International Journal of Artificial Intelligence,
ISSN 0974-0635; Int. J. Artif. Intell.
2026, Volume 24, Issue Number 2 [Published on 29/08/2026]

Deep Learning Algorithms for Real-time Antivirus Systems: A New Approach to Detect and Mitigate Cyber Threats

K. Asgarov[1]

[1]Department of Engineering Mathematics and Artificial Intelligence, Azerbaijan Technical University, AZ1073, 25 H. Javid Ave., Baku, AZERBAIJAN
Email : asgarovkamran2@gmail.com

ABSTRACT

The purpose of the study was to develop methods for integrating deep learning algorithms into antivirus systems to increase the effectiveness of detecting and neutralising cyber threats. Modern deep learning algorithms were analysed and tested, including convolutional neural networks, variational autoencoders, generative-adversarial networks, recurrent neural networks, and transformers. Each of the algorithms has been adapted for the tasks of detecting and neutralising threats, and integrated into prototypes of antivirus systems. The main results included antivirus platforms such as CrowdStrike, Sophos, Darktrace, SentinelOne, and CylancePROTECT. Convolutional neural networks have shown high efficiency in analysing programme behaviour and classifying network traffic, which makes them suitable for identifying known and new threats. Autoencoders have demonstrated their effectiveness in detecting anomalies, which increases the accuracy of recognising rare and unknown attacks. Generative-adversarial networks have made it possible to model threat scenarios and improve the adaptability of antivirus systems. Recurrent neural networks with their ability to analyse time dependencies have provided accurate predictions based on programme behaviour. Transformers proved to be the most effective for analysing the structure of the code, which made it possible to identify complex malware. In other words, the results of the study confirm that the implementation of deep learning algorithms in antivirus systems such as Darktrace and SentinelOne provides a higher level of protection and adaptability. Sophos and CylancePROTECT, due to the use of autoencoders and transformers, have shown their effectiveness in detecting complex threats with a minimum number of false positives. Thus, the proposed approach expands the capabilities of modern antivirus systems, making them more reliable and flexible in the face of rapidly changing cyber threats.

Keywords: Convolutional neural networks, variational autoencoders, generative-adversarial networks, recurrent neural networks, generative transformers.

Mathematics Subject Classification: 68T07, 68M25, 68T10

Journal of Economic Literature (JEL) Classification: C45, C63, L86

1. INTRODUCTION

The modern digital age is accompanied by an increase in data volumes and the complexity of their processing, which makes artificial intelligence (AI) technologies key for information analysis. Machine Learning (ML), in turn, is an approach in which algorithms are trained on data, identifying patterns and making decisions without explicit programming. Deep Learning (DL), being a subcategory of ML, uses

multilevel artificial neural networks that can extract complex characteristics from data. This allows DL to handle high-complexity tasks such as image, text, time series, and network data processing (Rudenko et al., 2024).

As for antivirus systems, they face a number of problems related to the rapid evolution of cyber threats, the high complexity of attacks and large amounts of data for analysis. Conventional threat detection methods are no longer able to effectively deal with new, rapidly changing viruses and programmes with unknown behaviour (Aizstrauta and Ginters, 2016; Ginters et al., 2020). This leads to a high number of false positives and missed threats. The problem lies in the need to develop more adaptive and accurate systems capable of responding quickly to new threats (Guliyev, 2019; Kondratenko and Kondratenko, 2014). In this context, DL algorithms can significantly improve the effectiveness of antivirus platforms by reducing the number of false positives and increasing adaptability to new types of threats.

It is worth mentioning the study by Huseynova (2022), which examined DL methods, especially their classification and structuring. The article summarised modern approaches to using deep neural networks, such as Convolutional Neural Network (CNN), to solve cybersecurity problems, with an emphasis on their advantages in processing large amounts of data and superiority over conventional ML methods. Zhou and Liang (2024) investigated the applications of AI technologies, including ML and DL, to monitor the cybersecurity of computer networks. They developed models for detecting intrusions, anomalies, and malware, the effectiveness of which was evaluated based on open datasets. In addition, Sufi (2024) investigated the importance of cyber-threat intelligence in a dynamically changing threat environment. The research considered the types, processes, and platforms of intelligence.

On the other hand, Bala et al. (2022) proposed an image classifier based on learning transfer using CNN. The model achieved 97.24% accuracy, surpassing the hybrid model based on deep probabilistic and recurrent networks with short-term memory, as well as conventional ML methods. Shaikh et al.(2024) developed a hybrid Long Short-Term Memory (CNN-LSTM) model based on DL for detecting Distributed Denial-of-Service (DDoS) attacks. This model achieved 99.86% accuracy on the dataset used. Moreover, Aziz and Alfoudi (2023) conducted a review of modern intrusion detection models using ML and DL methods, and feature classification and selection strategies. The researchers emphasised the effectiveness of MultiTree and adaptive voting algorithms, which demonstrated 99.98% accuracy. Wu et al. (2022) proposed a stack-based DL method for detecting cyber-attacks in systems of Supervisory Control and Data Acquisition (SCADA), superior to standard algorithms and conventional methods of protection. The use of real data from laboratory SCADA systems showed the high accuracy of the model and allowed it to be optimised by analysing the significance of the features (Gospodinova, 2022; 2023).

In turn, Ikpe et al. (2024) reviewed the application of DL algorithms in industrial machines, highlighting their potential to improve productivity and product quality. The study identified key advantages such as big data analysis and accurate forecasts, and challenges related to high demands on data, resources, and expertise. Doris and Gracias (2024) investigated the use of DL in video surveillance, emphasising its importance for improving real-time security. They considered algorithms such as CNN, RNN, transformers, and lightweight models for performing tasks, including object detection and anomalies,

with an emphasis on the challenges of computing resources and a limited amount of labelled data. Additionally, Jiang et al. (2024) presented the DL-based OpenPose algorithm in real time, which effectively tracks human postures in various conditions. Using the YOLO network to create a dataset and the OpenPose method to detect key body points, the algorithm also applies the Kalman filter to multitask tracking and predicting the states of objects in the shaded area.

Thus, this study focused on creating an approach to integrating DL algorithms into antivirus systems to increase the effectiveness of detecting and neutralising cyber threats. However, the above-mentioned articles mainly focused on the use of DL to detect cyber threats, but without integrating such algorithms into antivirus platforms. The research tasks included the analysis of DL algorithms in the context of cybersecurity, their testing, and integration into antivirus systems.

2. MATERIALS AND METHODS

A comprehensive analysis of DL algorithms and their integration into antivirus systems was carried out. The research was based on the use of the Python programming language, using the Online Python compiler (2025), which provided convenience for development, testing, and visualisation of results. The focus was on the application of modern approaches such as CNN, Variational Autoencoder (VAE), Generative Adversarial Nets (GAN), Recurrent Neural Network (RNN), and transformers, in particular, Bidirectional Encoder Representations from Transformers (BERT) and Generative Pre-trained Transformer (GPT). CNN were chosen because of their ability to effectively analyse spatial dependencies in data, VAE were used to detect anomalies because they are good at detecting deviations from normal behaviour, and GAN were chosen because they allow threat scenarios to be modelled and cybersecurity systems to be trained on synthetic data. RNN, in particular LSTM, were also used to analyse time dependencies in network traffic and malware behaviour, while transformers, in particular, BERT and GPT, were used to analyse the code structure.

