UDC: 004.627 https://doi.org/10.15407/jai2024.03.075

DEVELOPMENT OF A CONDITIONAL VARIATIONAL AUTOENCODER FOR HANDWRITTEN DIGIT RECOGNITION

B. Podoliak[1], T. Filimonova[2], Yu. Yurchenko[3]
[1,2,3]State University of Trade and Economics, Ukraine
19, Kyoto str., Kyiv, 02156
[1]b.podolyak.fit.122.20@knute.edu.ua
[2]t.filimonova@knute.edu.ua
[3]y.yurchenko@knute.edu.ua
[1]https://orcid.org/0009-0002-6667-3278
[2]https://orcid.org/0000-0001-9467-0141
[3]https://orcid.org/0000-0002-8047-7647

Abstract. This study examines the performance of Conditional Variational Autoencoder (CVAE) in handwritten digit recognition. Using the MNIST dataset, two variants of the CVAE models — convolutional and multilevel architecture — were developed and compared. The research methodology includes comprehensive data preprocessing, architecture design, training, and thorough evaluation processes. The obtained data highlight the better performance of the convolutional model-based CVAE in achieving recognition accuracy compared to its multilayer counterpart. Evaluation metrics include analysis of original and reconstructed images, comparison of hidden layer vector distribution patterns, and visualization of loss function dynamics. In addition, the study highlights the practical implications of CVAEs in various fields, highlighting their performance in digit recognition tasks due to their inherent robustness and extraordinary generalizability.

Keywords: neural network, autoencoder, encoder, decoder, hidden layer, image recognition.

Introduction

The ability to quickly and efficiently recognize handwritten numbers is an important element in the development of many areas of activity, such as banking, retail, health care, education and others. In this regard, scientists began to look for different methods to solve the problem of automating the recognition of handwritten digits and their rapid analysis. One of these methods is the use of artificial neural networks. Regardless of the problems that neural networks face when solving various problems, they remain a leading element for the further development of society. There are many types of artificial neural networks that can be used specifically for recognizing handwritten digits, but one of the best is the variational autoencoder (VAE). VAE is a type of artificial neural network that compresses the input data into a limited space (latent space), and after encoding, returns the output data to normal sizes [1]. It consists of two main parts, namely: an encoder and a decoder. The first part is responsible for reducing the dimensionality of the input data and transferring it to the latent space, which describes the probability distribution of the data, and the second part, in turn, is responsible for decoding the latent code and restoring the input data with maximum accuracy. This allows the VAE to generate new data similar to what was trained.

By itself, the variational autoencoder is quite a powerful tool for solving various problems, including the recognition of handwritten digits, but they, like any other types of artificial neural networks, have their drawbacks, such as high computational complexity, problems with generating realistic samples data and sensitivity to hyperparameters. In order to improve the performance of VAE and close these gaps, a condiitonal variational autoencoder (CVAE) was developed. CVAE is a modified version of the variational autoencoder that uses different techniques to improve its performance. Techniques used to improve the results of VAE may include changing the architecture of the encoder and decoder, regularization, or adding additional data, as CVAE can be trained on a larger dataset.

Analysis of recent research and publications

Autoencoders (AE) are powerful machine learning tools that have been successfully applied to tasks such as compression, layered learning, and data

generation. They are the basis of many deep learning models and are used in areas such as computer vision, natural language processing, and others. Recent research in this field reveals a variety of approaches to improve the performance and capabilities of autoencoders. Below is an overview of some key publications in recent years. The work [2] presents an improved information-theoretic methodology for understanding the dynamics of learning and designing AE, which affects the development of the optimal architecture of autoencoders, alternative methods of direct propagation learning. The theoretical foundations of the variational encoder are discussed in [3, 4]. The use of the noise cleaning criterion to improve the quality of generated images in noise-eliminating encoders (DAE) is proposed in the article [5]. Theoretical aspects, principles of development, advantages and disadvantages of CVAE are considered in work [6]. In these works, the advantage belongs to the theoretical aspects of the development of autoencoders of various types. In [7], an AE architecture was built based on a multilayer model, the distribution of hidden layer vectors, the loss function during training was investigated, and the recognition results were presented. VAE based on a multilayer model was implemented in work [8]: the architecture of the encoder and decoder, the distribution of the hidden layer vectors, the graph of the loss function, and the recognition result were visualized. The advantages and disadvantages of VAE compared to AE are also analyzed.

