applied sciences

MDPI

Article

Improving OCR Accuracy for Kazakh Handwriting Recognition Using GAN Models

Arman Yeleussinov [1], Yedilkhan Amirgaliyev [2] and Lyailya Cherikbayeva [1,2,*]

[1] Faculty of Information Technology, Department of Computer Science, Al Farabi Kazakh National University, Almaty 050010, Kazakhstan
[2] Institute of Information and Computational Technologies, Almaty 050010, Kazakhstan; amir_ed@mail.ru
* Correspondence: cherikbayeva.lyailya@gmail.com; Tel.: +7-(778)-220-63-64

Abstract: This paper aims to increase the accuracy of Kazakh handwriting text recognition (KHTR) using the generative adversarial network (GAN), where a handwriting word image generator and an image quality discriminator are constructed. In order to obtain a high-quality image of handwritten text, the multiple losses are intended to encourage the generator to learn the structural properties of the texts. In this case, the quality discriminator is trained on the basis of the relativistic loss function. Based on the proposed structure, the resulting document images not only preserve texture details but also generate different writer styles, which provides better OCR performance in public databases. With a self-created dataset, images of different types of handwriting styles were obtained, which will be used when training the network. The proposed approach allows for a character error rate (CER) of 11.15% and a word error rate (WER) of 25.65%.

Keywords: optical character recognition; generative adversarial network; character error rate; word error rate

Citation: Yeleussinov, A.; Amirgaliyev, Y.; Cherikbayeva, L. Improving OCR Accuracy for Kazakh Handwriting Recognition Using GAN Models. *Appl. Sci.* **2023**, *13*, 5677. https://doi.org/10.3390/app13095677

Academic Editor: Dariusz Frejlichowski

Received: 28 March 2023
Revised: 28 April 2023
Accepted: 2 May 2023
Published: 5 May 2023

1. Introduction

A large number of manuscripts have been collected over the years and continue to be written today. They have many applications and require automatic processing. Handwritten texts are still widely used in many fields, such as healthcare, library work, personnel management, archives, and other fields. The need to make them available in modern information exchange and search systems is growing. Automation greatly reduces the labor involved in processing correspondence and surveys, as well as transcribing historical manuscripts. The current OCR seems to be advanced enough to handle printed text [1–3], but the handwriting recognition system (HTR) still needs improvement.

The differences between handwritten Cyrillic Kazakh and handwritten Latin Kazakh are less pronounced than the differences between the printed versions. However, there are still some differences in the way the characters are written. Handwritten Cyrillic Kazakh tends to be more angular and blocky, while handwritten Latin Kazakh is more cursive and rounded. In recent years, there has been a movement to switch from Cyrillic Kazakh to the Latin script. The Kazakh government has set a goal of fully transitioning to the Latin script by 2025, citing the Latin script's familiarity and ease of use as reasons for the change [4,5].

While there is a movement to transition to the Latin script, Cyrillic Kazakh will continue to be used for some time and is an important part of the Kazakh cultural heritage.

Handwritten text has several characteristics related to variability both in class and outside of class: different instances of the same word written by different people may look different, while the same character written by the same author may look different. It varies depending on the context in which it is written. We attribute this gap to the lack of fully annotated manuscripts and the difficulty in obtaining them. In this article, we attempt to fill this gap by creating a general-sized visual text, reducing the need for annotations, and enriching the diversity of training data in both style and vocabulary.

The Generative Adversarial Network (GAN) is an effective way to solve this problem. GANs offer the right way to learn deep concepts without extensive use of labeled learning data. This approach has caught the attention of many computer vision researchers because it allows them to generate large amounts of data without accurately changing the probability density function (PDF). In GAN, the generative model is evaluated using a competitive process in which generating and discriminating networks are trained simultaneously. The generator learns to extract reliable data, and the discriminator learns to distinguish fake data generated by the generator from real data patterns. Considering the fast growth of GANs over the past few years and their use in various fields, these networks need to be carefully studied.

This article presents the main concepts of GAN, compares several models of deep SOTA generation, and explains the evaluation indicators used in the literature and the problems associated with GAN. Moreover, the most notable GAN architectures are classified and discussed.

2. Materials and Methods

Generative adversarial networks (GANs) are part of the group of generative models. Generative models attempt to learn a probability density function from a training set and then derive new models from that distribution. GAN created new data in a synthetic way, similar to real data, by comparing two neural networks (a generator and a discriminator). The generator tries to determine the actual distribution of the data to create new models [6]. In [7], the authors show how generative modeling using GAN architectures can be used to reproduce small galaxy image datasets.

GAN's field of application has also expanded. In paper [8], an overview of GAN and applied solutions are presented that are interesting for researchers in the fields of computer vision and artificial intelligence in relation to healthcare tasks. Ref. [9] offered GAN analysis and applications that are important for computer vision and artificial intelligence researchers in an applied form. Examples of the use of the GAN are given, such as image classification and transformation, image fusion and image insertion, image-to-image conversion, maximum resolution, and point capture.

Recently, there has been a lot of literature describing various methods related to deep learning that have shown good results in image segmentation. One of the most promising directions in this field is the use of a GAN, in which an image generator of handwritten words is constructed. In the work [10], the problem of generating plausible (difficult to separate from real) X-ray images of a normal human chest is considered. This problem is solved using generative-adversarial neural networks (generative adversarial nets). The degree of plausibility of the results obtained is assessed both visually and quantitatively by comparing image structure descriptors based on local binary patterns.