At the stage of algorithm analysis, synthetic data was generated to test various models. VAE were used to detect anomalies. Data was created that mimics normal behaviour and anomalies, after which a simplified autoencoder capable of detecting deviations was developed and trained. The model looked for abnormal elements that were identified by a given threshold. In turn, CNNs were used to classify network data, where each sample represented simplified network traffic. The CNN model analysed the data, identifying features specific to threats, and demonstrated predictions based on test samples. A generator and discriminator were developed for GAN, which were also trained on synthetic data. They created new data samples similar to real ones, which allowed simulating potential threats and training the discriminator to distinguish real data from artificially created ones. During the GAN training, the parameters of both models were updated, and the results were evaluated based on the losses of the generator and discriminator.

For time series analysis, RNN were used, which were trained on time-dependent data sequences and also included LSTM. Time series made it possible to simulate the dynamic behaviour of systems, and the results of the network included predictions of hidden states. Transformers were used to analyse

sequential data. This model allowed considering global relationships within the data, providing a more accurate transformation of sequences. In addition, the study included a comparison of all the algorithms considered according to their advantages and limitations. An example of their application in the tasks of detecting and neutralising cyber threats was proposed for each of them. In addition, the study included a comparison of all the algorithms considered according to their advantages and limitations. The comparison criteria included the effectiveness of detecting cyber threats, adaptability to new attacks, feature selection, ability to process various types of data, interpretability, learnability, and computational complexity.

In particular, the possibilities of adapting these algorithms for antivirus systems that provide protection from both known and new threats were investigated. Five leading platforms were considered for this purpose. For example, CrowdStrike (2024) used CNN to analyse files and programme behaviour, which helped to identify both known and unknown threats. Sophos (2024) used autoencoders to extract features and detect anomalies in data, which increases the effectiveness of protection. The Darktrace platform (2024) implemented GAN to create synthetic data and simulate attacks, which helps training and improves the system's ability to identify complex threats. On the other hand, SentinelOne (2024) used RNN to analyse time dependencies, which makes it possible to predict threats based on programme behaviour. CylancePROTECT (2024) used transformers to analyse the structure of the code, which helps to detect complex malware. The ratios of algorithms and examples of platforms were visualised for each antivirus platform.

3. RESULTS

3.1. Autoencoder and CNN algorithms in the context of cybersecurity

Modern cyber threats are becoming more complex, and their detection requires the use of advanced technologies. DL algorithms provide powerful tools for analysing large amounts of data and detecting threats in real time. It is worth considering the existing DL algorithms used in cybersecurity, for example, VAE. In general, autoencoders are neural networks trained on the task of reconstructing input data (Sokoliuk et al., 2021). They consist of two parts: an encoder and a decoder. The encoder compresses the input data into a compact representation (feature vector), and the decoder reconstructs the original data from this representation (Figure 1).

```python
import numpy as np

np.random.seed(42)
normal_data = np.random.normal(0, 1, (100, 128))
anomalous_data = np.random.uniform(-5, 5, (5,
128))
full_data = np.vstack([normal_data,
anomalous_data])

class Autoencoder:
    def __init__(self, input_dim, encoding_dim):
        self.input_dim = input_dim
        self.encoding_dim = encoding_dim
        self.weights_encoder = np.random.normal(0,
0.1, (input_dim, encoding_dim))
        self.weights_decoder = np.random.normal(0,
0.1, (encoding_dim, input_dim))

    def encode(self, x):
        return np.dot(x, self.weights_encoder)

    def decode(self, encoded):
        return np.dot(encoded,
self.weights_decoder)

    def train(self, data, epochs=100, lr=0.01):
        for epoch in range(epochs):
            encoded = self.encode(data)
            reconstructed = self.decode(encoded)
            loss = np.mean((data - reconstructed)
** 2)
            grad_decoder = -2 * np.dot(encoded.T,
(data - reconstructed))
            grad_encoder = -2 * np.dot(data.T,
(data -
reconstructed).dot(self.weights_decoder.T))
            self.weights_decoder -= lr *
grad_decoder / data.shape[0]
            self.weights_encoder -= lr *
grad_encoder / data.shape[0]
            if epoch % 10 == 0:
                print(f"Epoch {epoch}, Loss:
{loss:.4f}")

    def predict(self, x):
        encoded = self.encode(x)
        reconstructed = self.decode(encoded)
        return reconstructed

input_dim = 128
encoding_dim = 32
autoencoder = Autoencoder(input_dim, encoding_dim)
autoencoder.train(normal_data, epochs=50)

reconstructions = autoencoder.predict(full_data)
loss = np.mean((full_data - reconstructions) ** 2,
axis=1)
threshold = 1.5
anomalies = loss > threshold

print(f"Detected {np.sum(anomalies)} anomalies.")
```

Figure 1. An implementation code for an auto-encoder for detecting anomalies.

Source: compiled by the author.

The programme trains the autoencoder on normal data, minimising the recovery error. After the training is completed, the data is passed through the model and the recovery error is calculated for each sample. If the error exceeds the specified threshold, the sample is classified as an anomaly (Zinchenko et al., 2020). During the training process, there is a decrease in the recovery error, which indicates successful training of the model. In the final data, the programme finds 5 anomalies, which corresponds to the number of abnormal samples in the test dataset (Figure 2).

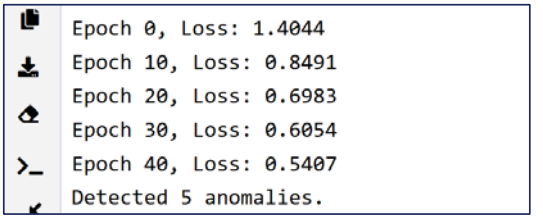

Figure 2. Result of the auto-encoder programme.

Source: created by the author based on Online Python (2025).

To increase the effectiveness of detecting and neutralising cyber threats in real time, it is proposed to use improved autoencoders integrated with streaming data processing mechanisms, which will allow analysing input data without delay. To increase accuracy, hybrid architectures can be implemented that combine autoencoders with neural networks, which will ensure that time dependencies are considered in the analysed data. It is also recommended to include metrics of abnormal behaviour based on the unique characteristics of network attacks in the learning process (Kozhakhmetova et al., 2024). This approach will be further enhanced by the use of active learning, in which the model adapts based on new threats identified in real time, which will significantly increase the level of security of information systems.

In turn, it is worth considering CNNs, which are designed to analyse data with a spatial structure, such as images or time series. They extract high-level features from the input data thanks to convolutional and pulling layers (Figure 3).

```python
import numpy as np

np.random.seed(42)
x_train = np.random.random((100, 32, 32))
y_train = np.random.randint(2, size=(100,))

class SimpleCNN:
    def __init__(self, input_shape):
        self.weights = np.random.normal(0, 0.1,
(3, 3))
        self.bias = np.random.normal(0, 0.1, 1)

    def convolve(self, input_data):
        output = np.zeros((input_data.shape[0] -
2, input_data.shape[1] - 2))
        for i in range(output.shape[0]):
            for j in range(output.shape[1]):
                patch = input_data[i:i+3, j:j+3]
                output[i, j] = np.sum(patch *
self.weights) + self.bias
        return output

    def flatten(self, input_data):
        return input_data.flatten()

    def classify(self, input_data):
        flattened = self.flatten(input_data)
        score = np.sum(flattened)
        return 1 if score > 0 else 0

    def predict(self, data):
        predictions = []
        for sample in data:
            conv_output = self.convolve(sample)
            prediction =
self.classify(conv_output)
            predictions.append(prediction)
        return np.array(predictions)

model = SimpleCNN(input_shape=(32, 32))
y_pred = model.predict(x_train[:10])
print(f"Predictions: {y_pred}")
```

Figure 3. A CNN implementation code for classifying network traffic.

