UDC: 004.8:004.93 https://doi.org/10.15407/jai2024.02.032

L. Korotka[1], D. Klevzhyts[2], D. Shvydko[3]
[1,2,3]Ukrainian State University of Science and Technology, Ukraine
2, Academik Lazaryan st., Dnipro, 49010
[1]larysakorotka@gmail.com
[2]dimaklevzhts8@gmail.com
[3]shvydkodan@gmail.com
[1]https://orcid.org/0000-0003-0780-7571
[2]https://orcid.org/0009-0007-2739-6286
[3]https://orcid.org/0009-0008-6988-2597

USE OF GENERATIVE-ADVERSARIAL NETWORKS WHEN CREATING CONTENT

Abstract. The application of generative-adversarial networks in the creation of content is studied. Monitoring of training, analysis of architectures, determination of internal processes at the level of layers, research of properties of latent space, and interaction with it are carried out. Variants of using the specified networks in image generation are considered. Special attention is paid to practical implementation aspects, including selecting optimal parameters and data processing. The difference between a classifier and a discriminator is formulated. The principles of generative-adversarial networks and their influence on the efficiency and quality of generated images are studied. The advantages and limitations of using GANs in content creation are considered.

Keywords: generative adversarial networks, discriminator, generator, image generation, TensorFlow 2.

Introduction

A popular tool in the modern technology industry for content creation is the use of generative-adversarial networks (GAN) [1]. Algorithms based on machine learning principles are capable of generating high-quality visual, text and audio content that is difficult to distinguish from human-made content.

Analysis of recent research and publications

GANs are used in various fields, from art and entertainment to marketing and scientific research, opening new horizons for creativity. Some of the models can be called "creative". That is why one of the most promising areas for generative adversarial networks is art and fashion. Well-trained GANs can be used to create paintings, songs, clothes, and even poems [2].

Generative-adversarial networks can be used to improve the clarity of images based on statistical distributions: they are able to predict missing fragments, and generate the corresponding pixel values, which will improve the quality of photos taken by telescopes or microscopes where missing, lost, damaged fragments are present [3-5]. GANs can be used to predict computational bottlenecks in scientific research projects as well as in industrial applications. Obviously, this is far from a complete list of applications of these networks. Despite the interest and progress in generative image modeling, the successful creation of various high-resolution samples from complex datasets is far from complete [6, 7].

The goal and task of the research: to investigate the generative-adversarial network, and its internal behavior during generation.

The task of the work is to generate content using GAN. It is necessary to find architectures and recommendations that will improve the quality of the received data.

Investigate the processes of learning models and generating images, and consider their internal behavior. Need to: find and test different ways to monitor GAN training; find out what latent/hidden space is, what its properties are, and how to interact with it; determine the processes that take place inside the obtained neural networks at the level of layers; establish and describe the features of the mathematical basis of the discriminator; formulate the difference between a classifier and a discriminator; collect experimental data and conduct their analysis.

Research materials and methods

Any learning of neural networks, including deep learning, is not a simple task for a personal computer (PC). Especially if you do not configure the use of GPU (Graphics Processing Unit) for this process.

The work uses the Python programming language. The TensorFlow framework [7, 8] can work in both modes: if there are no settings for the use of a graphics card, then the calculations are performed on the processor.

For small networks, the CPU (Central Processing Unit) will suffice, but due to its operational characteristics, it may take significantly longer. It is still recommended to make an effort to install the appropriate programs and drivers required for the TensorFlow framework [7, 8]. It is worth noting that since version 2.11 TensorFlow is not supported by the Windows operating system, so further use requires additional settings or switching to another system.

There are two options for solving the problem: using your PC or an online service, for example, Google Colab [9]. The latter approach does not require additional settings, has free access to graphics processors, and allows you to distribute documents to other people. Proposed technical characteristics of the free version: T4 GPU, 16 GB of GDDR6 memory at 10 GHz; 12 GB of RAM; CPU, 2 cores.

Presenting main material

As is well known, Generative Adversarial Networks are a type of deep learning model that consists of two competing neural networks: a generator (G) and a discriminator (D). Random noise is applied to the input of the generator, and either the output G or real data is supplied to the discriminator [10]. The output D is the label (probability value) of the class - the data is real or artificial. The utility function is maximized:

$$U(D,G) = Ex\sim Px(x)[logD(x)] + EZ\sim Pz(z)[log(1-G(z))]. \quad (1)$$

Various training data are used for experimental research, including the MNIST (Modified National Institute of Standards and Technology) dataset [11].

By definition, the input to GAN networks is a vector or other form of array of random data, on the basis of which images are generated (Fig. 1).

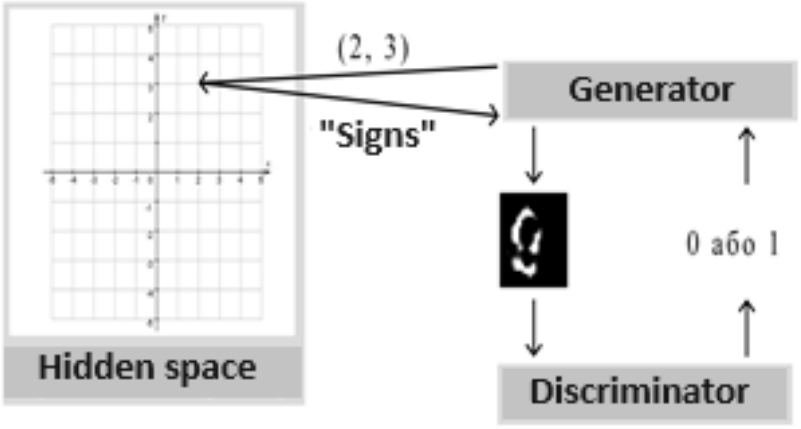

Fig. 1. Scheme of the process of recording features in the hidden space

GAN research will begin with the architecture selection process. In the work, experiments were conducted with various models, let's focus on some of them. The peculiarity of the architecture of the network generator (Fig. 2) is the size of the input noise: the dimensionality of the input data is equal to two, which corresponds to two-dimensional coordinates. A size of one hundred or more is usually specified, which may simplify the training of the model, but complicate further study due to the dimensionality problem.