Data collection is a complex and expensive process, and subsequent labeling becomes even more difficult. One affordable way to reduce the burden of annotating data is through supervised learning. In addition to labeled data, semi-tracked methods use some unlabeled patterns to improve performance over fully-tracked methods. Thus, these methods can adapt to patterns that are not visible during testing. The paper [11] introduced scrabblegan, a semi-supervised approach to synthesizing handwriting images common in both style and vocabulary. The scrabble game is based on a new generative model that allows the creation of word images of any length [12]. Handwritten text recognition (HTR) has been proposed that outperforms current methods. The results were compared with the three most commonly used datasets in HTR tasks: Ben-tham, IAM, and Saint Gall. In addition, results were presented for two recently introduced datasets: the Peter the Great manuscripts and the HKR dataset. The neural network architecture allows for the handwriting recognition tasks of modern and historical documents in different languages. In our paper, consider the recognition task—partially supervised learning. There are different approaches and algorithms to solve the problem of partially supervised learning. In this problem, only a part of the objects in the original sample have known class labels; it is required to classify

either the existing unlabeled objects or to form a decisive rule for the recognition of new objects. This problem is relevant for the following reasons: As a rule, unlabeled data are "cheap" (in the case where determining the class to which objects belong is a costly procedure); using unlabeled data together with labeled data allows one to attract additional information, which can significantly increase the level of learning [13].

The work is aimed at improving the accuracy of handwritten text recognition in the Kazakh language. Recently, the scientific works of many scientists have been published aimed at handwriting recognition and signature verification. In the paper [14], the authors propose a decision support system for the forensic analysis of dynamic signatures, while [15] other scientists propose a two-level ensemble approach for online verification of the signature, depending on the author. In [16], the authors propose an online signature verification system based on the extraction of composite features and shared convolution by depth. The methods used in these works vary, but all of them are based on machine learning algorithms to improve the accuracy of handwriting recognition and signature verification.

In the paper [17], the author presents the architecture of a handwriting recognition neural network model that can be trained to recognize complete handwritten or typed text without segmentation of the image. Based on the image-to-sequence transformation architecture, however, it is able to retrieve the texts present in the image and then position them correctly with no restrictions on the focus, location, or size of the text elements.

Image recognition plays a vital role in different areas, such as gesture recognition [18], autonomous tomato harvesting [19], medicine issues [20], image steganography [21], and remote sensing [22]. Image recognition and handwriting recognition neural networks face challenges related to the variability and complexity of the input images. To address these challenges, researchers often use similar techniques in both image and handwriting recognition, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs).

The article proposes a reconstruction of floor plans from point clouds within rooms, and the method employs a generative adversarial network in order to study a complex distribution of floor plan schedules and transform partial room masking into more regular segmentation areas [23].

The aim of the study is to prepare a model for the generation of handwritten text in the Kazakh language and to calculate its metrics. This model is used to improve the accuracy of the handwritten text recognition model.

To achieve the goal, the following objectives were formulated:

- To train a generative adversarial network with different loss functions, experimental determination hyperparameters, and formulation;
- To compare CER and WER metrics of handwritten text recognition models after training with generated data.

2.1. Loss Functions of GAN Models

The loss functions express the discrepancy between the trainable model's predictions and the actual problem instances. If the deviation between the predicted result and the actual result is too large, then the loss function will have a very high value. Gradually, with the help of some optimization functions, the loss function learns to reduce the prediction error.

Typeface classifier loss is a font classifier pre-trained on synthetic images and frozen during the training of the entire architecture. The classifier gives the generator gradients during training to better understand the essence of fonts. This loss aims to reduce the differences in font between the generated and target images.

Discriminative loss. While the discriminator is trained, it classifies both the real data and the fake data from the generator. It penalizes itself for misclassifying a real instance as

fake or a fake instance (created by the generator) as real by maximizing the below function.

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D(x^{(i)}) + \log(1 - D(G(z^{(i)}))) \right] \quad (1)$$

log(D(x)) refers to the probability that the generator is correctly classifying the real image; maximizing log(1 − D(G(z))) would help it correctly label the fake image that comes from the generator. ∇ is the stochastic gradient, θ_d is a hyperparameter of a multilayer perception that represents a differentiable function $G(z; \theta_d)$ that maps input noise z to data space.

Generator Los. When the generator is trained, it receives random noise and generates the output of that noise. The output then passes through a discriminator and is classified as "true" or "false", depending on the discriminator's ability to distinguish one from the other. The generator loss is then calculated based on the discriminator's classification; if it effectively fools the discriminator and is otherwise penalized, it is rewarded. The next equation boils down to training the generator.

log(D(x)) refers to the probability that the generator is correctly classifying the real image and maximizing log(1 − D(G(z))) would help it correctly label the fake image that comes from the generator.

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(z^{(i)})) \quad (2)$$

Category loss: To create high-quality images of fonts, it is important that models know not only their own styles but also the styles of other fonts. Thus, models must be able to learn multiple font styles simultaneously, so registering a symbol before going to the decoder uses categorical embedding by combining a unique Gaussian noise as a style embedding. Added the ability to lose multiple class categories to control the discriminator so that models do not mix styles with each other or create characters other than those represented, allowing you to predict the styles of created characters.

