) 712-724.
[28] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556, 2014.
[29] P. Krishnan et al., Textstylebrush: transfer of text aesthetics from a single example, arXiv preprint arXiv:2106.08385, 2021.
[30] L. Kang et al., GANwriting: content-conditioned generation of styled handwritten word images, in: Proceedings of the European Conference on Computer Vision, Springer, Cham, 2020, pp. 273-289.
[31] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, Gans trained by a two time-scale update rule converge to a local nash equilibrium, Advances in Neural Information Processing Systems (2017) 6626–6637.
[32] M. A. Hamada, K. Sultanbek, B. Alzhanov, and B. Tokbanov, Sentimental text processing tool for Russian language based on machine learning algorithms, in: Proceedings of the 5th International Conference on Engineering and MIS, ICEMIS '19, Association for Computing Machinery, New York, NY, USA, Article 37, 2019, pp. 1–6.
[33] M. Mirza and S. Osindero, Conditional generative adversarial nets, arXiv preprint arXiv:1411.1784, 2014.
[34] P. Krishnan and C. V. Jawahar, Generating synthetic data for text recognition, 2016.
[35] R. Reeve Ingle, Y. Fujii, T. Deselaers, J. Baccash, and A. C. Popat. A Scalable handwritten text recognition system, arXiv:1904.09150, 2019.
[36] R. Ali, U. Farooq, U. Arshad, W. Shahzad, and M. O. Beg, Hate speech detection on twitter using transfer learning, Computer Speech & Language 74 (2022) 101365, 2022.
[37] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, Generative Adversarial Text to Image Synthesis, in ICML, 2016.
[38] S. Hochreiter and J. Schmidhuber, Long Short- Term Memory, in: Neural Comput. 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. doi: 10.1162/neco. 1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997. 9.8.1735.
[39] S. Liyang, Crnn-pytorch, URL: https://github.com/- Holmeyoung/crnn-pytorch, 2019-11-01.
[40] M. D. Shaiq, M. D. A. Cheema, A. Kamal, Transformer based Urdu Handwritten Text Optical Character Reader, arXiv preprint arXiv:2206.04575, 2022.
[41] S. Ahmed, S. Naz, S. Swati, and M. Razzak, Character recognition using 1-dimensional blstm classifier, Neural Computing and Applications 31 (2019) 04.
[42] R. M. Saabni, J. A. El-Sana, Keywords image retrieval in historical handwritten Arabic documents, Journal of Electronic Imaging 22.1 (2013) 013016.
[43] T. S. F. Haines, O. M. Aodha, and G. J. Brostow, My Text in Your Handwriting, Transactions on Graphics, 2016.
[44] T. Bluche and R. Messina, Gated convolutional recurrent neural networks for multilingual handwriting recognition, in: 2017 14th IAPR International Conference on Document Analysis and Recognition, ICDAR, IEEE, vol. 1, 2017, pp. 646–651.
[45] 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 (2002) 39–46.