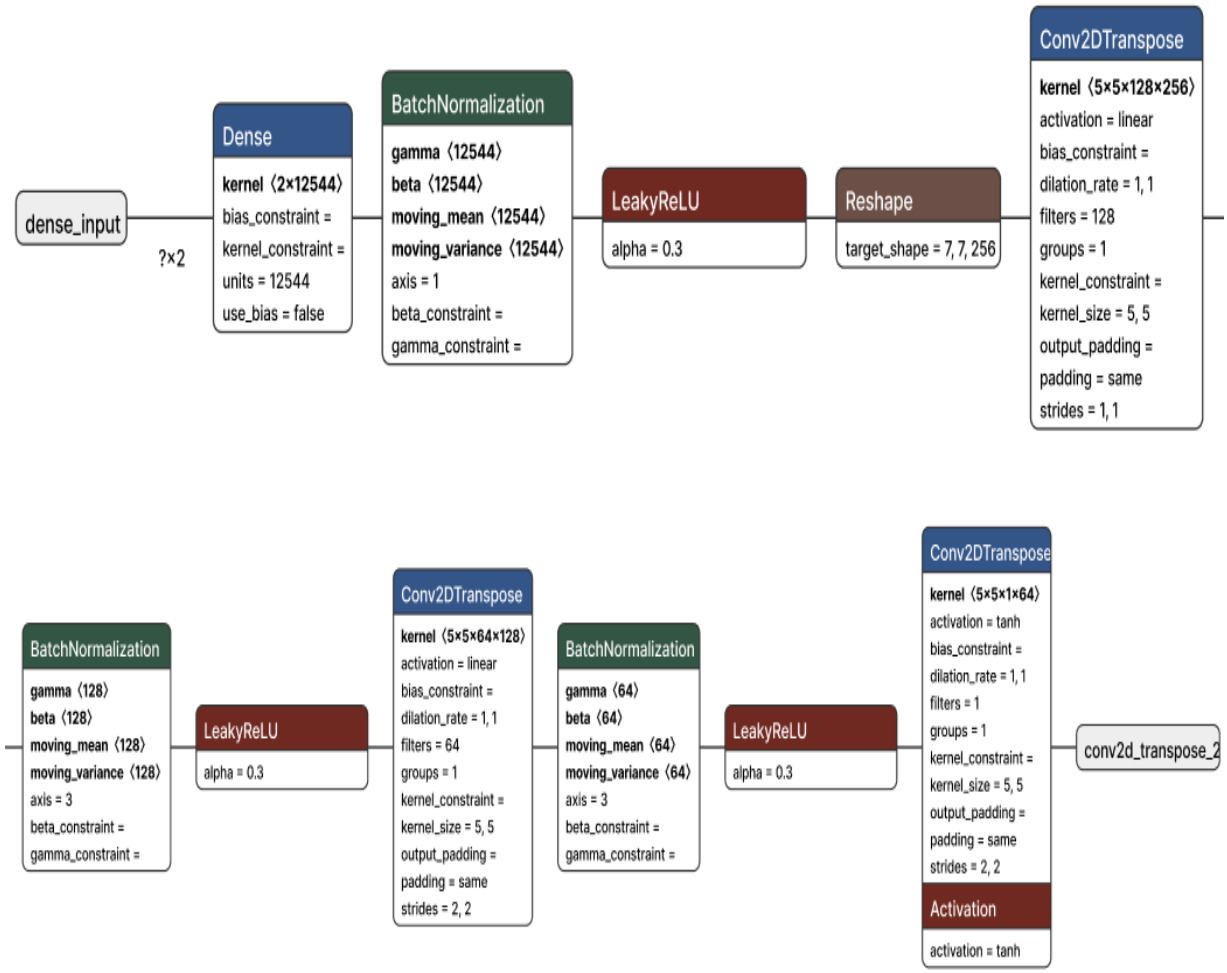

Fig. 2. Architecture of the 2D generator

The output value of the generator is equal to the data size of the MNIST dataset – 28 x 28 x 1, which corresponds to the input value of the discriminator (Fig. 3).

As for architectures of generators for 3D, 4D noises, their changes concern only the input layer to the values “?x3” and “?x4”, respectively. The discriminators, in turn, do not change.

Training any model can be done using the same approach. Only the input-output data, the structure of the models, the method of storing intermediate results, or their very type are changed.

The latter allows you to have a single template, a set of your libraries or packages, with which you can reduce code duplication. Therefore, 2D GAN training approaches are described below, which will work for other models as well, but with their modifications. Here are the main stages.

The work process begins with the preparation of input data: normalization or standardization. The formation of the generator and discriminator, the function of calculating their errors, takes place. The module is responsible for the direct training of models, updating their weights with the help of optimizers.

An important point is to save the state of the model to be able to pause the training and continue it later. The last template component will be the training launch block.

Individual changes and settings are made for each type of model [12].

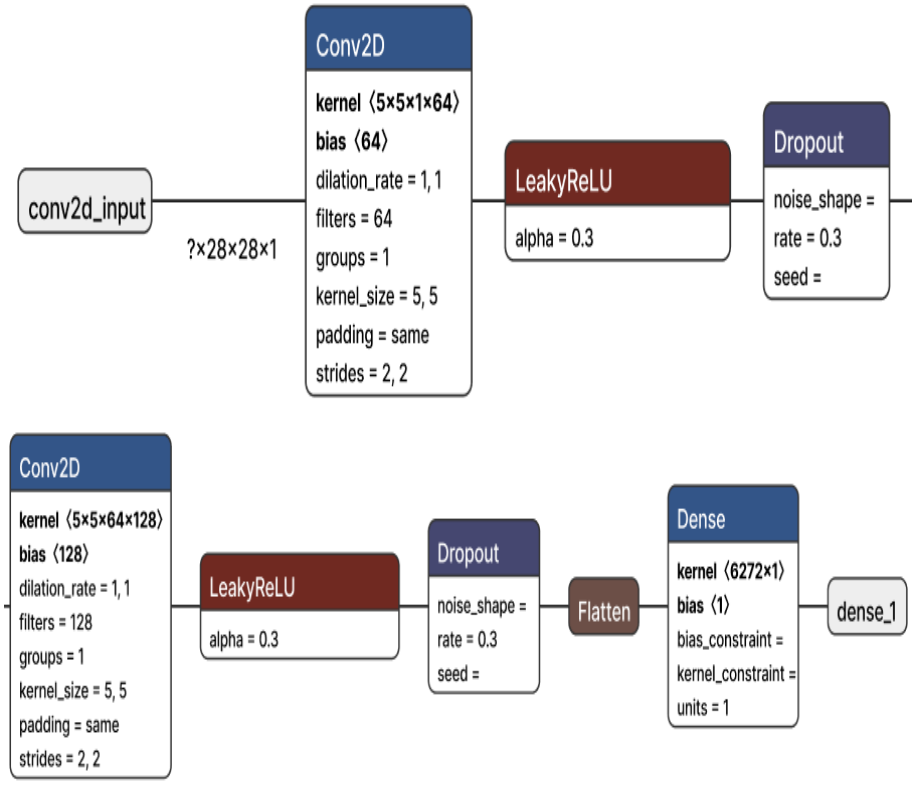

Fig. 3. Architecture of 2D, 3D, 4D discriminator

Results of experiments

The following results were obtained during direct GAN training:

– training was conducted for 15 hours, where one epoch accounted for approximately 10 seconds;

– a total of 500 iterations of 10 epochs were passed, from each iteration 16 photo numbers with the same set of noise from 16 vectors were saved (images from the first and last iteration are shown in Fig. 4 a and 4 b, respectively), maps of areas obtained by classification of 10,000 generated images by a combination of a hundred points in the range from -1 to 1 along the abscissa and ordinate axes (therefore, in a combination of 10,000 noise values). The initial and final state of the class map is shown in Figure 5a and Figure 5b, respectively;

– the accuracy of the strong independent classifier (IC) at the thousandth epoch was 0.49% on the training set and 0.55% on the test set, at the end of training (five thousandth epoch) the values were 0.73% and 1.54%, respectively. IC is essentially a trained discriminator, but it is an independent model whose task is to distinguish the generated images from the real ones with the maximum percentage. The latter distinguishes him from a discriminator who does not learn to the end. The training schedule of the last strong IC (Fig. 6);

– the accuracy value of a small network of an independent classifier (Fig. 7);

– the history of generator and discriminator errors separately (Fig. 8);

– the result of the generation of the trained model (Fig. 9).

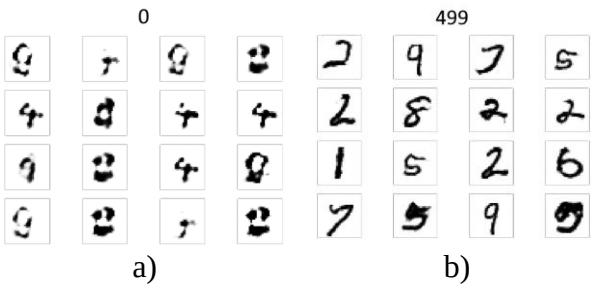

Fig. 4. Generated images on the first and last iteration

It is not difficult to notice that the quality of the numbers has become better over time (Fig. 4). From epoch to epoch, the generator improved its output step by step, until it reached a stable version, although some numbers look rather imprecise, unclear, have extra pixels.

