sciendo

JAISCR, 2022, Vol. 12, No. 2, pp. 149 – 163
10.2478/jaiscr-2022-0010

AN AUTOENCODER-ENHANCED STACKING NEURAL NETWORK MODEL FOR INCREASING THE PERFORMANCE OF INTRUSION DETECTION

Csaba Brunner[1], Andrea Kő[1], Szabina Fodor[2,*]

[1]*Department of Information Systems, Corvinus University of Budapest*
Fővám tér 13-15, 1093 Budapest, Hungary

[2]*Department of Computer Science, Corvinus University of Budapest*
Fővám tér 13-15, 1093 Budapest, Hungary

[*]*E-mail: szabina.fodor@uni-corvinus.hu*

Submitted: 15th December 2021; Accepted: 30th January 2022

Abstract

Security threats, among other intrusions affecting the availability, confidentiality and integrity of IT resources and services, are spreading fast and can cause serious harm to organizations. Intrusion detection has a key role in capturing intrusions. In particular, the application of machine learning methods in this area can enrich the intrusion detection efficiency. Various methods, such as pattern recognition from event logs, can be applied in intrusion detection. The main goal of our research is to present a possible intrusion detection approach using recent machine learning techniques. In this paper, we suggest and evaluate the usage of stacked ensembles consisting of neural network (SNN) and autoencoder (AE) models augmented with a tree-structured Parzen estimator hyperparameter optimization approach for intrusion detection. The main contribution of our work is the application of advanced hyperparameter optimization and stacked ensembles together.

We conducted several experiments to check the effectiveness of our approach. We used the NSL-KDD dataset, a common benchmark dataset in intrusion detection, to train our models. The comparative results demonstrate that our proposed models can compete with and, in some cases, outperform existing models.

Keywords: intrusion detection, neural network, ensemble classifiers, hyperparameter optimization, sparse autoencoder, NSL-KDD, machine learning

1 Introduction

Computer networks face various dynamic security threats and intrusions affecting the availability, confidentiality and integrity of resources and services. To counteract these threats, many organizations designed and implemented intrusion detection systems. In the context of information systems, an intrusion is a deliberate unauthorized attempt to access and manipulate information in order to render a system unreliable or unusable. The goal of intrusive behavior is to compromise the security of computer and network components in terms of confidentiality, integrity and availability [13]. Intrusion detection is a set of actions to detect intrusive behavior, to raise alerts, and to provide information to prevent intrusive behavior. The key assumption of intrusion detection is that attacks are significantly discernible from normal activities. Intrusion detection is defined as the task of identifying individuals who are

either using a computer system without authorization (i.e., crackers) or those who have legitimate access to the system but are exceeding their privileges (i.e., insider threats) [45, 15]. According to ISACA [20], intrusion detection is the "process of monitoring the events occurring in a computer system or network to detect signs of unauthorized access or attack". Intrusion detection is a complex task that can be supported with various methods, such as statistical analysis, expert knowledge and pattern recognition from event logs. Intrusion detection systems can be organized by the protected system component or by the type of pattern recognition applied to the task [21, 39, 31]. Regarding protected system components, one can consider network-based (NIDS) or host-based (HIDS) intrusion detection. An NIDS identifies attacks within a monitored network using potential alerts raised to the system operator. An HIDS, however, is configured for a specific server environment and will monitor the internal resource utilization of the operating system to warn of a possible attack. Intrusion detection systems can detect modifications in the code of executable programs, detect unauthorized deletions of files and issue warnings when an unauthorized use of a privileged command is attempted. In further sections of this article, our primary focus will be on network intrusion detection. Regarding the type of pattern recognition applied to the task, IDSs can be classified as misuse/signature, anomaly and hybrid detection systems. A misuse/signature-based IDS raises alerts when a known intrusive pattern in packed data is detected. These known patterns can be detected reliably; however, these systems struggle with new, unseen attack patterns, and they require information on the attack type first, which is not always available. Anomaly detection triggers alerts when the network traffic behaves in a significantly different way than predetermined normal traffic patterns. Trained using only normal traffic, anomaly detectors can detect new attack patterns; however, they often make mistakes with normal, albeit unusual, network traffic patterns. Hybrid detection approaches combine the benefits of both signature detection and anomaly detection, such as by performing anomaly detection on traffic classified as normal by the signature detector and vice versa.

Applying data mining and machine learning methods to intrusion detection has been suggested in many previous works [51, 5, 19]. Several researchers have explored new methods to detect these cyberattacks [17, 3, 8]. The application of machine learning algorithms benefits intrusion detection research in particular as the volume of network traffic makes earlier analysis methods less effective and time-consuming. Several other approaches, such as collaborative intrusion detection systems (CIDSs), have also been published to develop more efficient intrusion detection systems (IDSs) [14, 43].

The main goal of our research is to offer a machine learning method for intrusion detection. We suggest a stacked ensemble neural network (SNN) combined with an autoencoder (AE) model optimized with tree-structured Parzen estimators trained on the NSL-KDD benchmark dataset. We found only a limited number of similar solutions in the existing intrusion detection literature [3]; however, these approaches provide promising intrusion detection results.