Main Material

As a result of the development of AE and VAE, based on a multilayer model [7, 8], sufficiently clear images were obtained. Let's try to improve the recognition ability. For this purpose, we will build an extended variation encoder. In order to figure out which type of model would be better suited for this task, convolutional model-based and multilayer model-based CVAE were developed.

The training sample of the MNIST dataset has class labels. We use this information and transfer it to the encoder and decoder (Fig. 1).

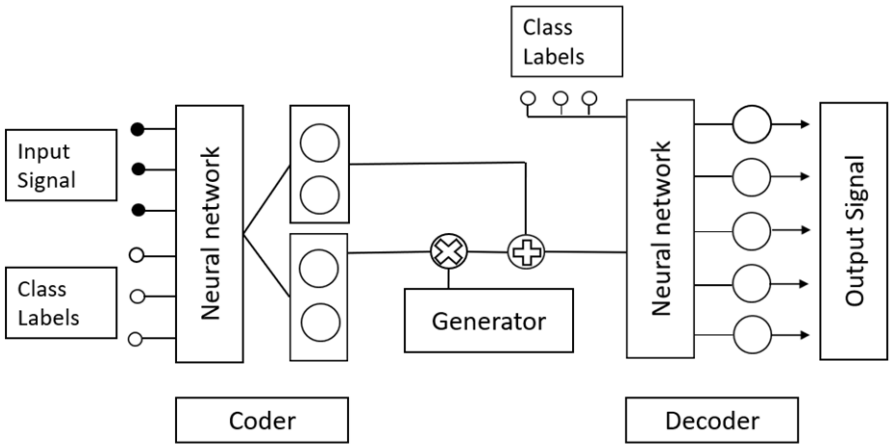


Fig. 1. The principle of operation of the conditional variational autoencoder

The first step in building such an autoencoder is adding the necessary libraries and normalizing the data that will be used for training. Next, you need to specify the number of neurons in the hidden layer of the encoder and decoder, as well as set a larger dimension to the hidden space than for the usual VAE to reproduce more information. This is followed by the creation of encoder (Fig. 2) and decoder (Fig. 3) architectures, in which Conv2D layers were added to reduce information loss and improve the quality of data reconstruction [9].

```
#Encoder Architecture
input_img = Input((28, 28, 1))
x = Conv2D(16, (3, 3), activation='relu', padding='same')(input_img)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(8, (3, 3), activation='relu', padding='same')(x)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Flatten()(x)
z_mean = Dense(hidden_dim)(x)
z_log_var = Dense(hidden_dim)(x)
```

Fig. 2. Encoder architecture

```
#Decoder Architecture
input_dec = Input(shape=(hidden_dim,))
d = Dense(7*7*8, activation='relu')(input_dec)
d = Reshape((7, 7, 8))(d)
d = Conv2D(8, (3, 3), activation='relu', padding='same')(d)
d = UpSampling2D((2, 2))(d)
d = Conv2D(16, (3, 3), activation='relu', padding='same')(d)
d = UpSampling2D((2, 2))(d)
decoded = Conv2D(1, (3, 3), activation='sigmoid', padding='same')(d)
```

Fig. 3. Decoder architecture

Having an encoder and a decoder, it is also necessary to specify the size of the validation sample to evaluate the overall performance and generating ability of the autoencoder, after which the model can be trained. In the course of training, we will additionally transfer class labels to the entrance. At the end of the work, the results of the CVAE work should be displayed in the form of 10 original and reconstructed images (Fig. 4).

In addition, the graphs of the distribution of the hidden layer vectors (Fig. 5) and the graphs of the loss function (Fig. 6) will be displayed based on both types of models in order to independently compare the quality of the work of autoencoders.