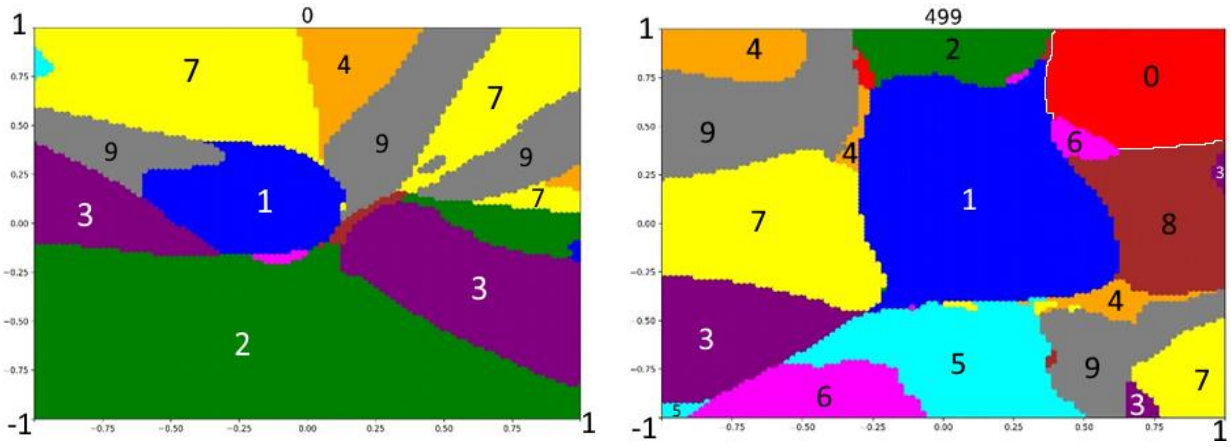

Fig. 5. Maps of class areas on the first and last iteration

Analysis of the obtained results makes it possible to note an interesting fact regarding the map of class regions: the number “1” was often found in the center between different 2D GAN models. This is what was discovered during the restart of training and the process of preparing for the final version of the network. The explanation may be that this figure has the greatest connection with others, and its shape easily transforms into most others.

It can be concluded that in the center of the map of the hidden space of the model, the most common version of the image is most often found, which is the carrier of most of the characteristics and features inherent in all other objects.

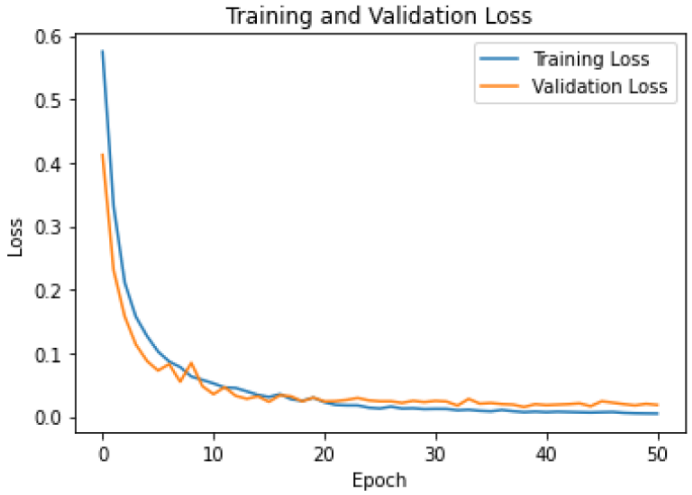

Fig. 6. The process of strong learning independent classifier

After the hundredth epoch, weak ICs stopped rapidly increasing in value, as shown by the graph in Fig. 7.

One would assume that the training is complete, but this is a misleading impression. The ideal variant of the error value in such a problem is equal to 50%, so that it can mean “guessing” by the classifier: where is one class and where is the other.

The classifier quickly learns to recognize which image is real and which is artificial (Fig. 6).

After the two thousandth epoch, the errors gradually move away from each other (Fig. 8), which indicates the stabilization of learning. The model of a weak independent classifier gradually deteriorates the ability to separate real images from generated ones, therefore, the model has learned (Fig. 7).

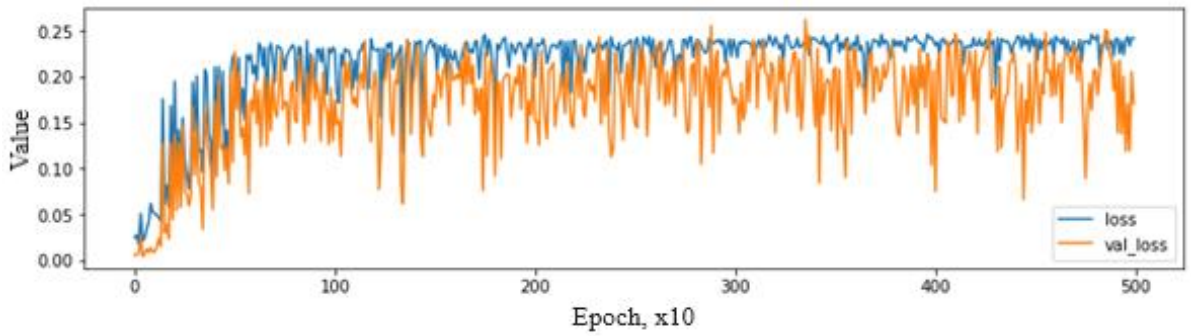

Fig. 7. Accuracy of independent classifiers every ten epochs

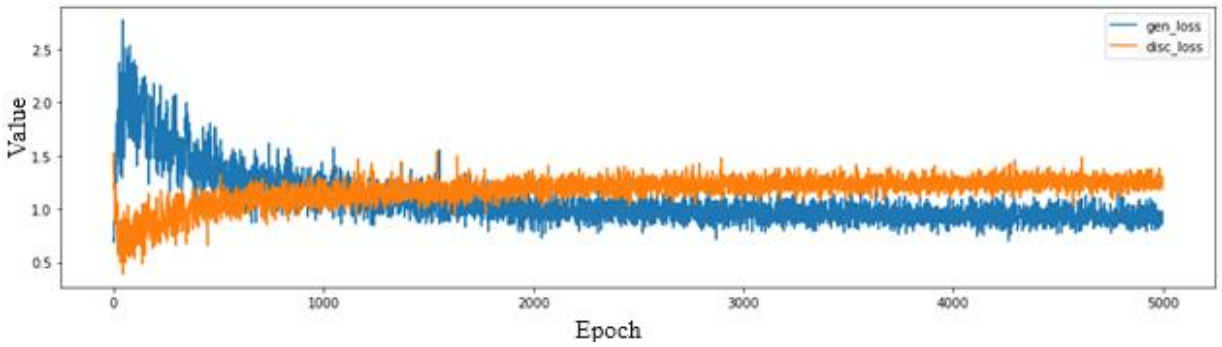

Fig. 8. History of generator and discriminator errors

Testing on the validation set with a strong classifier showed an error of 0.55% at 1000 epochs and 1.54% at 5000 epochs, which may indicate an improvement in generation. The learning stop occurred due to a decrease in the intensity of changes in the last epochs, which are visible in the sixteen image generations from each epoch and in the class regions.

The description of the 3D GAN training process now does not require such detail, but only requires specific key changes. Thus, the convenient view of a rectangular map of class areas in 2D space is replaced by a 3D cube, which makes it much more difficult to view. Such a figure is not easy to visualize; therefore, the concept of sections is introduced: 16 “sections” are made along all axes of the cube (16 points from -1 to 1 with the corresponding step) forming three projections. With such images at each epoch or iteration, the process of changing the map of class areas can be displayed.

The results of the experiments make it possible to draw the following conclusions: training the model took less time compared to the 2D version, while the quality of the generated images improved. The numbers have become clearer with neat shapes and a realistic look. The errors of weak independent classifiers have grown in the same way as in the previous model.

As for the errors of strong classifiers, the values are quite different. The 2D generator was able to achieve a result of 1.54%, and the 3D - 3.2%.

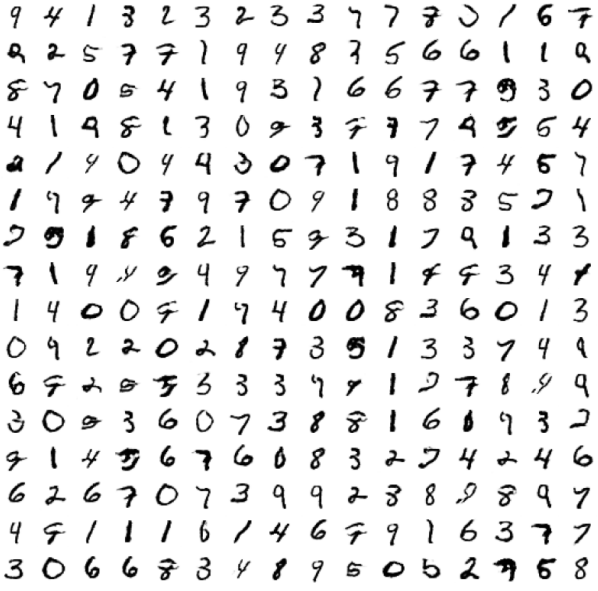

Fig. 9. An example of generating numbers by the GAN model using two-dimensional noise (five thousand epochs)

The dynamics of errors directly of the generator and discriminator models does not show any problems.