Source: compiled by the author.

The programme uses a simplified CNN model that applies convolution to the data, extracts features, and classifies them based on their combined values. The model classifies the generated data, demonstrating the basic principles of convolutional networks in structured data analysis. The result shows that the model classified all the data as belonging to the same class, since the total values of the extracted features in each sample exceed the set threshold (Figure 4).

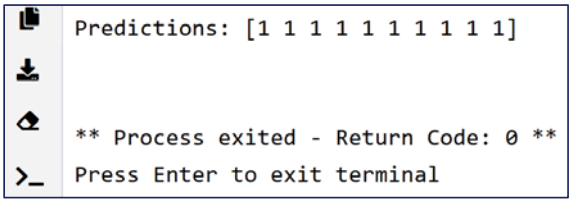

Figure 4. Result of the CNN programme.

Source: created by the author based on Online Python (2025).

It is recommended to use CNN to analyse network traffic and detect cyber threats, with an emphasis on the use of layered architectures. These architectures can be integrated with dynamic weight update mechanisms, which will allow the model to be adapted to changing patterns of attacks in real time. In addition, it is proposed to use deep CNNs with attention mechanisms to analyse complex network data, such as time series with multiple parameters. To increase the system's resilience to security circumvention attempts, it is recommended to combine CNNs with data augmentation methods that generate training sets that are as close as possible to real threats. This approach will enhance the ability of models to identify important spatial signs of attacks and ensure high accuracy in detecting malicious activity.

3.2. Deep learning: GAN and RNN algorithms

It is necessary to disassemble GAN, which are a class of DL algorithms consisting of two parts: a generator and a discriminator. The generator creates fake data in an attempt to simulate real data, while the discriminator evaluates whether the data is real or generated. These networks compete with each other, which allows the generator to improve in creating more realistic data (Figure 5).

```
import numpy as np

class Generator:
    def __init__(self, noise_dim, output_dim):
        self.noise_dim = noise_dim
        self.output_dim = output_dim
        self.weights = np.random.normal(0, 0.1,
(noise_dim, output_dim))

    def generate(self, noise):
        return np.dot(noise, self.weights)

    def update_weights(self, grad, lr):
        self.weights -= lr * grad

class Discriminator:
    def __init__(self, input_dim):
        self.input_dim = input_dim
        self.weights = np.random.normal(0, 0.1,
(input_dim, 1))

    def discriminate(self, data):
        return np.dot(data, self.weights)

    def update_weights(self, grad, lr):
        self.weights -= lr * grad

def train_gan(generator, discriminator,
epochs=100, batch_size=32, lr=0.01):
    for epoch in range(epochs):
        noise = np.random.normal(0, 1,
(batch_size, generator.noise_dim))
        generated_data = generator.generate(noise)

        real_data = np.random.normal(0, 1,
(batch_size, discriminator.input_dim))

        real_output =
discriminator.discriminate(real_data)
        fake_output =
discriminator.discriminate(generated_data)

        loss_real = np.mean((real_output - 1) **
2)
        loss_fake = np.mean(fake_output ** 2)
        loss_discriminator = loss_real + loss_fake

        grad_real = (real_data.T @ (real_output -
1)) / batch_size
        grad_fake = (generated_data.T @
fake_output) / batch_size
        discriminator.update_weights(grad_real +
grad_fake, lr)

        fake_output =
discriminator.discriminate(generated_data)

        loss_generator = np.mean((fake_output - 1)
** 2)

        grad_generator = (noise.T @ (fake_output -
1)) / batch_size
        generator.update_weights(grad_generator,
lr)

        if epoch % 10 == 0:
            print(f"Epoch {epoch}, Loss
(Discriminator): {loss_discriminator:.4f}, Loss
(Generator): {loss_generator:.4f}")

generator = Generator(noise_dim=100,
output_dim=128)
discriminator = Discriminator(input_dim=128)

train_gan(generator, discriminator, epochs=100)
```

Figure 5. GAN implementation code for generating fake data.

Source: compiled by the author.

This code implements the GAN architecture, which includes a generator and a discriminator. The generator creates synthetic data based on random noise, and the discriminator tries to distinguish real data from generated data. Both models are trained by optimising their weights: the discriminator minimises classification errors, and the generator is trained to create data that the discriminator cannot distinguish from the real ones. As a result, both models demonstrate a gradual reduction in errors during the learning process, illustrating the dynamic confrontation between the generator and the discriminator (Figure 6).

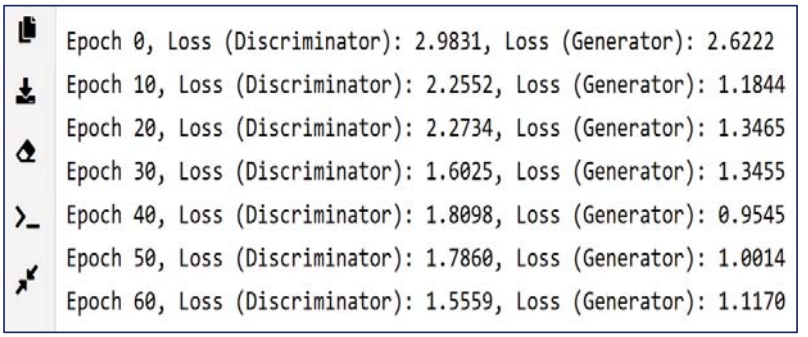

Figure 6. Result of the GAN programme.

Source: created by the author based on Online Python (2025).

It is recommended to use GAN for modelling and analysing cyber threats in real time, as such a generator can be configured to create synthetic data that mimics the behaviour of various cyber-attacks, which will allow anti-virus systems to be trained and tested on a larger number of scenarios. The discriminator can be used to classify new data and detect previously unknown threats. It is proposed to integrate GAN with real-time monitoring systems, which will allow detecting anomalies as they occur. To increase efficiency, advanced learning methods such as gradient coupling or data mixing should be used, and architectures with multi-level attention that can adapt to complex and dynamically changing threats (Biloshchytskyi et al., 2026). This approach will ensure higher accuracy and timely detection of attacks.

On the other hand, RNN are a class of neural networks specifically designed to work with sequences of data. They use hidden states that preserve information about previous steps, making them suitable for time series analysis (Figure 7).

```
import numpy as np

np.random.seed(42)
time_series = np.sin(np.linspace(0, 100, 200)) +
np.random.normal(0, 0.1, 200)

class SimpleRNN:
    def __init__(self, input_size, hidden_size):
        self.hidden_size = hidden_size
        self.weights_input = np.random.normal(0,
0.1, (input_size, hidden_size))
        self.weights_hidden = np.random.normal(0,
0.1, (hidden_size, hidden_size))
        self.bias = np.zeros(hidden_size)

    def forward(self, x):
        h = np.zeros(self.hidden_size)
        for t in range(x.shape[0]):
            h = np.tanh(np.dot(x[t],
self.weights_input) + np.dot(h,
self.weights_hidden) + self.bias)
        return h

rnn = SimpleRNN(input_size=1, hidden_size=10)
output = rnn.forward(time_series.reshape(-1, 1))
print(f"Final hidden state: {output}")
```

Figure 7. RNN implementation for time series analysis.