The main contribution of our work is the application of advanced hyperparameter optimization and stacked ensembles together. Application of more advanced hyperparameter search strategies resulted that we managed to achieve performance comparable to more recent variational autoencoder (VAE) and conditional variational autoencoder (CVAE) based outcome. We compared our results with those of similar initiatives; and in terms of some validation metrics, the proposed models outperformed existing models. We achieved a higher perclass recall rate on minority classes. Two approaches were provided to deal with imbalanced data, which is common in IT security cases. First, we applied a synthetic oversampling methodology (SVM SMOTE) to eliminate class imbalance, second, we used autoencoder models.

Our work first provides an overview of related works on hyperparameter optimization, AE networks and IDSs. The following section describes the suggested models followed by the achieved results and a discussion on how our models performed compared to contemporary literature. Finally, the last section provides a conclusion, including the potential application of our findings and further research opportunities.

2 Related work

Artificial neural networks (ANNs) are machine learning models inspired by the learning process of the human brain. They are widespread in business applications, classification and forecasting due to their advantages, such as possessing a high tolerance to noise, solving nonlinear and ill-defined problems based on parallel composition and not being restricted by normality and/or independence assumptions. ANNs can be distinguished by the application area, network architecture and learning algorithm. Recently, the utilization of ANNs has increased [2, 52, 7, 50].

Tian et al. [49] applied a distributed neural network learning algorithm (DNNL) for intrusion detection. They compared their approach with other works on the KDD Cup 1999 benchmark dataset [46], and the proposed model achieved a higher detection rate and lower false alarm rate. Beghdad studied five neural network types to classify the normal and attack patterns using a sample of the KDD Cup 1999 dataset containing 18,285 manually selected records [8]. The main contribution of their approach is the investigation of the performances of multilayer perceptron (MLP), generalized feed forward (GFF), radial basis function (RBF), self-organizing feature map (SOFM) and principal component analysis (PCA) neural networks at detecting attacks and classifying attacks into one or more classes. GFF resulted in the best confusion matrix in the multiclass case.

Another valid approach is the use of ensemble models to improve classification performance. Three approaches exist for creating model ensembles: bagging, boosting and stacking [36, 44]. Bagging, or bootstrap aggregation, combines majority voting with machine learning models to improve predictions. Boosting sequentially trains weak prediction models, measures the error between predicted and expected outcomes, assigns weights to observations based on the error, and then trains a new model, thus creating a more powerful model. Stacking combines multiple machine learning models using a meta-classifier. The base-level models are first trained on the training data, and then the meta-classifier is trained on the predictions of the base models. Stacking, compared to boosting and bagging, can reduce the model variance and bias at the same time, providing powerful aggregate prediction models. This improvement stems from the heterogeneity of the base models, which could be achieved by training the same type of models on different data features or by training different models. Considering the advantageous property of simultaneously reducing the variance and bias in model predictions, we decided to use this ensemble design for our intrusion detectors. A drawback of ensemble models is increased complexity as multiple models must be trained and maintained.

2.1 Hyperparameter optimization

Machine learning models require setting the parameters prior to training. These parameters could directly influence the performance achieved by a model; therefore, an automated approach for selecting these parameters is crucial. This approach is called hyperparameter optimization, a method encompassing the regular training-testing-evaluation process of machine learning.

The two most common methods for hyperparameter optimization are grid and random search, but these are not suitable for deep neural networks as both methods have issues, either with execution time or with performance. Other approaches use dedicated algorithms, such as Bayesian optimization [41], gradient-based optimization or evolutionary optimization, to find the best set of parameters.

In our study, we used the tree-structured Parzen estimator (TPE) [12, 11], a method used to solve expensive single-objective optimization problems. This method works by replacing the distributions of the prior parameter settings with nonparametric densities. This surrogate naturally handles continuous, discrete, categorical, and conditional variables. Furthermore, this surrogate has lower computational complexity than Bayesian optimization and can scale to tens of variables and thousands of parameter samples [37]. The tree-structured Parzen estimator has been adopted as the main model in hyperopt [10, 9], a Python framework designed for hyperparameter optimization.

2.2 Autoencoder networks

Autoencoder networks are unsupervised neural network algorithms created when the target vectors are set to be identical to the input vectors. An AE

can be divided into three parts: an encoder, learning interesting patterns in the input data; a bottleneck creating a limited representation; and a decoder reconstructing the input from this limited representation. Training an AE, performed using a forward pass followed by back propagation, is similar to that of a fully connected neural network. The most important differences are in the network architecture; the choice of the expected output to compare predictions to; and in the case of intrusion detection, whether the training data have been filtered prior, for example, by an intrusion class. Then, the AE reconstruction error (the MSE between original and predicted inputs) will be lower for that class and greater for all the remaining classes, which can be exploited for anomaly detection purposes.