The increase in the noise dimension had a positive effect on the quality of the models, while there was a decrease in the time spent and the number of epochs (the latter would reduce the number of photos if they were stored at each epoch).

We present the results of experiments investigating various properties of the resulting networks. Without further delay, let's start with two-dimensional space.

A point can be added to the resulting map of the class areas of the two-dimensional GAN model, the movement of which allows you to explore the hidden space (Fig. 10).

There are several areas in the program interface: directly generated image "Generated image"; a sequence of images "Activations for each layer (average values)" showing intermediate generation results starting from the first layer of the neural network to the last; map of class areas "Map of class areas"; slider for "*x*" and "*y*", which can be used to change the coordinates of the point.

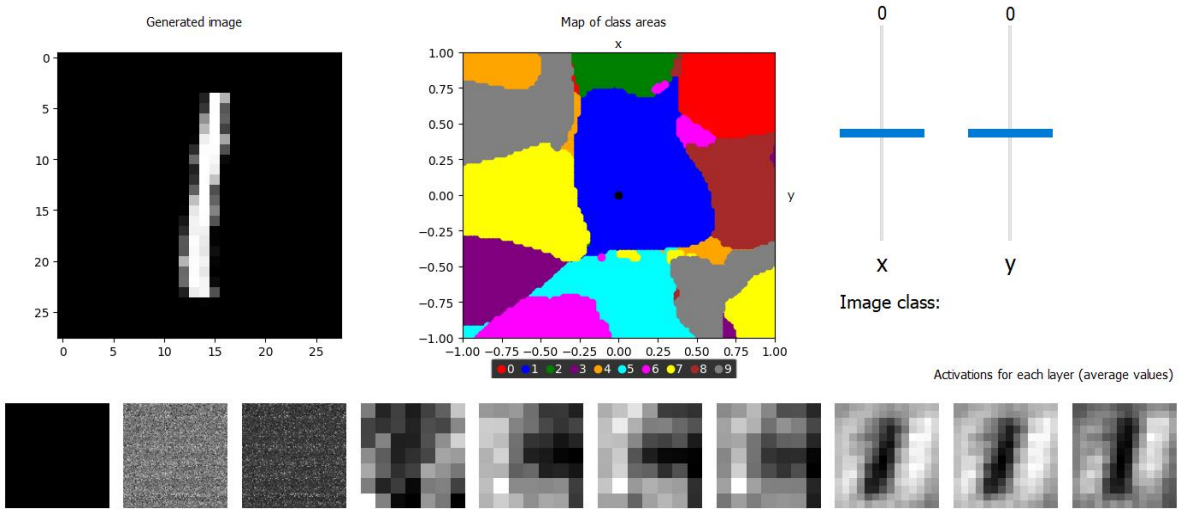

Fig. 10. Interface for exploring 2D hidden space

The state of the system is updated when the sliders move (Fig. 11): the point with coordinates (-0,84; 0,54) occupied a new area on the map corresponding to the number "9"; coordinates (0; 0) correspond to the number "1".

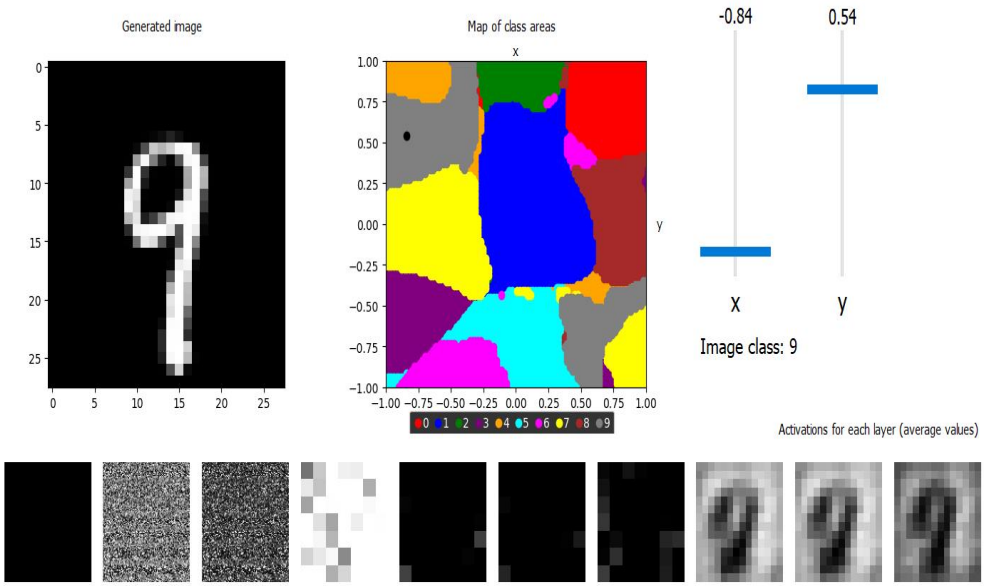

Fig. 11. New system status

Other areas, taking into account the black and white version (Fig. 12).

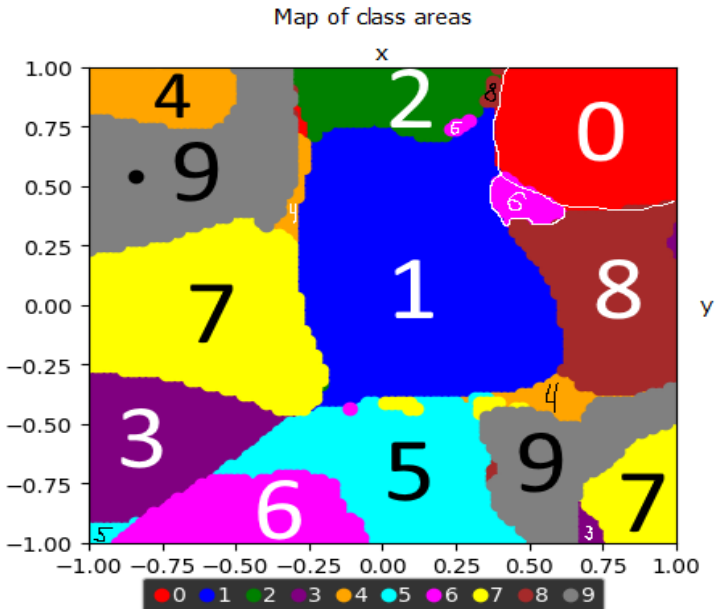

Fig. 12. Text designation of class areas

The work implements the so-called “smooth flow” from one point “A” to another point “B” (Fig. 13).