Pixel-wise loss: the most commonly used pixel loss for generated images and real images is L_2 and L_1 distances [24]. Distance L_1, not L_2, because L_1 reduces blurring. The formulation is below:

$$\Re_{L1} = E_{x \in p_{data}, z \in p_{input}} \parallel x - G(z) \parallel_1 \quad (3)$$

2.2. Evaluation Metrics of GAN Algorithms

GAN algorithms are evaluated using a variety of metrics, including the Inception Score (IS), the Frechet Inception Distance (FID), the Structural Similarity Index (SSIM), and the Fréchet Distance (FD) [25,26]. The IS measures the quality of the generated images, while the FID and the SSIM measure the similarity between the real and generated images. The FD measures the similarity between the distributions of real and generated images. Additionally, GAN algorithms can be evaluated using user studies or metrics such as the Human Visual System (HVS) or classification accuracy.

The Inception Score (IS) is a metric that measures the quality of generative models, specifically synthetic images output by generative adversarial network models. It is based on a convolutional network and is calculated by taking the mean of the KL-divergence [27] between the predicted label distributions for a set of generated images and the label distributions for the ground truth. The quantitative evaluation of IS is calculated with the equation

$$I = \exp(E_x D_{KL}(p(y|x)||p(y))) \quad (4)$$

where x denotes one generated sample and y is the label predicted by the model.

The Frechet Inception Distance (FID) is a metric used to measure the similarity between two sets of images. The FID score is calculated by comparing the statistics of the feature representations of the two image sets. The lower the FID score, the more similar the two image sets are considered to be. Frechet distance is used to compute the distance

between two "multivariate" normal distributions. For a "univariate" normal distribution, the Frechet Distance is given as,

$$d(X, Y) = (\mu_X - \mu_Y)^2 + (\sigma_X - \sigma_Y)^2 \quad (5)$$

where μ and σ are the mean and standard deviation of the normal distributions, and X and Y are two normal distributions. In common, the FID metric is the squared Wasserstein metric between two multidimensional Gaussian distributions: N(μ, Σ), the distribution of some neural network features of the images generated by the GAN, and N(uw, Σw), the distribution of the same neural network features from the "world" or real images used to train the GAN:

$$\text{FID} = \| \mu_X - \mu_Y \|^2 - T_r\left(\sum_X + \sum_Y - 2\sqrt{\sum_X \sum_Y}\right) \quad (6)$$

where X and Y are the real and fake embeddings assumed to be two multivariate normal distributions. μ_x and μ_y are the magnitudes of the vectors X and Y. Tr is the trace of the matrix and ∑X and ∑Y are the covariance matrix of the vectors.

The Structural Similarity Index (SSIM) is an index designed to assess visual differences between people instead of numerical errors. This is because human vision specializes in extracting structural information from images, so distortion of structural information has a large impact on perception. Using this approach, image quality is evaluated based on three variables: luminance, contrast, and structure. These three methods are based on the fact that they have the greatest impact on the human visual system [28]:

$$l(x, y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \quad (7)$$

$$c(x, y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \quad (8)$$

$$s(x, y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \quad (9)$$

where l(x, y), c(x, y), and s(x, y) represent brightness, contrast, and the structure of an image, respectively. C_1, C_2, and C_3 are weighting constants to prevent denominators from being zero. C_1, C_2, and C_3 are weighting constants to prevent denominators from being zero. It is in general expressed as $C_1 = (K_1L)^2$ and $C_2 = (K_2L)^2$, and the nominal values are given by $K_1 = 0.01$, $K_2 = 0.03$, and $L = 255$, respectively. Moreover, μ and σ denote the mean value and standard deviation defined as follows, respectively:

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i \quad (10)$$

$$\sigma_x = \left(\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2\right)^{\frac{1}{2}} \quad (11)$$

where N is the overall number of pixels, and μ_x and σ_x are the average value and standard deviation of x, respectively. In addition, σ_{xy} is the variance of the covariance between x and y, determined as follows:

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y) \quad (12)$$

Note that $C_3 = C_2/2$. Then, the structural similarity index measure is defined as follows:

$$\mathrm{SSIM}(x, y) = l(x, y) \times c(x, y) \times s(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad (13)$$

3. Results

3.1. Kazakh Handwritten Dataset Description and Presettings

For evaluation of the proposed approach, we use the dataset that was previously collected and annotated and was presented in the paper [29]. Handwritten texts were collected from 135 local authors. To collect images, authors are invited to write texts in Kazakh from various sources. Collected handwritten images were scanned and saved in PNG format. In this work, we used a word-labeled annotation format where words were cropped by their bounding boxes (Figure 1).

Figure 1. Example of handwritten text with their annotations: bounding boxes—coordinates of words, numbers—IDs of words.

Our strategy for dividing the data set into training and test sets was to randomly select rows from each page. 80% was allocated to training and 20% to testing. This method of randomization generates a good variety of training data and, therefore, a good representation of the data, which is important for model building. The total training sample was 3425 lines containing 45,211 words and 234,576 characters, and the test sample was 685 lines containing 12,312 words and 77,713 characters.

The experiment was conducted on a PC with an Intel Core i7 12.90 GHz × 16, a NVIDIA RTX3060 12 GB GPU, and 32 GB of memory. The PC runs the Ubuntu 20.04.5 LTS operating system, and the deep learning framework used is PyTorch 1.9.0.