The base version of an AE consists of three layers: the input acting as an encoder, the hidden layer as a bottleneck and the output layer as a decoder. This setup can be extended with additional hidden layers to create deep AEs (DAEs). These hidden layers may contain fewer neurons than preceding (or following, in the case of decoder) layers. AEs with layers designed in this way are called undercomplete AEs, and they learn a compressed representation of the data. AEs that have no such constraints on the hidden layers are called overcomplete AEs. Overcomplete AEs have a tendency to learn the identity function and thus have reduced usability. To overcome this, the activations of overcomplete AEs are regularized to provide a sparse representation of the data. These AEs are called sparse AEs (SAEs). Sparsity is achieved using the Kullback–Leibler divergence (KL divergence) [27], with the following formula

$$\sum_{q=1}^{N^l} KL(\rho||\hat{\rho}_q) = \sum_{q=1}^{N^l} \rho \log \frac{\rho}{\hat{\rho}_q} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_q}$$

where $\hat{\rho}_q = \frac{1}{n} \sum_{i=1}^{n} a_q^l(x_i)$ is the average activation of neuron q over all inputs x_i, N^l is the number of neurons for hidden layer l and ρ is the rate used to enforce activation sparsity. KL divergence tends to infinity when the average activation of neuron q is greater or lower than the threshold ρ. As the average activation of a neuron with sigmoid activation function is only small if most of the activations are close to zero, the KL divergence is an appropriate function to enforce sparsity.

The variational autoencoder (VAE) is a generative model suggested by Kingma and Welling [25]. VAE models are tasked to generate the latent distribution of the input, captured by a standard deviation and a mean vector, each generated by two hidden layers simultaneously at the bottleneck. We call VAEs generative models as the decoder, together with a random sample from a multivariate Gaussian distribution fed to the decoder, can generate new synthetic observations. A drawback of VAEs is that they can only generate new data for one class, which is a challenge if multiclass classification is expected in following connected model components. The conditional variational autoencoder (CVAE) is an extension of the VAE [26] that settles this challenge. A CVAE converts the unsupervised training model of VAEs into a supervised training model by feeding expected class outputs as inputs to the VAE model.

VAE and CVAE models have been recently applied for anomaly detection [23, 47, 16, 35, 29, 28, 53]. VAE models were applied for simulating network attacks [16] and for intrusion detection [35]. Lopez-Martin et al. suggested an intrusion detection CVAE (ID-CVAE) classifier to perform classification and feature recovery [29]. The ID-CVAE applies the nearest neighbour method based on the Euclidean distance to classify the test samples. In a later study, [28] compared a VAE and a CVAE model applied to synthetic oversampling methods and reported increased prediction performance using the VAE models, especially with the CVAE model labeled the variational generative model (VGM).

Yang et al. [53] proposed a novel intrusion detection model ICVAE-DNN, which combines an improved conditional variational AE (ICVAE) with a deep neural network (DNN) model. The role of the ICVAE is to learn and explore potential sparse representations between network data features and classes. A DNN was used to automatically extract high-level features and adjust network weights using back propagation and fine-tuning to better address the problem of the classification of complex, large-scale and nonlinear network traffic. The article evaluates the performance of the ICVAE-DNN using the NSL-KDD dataset. The proposed ICVAE-

DNN provides higher detection rates in minority attacks (i.e., U2R, R2L, shellcode and worms) than six other well-known classification algorithms: the KNN, multinomial NB, RF, SVM, DNN and DBN.

Ludwig [30] developed an ensemble of 4 different neural network models (AEs, DBNs, DNNs and extreme learning machines (ELM)) with the results aggregated using a simple majority vote mechanism. The article compared predictions differentiating between normal traffic and the 4 attacks individually on the NSL-KDD dataset. The work reported high accuracy with each comparison and a higher than average recall for minority classes.

2.3 Related work in IDS domain

Yao et al. [53] introduced hybrid multilevel data mining, a system for the multiclass classification of unbalanced intrusion data. The system consists of three components: a preprocessing component for data encoding, data normalization and generating one vs. rest subsets for feature selection and classification. The data mining module applied k-means clustering followed by a support vector machine, an artificial neural network and a decision tree-based classification for each cluster. The third phase involved correcting classifications by applying a decision tree classifier to previously classified, randomly sampled data. The system selected the best performing model for each sample vs. the rest of the data sample and cluster. Performance was measured using the precision, recall, F-score and accuracy metrics. The authors claimed that that the proposed method achieved high performance on DOS and R2L classes while the performance on normal and probe classes were average compared to the results of other works in the field.

Yin et al. [54] proposed a deep learning approach for intrusion detection using recurrent neural networks (RNN-IDS). Al-Qatf et al. [4] combined sparse autoencoders with an SVM classifier. This was achieved by training an SAE on unlabeled data to generate a low-dimensional representation. Then, new data with target labels were fed to the encoder layers only. The reduced dimension explanatory features were then fed to the SVM classifier. The authors not only reported improved performance but also improved the memory footprint and lowered the training time for the SVM model. Similarly, Javaid et al. [22] combined an autoencoder with a multiclass logistic regression. Both studied reported classification performances better than those of ensemble models.

Based on the literature we reviewed, we found two areas that could be improved. First, the sampling methodology used by [33, 38, 55] is questionable as both the training and test samples were created separately from the same dataset based on the same stratified sampling methodology: all target classes were sampled proportionately to their size except for the underrepresented U2R and R2L classes, 100% of which were sampled. The target class is unavailable in a real environment, and assumptions about the class distribution of the test set inherently hold the threat of information leakage. The second issue we found with most articles, especially [33, 38, 55], is the prominent use of the accuracy as a performance metric. The accuracy works best as a metric when all target classes are balanced. This is not the case for network intrusion detection, where there are large imbalances in the data, with a disproportionate amount of good or normal traffic data and very few attack cases in most cases [40]. The best metrics for classification on imbalanced datasets are the precision, recall (referred to as detection rate in some papers), false-positive rate, specificity and AUC based on ROC curves. Most of the metrics listed are applicable in multiclass classification, except the AUC, which is only available in binary or one vs. all contexts.