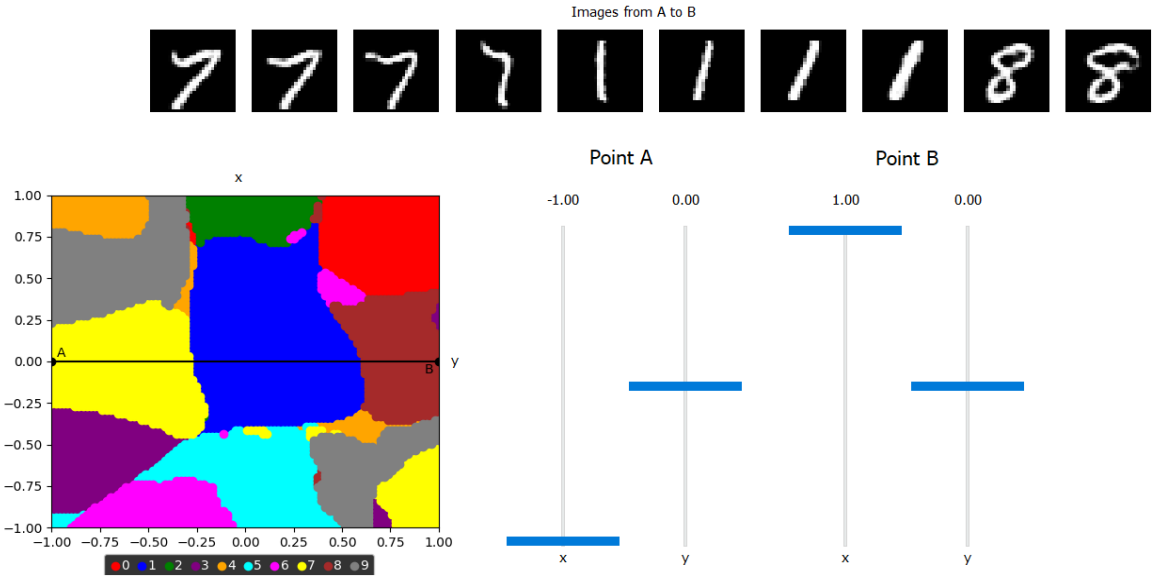

Fig. 13. “Smooth flow” of images between two points

The logic of working with three-dimensional space is similar to working with two-dimensional space. The proposed software interface is presented in fig. 14. As in the previous approaches, there are regions of the generated image in the space, sequences of generation by network layers, a cube map of class regions, sliders for changing the coordinates of a point, the location of which is now tracked not only by rotating the cube, but also by using three projections on the corresponding planes.

If you change the coordinates of the point, the state of the system will change and a new one will be obtained (Fig. 15). The point now occupies the corner of the cube corresponding to the number eight.

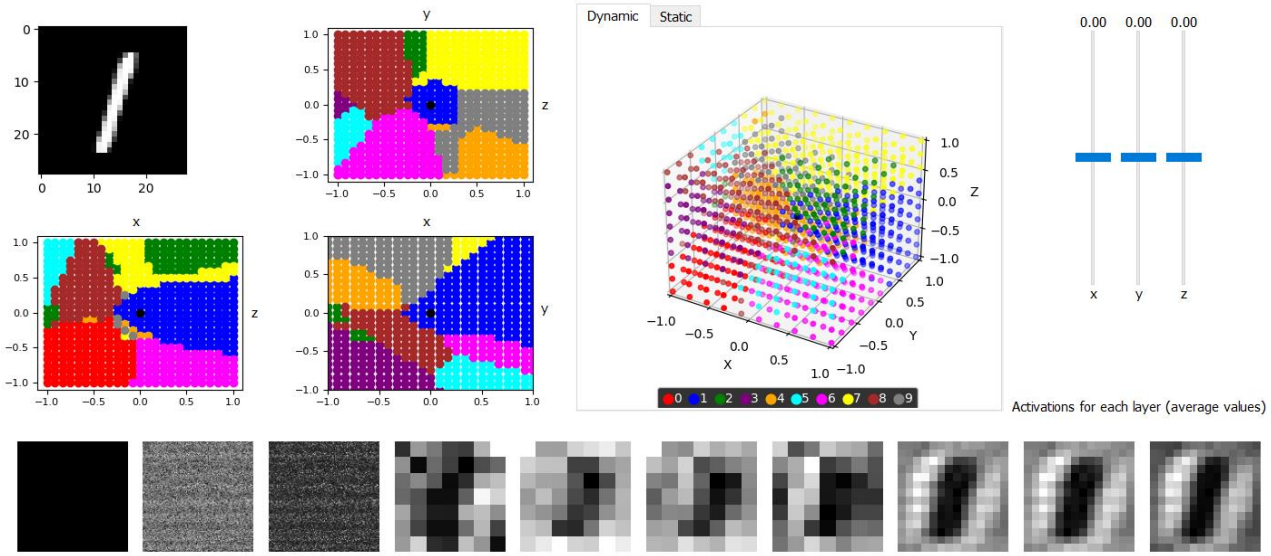

Fig. 14. Program interface for working with one point of three-dimensional space

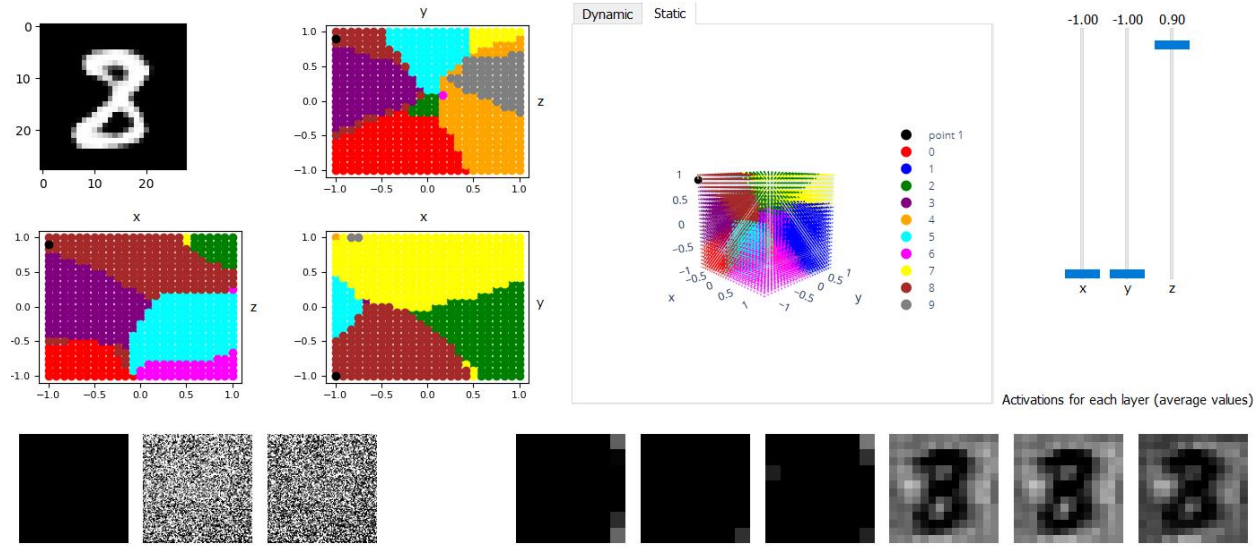

Fig. 15. Changing the position of a point in space

A “smooth flow” between two points is also implemented (Fig. 16).

In the work, a study of CGAN networks (Conditional Generative Adversarial Networks) is carried out – a type of generative-adversarial networks, which during generation receive not only noise as input but also a condition in the form of a class label of the image to be obtained.

The resulting universal model, which can generate more different images per unit of noise, theoretically increases the generation options several times.

In a regular GAN, the number of points is equal to 10,000, in which all classes are distributed, in CGAN – 10,000 for each class separately.

As disadvantages, one can note: that a significant number of images are not unique or do not look natural, but this is more a matter of the process, learning methods, and the structure of the neural network model itself.

As part of the continuation of the research topic, the issue of choosing the activation function (FA) was considered [13]. A series of experiments with the same model architecture, but the different most widely used FAs, in particular: ReLU, LeakyReLU, PReLU, and ELU, were conducted and analyzed.