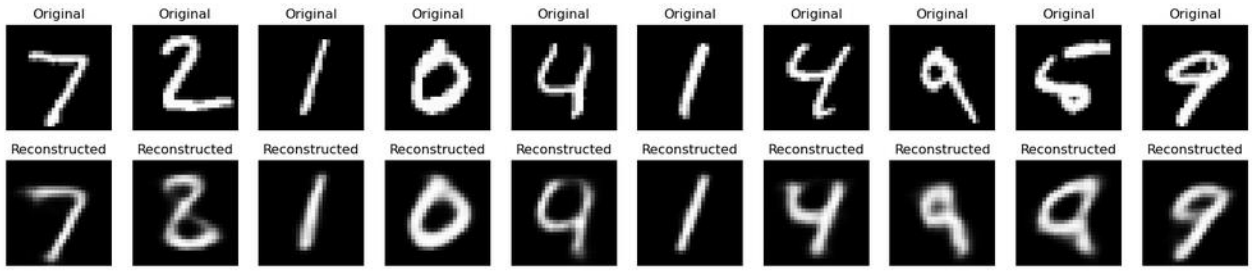
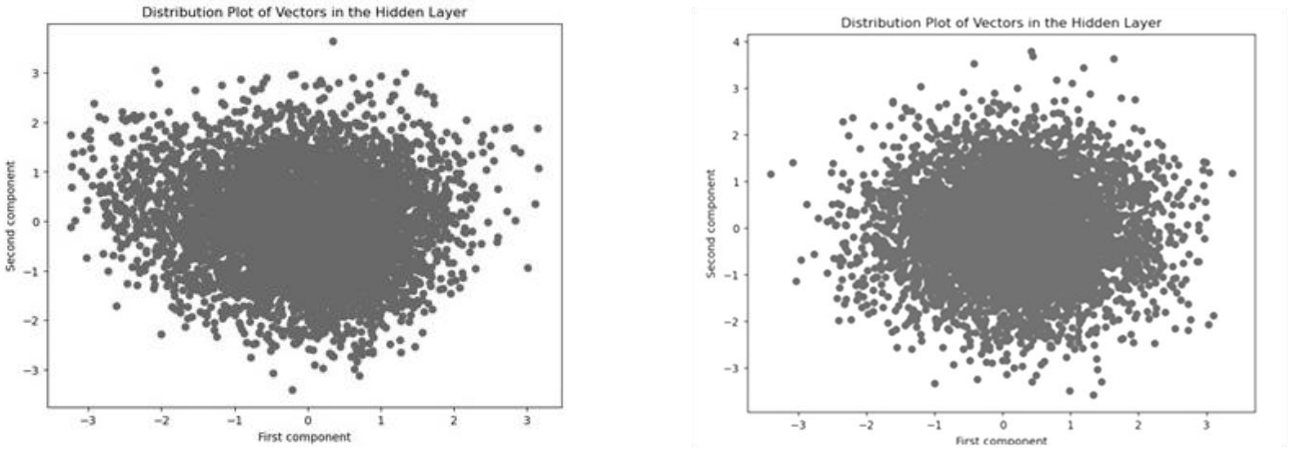


Fig. 4. Autoencoder Operation Visualization

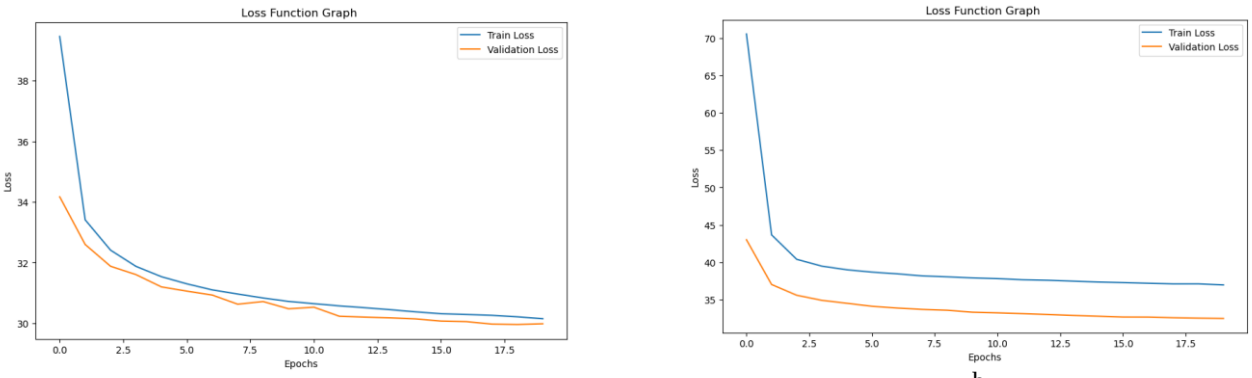


a b

Fig. 5. Distribution Plot of Vectors in the Hidden Layer
based on the Convolutional Model (a) and Multilayer Model (b)

By looking at fig. 4 it can be seen that the results are obtained with small differences. Image details are slightly blurred.