Source: compiled by the author.

This programme implements a simple RNN for analysing synthetic time series. This RNN uses weights for the input data, hidden state, and offset, which are updated at each time step. At each step, the hidden state is calculated, combining the current input data and the previous state through the hyperbolic tangent activation function. This structure allows the model to account for the relationship between successive points in the time series. As a result, the final hidden state is a vector that contains generalised information about the entire sequence of the time series processed by the model, and can be used for further analysis or classification (Figure 8).

```
Final hidden state: [ 2.04170113e-04 -3.71531735e-03 -1.08265084e-01 -8.48420515e-02
  5.04910765e-02  4.60692637e-02 -2.62396044e-02 -2.36026270e-02
 -6.53990935e-02 -2.39775317e-01]


** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

Figure 8. Result of the RNN programme.

Source: created by the author based on Online Python (2025).

RNN are recommended to be adapted for cybersecurity, combining them with attention mechanisms and advanced architectures such as LSTM or improved recurrent blocks. This will allow for a more accurate analysis of time series, for example, network traffic or event logs, considering their time dependencies. It is also proposed to use RNN to simulate attack sequences, which will help to predict further actions of intruders and ensure prompt response to threats.

3.3. Transformers and comparison of DL algorithms

Transformers (for example, BERT and GPT) are a class of DL models designed to work with sequences of data, but without using recursion. They are based on an attention mechanism that allows the model to focus on important parts of the sequence when processing data. Transformers have found wide application in natural language processing tasks, but they can also be adapted for cybersecurity applications (Figure 9).

```
import numpy as np

np.random.seed(42)
sequence = np.random.random((10, 64))

class Transformer:
    def __init__(self, input_dim, model_dim,
num_heads):
        self.weights_query = np.random.normal(0,
0.1, (input_dim, model_dim))
        self.weights_key = np.random.normal(0,
0.1, (input_dim, model_dim))
        self.weights_value = np.random.normal(0,
0.1, (input_dim, model_dim))
        self.num_heads = num_heads
        self.scale = np.sqrt(model_dim)

    def scaled_dot_product_attention(self, query,
key, value):
        scores = np.dot(query, key.T) / self.scale
        weights = np.exp(scores) /
np.sum(np.exp(scores), axis=-1, keepdims=True)
        return np.dot(weights, value)

    def forward(self, x):
        query = np.dot(x, self.weights_query)
        key = np.dot(x, self.weights_key)
        value = np.dot(x, self.weights_value)
        return
self.scaled_dot_product_attention(query, key,
value)