3 Proposed approach

In this section, we present the NSL-KDD dataset and the architecture and functioning of our three proposed models. Each model follows an ensemble intrusion detection approach by having one model for each feature group, with the final class labels provided by a separate aggregation model gathering the class labels of each base model.

3.1 Dataset and data preprocessing

We selected the NSL-KDD dataset [48] as the benchmark dataset for intrusion detection model comparison. Although the dataset has been available for a long time, it is still widely used as a standard for the evaluation of different IDSs. This dataset is a revised version of the KDD Cup 1999

dataset [46] for fixing the problem of large numbers of redundant observations.

The NSL-KDD dataset contains 125,973 and 22,544 records in the training and test sets, respectively. The test set does not have the same probability distribution as the training set, and it includes unknown attack types that do not exist in the training set. According to [46], the purpose of this was to simulate the appearance of new types of intrusions over time; thus, the dataset still has value despite its age.

Each record contains 41 different features with the 42nd feature containing information on the various intrusion attempts to which the traffic observation was connected. These techniques can be assigned into one of 5 classes: normal and 4 attacks. The descriptions of these attack classes are as follows:

- DoS (Denial of Service): an attacker tries to prevent legitimate users from using a service
- Probing: network surveillance and other probing attacks
- R2L (Remote to Local): unauthorized access from a remote machine
- U2R (User to Root): unauthorized access to local super user (root) privileges

NSL-KDD is a highly imbalanced dataset for intrusion detection; therefore, data preprocessing had to be implemented. The outline of this process is given in Figure 1. Some of the independent features had to be changed from numerical to numerically encoded categorical representations. The original class labels in NSL-KDD are too detailed and were joined together into 5 categories based on conclusions from [46]. Feature selection based on the relative deviation of independent features was performed. Depending on the feature category, we applied joint one-hot encoding on categorical features and min-max normalization on numerical input features and transformed the target feature to an integer representation. To reduce the effect of the class imbalance, we resampled the data using the SVM synthetic minority oversampling technique (SMOTE) [34, 6]. This step was conducted only for the training sample of the NSL-KDD dataset as synthetic resampling is irrelevant for calculating model performance metrics. Finally, as we have already outlined in Section 3, we split the data into four feature groups according to [46]. These feature groups were intrinsic, time-based traffic, host-based traffic and content features. Following these preprocessing steps, the data are prepared to train our model proposals.

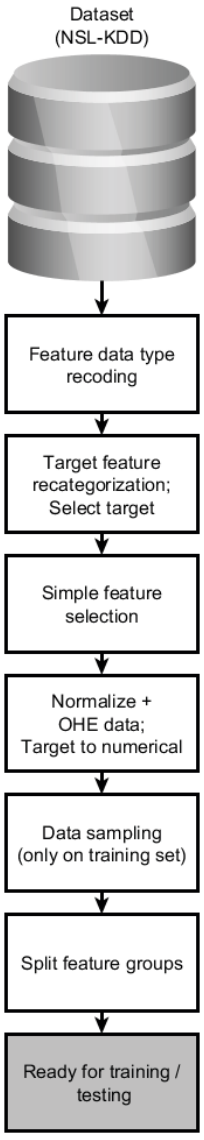

Figure 1. Data preprocessing steps for the proposed models.

3.2 Model 1: Stacked neural network (SNN)

Our first proposed model is a stacked neural network built on the TensorFlow [1] and Keras [18] open-source libraries (see in Figure 2).

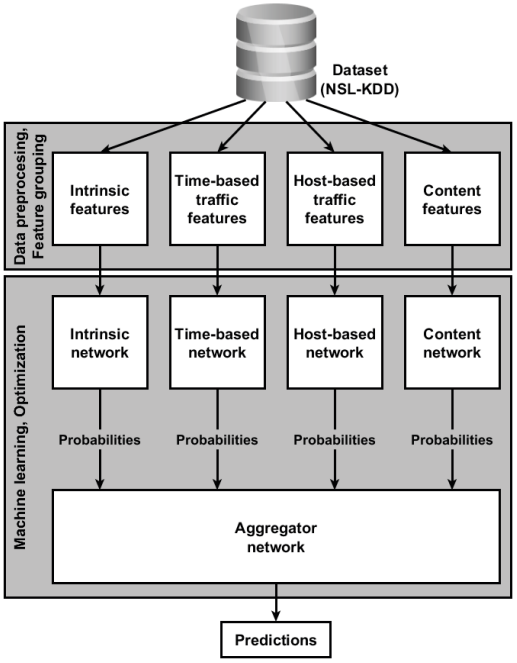

Figure 2. SNN model architecture.

Table 1. TPE hyperparameter settings for the SNN and AE-SNN models.