Figures 5 (a and b) show a characteristic picture of the formation of the input signal reflection area in the space of the hidden layer. If we choose arbitrary points within the formed area, we will get a meaningful image at the output of the decoder. When selecting a point outside this area, we expect to get an indeterminate image. From the figures, it can be concluded that CVAE based on the convolutional model is able to recognize images better. In principle, when developing CVAE based on the convolutional neural model and using the Keras multilayer model, we obtained satisfactory results, with small differences.



a b

Fig. 6. Loss Function Graph based on the Convolutional Model (a) and Multilayer Model (b)

From Figure 6, it can be concluded that the CVAE based on the convolutional model for the training and test data has smaller values at the corresponding training epochs compared to the CVAE based on the multilayer model using KERAS. Note that the value of the loss function can be reduced by selecting the hyperparameters of the model. A web service can be created to effectively display the operation of the autoencoder. This service can allow users to visually view the output of the autoencoder, such as the original and reconstructed images, and facilitate its use in various applications.

The created web service is the result of working with Flask, one of the powerful frameworks for Python. Initial setup includes creating a Flask application object using the ‘Flask(name)’ constructor, setting the file upload path, where the images will be stored (UPLOAD_FOLDER), configuring Flask, and defining a list of allowable file extensions (ALLOWED_EXTENSIONS) (Fig. 7).

```
app = Flask(__name__)

UPLOAD_FOLDER = 'static\\uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER


ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg'}
```

Fig. 7. Creating and configuring the application

This web application implements two key functions: ‘allowed_file(filename)’ and ‘index()’. The first function is used to check whether a file extension is allowed. It plays an important role in filtering downloaded files to avoid possible security threats. The second one is responsible for handling GET and POST requests. It interacts with client requests, receives data from users, processes them and provides a response (Fig. 8).

```
def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS


autoencoder = load_model('modelCifar100_83percents.h5')

@app.route('/', methods=['GET'])
def home():
    return render_template("index.html")

@app.route('/archive')
def archive():
    return render_template('archive.html', archive_data=archive_data)

archive_data = []

@app.route('/', methods=['GET', 'POST'])
def index():
    original_img = None
    decoded_img = None
```

Fig. 8. Application functions

When the app.py file is run directly, it automatically creates a local web server using the ‘app.run(debug=True)’ method. In debug mode (debug=True), Flask provides automatic page updates after changes to the source code. This is very convenient during development, as it allows you to immediately see the results of changes (Fig. 9).

```
if __name__ == '__main__':
    app.run(debug=True)
```

Fig. 9. Launching the application

The HTML markup language is used to create the display of a web page. CSS is used to stylize and improve the appearance of the web interface. The page interface has a simple and clear structure, with two main buttons: one for uploading images and the other for their processing and recognition.

After the images are uploaded, they go through an automatic processing process in the autoencoder, and the user receives the result in the form of a reconstructed image. As an experiment, noise was added to the input image in order to test the stability of the autoencoder to distortions. The obtained result can be seen in the figure (Fig. 10).