transformer = Transformer(input_dim=64,
model_dim=128, num_heads=4)
output = transformer.forward(sequence)
print(f"Transformed sequence: {output}")
```

Figure 9. Implementing a transformer for sequence analysis.

Source: compiled by the author.

The code implements a simplified transformer that includes an attention mechanism that allows the model to process sequences of data, revealing important connections between elements. The model calculates queries, keys, and values, and then applies an attention mechanism to transform the input data. As a result, the programme returns a transformed sequence in which each element contains information about its relationship with other elements of the original sequence (Figure 10).

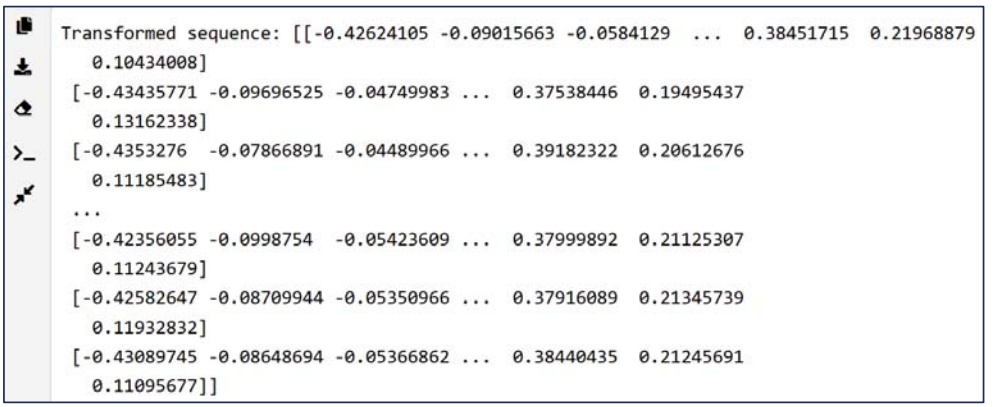

Figure 10. Result of the transformer programme.

Source: created by the author based on Online Python (2025).

It is recommended to adapt transformers for cybersecurity tasks, including processing large amounts of data and detecting complex attacks. The combination of transformers with network traffic processing, log analysis, or anomaly detection methods can ensure high accuracy and speed of threat detection. A particularly promising area is the use of pre-trained transformers such as BERT and GPT to accelerate the analysis and development of specialised models for specific threats.

For a better understanding, it is worth comparing the considered algorithms, indicating their advantages and disadvantages (Table 1).

Table 1: Comparison of DL algorithms.

Algorithm	Advantages	Limitations
VAE	Ease of implementation and interpretation	Limited ability to account for time dependencies
	Work well for detecting anomalies.	
	Effective for working with large datasets	Require setting a threshold for classifying anomalies.
CNN	Do an excellent job of analysing data with spatial structure.	Can be resource-intensive when learning deep architectures.

	They are good at extracting features from network traffic and other complex data.	Less suitable for sequential data
GAN	Effective for data generation	Sensitive to hyperparameter settings
	Can improve model training by using synthetic data.	Difficulty and slowness of learning due to the competitive nature
RNN	Time dependencies are considered	Prone to disappearing/exploding gradients
	Suitable for time series analysis	High computational complexity
BERT, GPT	Do not suffer from disappearing gradients	May be less effective for long sequences.
	Suitable for tasks with a large amount of data	Require a lot of data for training

Source: compiled by the author

Thus, it is better to use VAE if it is necessary to detect anomalies in the data, for example, to identify unusual user behaviour or deviations in network traffic metrics. CNNs are suitable for data analysis tasks with explicit spatial structure, such as network traffic visualisation, data clustering, or time series analysis with spatial partitioning. Moreover, GAN are suitable for generating data simulating various types of attacks, which is useful for testing cybersecurity systems. RNN are effective for sequential data analysis, for example, to predict the next event in the action log or to identify attack patterns in time series. And transformers (BERT, GPT) are suitable for processing large amounts of data, for example, analysing event logs or data streams.

3.4. Integration of deep learning algorithms into antivirus systems

Conventional antivirus systems rely on two main methods of threat detection: signature and heuristic. Signature detection is based on checking files and software images against pre-known virus signatures. The heuristic method searches for suspicious programme behaviour, which allows identifying unknown threats, but can also lead to more false positives. Both of these methods are limited in their ability to effectively respond to rapidly changing and complex threats, leading to the need to integrate more advanced technologies such as DL algorithms.

Overall, DL has become one of the most promising approaches for improving antivirus systems, providing higher detection accuracy and the ability to adapt to new threats (Brescia et al., 2023a; Petrov et al., 2023). In this context, it is worth considering examples of antivirus solutions using DL algorithms. One example is CrowdStrike (2024). CrowdStrike is a cloud-based endpoint protection platform that uses AI technologies to provide real-time security. This system actively uses behavioural analysis and ML to analyse programme behaviour and detect anomalies, preventing both known and unknown threats. CrowdStrike uses CNN to classify and analyse files, programme behaviour, and to prevent attacks. CNN analyses the features and patterns extracted from the data, which allows effectively

identifying suspicious activities, even if they were not previously registered in databases (Kabdoldina et al., 2022).

Sophos, for its part, is an endpoint protection solution that integrates multiple technologies to protect against complex threats (2024). It includes proactive protection against exploits, the use of AI to detect threats, and automatic incident response. Sophos uses autoencoders to automatically extract features from large amounts of data, which helps to effectively detect anomalies and new viruses. This model can identify unknown threats by analysing deviations from normal behaviour, and simultaneously has a minimal number of false positives.

The Darktrace platform is an ML-based cyber defence solution that detects anomalies in real time (2024). The system actively uses self-learning algorithms to analyse behaviour and identify attacks, even if they have not been registered before. Darktrace uses GAN to generate synthetic data and create behavioural models, which helps in training and improving the accuracy of the system in detecting complex attacks and new types of threats. GAN allow simulating possible attack scenarios, which improves the system's ability to adapt and predict threats (Brescia et al., 2023b).

In turn, SentinelOne (2024) is an endpoint protection system that uses both static and dynamic analysis to protect against a variety of threats, including viruses, rootkits, and Trojans. This platform is known for its high efficiency in threat detection and real-time response. SentinelOne uses RNN to analyse time dependencies in data, which helps to effectively detect sequences of actions characteristic of attacks. These networks can analyse a series of events and predict the likelihood of attacks based on the behaviour of programmes and users during execution.

In addition, mention should be made of the CylancePROTECT platform, which uses AI technologies to prevent attacks before they occur (2024). The platform is based on the use of ML algorithms to block malicious programmes even before they are executed. CylancePROTECT uses transformers to analyse the structure of the code and predict potential threats based on the context of the code. Transformers can consider global dependencies and connections in data, which helps to more accurately identify complex malware based on non-standard approaches.

Thus, each antivirus platform has its own machine/deep learning algorithms (Figure 11).

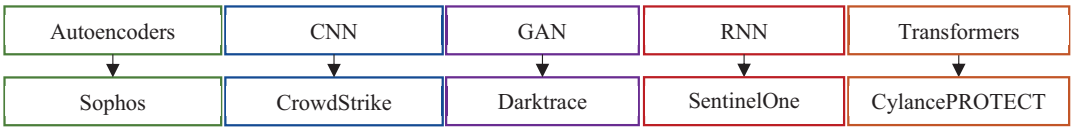

Figure 11. Relationship between algorithms and antivirus platforms.

Source: compiled by the author.

As part of the discussion of DL algorithms in the context of antivirus systems, each of the algorithms considered has its own unique role in ensuring security. For example, CNNs are used in CrowdStrike to analyse programme behaviour and classify files, which helps to identify known and unknown threats. Autoencoders are used in Sophos to extract features and detect anomalies in data, which increases

the effectiveness of protection against new viruses. GAN are being implemented in Darktrace to generate synthetic data, improving the accuracy of detecting complex attacks and modelling the behaviour of attackers. RNN in SentinelOne analyse sequences of events, helping to predict possible threats based on time dependencies. Ultimately, transformers in CylancePROTECT help to analyse the code structure and predict threats based on global dependencies, which increases the accuracy in detecting non-standard malware. All these methods allow modern antivirus platforms to adapt to new, rapidly changing threats, increasing their overall effectiveness, and reducing the number of false positives.