3.2. Parameters of Proposed Models

In our experiments, we use the three models with different loss functions and evalute them using CER and WER metrics.

Experiment 1. In this experiment, we train the TextStyleBrush model [30]. The model consists of 7 layers, including 5 loss grids. The content encoder and style encoder are both ResNet34 backbones. Moreover, the text is not presented in the form of encoded letters but simply as a separate picture; it is printed on a white background using a standard font. Style mapping network—the mesh transforms the style encoder's output vector into many

individual vectors. Then they are fed on different layers to the input of the generator as parameters in the AdaIN layers [31].

$$\mathrm{AdaIN}(z, \alpha, \beta) = \alpha\left(\frac{z - \mu(z)}{\sigma(z)}\right) + \beta \tag{14}$$

where $z \in F$ (the combination of the calligraphic style attributes and the textual content information character-wise), μ and σ are the channel-wise mean and standard deviations. The global content information is injected four times ($p = 4$) during the generative process by the AdaIN layers. Their parameters α and β are obtained by splitting fc into four pairs.

$$x = H(t, X_i) = G(C(t), S(X_i)) = G\left(g_1\left(\tilde{t}\right), g_2\left(\tilde{t}\right), S(X_i)\right), \tag{15}$$

where $\tilde{t} = [e(c); \forall c \in t]$ is the encoding of the string t. This allows the generator to better capture the style of the image at different levels. The generator is StyleGAN [32], which the authors of the article have modified in such a way that it accepts as input the results of two encoders: text and style. The output generator, in addition to the picture, predicts a text mask, which is then used in losses (there is no markup for masks; the grid itself learns to predict them). These masks help the architecture better separate text from background/style. The network architecture seems overly complicated, and if you need to fix something, you will have to tune the mesh to adjust for the large number of components inside. Due to its architecture, TextStyleBrush can only generate styles/handwriting of which there are already examples and cannot create completely new examples or some kind of "averaging" of existing ones. Only one picture as the input style and one picture as the output with the same style and new text. The input generator takes text as a picture printed in a simple font on a white background and then upscales it. Additionally, the classifier of standard printed fonts acts as one of the losses. This architecture is more suited to generating images with printed text than with handwritten text.

Experiment 2. We further train neural networks with an architecture based on GanWriting [32]. The content encoder takes the text as a one-hot matrix (unlike the TextStyleBrush, which rendered the text in a standard font on a white background), then the encoder is divided into two heads, g_1 and g_2. 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 g_2 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).

Experiment 3. In this experiment, we use the architecture of the ScrabbleGAN [11]—a model that is simpler than the ones we have specified before. There is only a generator, discriminator, and OCR. A little about the architecture: one-hot text is multiplied by a random noise vector, passes through linear layers, and is then upscaled by a generator BigGAN model [33]. The noise vector is responsible for the style of writing: handwriting/thickness/italics, and so on. The generated image is fed into the discriminate input, which helps to improve the overall quality of the image, and to a recognizer (OCR), which makes the text readable. The authors of the article store only convolutional layers in OCR models for the reason that recurrent ones can learn the implicit language model of the data set and predict the correct description of words, while in the picture it is written erroneously (implicit language model in LSTM for OCR). This can get in the way when OCR is used as part of the GAN architecture, because here OCR should read exactly the text that the generator wrote without thinking it through. In the table, the results of each model with different losses.

For all models, we use the Adam optimizer with $\beta 1 = 0$ and $\beta 2 = 0.8$ for training. By default, the learning rate for the discriminator is 0.0004, and the learning rate for the generator is 0.0001.

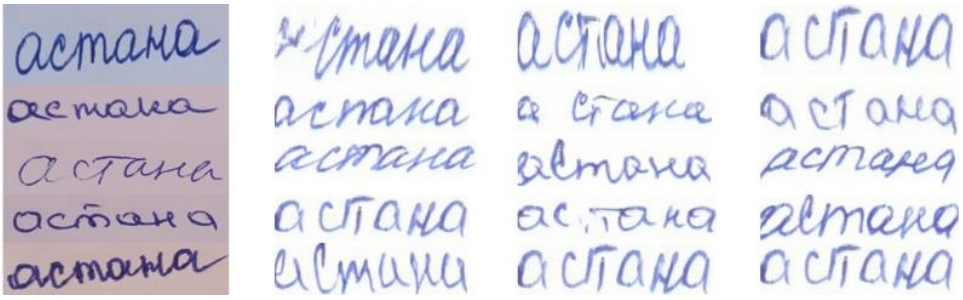
As shown in Table 1, the model ScrabbleGAN achieves the best IS, FID, and SSIM scores among all generative models.

Table 1. Comparing the influence of different loss functions on GAN models' results.

Model	IS	FID	SSIM
TextStyleBrush	35.24	32.11	0.2
GanWriting	49.33	29.23	0.38
ScrabbleGAN	51.21	20.14	0.43

The model significantly improves the best published Inception score from 35.24 to 52.21. We found that different loss functions help improve classifier sensitivity to imperfect GAN outputs. Accordingly, as generative modeling technology advances in the future to generate better reconstructions, similar GAN upscaling and merging strategies may lead to even greater improvements.