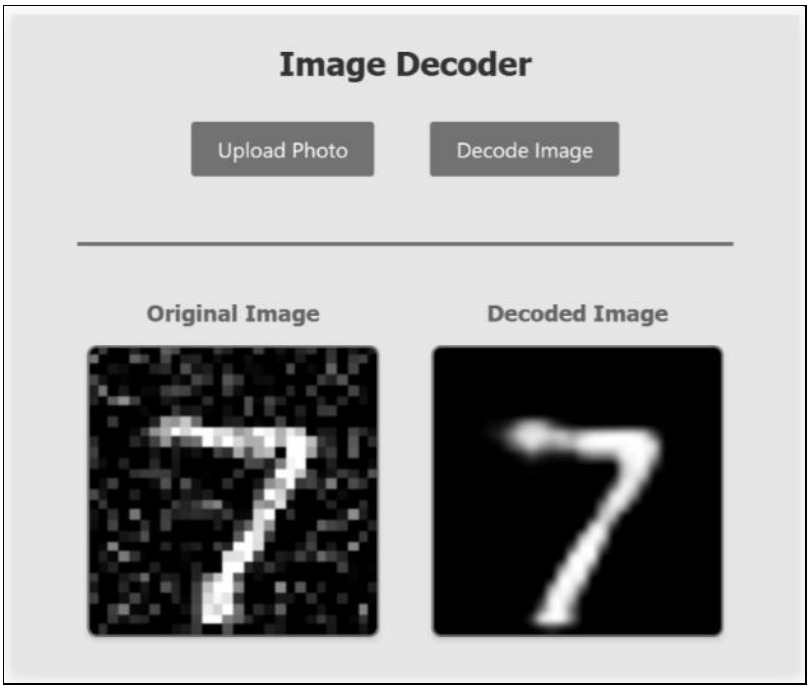


Fig. 10. Application interface and operation

Conclusion

As a result of the study, the conditional variational autoencoder for recognition of handwritten digits was built based on a convolutional neural network and a multilayer model using Keras. For both models, the graphs of the loss function for the training and test samples, the distribution of the hidden layer vectors, are constructed. As a result of comparing the models, it can be concluded that the extended variational autoencoder that uses the convolutional model has higher accuracy in recognition, which makes it more efficient to use.

Therefore, the use of CVAE is primarily intended for solving complex problems. It is used in various industries to recognize bank card numbers, check numbers, postal codes and telephone numbers. Due to its increased accuracy, robustness to noise and distortion, and better generalization to new data, this type of artificial neural network is one of the best options for use in handwritten digit recognition.

Furthermore, a web service was developed for the use of CVAE, which increases the accessibility and usability of this type of neural network. Users can conveniently upload images containing handwritten digits via a web interface, allowing CVAE to quickly and accurately identify existing digits. This web service opens the door to many possibilities of use for different fields of activity. From personal use for quick digit recognition in daily tasks to integration into business processes to automate data processing tasks, this service can become an indispensable tool for anyone who needs efficient recognition of handwritten digits.

References

1. Chollet F. Deap Learning with Python / F. Chollet – Manning Publications Co., 2021. – 504 p.
2. Shujian Yu. / Principe Understanding autoencoders with information theoretic concepts / Yu Shujian, Jose C. Principe – Neural Networks Volume, 2019. – pp. 104 – 123.
3. Hoegh. On Learning Useful Variational Autoencoder Representations: Applications in Audio Modelling and Hearing Loss Treatment / Hoegh, Rasmus M. Th. – Technical University of Denmark, 2022. – 157 P.
4. Doersch C. / Tutorial on Variational Autoencoders / C. Doersch, C. Mellon. – 3 Jan. 2021, 23P.
5. Jiwoong Im. D. / Denoising Criterion for Variational Auto-Encoding Framework / D. Jiwoong Im, S. Ahn, R. Memisevic, Yo. Bengio – Montreal Institute for Learning Algorithms. University of Montreal, 4 Jan. 2016. – pp. 1–14.
6. Kaae Sonderby C. / Ladder Variational Autoencoders / C. Kaae Sonderby, N. Raiko, L. Maaloe, S. Kaae Sonderby, O. Winther. – Advances in neural information processing systems, 2016. – pp. 1 – 9.
7. Podolyak B. Y., Filimonova T. O. Development of an autoencoder for handwritten digit recognition. Collection of abstracts of the XX International Scientific and Practical Conference «Mathematical Software for Intelligent Systems. MSIS – 2022», O. Honchar DNU, Dnipro, November 22-24, 2023. - P. 243 - 244.
8. Podoliak B., Filimonova T. Development of a variational autoencoder for handwritten digit recognition. Information Technologies: Theory and Practice. I (VII) International Scientific and Practical Conference of Higher Education Applicants and Young Scientists «Information Technologies: Theory and Practice». Abstracts of Papers (Dnipro, March 20-22, 2024) / Ministry of Education and Science of Ukraine, National Technical University «Dnipro Polytechnic». - Dnipro: Svidler A.L., - 2024. - P. 160 - 162. How can you optimize performance for VAEs?: Website. URL:https://www.linkedin.com/advice/0/how-can-you-optimize-performance-vaes-7xj1f (Accessed: 16.02.2024).

The article has been sent to the editors 07.08.24.
After processing 14.08.24.
Submitted for printing 30.09.24.

Copyright under license CCBY-NC-ND