4. DISCUSSION

The study results include various neural networks, including CNN, to detect and neutralise cyber threats. Muñoz et al. (2024) proposed an approach using CNN to analyse ultrasound images of the lungs, automating the identification and interpretation of key features. In both studies, CNNs demonstrated high efficiency in processing complex data, but the results obtained in the current study show an advantage in adaptability and versatility, confirming that DL algorithms can be successfully integrated into various fields to improve the accuracy of analysis and automate processes. In addition, the results of the study demonstrate the use of DL to solve various tasks, such as data analysis and threat detection. Unlike Abo El-Ela and Hamdi (2024), who used DL models such as Multi-Layer Perceptron (MLP) and Gated Recurrent Units (GRU) to optimise routes, the current used CNN and RNN to analyse programme behaviour and detect anomalies. Although both studies demonstrate the high efficiency of models in data analysis, the study showed that neural networks are better suited for threat classification, while the methods used in the mentioned study proved to be effective specifically in adaptive routing tasks.

The results showed that DL algorithms are successfully used to analyse programme behaviour and neutralise cyber threats in real time, providing high accuracy and adaptability in changing conditions. Nithya Raju et al. (2024) also used DL methods for real-time analysis, namely for automatic diagnosis of diseases. Despite the differences in the context, the results of the mentioned work complement the current ones, confirming the versatility and potential of using these technologies in various fields. As in the conducted research, Shenge (2024) used DL to solve problems in the field of cybersecurity. However, unlike the approach of the above study, which is limited to a single task, the current study applied more versatile DL algorithms to solve a wide range of tasks in this field, including anomaly detection and programme behaviour analysis. This allows the considered methods to be more flexible and adaptable to various threats in real time, which makes the results of this study more scalable and multitasking.

In contrast to this study, where the main focus is on the application of DL algorithms to detect cyber threats, the study by Patel (2024) focused on optimising training and reducing the training time of deep network models. While the mentioned study covers the theoretical aspects of improving the accuracy and performance of training algorithms, the current study confirms that DL can effectively solve applied tasks in real time, offering more specific solutions for adapting to dynamically changing threats. Similarly to this article, which explores the possibilities of integrating DL into antivirus systems to analyse

programme behaviour and detect anomalies, the study by Ramesh et al. (2023) proposed an intelligent DL model for virus detection and processing, and antivirus security systems were also considered. However, the current study demonstrates a more versatile approach, allowing adaptation to a wide range of threats and providing more flexible and scalable protection.

In this study, DL algorithms were used to analyse programme behaviour and detect anomalies in real time, ensuring effective detection and neutralisation of cyber threats. On the other hand, Chen (2025) proposed a DL-based system for forecasting and monitoring crop growth conditions using wireless sensor networks and cloud analysis. Both studies demonstrated the successful application of real-time DL technologies for complex data analysis. However, the current study focuses on adaptability to rapidly changing threats, while other study focuses on predictability and increased productivity in a stable environment, but is less suited to solving tasks requiring high flexibility and adaptation. In contrast to the conducted study, which focuses on the use of DL for antivirus systems, Rane et al. (2024) focuses on the scalability and adaptability of DL in distributed systems such as federated learning and concurrency models for big data processing. Nevertheless, the current study surpasses that mentioned in terms of application orientation, as it is focused on solving specific cybersecurity tasks.

Faruk Akmeşe et al. (2024) used DL to classify time series of discrete chaotic systems, which is similar to the current study using DL to analyse cyber threats. Both studies focus on complex and unpredictable data, but on different tasks. Nevertheless, the results of the mentioned study confirm the conclusions of the current one, as they emphasise the ability of DL to effectively cope with unstable and dynamic data, which is important both for analysing cyber threats and for classifying chaotic systems. Moreover, Khan (2024) used a combination of the Region Proposal Network (RPN) and CNN to actually detect objects from a webcam. This approach is based on deep neural networks that perform both classification and detection without the need for manual algorithms (Raj et al., 2025). In the current study, CNN and RNN were used to analyse and classify cyber threats, which allows effectively identifying anomalies in programme behaviour and prevent cyber-attacks. Thus, although both approaches use DL to work with dynamic data, the current approach is more focused on adaptability and protection from cyber threats, which makes it more specialised for security tasks.

Whereas the current study uses the Python language for DL algorithms used in antivirus systems, Lima et al. (2023) focuses on creating an antivirus using ML and JavaScript to detect malware based on real-time behaviour. The results of the study reviewed confirm the effectiveness of ML for threat detection, but the current study is expanding this area by analysing antivirus systems that use DL for a wider range of cyber threats. In turn, Zaidi et al. (2024) used the DL method to detect malicious software using Graph Neural Networks (GNN) and Graph Convolutional Networks (GCN). This approach focuses on converting files to graph formats, which allows analysing connections in the application code and effectively identifying new types of malware, which is a limitation of conventional methods (Al-Hawary et al., 2024; Amourah et al., 2025). In contrast, the current study used other networks, namely CNN and RNN, to analyse and classify cyber threats. Nevertheless, this study complements the current one, demonstrating the versatility and effectiveness of various networks for solving specific cybersecurity tasks, such as application protection and adaptive threat identification.

This study analysed existing antivirus systems using DL algorithms, whereas the Shaikh et al. (2024) introduced the Intrusion Detection System (IDS), which uses CNN and Vision Transformers (ViT) to detect DDoS attacks, achieving 99.99% accuracy. Although both researches examined CNN and transformers, the current study covered a broader range of DL algorithms, including VAE, RNN, and GAN, allowing analysis of various aspects of cyber threats such as anomaly detection, behavioural analysis, and attack scenario modelling. In addition, Wajahat et al. (2024) presented a new approach to threat detection using the Deep Neural Decision Forest (DNDF) classifier, which showed 99% accuracy, which underlines its effectiveness for modern threats with obfuscation. The current study focused on the analysis of existing antivirus systems using CNN, RNN, BERT and GPT, and VAE and GAN for threat detection. Thus, the conducted research expands on the mentioned, providing a comprehensive analysis of various algorithms and methods for protection against cyber threats.