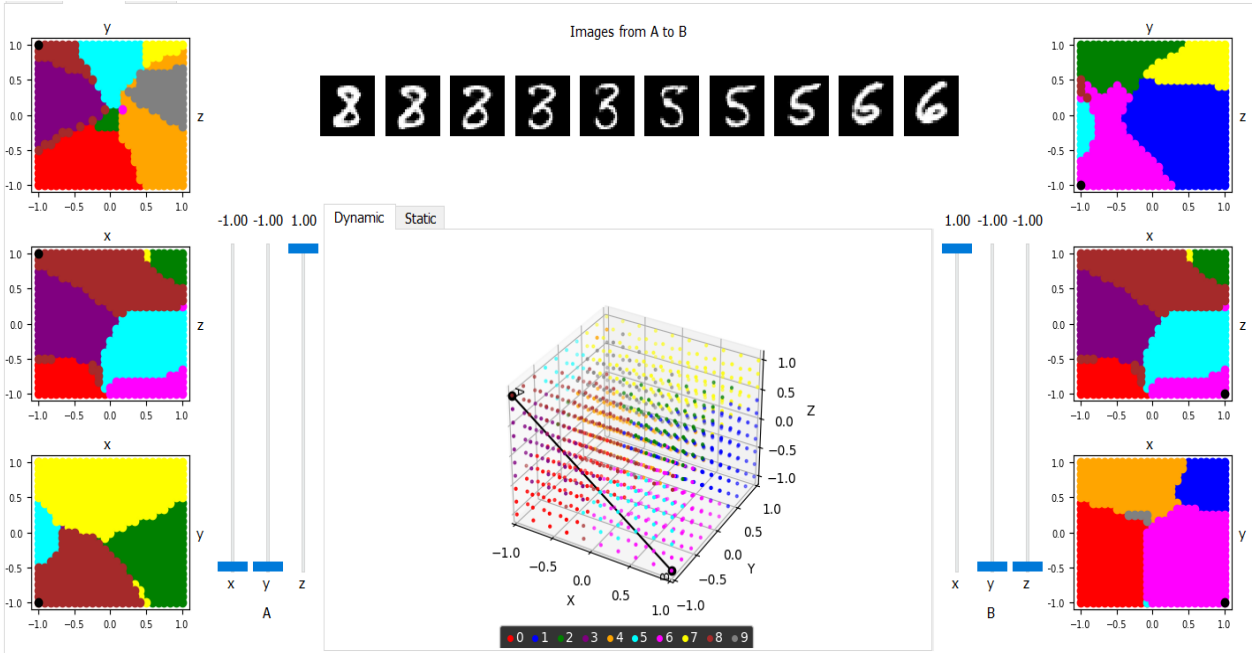

Fig. 16. An interface for studying the 3D "flow" of an image between two points

Testing was performed on 500 epochs for each model with the same MNIST dataset (Table 1).

Note that the data in the table are obtained using a strong independent classifier. Obviously, during training, the results of all models grow (the best value is 0.1561), which corresponds to a classifier error of 15.61% on the test set (Fig. 17).

Table 1. Results of the experiment with activation functions

Epoch	ReLU	Leaky ReLU	PReLU	ELU
100	0,0064	0,0065	0,0068	0,0186
200	0,0198	0,0538	0,0473	0,0369
300	0,0810	0,0502	0,0670	0,0967
400	0,0631	0,0821	0,0263	0,0706
500	0,1561	0,1428	0,0789	0,0526

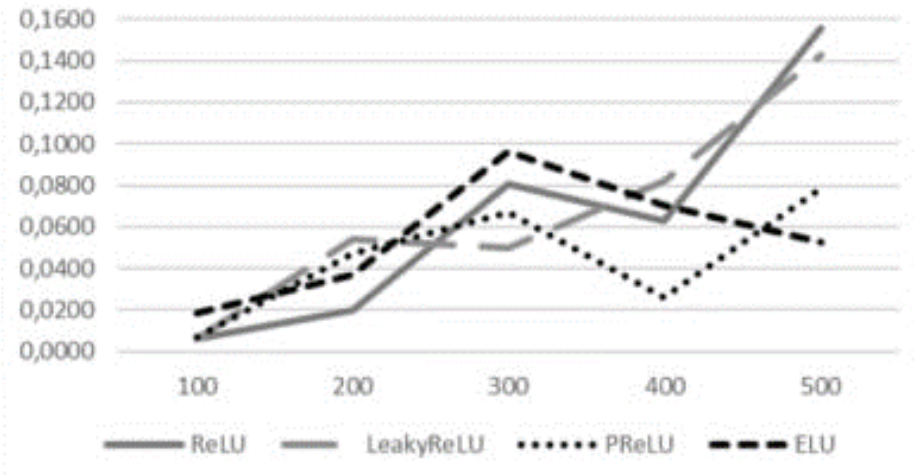

Fig. 17. Chart of results

At the end of training, the ReLU activation function achieved the best result compared to the others. Thus, the choice of FA depends on the problem to be solved and the designer of the neural network.

In the work, experiments were conducted with different FAs in Landscape GAN when generating landscape images.

Before that, the model was trained using the LeakyReLU activation function, with a maximum error of an independent classifier of approximately 8%.

Before the experiment, its architecture was taken as a basis and improved, the number of convolution filters was increased.

Table 2 shows some of the experimental results obtained.

With the best result is the ReLU activation function, which has 11.45% independent classifier error, the result is achieved at the four hundred and fiftieth epoch. ELU and LeakyReLU are next with respective results.

Each model learned about 20% longer than the previous version (100 hours). In general, a total of 480 hours were spent on the experiment (under the condition of parallel training).

Table 2. Results of the experiment

Epoch	ReLU	Leaky ReLU	ReLU	ELU
10	0,0010	0,0036	9,86E-04	0,0019
50	0,0288	0,0189	0,0057	0,0233
100	0,0541	0,0626	0,003	0,052
300	0,0689	0,0593	0,0063	0,067
400	0,0967	0,0725	0,0052	0,0587
450	0,1145	0,0636	0,0062	0,084
500	0,1086	0,0654	0,0035	0,0795
800	0,0853	0,0458	0,0036	0,099
1000	0,0397	0,0718	0,0059	0,0825

Further, the paper investigates the behavior of the internal system of models: the type of activation between layers in the best photo-landscape models with the corresponding functions.

The result is that all activation functions try to strongly “cancel” the inputs to the neurons.

Black means the activation value is close to zero, and white means strong response.

It turns out that the final image generated by the model is created by a large number of “dot-sticks”, which are then grouped into a photo as the last layer.

It can be concluded that when changing, for example, the value of activations after the penultimate layer, you can generate separate areas of the photo to improve it (remove defects or other artifacts).

Table 3 contains information about the training time of the models, the number of epochs, the size of the dataset and images, the quality of generation according to the assessment of the independent classifier, the number of parameters of the models.

Table 4 summarizes information about testing with various activation functions.

Obviously, the ReLU activation function ranks first in both types of generation problems.

For a better understanding of the operation of the generator and the hidden space, the variational autoencoder (VAE) is considered, which, with the help of root mean square error and Kullback-Leibler (KL) divergence, distributes vertices in the space next to each other in order of similarity.

Table 3. Results of experiments with models regarding the quality of generation, which were evaluated by independent classifiers

Generator			Numbers (noise)			
			2D	3D, CGAN		4D
t learning, hours			15	10	5.5	3
t_{avg} to 1 epoch, s			10	20	0.125	10
n epoch, thousands			5	2	60	1
Dataset, thousands, shape			60, 28 x 28 x 1			
Memory, GB			0,18			
IC, ε, %			1,5	3,2	0,084	8,29
params	G	Total	1 101 632	1 114 176	1 114 206	1 126 720
		Trainable	1 076 160	1 088 704	1 088 734	1 101 248
		Non-trainable	25 472			
	D	Total	212 865	212 865	220 705	212 865

Table 4. Results of testing models with different activation functions

Numbers				
Activation function	ReLU	LeakyReLU	PReLU	ELU
The biggest error, %	15,61	14,28	7,89	9,67
Study time: one epoch, s – total time, hours	9 s – 1 hours 15 min 06 s	10 s – 1 hours 22 min 37 s	10 s – 1 hours 15 min 14 s	9 s – 1 hours 19 min 32 s
Generator parameters	1 126 720			
Discriminator parameters	212 865			
Landscapes				
The biggest error, %	11,45	9,74	0,92	10,61
Study time: one epoch, s	410 s	410 s	430 s	420 s
Generator parameters	10 729 344			
Discriminator parameters	2 563 713			

Assume that the GAN implements this behavior without direct instructions. Kullback-Leibler divergence [14]:

$$D_{KL}[N(\mu,\sigma||N(0,1)=0,5\sum(1+\log(\sigma^2)-\mu^2).$$

The online service of the Kaggle platform was used to search for datasets and train networks.