Parameter	Generator function
Learning rate	hp.loguniform(10^{-3}, 10)
Dropout rate	hp.loguniform(10^{-3}, $5 \cdot 10^{-1}$)
Learning rate decay	hp.uniform(10^{-1}, $5 \cdot 10^{-1}$)
The number of hidden layers	hp.choice(1, 5)
Neurons per layer	hp.quniform(5, 50, $q = 1$) converted to integer
Activations per layer	hp.choice(sigmoid, ReLU, tanh)

The neural network-based predictor model was constructed using a stacked ensemble. The four base models were trained on one of four feature groups. The flexibility of TensorFlow and Keras allowed model training to explore a wider range of hyperparameters, such the parameters of the number of hidden layers, the number of neurons per hidden layer, the activation function for each hidden layer, the learning rate and the learning rate decay over time. We used the TPE algorithm for hyperparameter optimization as it possessed advantageous properties compared to Gaussian process optimization. The target measure to optimize the hyperparameters was the sparse categorical cross entropy achieved by the model. The distributions for TPE to sample from were defined according to the suggestions of [12, 10, 9] (presented in Table 1).

The distributions sampled from were log uniform for learning and dropout rates and uniform for the learning rate decay. We set the number of hidden layers to be randomly picked from a list of numbers between 1 and 5. The number of hidden layers also determined the numbers of neurons and types of activations functions per layer for each hidden layer. The number of neurons per hidden layers was sampled from a quantized uniform distribution converted to an integer value. The activation function was chosen from a list consisting of the sigmoid, ReLU and tanh functions. This dependent hyperparameter value selection is one of the many advantages of the TPE algorithm over Gaussian processes.

Other neural network parameters were set as their default values. For example, the number of epochs during training was set to 100, the batch size was set to 1024 and the lower boundary for learning rate reduction was set to 10^{-3}. The learning rate reduction and an early stopping criterion with patience set to the square root of the number of epochs were added as callback policies expanding the capabilities of the training process and reducing the execution time. Another unaffected parameter was L2 regularization, the coefficient of which was fixed at 10^{-3}; and we used the Adam solver of [24] for training.

3.3 Model 2: Autoencoder enhanced stacked neural network (AE-SNN)

The AE-SNN consisted of the earlier SNN extended with DAEs on the base classifier level (Figure 3). Each of these AEs was first trained only on normal traffic; then, before training the base models of the SNN, these AEs were used to predict all observed network connection data.

When attack connections were predicted as if they were normal traffic, we expected the squared difference between the actual and predicted features to be higher for attacks than for normal traffic. This difference can be calculated at both the observation and feature levels, transforming both the training and test data in a way that makes the SNN com-

ponent better at detecting differences between the attack categories and normal traffic. The rest of the model training was the same as with the SNN model. We used TPE for hyperparameter optimization with the hyperparameter settings shown in Table 1. The parameterization of the DAE model is shown in Table 2.

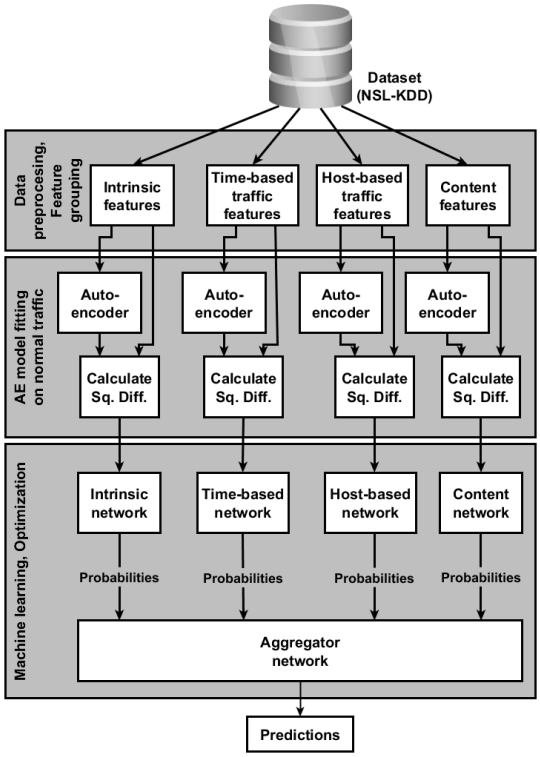

Figure 3. AE-SNN model architecture.

We used a linear activation function and the Adam optimizer with a learning rate of 10^{-3} and an early stopping criterion ending optimization after no improvement was achieved over a number of epochs equal to the square root of the total epochs. We did not perform regularization on the hidden layers of the autoencoder. In this model, the bottleneck was determined as a rounded integer of the square root of the number of input features. Finally, a sequential layer reduction rate, which decreases the number of neurons for each consecutive hidden layer in the encoder up to the bottleneck layer and is then reversed for the decoder layer, enforcing an undercomplete AE, is introduced.

3.4 Model 3: Sparse Autoencoder Stacked Neural Network (SAE-SNN)

In this model proposal, we applied a sparsity condition to the activations of each hidden DAE layer. Furthermore, instead of squared differences between actual and predicted observations, we used the output of latent features of each SAE to train the base classifiers of the SNN component. Apart from this, no other changes were applied to data preprocessing or to the rest of the model training.