Similar to the study that uses DL algorithms to detect cyber threats, Rane et al. (2024) consider such DL algorithms as CNN, RNN, GAN and transformers, including LSTM and self-attention mechanisms. In addition to the mentioned algorithms, autoencoders were also used in the current study, which confirms the similarity in approaches to using DL for processing complex data. In other words, the results confirm the current conclusions about the high efficiency of these algorithms in data analysis and their application, which highlights the versatility of DL algorithms and their adaptability in various contexts, including cybersecurity. If the conducted research uses CNN and RNN to neutralise cyber threats, then the study by Baniya and Rush (2024) considers the use of Deep Neural Network (DNN) in the context of protection against malicious packets on the network. While this approach focuses on an ensemble of different classifiers and deep neural networks to detect attacks, the current study focuses on using CNNs to analyse programme behaviour and RNN to identify attack sequences. Therefore, it deepens the analysis by applying different types of neural networks to more accurately study and classify threats based on their behaviour and time dependencies (Smailov et al., 2025).

Whereas the conducted research uses DL to analyse threat behaviour, Suparman et al. (2024) explore its application to threat detection and examine the challenges of implementing AI in cybersecurity, including issues of privacy, computing resources, and model updates. The current study deepens this approach by demonstrating the practical integration of DL algorithms for more accurate and adaptive threat detection. In a similar way to the study conducted, which examines antivirus systems, Saputra et al. (2024) conducted a comparative analysis of antivirus solutions specifically for smartphones. In both cases, the focus is on identifying the most effective approaches to protecting against malicious threats. However, unlike the mentioned article, the current study is based on a comparison of antivirus systems using DL algorithms, which expands the coverage of the threats under consideration and makes the results more universal for various platforms, not just smartphones.

Unlike the study by Drakulić and Mujčić (2023), where a comparative analysis of conventional antivirus solutions (Avast, Avira, AVG, Eset Nod32, Panda, Malwarebytes) is conducted, the current study focuses on antiviruses using DL algorithms such as Sophos, CrowdStrike, Darktrace, SentinelOne, and CylancePROTECT. While the study evaluates the performance of antiviruses based on virus detection, deletion, and memory usage parameters, current operating systems use DL algorithms, which allows

them to more effectively identify new threats and anomalies in programme behaviour, improving the accuracy and adaptability of protection. Ultimately, Tavares-Silva et al. (2024) have developed an antivirus for dynamic malware analysis based on neural networks and the Internet of Things (IoT). The current study focuses on antivirus systems using neural networks, however, in the context of DL, rather than IoT, which allows expanding the scope of algorithms and improving the adaptability of protection for various platforms, with more accurate threat detection in real time.

Thus, the conducted research confirmed the high potential of integrating DL algorithms into antivirus systems, emphasising their importance for improving the effectiveness of protection against cyber threats. These technologies provide more accurate and rapid threat detection, adapting to changes in programme behaviour and the emergence of new types of attacks.

5. CONCLUSIONS

As a result of the conducted research, it was found that DL algorithms such as CNN, VAE, GAN, RNN, GPT and BERT significantly increase the effectiveness of detecting and neutralising cyber threats. CNN has demonstrated excellent results in analysing programme behaviour and classifying network traffic, which allows not only effective detection of known, but also new threats. Autoencoders have proven useful for detecting anomalies, providing high accuracy in detecting rare and complex attacks. GAN have improved the ability of antivirus systems to simulate potential threats, which increases their adaptability. RNN efficiently handle time dependencies, which contributes to accurate predictions based on programme behaviour. And transformers have shown the greatest efficiency in analysing the code structure, which allows for more accurate identification of complex malware.

An analysis of antivirus systems such as CrowdStrike, Sophos, Darktrace, SentinelOne, and CylancePROTECT has shown that the implementation of DL algorithms significantly improves their performance. Systems using CNN, VAE, and GAN demonstrate a high level of protection, allowing effective detection of both known and previously unreported threats. In particular, CrowdStrike and SentinelOne effectively use CNN and RNN to analyse programme behaviour and predict attacks, while Sophos and CylancePROTECT, using autoencoders and transformers, and focus on results in detecting complex threats with minimal false positives.

The integration of DL algorithms into antivirus systems has demonstrated a significant increase in the effectiveness of detecting and neutralising cyber threats, especially due to the adaptability of models to new attacks. However, high computational costs and dependence on data quality remain key constraints affecting the scalability of solutions. In the future, it is necessary to focus on optimising algorithms and exploring promising approaches such as quantum computing and multi-agent systems to increase their efficiency and adaptability.

6. REFERENCES

Aizstrauta, D., and Ginters, E., 2016, Using Market Data of Technologies to Build a Dynamic Integrated Acceptance and Sustainability Assessment Model. *Procedia Comput. Sci.*, **104**, 501–508.

Akmeşe, Ö.F., Emin, B., Alaca, Y., Karaca, Y., and Akgul, A., 2024, The Time Series Classification of Discrete-Time Chaotic Systems Using Deep Learning Approaches. *Mathematics*, **12(19)**, 3052.

Al-Hawary, T., Amourah, A., Salah, J., and Yousef, F., 2024, Two Inclusive Subfamilies of bi-univalent Functions. *Int. J. Neutrosophic Sci.*, **24(4)**, 315–323.

Amourah, A., Frasin, B., Salah, J., and Yousef, F., 2025, Subfamilies of Bi-Univalent Functions Associated with the Imaginary Error Function and Subordinate to Jacobi Polynomials. *Symmetry*, **17(2)**, 157.

Aziz, M.R., and Alfoudi, A.S., 2023, Different mechanisms of machine learning and optimization algorithms utilized in intrusion detection systems, in *AIP Conference Proceedings (Vol. 2839, No. 1, p. 040005)*. AIP Publishing LLC.

Bala, Z., Zambuk, F.U., Imam, B.Y., Gital, A., Shittu, F., Aliyu, M., and Abdulrahman, M.L., 2022, Transfer Learning Approach for Malware Images Classification on Android Devices Using Deep Machine Learning. *Procedia Comput. Sci.*, **212**(4), 429–440.

Baniya, B.K., and Rush, T., 2024, Intelligent Anomaly Detection System Based on Ensemble and Deep Learning, in *Proceedings of the 26th International Conference on Advanced Communications Technology*. Institute of Electrical and Electronics Engineers (IEEE).

Biloshchytskyi, A., Chupryna, I., Tormosov, R., and Fedorchenko, M., 2026, Applied Modules of System Performance of Changes in the Operational System of Construction Enterprises. *CEUR Workshop Proc.*, **4193**. Almaty: CEUR-WS.

Brescia, E., Massenio, P.R., Nardo, M.D., Cascella, G.L., Gerada, C., and Cupertino, F., 2023a, Nonintrusive Parameter Identification of IoT-Embedded Isotropic PMSM Drives. *IEEE J. Emerg. Sel. Top. Power Electron.*, **11(5)**, 5195–5207.

Brescia, E., Vergallo, P., Serafino, P., Tipaldi, M., Cascella, D., Cascella, G.L., Romano, F., and Polichetti, A., 2023b, Online Condition Monitoring of Industrial Loads Using AutoGMM and Decision Trees. *Machines*, **11(12)**, 1082.

Chen, X., 2025, Smart Agricultural Data Real-Time Monitoring System Based on Deep Learning, in *Smart Infrastructures in the IoT Era.* Springer.

CrowdStrike, 2024, The CrowdStrike State of AI in Cybersecurity Survey. https://www.crowdstrike.com/en-us/state-of-ai-survey/.

CylancePROTECT, 2024, CylancePROTECT. https://www.blackberry.com/id/id/products/cylance-endpoint-security/cylance-protect.

Darktrace, 2024, Darktrace Releases Annual 2024 Threat Insights. https://darktrace.com/blog/darktrace-releases-annual-2024-threat-insights.

Doris, L., and Gracias, A., 2024, Applying Deep Learning to Detect and Respond to Cyber-Attacks on Sensor Fusion Systems, Such as LiDAR and Camera Inputs. https://www.researchgate.net/publication/386573013_APPLYING_DEEP_LEARNING_TO_DETECT_AND_RESPOND_TO_CYBER-ATTACKS_ON_SENSOR_FUSION_SYSTEMS_SUCH_AS_LIDAR_AND_CAMERA_INPUTS.