From the results of images generated on the TextStyleBrush, GanWriting, and ScrabbleGAN models, it appears that TextStyleBrush generates an image with little handwriting variation, which is very important for the OCR model. GanWriting has difficulty with similar letters such as "м", "ш", "щ". ScrabbleGAN is very good in terms of handwriting variation, but there are certain difficulties with capital letters; this is due to the fact that their ratio was very small in the dataset. In the next experiment, we will use only the ScrabbleGan model for generating images due to the model's good results (Figure 2).



(**a**) (**b**) (**c**) (**d**)

Figure 2. Visual comparison results of GAN models for the Kazakh word: (**a**) real data; (**b**) TextStyle Brush; (**c**) GanWriting; (**d**) Scrabble Gan.

Experiment 4. The purpose of the experiment is to evaluate how our data generation approach will help improve the accuracy of handwriting recognition on real data depending on the quantity and quality of images. In this experiment, we present a transformer-based architecture for handwriting text recognition with a lightweight convolutional encoder (Figure 3). Transformed-based models use data augmentation techniques to transform the original dataset into a larger set of training examples [34]. This technique helps improve the model's accuracy by reducing overfitting and increasing the diversity of the training examples. The size of the dataset is a critical factor in determining the performance of transformed-based models.

To evaluate the model, we use character error rate (CER) and word error rate (WER) metrics [35]. To assess how much generated data helps for metrics, we conducted several experiments adding different amounts of data to the real dataset. First, we trained our model only on real data with a total of 45,211 words. After that, we generated an image by adding 30%, 50%, and 100% for the number of each word in the dataset; in total, we obtained 14 k, 22 k, and 44 k generated images of words, respectively. On different amounts of data, our model was trained, and the results are shown in the table below.

From Table 2, we can see that the average WER (%) and CER (%) of the trained models tested on different combinations of datasets. The WER and CER results showed 25.65% and 11.15%, respectively.

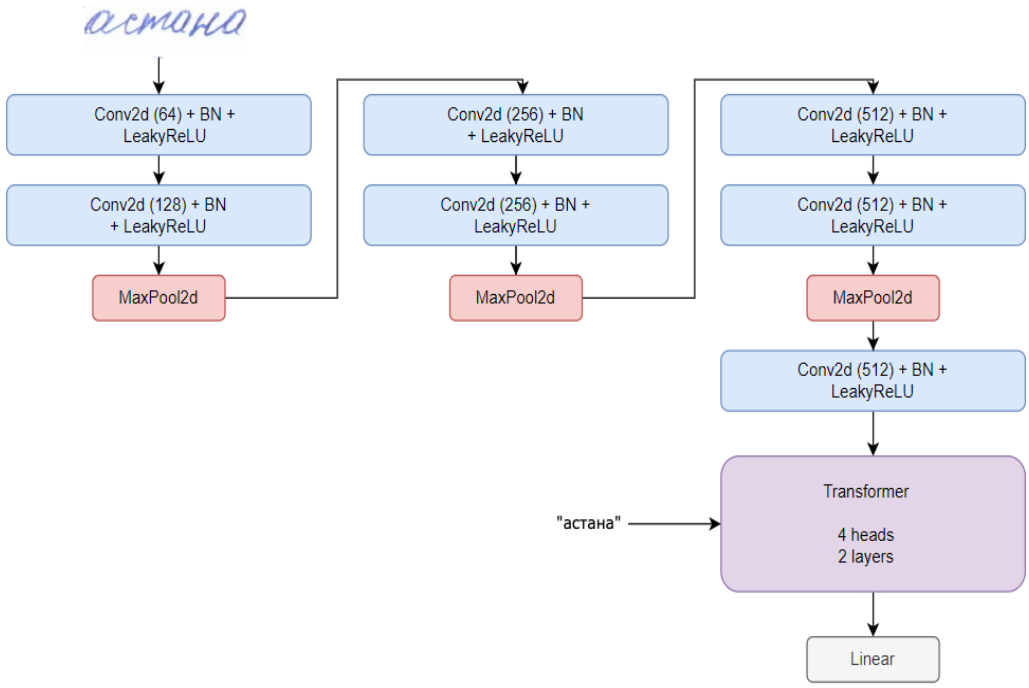


Figure 3. Transformer-based architecture of the OCR recognition model for Kazakh language.

Table 2. WER and CER results of the model for the different combinations of datasets.

Dataset	WER (%)	CER (%)
real 45 k	25.65	11.15
real + generated 59 k	23.21	9.24
real + generated 67 k	17.34	9.01
real + generated 89 k	15.21	8.13

WER and CER on real data showed 25.65% and 11.15%, respectively.

As can be seen from Table 2, the results of the trained model with the generated data show a significant improvement in WER and CER of 10.44% and 3.02%, respectively.

Experiment 5. In this experiment, we tested the accuracy of the model on real data after training it on the generated data. Below are graphs of the values of the loss function and the CER (character accuracy rate) metric during the training process (Figure 4). The CAR (Character Accuracy Rate) metric was calculated for each type of symbol in order to visually see which symbols the model classifies well and which ones have difficulties (Figure 5). The average CAR was 82.13%.

Looking at specific examples, you can make sure that the model copes even with hard-to-read handwriting options (Figure 6).

On the test dataset, the model achieved results of 17.11% CER. The obtained results are sufficient to use the models in projects that do not require absolute accuracy. Additionally, it is worth remembering that applying post-processing techniques to the strings received from the neural network can increase the accuracy by another couple of percent (mostly when working with dictionaries, since many words are more or less known and the output of the sequence of letters is rarely random).