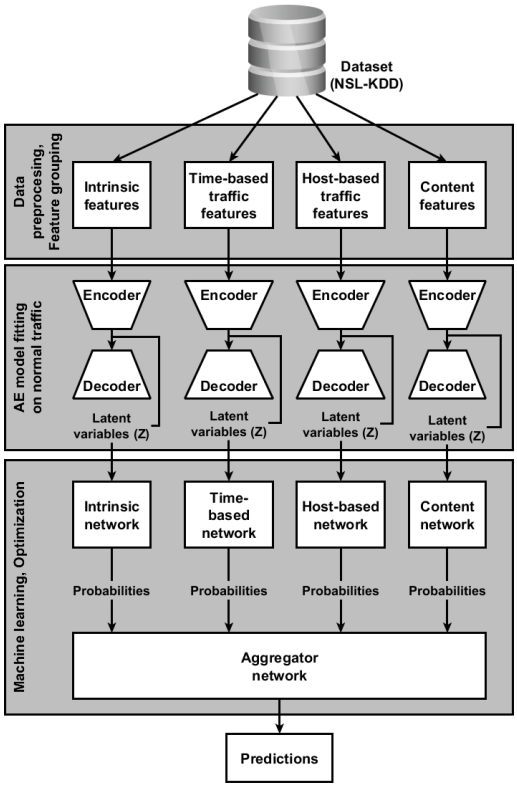

Figure 4. SAE-SNN model architecture.

The model architecture changed to accommodate the updated autoencoder models (see in Figure 4). The encoder bottleneck generates the latent features (Z) based on the actual inputs (X) provided to the AE. The decoder reconstructs the values of X, or at least a close approximation ($\hat{X}$). At a later step, the base models of the SNN were trained not on $\hat{X}$, but on the latent features Z. Here, we used the ability of AE models to generate a reduced dimensional representation of the original features.

Additional changes to the autoencoders compared to the AE-SNN were a different layer configuration; a different number of neurons for each hidden layer, except the bottleneck; and the num-

Table 2. Autoencoder parameter settings.

Parameter	Parameter setting
Activation function	Linear
Layer reduction rate	2
Optimizer	Adam (LR=10^{-3})
Number of bottleneckneurons	$round(\sqrt{\lvert number\_of\_inputs \rvert})$
Number of epochs	10^2
Early stopping patience	$round(\sqrt{epoch})$

Table 3. SAE parameter settings.

Parameter	Parameter setting
Activation function	Sigmoid
Number of hidden layers	$\lfloor \log_2(number\_of\_inputs) \rfloor$
Number of hidden neurons per layer	$3 \cdot (number\_of\_inputs)$
Number of bottleneck neurons	$\lfloor \log_2(number\_of\_inputs) \rfloor$
Hidden layer activity	KL divergence
regularization	$(\lambda = 10^{-3}, \rho = 5 \cdot 10^{-2})$
Optimizer	Adam $(LR = 10^{-3})$
Number of epochs	10^2
Early stopping patience	$\lfloor \sqrt{epochs} \rfloor$

ber of neurons (latent feature Z)) for the bottleneck layer (Table 3).

We changed the activation function for each hidden layer to the sigmoid function in order to effectively regularize them with the Kullback–Leibler divergence. The optimizer we used was Adam with a learning rate of 10^{-3}.

4 Results and discussion

This section aims to evaluate the proposed intrusion detector models introduced in the previous section.

We performed the assessment by giving an overview of some of the most important classification metrics of our three model proposals (SNN, AE-SNN and SAE-SNN) (Table 4); then by comparing the accuracies and recalls of the three models with those of the models studied in the contemporary intrusion detection literature (Table 5). Furthermore, Yang et al. [53] provided detailed per-class recalls, which allowed to perform the comparisons (Table 6).

Table 4 shows the accuracy, recall, precision, F1 score and false positive rate (FPR) of each of our model proposals. Apart from accuracy, each metric has been macro-averaged from per class metrics. This is especially true for F1 score, explaining why it does not fall between the recall and precision scores. The SNN model proved to be the best for accuracy, precision and F1 score, while AE-SNN was the best according to recall and FPR metrics. While it did not excel for any metric, SAE-SNN was the second best model for recall, precision and F1 score. We presented metrics, we analyze accuracy and recall scores in more detail in the following paragraphs.

Table 5 shows the results compared with articles also studying intrusion detection. The works listed here can be divided into three categories: single model intrusion detectors, detecting network attacks using only one model; models enhanced with synthetic sampling and models enhanced with AEs. Most of the listed works studied AE network performance primarily for intrusion detection while including non-AE models as references. The mean accuracy of the collected models was 77.72%. The SNN model outperformed this, and the AE-SNN and SAE-SNN achieved lower than average accuracy.

Table 4. Overall performance metrics for the proposed models.

Metrics	**Model**		
	SAE-SNN	AE-SNN	SNN
Accuracy	73.21%	74.26%	77.75%
Recall	63.70%	65.82%	59.23%
Precision	61.73%	57.44%	73.54%
F1 Score	59.50%	54.90%	62.85%
False Positive Rate	8.16%	6.56%	7.27%

Table 5. External comparisons in terms of accuracy and recall. The gray background indicates the results of our method.