Drakulić, U., and Mujčić, E., 2023, A Comparative Performance Analysis of Various Antivirus Software, in *Advanced Technologies, Systems, and Applications VIII: Proceedings of the International Symposium on Innovative and Interdisciplinary Applications of Advanced Technologies (IAT 2023)*. Springer.

El–Ela, M.H.A., and Hamdi, A., 2024, Deep heuristic learning for real-time urban pathfinding, in *2024 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)*. IEEE.

Ginters, E., Gutiérrez, J.M., and Mendívil, E.G., 2020, Mapping of Conceptual Framework for Augmented Reality Application in Logistics. In: *Proc. 61st Int. Sci. Conf. Inf. Technol. Manage. Sci. (ITMS 2020)*, **1**, Article 9259302.

Gospodinova, E., 2022, Analysis and Development of an Algorithm to Increase the Energy Efficiency of Electrical Street Lighting Systems Using an Artificial Neural Network. In: *Proc. 6th Eur. Conf. Electr. Eng. Comput. Sci. (ELECS 2022)*, 145–150.

Gospodinova, E., 2023, Mathematical Modeling and Planning of Energy Production Using a Neural Network. *WSEAS Trans. Power Syst.*, **18**, 39–48.

Guliyev, H.B., 2019, Reactive Power Adaptive Control System in Networks with Distributed Generation Based on Fuzzy Set Theory. In: *Proc. Int. Conf. Artif. Intell. Data Process. Symp. (IDAP 2019)*, Article 8875963.

Hüseynova, N., 2022, Deep Learning as One of the Main Areas of Artificial Intelligence. *Proc. Azerb. High Tech. Educ. Inst.*, **23(12)**, 113–121.

Ikpe, A.E., Ekanem, I.I., and Ikpe, E.O., 2024, A Review of Deep Learning Algorithm in Real-Time Performance of Intelligent Industrial Machines, in E*FES International Scientific Research and Innovation Congress.* UBAK Publications.

Jiang, Y., Hang, R., Huang, W., Wu, Y., Pan, X., and Tao, Z., 2024, A Real-Time Posture Detection Algorithm Based on Deep Learning. *Int. J. Comput. Intell. Appl.*, **24(3)**, 1–15.

Kabdoldina, A., Ualiyev, Z., Smailov, N., Malikova, F., Oralkanova, K., Baktybayev, M., Arinova, D., Khikmetov, A., Shaikulova, A., and Bazarbay, L., 2022, Development of the Design and Technology for Manufacturing a Combined Fiber-Optic Sensor Used for Extreme Operating Conditions. *East.-Eur. J. Enterp. Technol.*, **5(5-119)**, 34–43.

Khan, M.S., 2024, Real Time Object Detection Using Deep Learning. http://dx.doi.org/10.13140/RG.2.2.23126.46408.

Kondratenko, Y., and Kondratenko, V., 2014, Soft Computing Algorithm for Arithmetic Multiplication of Fuzzy Sets Based on Universal Analytic Models. *Commun. Comput. Inf. Sci.*, **469**, 49–77.

Kozhakhmetova, D., Kaliyeva, S., Sugurova, L., Sugur, Z., Wójtowicz, R., and Zhylkybayev, T., 2024, Analysis of the Distillation Column of a Catalytic Cracking Unit Using Fuzzy Input Information. *Energies*, **17(17)**, 4446.

Lima, S., Souza, D.M., Pinheiro, R., Silva, S.M., Lopes, P.G., de Lima, R.D.T., de Oliveira, J.R., Monteiro, T., Fernandes, S.M.M., Albuquerque, E., da Silva, W.W.A., and Dos Santos, W.P., 2023, Next-Generation Antivirus for JavaScript Malware Detection Based on Dynamic Features. *Knowl. Inf. Syst.*, **66(2)**, 1337–1370.

Muñoz, M., Rubio, A., Cosarinsky, G., Cruza, J.F., and Camacho, J., 2024, Deep Learning-Based Algorithms for Real-Time Lung Ultrasound Imaging. *Appl. Sci.*, **14(24)**, 11930.

Nithya Raju, N., Suresh, T., Rajaram, G., Malleswari, M., Keerthika, E., and Somasundaram, K., 2024, Deep Learning Algorithm for Real-Time Disease Detection and Classification in Radiological Images with Enhanced Diagnostic Accuracy, *J. Electr. Syst.*, **20(5s)**, 2042–2050.

Online Python, 2026, Online Python. https://www.online-python.com/ZYsXyNdltb.

Patel, J.M., 2024, A Conceptual Framework for Deep Learning Algorithms and Their Applications. *Int. J. Multidiscip. Res.*, **6(6)**.

Petrov, N., Sydykova, G., Dimitrova, K., Gospodinova, E., Tlegenov, A., and Shegenbaeva, R., 2023, Study of the Sustainability of Functioning of Electronic Apparatus. *AIP Conf. Proc.*, **2889(1)**, 050006.

Raj, L.V., Sasi, S., Rajeswari, P., Pushpa, B.R., Kulkarni, A.V., and Biradar, S., 2025, Design of FBG-based Optical Biosensor for the Detection of Malaria. *J. Opt. (India)*. DOI: 10.1007/s12596-025-02780-x.

Ramesh, G., Aravindarajan, V., and Logeshwaran, J., 2023, The Performance Evolution of Antivirus Security Systems in Ultradense Cloud Server Using Intelligent Deep Learning. *J. Comput. Intell. Commun. Netw.*, **1(1)**, 21–25.

Rane, N., Mallick, S.K., Kaya, Ö., and Rane, J., 2024, Techniques and Optimization Algorithms in Deep Learning: A Review, in *Applied Machine Learning and Deep Learning: Architectures and Techniques.* Deep Science Publishing.

Rane, N.L., Rane, J., Mallick, S.K., and Kaya, Ö., 2024, Scalable and Adaptive Deep Learning Algorithms for Large-Scale Machine Learning Systems, in F*uture Research Opportunities for Artificial Intelligence in Industry 4.0 and 5.0.* Deep Science Publishing.

Rudenko, O.V., Anop, D.K., Shevlyakov, V.F., and Chernavin, V.Y., 2024, Manufacturing Small Architectural Forms with 3D Printing. *Int. J. GEOMATE*, **27(123)**.

Saputra, H., Zahra, A., Faldi, F., Rahman, F.F., Harits, S., Pranoto, W.J., and Rahman, F., 2024, Recommendation Mobile Antivirus for Android Smartphones Based on Malware Detection. *Int. J. Artif. Intell.*, **13(3)**, 3559–3566.

SentinelOne, 2024, World-Leading Cybersecurity, Powered by AI. https://www.sentinelone.com/.

Shaikh, J., Butt, Y.A., and Naqvi, H.F., 2024, Effective Intrusion Detection System Using Deep Learning for DDoS Attacks. *Asian Bull. Big Data Manag.*, **4(1)**, 168–183.

Shaikh, J., Syed, T.A., Shah, S.A., Jan, S., Ain, Q.U., and Singh, P.K., 2024, Advancing DDoS Attack Detection with Hybrid Deep Learning: Integrating Convolutional Neural Networks, PCA, and Vision Transformers. *Int. J. Smart Sens. Intell. Syst.*, **17(1)**.

Shendge, D., 2024, Real-Time Criminal Detection System Using Deep Learning Technique. *Int. J. Res. Appl. Sci. Eng. Technol.*, **12(10)**, 700–703.

Smailov, N., Akmardin, S., Ayapbergenova, A., Ayapbergenova, G., Kadyrova, R., and Sabibolda, A., 2025, Analysis of VLC Efficiency in Optical Wireless Communication Systems for Indoor Applications. *Inform. Autom. Pomiary Gospod. Ochr. Środ.*, **15(2)**, 135–138.

Sokoliuk, A., Kondratenko, G., Sidenko, I., Kondratenko, Y., Khomchenko, A., and Atamanyuk, I., 2021, Machine Learning Algorithms for Binary Classification of Liver Disease. In: *Proc. IEEE Int. Conf. Probl. Infocommun. Sci. Technol. (PIC S&T 2020)*, 417–421.

Sophos, 2024, Year in Review 2024: The Major Headlines and Moments from Sophos This Year. https://news.sophos.com/en-us/2024/12/17/year-in-review-2024-the-major-headlines-and-moments-from-sophos-this-year/.

Sufi, F.K., 2024, Open-Source Cyber Intelligence Research Through PESTEL Framework: Present and Future Impact. *Soc. Impacts*, **3**, 100047.

Suparman, A., Akhmad, E.P.A., and Dinata, B.M., 2024, Leveraging Artificial Intelligence for Enhancing Cybersecurity: A Deep Learning Approach to Real-Time Threat Detection. *J. Acad. Sci.*, **1(7)**, 835–842.

Tavares-Silva, S.H.M., Lima, S., Paranhos-Pinheiro, R., Santiago-Abreu, L.M., Toscano-Lima, R.D., and Fernandes, S.M.M., 2024, Antivirus Solution to IoT Malware Detection with Authorial Next-Generation Sandbox. *J. Supercomput.*, **81(1)**, 151.

Wajahat, A., He, J., Zhu, N., Mahmood, T., Nazir, A., Ullah, F., Qureshi, S., and Osman, M., 2024, An Effective Deep Learning Scheme for Android Malware Detection Leveraging Performance Metrics and Computational Resources. *Intell. Decis. Technol.*, **18(6)**, 33–55.

Wu, W., Harrou, F., Bouyeddou, B., Senouci, S.-M., and Sun, Y., 2022, A Stacked Deep Learning Approach to Cyber-Attacks Detection in Industrial Systems: Application to Power System and Gas Pipeline Systems. *Cluster Computing*, **25(4)**, 561–578.

Zaidi, A.R., Khan, T.A., Ramay, S.A., Nawaz, A., Ameen, K., and Irfan, M., 2024, Deep Learning-Based Detection of Android Malware Using Graph Convolutional Networks (GCN)*, Stat. Comput. Interdiscip. Res.*, **6(1)**, 57–73.

Zhou, Y., and Liang, Y., 2024, Application of Artificial Intelligence Technology in Network Security. *Highlights Sci. Eng. Technol.*, **92**, 479–485.

Zinchenko, V., Kondratenko, G., Sidenko, I., and Kondratenko, Y., 2020, Computer Vision in Control and Optimization of Road Traffic. In: *Proc. IEEE Int. Conf. Data Stream Min. Process. (DSMP 2020)*, 249–254.