The architecture of the U-Net network was used as an experiment. The latter is done to get the “noise” to deep levels.

Special attention is paid to the selection of the dataset. As already noted, the MNIST dataset is quite clear and easy to use, but not only it was used in the work.

The Kaggle platform is known to contain a sufficient number of available datasets for various machine-learning tasks. Of course, there are other online services, for example, the possibility of using GitHub repositories or Alibaba's Tianchi platform for this.

Six thousand images of ships with a size of 32x32 pixels were used as a dataset. An independent classifier can indicate the quality of the generation: the worse the division into two classes is, the better the trained GAN model.

An experiment on the convolution kernels showed that the best result, with a 13% IC error, was the value of the convolution kernels “5-5”. Therefore, the following U-Net-2 model was trained over twenty thousand epochs, which had erratic errors, low-quality signal generation, and a test with an independent classifier showing a low 8%.

For further experiments, it is suggested to consider a linear model and drawings of landscapes with a size of 64x64 pixels. To improve the quality of training, it is proposed to increase the volume of the dataset from five to twenty thousand.

The following model is trained on ninety thousand real 64x64 landscape photos.

The generated color images have visually clear images and clearer boundaries of objects (Fig. 18).

As a result, an image with sufficient detail and image clarity was obtained. During testing with an independent classifier, an error of 11% was obtained (Fig. 19), which is quite good compared to past experiments.

Note that the following layers are used in the generator model: Dense, BatchNormalization, LeakyReLU, Reshape, and Conv2DTranspose.

As already mentioned earlier, as part of the work, tests were conducted with various functions of activations in the models.

The best version of the model has ReLU as the main FA, and the hyperbolic tangent (tanh) on the last layer. Diagram of errors (Fig. 20).

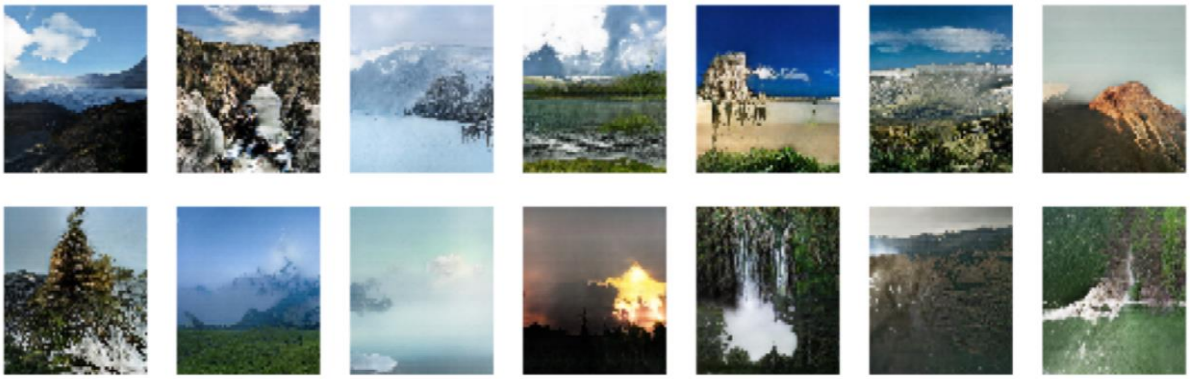

Fig. 18. Generated Images

Epoch 47/600
71/71 [==============================] - 3s
40ms/step - loss: 0.0083 - accuracy: 0.9982 - val_loss:
0.1125 - val_accuracy: 0.9690 - lr: 5.2429e-05

Fig. 19. IC test

The intermediate state of the model at the four-hundred-and-fiftieth epoch (Fig. 21) and the final state of the model (Fig. 22) are given below.

When visually comparing (Figs. 21 and 22) the intermediate and final state of the network, it can be stated that at the time of the four hundred and fiftieth epoch, the images have a better display of images than the final result.

The results of experiments conducted on training neural networks are given in [7].

Note that different network architectures, their parameters and layers, activation functions and datasets are used. The resulting data can be used to train networks to generate images larger than 64x64x3.

Conclusions

The work included: the behavior of generative-adversarial networks was investigated, the hidden space and interaction with it were considered.

The template architecture of the generator and discriminator is given. The learning processes of 2D, 3D and 4D GAN networks are described.

The behavior of the final versions of the models after their training is considered and investigated. The features of the generative-adversarial network with the CGAN condition

are given, and the representation of their hidden spaces is considered.

Experiments were carried out with activation functions when solving the problem of generating images of handwritten numbers and photos of real landscapes.