We were not able to objectively compare the accuracy of the model with other works due to the lack of work on handwriting recognition, specifically in the Kazakh language. Because of this, we compared them with works for Cyrillic handwritten Russian recognition, as they are similar with the exception of some characters (ә, ң, і, г, ү, ұ, һ, ө).

According to the experimental result, our model outperforms other approaches with 17.11% CER in the test set of our dataset (Table 3).

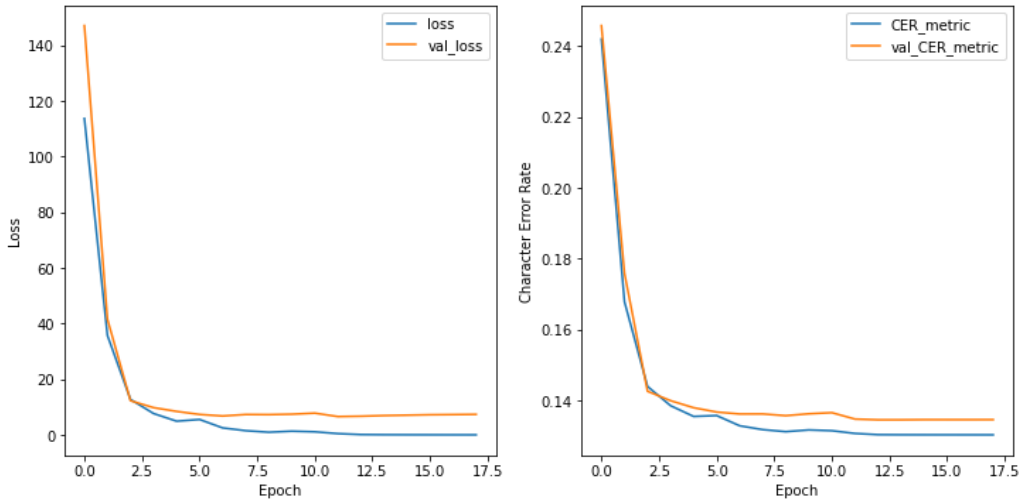


Figure 4. Results of model errors at different training epochs.

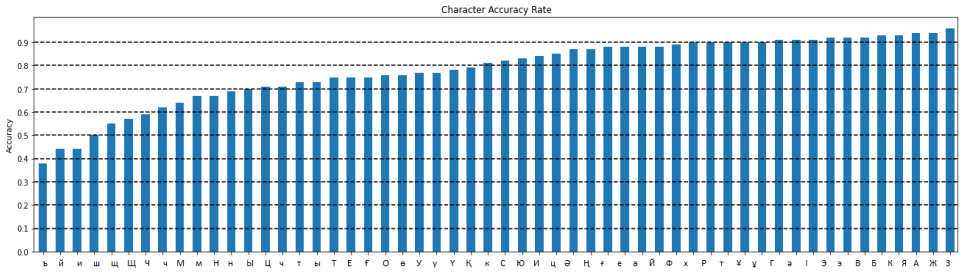
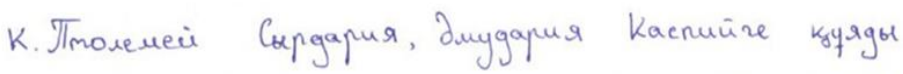


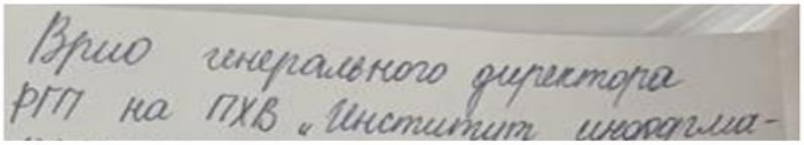
Figure 5. CAR metric for each Cyrillic character in Kazakh language.



(**a**)

Полемей Сырдария дмудария Каспийге куяды

(**b**)



(**c**)

Врио генерального директора РГП на ПХв Институт информа-

(**d**)

Figure 6. (**a**,**c**) Real handwritten sentences in Kazakh language; (**b**,**d**) recognition results.

Table 3. Comparative results.

Methods	Accuracy (%)	CER (%)
(Shonenkov et al., 2021) [12]	75.88	21.54
(Hamada et al., 2019) [36]	73.32	27.96
Our proposed method	78.17	17.11

4. Discussion

One approach that can be used to improve the accuracy of Kazakh handwriting recognition is the use of generative models such as generative adversarial networks (GANs). These models can be trained on a small dataset of Kazakh handwriting samples to create synthetic images that resemble real handwriting. In this work, three neural networks were trained, and their discriminators and error functions were also studied. After increasing the date set by 100%, the accuracy of the model on 9434 test data points reached a value of 87.3%, which is a good result.

Once trained, generative models can be used to augment the training data by increasing the variety and number of training samples. This can help improve the reliability of the recognition model, allowing it to better handle changes in handwriting style and quality.

In addition, generative models can also be used for data augmentation techniques such as style transfer. This includes converting the handwriting style of the sample to another style and generating additional training data that covers a wider range of Kazakh handwriting variations.

Finally, generative models can be used to synthesize additional training data for classes with a small number of examples. This can be especially useful in cases where there is limited training data, which is often the case for less frequently used characters or words in Kazakh handwriting.