Model	**Accuracy**	**Recall**
KNN [53]	76.51%	48.30%
Multinomial NB [53]	78.73%	47.69%
RF NB [53]	76.49%	48.84%
SVM [53]	72.28%	45.88%
DNN [53]	80.22%	52.77%
DBN[53]	80.82%	53.61%
ROS-DNN[53]	78.26%	49.59%
ADASYN-DNN[53]	80.10%	51.47%
ICVAE-DNN[53]	85.97%	62.66%
VGM + RF[28]	73.61%	–
VGM + Logistic Regression[28]	77.29%	–
VGM + Linear SVM[28]	77.23%	–
VGM + MLP[28]	79.26%	–
SMOTE + RF[28]	74.25%	–
SVM SMOTE + Logistic Regression[28]	76.29%	–
SVM SMOTE + Linear SVM[28]	77.99%	–
SVM SMOTE + MLP[28]	77.98%	–
Decision Tree[54]	74.60%	–
NB[54]	74.40%	–
RF[54]	72.80%	–
NB Tree[54]	75.40%	–
MLP[54]	78.10%	–
RNN[54]	81.29%	–
SAE + SMR[22]	79.10%	–
AE + SVM[4]	80.48%	–
SAE-SNN	73.21%	63.70%
AE - SNN	74.26%	65.82%
SNN	77.75%	59.23%

The authors of [53] and [30] also published model recalls. The mean recall of these was 51.23%. All our proposed models managed to outperform this value. In fact, both the AE-SNN at 65.82% and the SAE-SNN at 63.70% achieved better recalls, even compared to the best model in the referenced intrusion detection literature, the ICVAE-DNN, with a recall of 62.66%.

In addition to macroaveraged overall recall, Yang et al. [53] published per class recalls, enabling a more detailed comparison. The mean recalls based on the collected data were 95.5% for normal, 77.44% for DoS, 64.52% for probe, 13.84% for R2L and 4.85% for U2R classes. Our AE-enhanced model proposals did not manage to achieve good recalls on normal and DoS traffic connections compared to the measurements of [53] and [30], and they underperformed at detecting probe attacks compared to [30].

The AE-SNN and SAE-SNN performed better, especially on U2R attacks and with R2L attacks; and our proposed models performed well, only behind [53]. A likely explanation for the poor performance of the proposed models on majority classes and better performance on minority classes is that the AE-SNN and SAE-SNN traded off good recall on majority classes for an improvement in classifications on minority classes, which in turn explains the degraded performance measured by the accuracy as that metric can be influenced by biases originating from class imbalances. This trade-off became more apparent when we compared the SNN model with SVM SMOTE sampling to the two AE-enhanced proposals. With the SNN, we achieved better overall accuracy and better recall on normal and DoS classes and worse recall on probe, R2L and U2R. Comparisons with [30] confirmed this as well. Our SAE-SNN proposal achieved a 33.0% recall on R2L and 50.75% on U2R classes compared to 32.39% and 22.00%, respectively.

The likely cause of the significantly improved performance for models enhanced by AE networks are the AE networks themselves. Due to how they were trained only on normal data, they are better suited for differentiating minority attacks from the majority attacks and normal traffic.

This section compared several works from the related literature on their reported performance measurements with the performance of our proposed models. Based on certain per-class and aggregate measures, the proposed models can compete and outperform the works in the related literature.

The contribution of our research is in the combination of the following techniques:

Intrusion detection addresses imbalanced data, that is, when the volume of benign traffic is far greater than the volume of malicious activity. This article addresses imbalanced data in two ways: first, by applying SVM SMOTE, a synthetic oversampling methodology designed to eliminate class imbalance; and second, by using AE networks. AE networks are neural networks designed to learn hidden representations of data. AE networks can be used to perform intrusion detection if the task is treated as anomaly detection. In our article, we used two AE variations, DAEs with more than one hidden layer and SAE models, where the activations of the hidden layers were kept sparse with the use of the KL divergence. Following the AEs, we trained a stacked model of fully connected neural networks for the final intrusion predictions. More advanced variations of AEs, such as variational AEs (VAEs) and conditional VAEs (CVAEs), exist; however, to our knowledge, no article using these variations performed a more advanced hyperparameter search for fine-tuning further neural networks connected to the AE models. For hyperparameter search, we used tree-structured Parzen estimators to train the base neural network classifiers of the stacking ensemble. The point of TPEs is to use a more intelligent search strategy than grid and random search, thus converging on an optimal solution faster. Furthermore, the tree structure permits at least some level of neural architecture search on the classifier models.

5 Conclusion

Our tested AE-SNN and SAE-SNN models confirmed the effectiveness of autoencoder networks in the field of intrusion detection. Compared with other published results [53, 28, 30], our models achieved a higher per-class recall rate on minority classes and a lower recall rate on majority classes. This suits the requirements of intrusion detection, where the costs of misclassifying an attack in a minority is greater than the costs of misclassifying network traffic sent by a legitimate user. An interest-

Table 6. Recall comparison per class. The gray background indicates the results of our method.