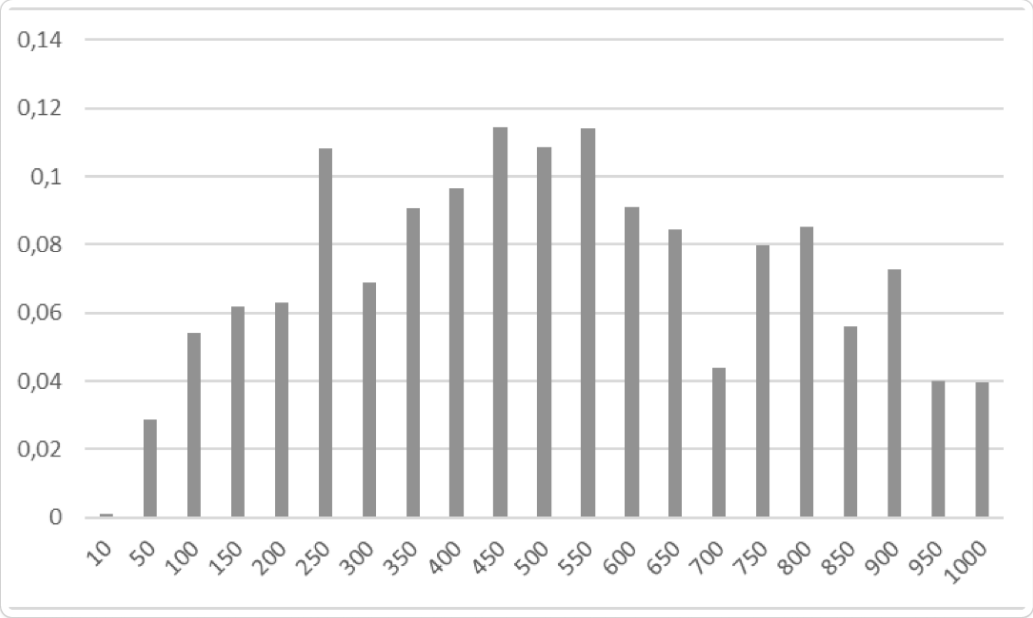

Fig. 20. Graph of errors by epoch

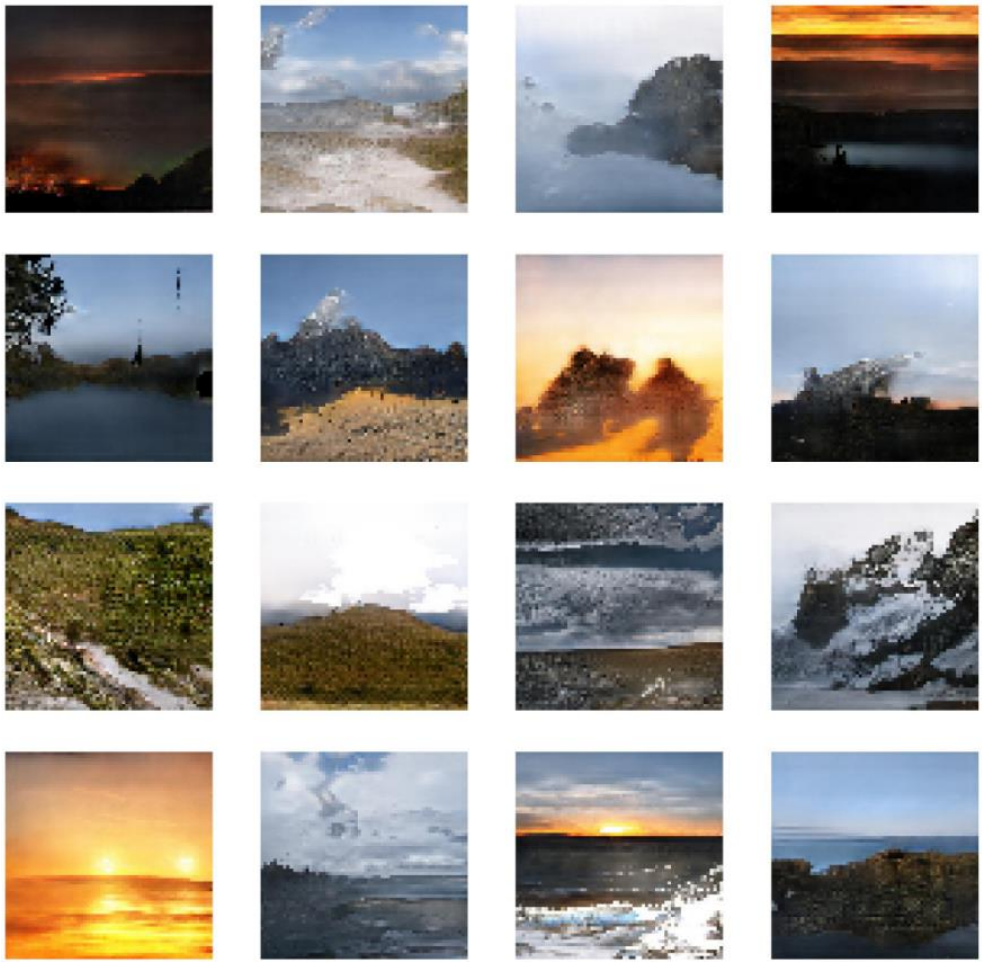

Fig. 21. Intermediate state of the model

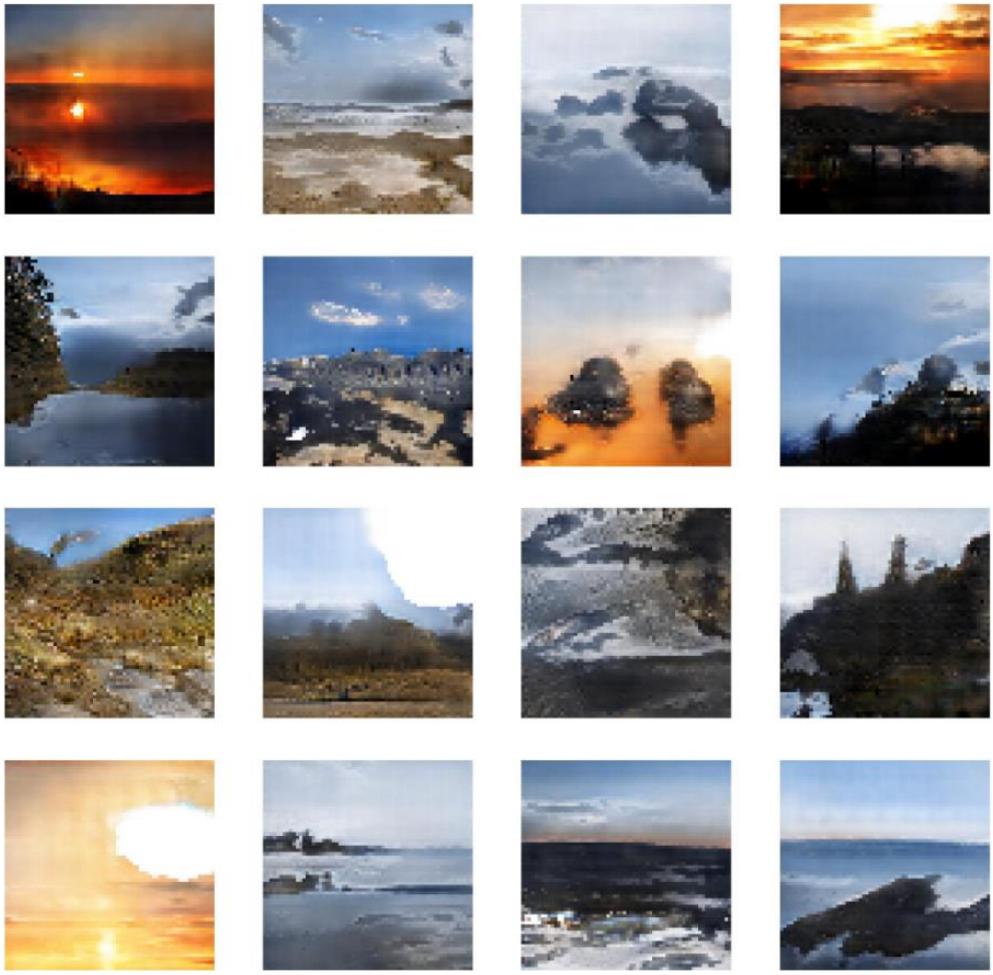

Fig. 22. Final state of the network

References

1. Ian J. Goodfellow et al. Generative Adversarial Networks. *Advances in Neural Information Processing Systems.* 2014. 3(11). P. 1-9 (DOI: 10.1145/3422622)
2. Course Goggle “Machine Learning”. GAN Variations. URL: https://developers.google.com/machine-learning/gan/applications?hl=ru (Date of application 05.02.2024).
3. Stratehiia rozvytku shtuchnoho intelektu v Ukraini. Za zahalnoiu redaktsiieiu A. I. Shevchenka. *Vydavnytstvo «Torpeda». Kyiv*. – 2023 r. S 305. (DOI: 10.15407/ development_strategy_2023).
4. Tero Karras, Samuli Laine, Timo Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. *Conference: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019. – P. 4396-4406. (DOI: 10.1109/CVPR.2019.00453)
5. Ian Goodfellow, Yoshua Bengio, Aaron Courville. Deep Learning (Adaptive Computation and Machine Learning series). *The MIT Press*. 2016. 800p.
6. Andrew Brock, Jeff Donahue, Karen Simonyan. «Large Scale GAN Training for High Fidelity Natural Image Synthesis». *Published as a conference paper at ICLR*. 2018. P. 1-35. (https://doi.org/10.48550/arXiv.1809.11096)
7. Pyvovar S.S., Korotka L.I. Mashynne navchannia dlia heneratsii hrafichnykh danykh z vykorystanniam bibliotek TensorFlow ta Keras. *Materialy VIII Mizhnarodnoi naukovo-tekhnichnoi konferentsii kompiuterne modeliuvannia ta optymizatsiia skladnykh system (1-3 lystopada 2023 roku m. Dnipro, Ukraina).* 2019. S. 128-130.
8. Prykladne mashynne navchannia za dopomohoiu Scikit-Learn ta TensorFlow: kontseptsii, instrumenty ta tekhniky stvorennia intelektualnykh system. *Orelen Zheron. Kyiv: «Dyalektyka»*, 2020. – 688s.
9. Online development service Google Colab. url: https://colab.research.google.com (Date of application 05.02.2024).
10. Klevzhyts D.D., Shvydko D.O., Korotka L.I. Heneratyvno-zmahalni merezhi u sferi stvorennia kontentu. *Shtuchnyi intelekt: dosiahnennia, vyklyky ta ryzyky. Mizhnarodna naukova konferentsiia (15-16 bereznia 2024 r., m. Kyiv).* 2024. S. 89-94.
11. Liangqu Long, Xiangming Zeng. Beginning Deep Learning with TensorFlow: Work with Keras, MNIST Data Sets, and Advanced Neural Networks. *1st ed. Edition*. 2022. 740 p.

12. Aurelien Geron. Hands-On Machine Learning with Scikit-Learn and TensorFlow. *O'Reilly Media, Inc.* 2017. 684 с.

13. RELU activation function. Official Tensorflow documentation. url: https://www.tensorflow.org/api_docs/python/tf/keras/layers/ReLU (Date of application 05.02.2024).

14. David Foster. Generative Deep Learning. *O'Reilly Media.* 2019. 327 с.

The article has been sent to the editors 03.06.24.
After processing 15.06.24.
Submitted for printing 28.06.24.