In general, the use of generative models to improve the recognition accuracy of the Kazakh manuscript has great potential. Using the capabilities of generative models, we can create more diverse and extensive training data that better reflects the nuances of Kazakh handwriting, which leads to improved recognition performance.

The study may offer a start on new architectures of neural networks for handwriting recognition, which will use GAN models for data augmentation already in the training phase. This is confirmed by the results of several experiments.

5. Conclusions

The results obtained allow us to state that generative-adversarial networks can be used to solve the problem of Kazakh handwriting recognition. The results showed that the proposed GAN-based model was able to improve the accuracy of Kazakh handwriting recognition by 11% compared to traditional recognition methods.

The paper discussed the problems of optical character recognition for Kazakh handwriting, including the lack of a standardized writing system and a wide variety of writing styles. The authors propose a GAN-based approach to generate synthetic Kazakh handwriting samples that can be used to train an OCR system. The GAN model is trained using a large dataset of real Kazakh handwriting samples to generate synthetic handwriting-like samples.

The authors evaluated the proposed approach using the Kazakh handwriting dataset and compared it with traditional OCR methods. The results of the experiment showed that the proposed approach significantly improved the accuracy of recognition of Kazakh handwriting. The study also presents a detailed analysis of the results and discusses the limitations of the proposed approach.

In conclusion, the paper shows that GANs can be an effective tool to improve the accuracy of OCR for Kazakh handwriting recognition. The proposed approach can be applied to other languages and handwriting styles to improve the accuracy of optical character recognition. The study provides valuable insights into the use of GANs for OCR and may be useful to researchers and practitioners working in the field of handwriting recognition.

Author Contributions: Conceptualization, conducted the experiments, software, A.Y., methodology, task management and checking results, Y.A., writing, draft preparation and editing, L.C. All authors have read and agreed to the published version of the manuscript.

Funding: The article was supported by grants AP14871625, BR18574144 of the Ministry of Science and Higher Education of the Republic of Kazakhstan.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shi, B.; Bai, X.; Yao, C. An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 2298–2304. [CrossRef]
2. Tran, B.H.; Le-Cong, T.; Nguyen, H.M.; Le, D.A.; Nguyen, T.H.; Le Nguyen, P. SAFL: A Self-Attention Scene Text Recognizer with Focal Loss. *arXiv* **2020**, arXiv:2201.00132.
3. Metzenthin, E.; Bartz, C.; Meinel, C. Weakly Supervised Scene Text Detection using Deep Reinforcement Learning. *arXiv* **2022**, arXiv:2201.04866.
4. Available online: https://astanatimes.com/2017/10/kazakhstan-to-switch-to-latin-alphabet-by-2025 (accessed on 25 April 2023).
5. Fedotov, A.; Tussupov, J.; Sambetbayeva, M.; Idrisova, I.; Yerimbetova, A. Development and implementation of a morphological model of kazakh language. *Eurasian J. Math. Comput. Appl.* **2015**, *3*, 69–79.
6. Dash, A.; Ye, J.; Wang, G. A review of Generative Adversarial Networks (GANs) and its applications in a wide variety of disciplines—From Medical to Remote Sensing. *arXiv* **2021**, arXiv:2110.01442.
7. Fussell, L.; Moews, B. Forging new worlds: High-resolution synthetic galaxies with chained generative adversarial networks. *Mon. Not. R. Astron. Soc.* **2019**, *485*, 3203–3214. [CrossRef]
8. Laino, M.E.; Cancian, P.; Politi, L.S.; Della Porta, M.G.; Saba, L.; Savevski, V. Generative Adversarial Networks in Brain Imaging: A Narrative Review. *J. Imaging* **2022**, *8*, 83. [CrossRef]
9. Park, S.-W.; Ko, J.-S.; Huh, J.-H.; Kim, J.-C. Review on Generative Adversarial Networks: Focusing on Computer Vision and Its Applications. *Electronics* **2021**, *10*, 1216. [CrossRef]
10. Kovalev, V.A.; Kozlovsky, S.A.; Kalinovsky, A.A. Generation of artificial chest X-ray images using generative-adversarial neural networks. *Informatics* **2018**, *15*, 7–17.
11. Fogel, S.; Averbuch-Elor, H.; Cohen, S.; Mazor, S.; Litman, R. ScrabbleGAN: Semi-Supervised Varying Length Handwritten Text Generation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 4324–4333.
12. Shonenkov, A.; Karachev, D.; Novopoltsev, M. StackMix and Blot Augmentations for Handwritten Text Recognition. *Comput. Vis. Pattern Recognit.* **2021**. [CrossRef]
13. Berikov, V.; Amirgaliyev, Y.; Cherikbayeva, L.; Yedilkhan, D.; Tulegenova, B. Classification at incomplete training information: Usage of clustering group to improve performance. *J. Theor. Appl. Inf. Technol.* **2019**, *19*, 5048–5060.
14. Mazzolini, D.; Mignone, P.; Pavan, P.; Vessio, G. An easy-to-explain decision support framework for forensic analysis of dynamic signatures. *Forensic Sci. Int. Digit. Investig.* **2021**, *38*, 301216. [CrossRef]
15. Bhowal, P.; Banerjee, D.; Malakar, S.; Sarkar, R. A two-tier ensemble approach for writer dependent online signature verification. *J. Ambient. Intell. Humaniz. Comput.* **2021**, *13*, 21–40. [CrossRef]
16. Vorugunti, C.S.; Pulabaigari, V.; Mukherjee, P.; Gautam, A. COMPOSV: Compound feature extraction and depthwise separable convolution-based online signature verification. *Neural Comput. Applic* **2022**, *34*, 10901–10928. [CrossRef]
17. Sumeet, S. Singh and Sergey Karayev. Full Page Handwriting Recognition via Image to Sequence Extraction. In *Book: Document Analysis and Recognition*; ICDAR: Lausanne, Switzerland, 2021; pp. 55–69.
18. Kenshimov, C.; Mukhanov, S.; Merembayev, T.; Yedilkhan, D. A Comparison of Convolutional Neural Networks for Kazakh Sign Language Recognition. *East.-Eur. J. Enterp. Technol.* **2021**, *5*, 44–54. [CrossRef]
19. Buribayev, Z.; Merembayev, T.; Amirgaliyev, Y.; Miyachi, T. The Optimized Distance Calculation Method with Stereo Camera for an Autonomous Tomato Harvesting. In Proceedings of the 2021 IEEE International Conference on Smart Information Systems and Technologies (SIST), Nur-Sultan, Kazakhstan, 28–30 April 2021; IEEE: Piscataway, NJ, USA; pp. 1–5.
20. Amirgaliyev, Y.; Shamiluulu, S.; Merembayev, T.; Yedilkhan, D. Using machine learning algorithm for diagnosis of stomach disorders. In Proceedings of the Mathematical Optimization Theory and Operations Research: 18th International Conference, MOTOR 2019, Ekaterinburg, Russia, 8–12 July 2019; Revised Selected Papers (pp. 343–355); Springer International Publishing: Cham, Switzerland, 2019.
21. Daiyrbayeva, E.; Yerimbetova, A.; Nechta, I.; Merzlyakova, E.; Toigozhinova, A.; Turganbayev, A. A Study of the Information Embedding Method into Raster Image Based on Interpolation. *J. Imaging* **2022**, *8*, 288. [CrossRef]
22. Merembayev, T.; Amirgaliyev, Y.; Saurov, S.; Wójcik, W. Soil Salinity Classification Using Machine Learning Algorithms and Radar Data in the Case from the South of Kazakhstan. *J. Ecol. Eng.* **2022**, *23*, 61–67. [CrossRef]
23. Jin, T.; Zhuang, J.; Xiao, J.; Xu, N.; Qin, S. Reconstructing Floorplans from Point Clouds Using GAN. *J. Imaging* **2023**, *9*, 39. [CrossRef]

24. Wright, J.; Ganesh, A.; Rao, S.; Peng, Y.; Ma, Y. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In *Advances in Neural Information Processing Systems*; Curran Associates: Red Hook, NY, USA, 2009; pp. 2080–2088.
25. Tran, N.T.; Bui, T.A.; Cheung, N.M. Dist-GAN: An Improved GAN using Distance Constraints. In *Book Chapter*; ECCV: Coburg, Victoria, 2018.
26. Ghojogh, B.; Karray, F.; Crowley, M. Theoretical Insights into the Use of Structural Similarity Index in Generative Models and Inferential Autoencoders. In *Image Analysis and Recognition*; Campilho, A., Karray, F., Wang, Z., Eds.; ICIAR 2020; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2020; Volume 12132. [CrossRef]
27. Shlens, J. Notes on Kullback-Leibler Divergence and Likelihood Computer Science. *arXiv* **2014**, arXiv:1404.2000.
28. Ho, Y.; Wookey, S. The Human Visual System and Adversarial AI//Computer Vision and Pattern Recognition (cs.CV); Machine Learning (cs.LG); Image and Video Processing (eess.IV). *arXiv* **2020**, arXiv:2001.01172.
29. Amirgaliyev, B.; Yeleussinov, A.; Taizo, M. Kazakh handwritten recognition. *J. Theor. Appl. Inf. Technol.* **2020**, *98*, 2744–2754.
30. Krishnan, P.; Kovvuri, R.; Pang, G.; Vassilev, B.; Hassner, T. TextStyleBrush: Transfer of Text Aesthetics from a Single Example. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**. [CrossRef]
31. Huang, X.; Belongie, S. Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization. In Proceedings of the IEEE, International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 1501–1510.
32. Karras, T.; Laine, S.; Aila, T. A Style-Based Generator Architecture for Generative Adversarial Networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–17 June 2019; pp. 4401–4410.
33. Brock, A.; Donahue, J.; Simonyan, K. Large Scale GAN Training for High Fidelity Natural Image Synthesis. *arXiv* **2018**, arXiv:1809.11096.
34. Li, M.; Lv, T.; Chen, J.; Cui, L.; Lu, Y.; Florencio, D.; Zhang, C.; Li, Z.; Wei, F. TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models. *arXiv* **2021**, arXiv:2109.10282.
35. Wick, C.; Reul, C.; Puppe, F. Calamari—A High-Performance Tensorflowbased Deep Learning Package for Optical Character Recognition. *Digit. Humanit. Q.* **2020**, *14*, 25–29.
36. Hamada, M.A.; Sultanbek, K.; Alzhanov, B.; Tokbanov, B. Sentimental text processing tool for russian language based on machine learning algorithms. In Proceedings of the ICEMIS'19: The 5th International Conference on Engineering & MIS, Astana, Kazakhstan, 6–8 June 2019; pp. 1–6.