Model	Normal	DoS	Probe	R2L	U2R
KNN [53]	92.78%	82.25%	59.40%	3.56%	3.50%
Multinomial NB [53]	96.03%	37.10%	82.61%	22.22%	0.50%
RF [53]	97.37%	80.24%	58.53%	7.55%	0.50%
SVM [53]	92.82%	74.85%	61.71%	0.00%	0.00%
DNN [53]	96.10%	85.40%	65.30%	14.56%	2.50%
ROS- DNN [53]	92.61%	80.32%	65.26%	12.75%	6.00%
SMOTE- DNN [53]	96.59%	82.19%	56.75%	10.93%	11.00%
ADASYN- DNN [53]	96.43%	83.28%	59.81%	9.84%	8.00%
ICVAE- DNN [53]	97.26%	85.65%	74.97%	44.41%	11.00%
SAE-SNN	85.28%	71.80%	77.65%	33.00%	50.75%
AE-SNN	83.67%	77.28%	77.32%	32.62%	58.21%
SNN	91.40%	84.38%	59.44%	31.09%	29.85%

ing result of our research is that despite using earlier AE models, namely, DAEs and SAEs, we managed to achieve performance comparable to more recent VAE- and CVAE-based results as our models benefited from more advanced hyperparameter search strategies.

A certain limitation of our research is the dataset used. The NSL-KDD dataset stems from the DARPA 1998 dataset, which was created over 20 years ago. Despite the best efforts of the original creators, much has changed since the inception of the dataset; and despite its usefulness as a benchmark, the proposed model could be evaluated on other datasets. In the future, we are planning to test our models on recently published datasets such as UNSW-NB15 [32] or either the 2017 or 2018 version of the CSE-CIC-IDS-20xx [42], and we will pay attention to increasing the recall rate on majority classes using VAEs and CVAEs in hyperparameter search.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems, 2016.

[2] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications: A survey. Heliyon , 4(11): e00938, 2018.

[3] Abdulla Amin Aburomman and Mamun Bin Ibne Reaz. A survey of intrusion detection systems based on ensemble and hybrid classifiers. Computers & Security , 65: 135–152, 2017.

[4] Majjed Al-Qatf, Yu Lasheng, Mohammed Al-Habib, and Kamal Al-Sabahi. Deep learning approach combining sparse autoencoder with SVM for network intrusion detection. IEEE Access, 6: 52843–52856, 2018.

[5] Wathiq Laftah Al-Yaseen, Zulaiha Ali Othman, and Mohd Zakree Ahmad Nazri. Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system. Expert Systems with Applications , 67: 296–303, 2017.

[6] Sikha Bagui and Kunqi Li. Resampling imbalanced data for network intrusion detection datasets. Journal of Big Data, 8(1): 1–41, 2021.

[7] Amelia A Baldwin, Carol E Brown, and Brad S Trinkle. Opportunities for artificial intelligence development in the accounting domain: the case for auditing. Intelligent Systems in Accounting, Finance & Management: International Journal, 14(3): 77–86, 2006.

[8] Rachid Beghdad. Critical study of neural networks in detecting intrusions. Computers & security, 27(5-6): 168–175, 2008.

[9] James Bergstra, Brent Komer, Chris Eliasmith, Dan Yamins, and David D Cox. Hyperopt: a python library for model selection and hyperparameter optimization. Computational Science & Discovery, 8(1): 14008, 2015.

[10] James Bergstra, Dan Yamins, and David D Cox. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In Proceedings of the 12th Python in science conference, pages 13–20. Citeseer, 2013.

[11] James Bergstra, Daniel Yamins, and David Daniel Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. 2013.

[12] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In Advances in neural information processing systems, pages 2546–2554, 2011.

[13] Monowar H Bhuyan, Dhruba Kumar Bhattacharyya, and Jugal K Kalita. Network Anomaly Detection: Methods, Systems and Tools. IEEE Communications Surveys & Tutorials, 16(1): 303–336, 2013.

[14] Nassima Bougueroua, Smaine Mazouzi, Mohamed Belaoued, Noureddine Seddari, Abdelouahid Derhab, and Abdelghani Bouras. A survey on multi-agent based collaborative intrusion detection systems. J. Artif. Intell. Soft Comput. Res., 11(2): 111–142, 2021.

[15] Anna L Buczak and Erhan Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Communications Surveys & Tutorials, 18(2): 1153–1176, 2015.

[16] Sarin E Chandy, Amin Rasekh, Zachary A Barker, and M Ehsan Shafiee. Cyberattack detection using deep generative models with variational inference. Journal of Water Resources Planning and Management, 145(2): 4018093, 2019.

[17] Zouhair Chiba, Noureddine Abghour, Khalid Moussaid, Amina El Omri, and Mohamed Rida. A novel architecture combined with optimal parameters for back propagation neural networks applied to anomaly network intrusion detection. Computers & Security, 75: 36–58, 2018.

[18] François Chollet. KERAS Documentation, 2015.

[19] Sumeet Dua and Xian Du. Data mining and machine learning in cybersecurity. CRC press, 2016.

[20] ISACA. CISA Review Manual. ISACA, 26 edition, 2015.

[21] ISACA. CISM Review Manual. ISACA, 15 edition, nov 2016.

[22] Ahmad Javaid, Quamar Niyaz, Weiqing Sun, and Mansoor Alam. A deep learning approach for network intrusion detection system. In Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS), pages 21–26, 2016.

[23] Yuta Kawachi, Yuma Koizumi, and Noboru Harada. Complementary set variational autoencoder for supervised anomaly detection. In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 2366–2370. IEEE, 2018.

[24] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. arXiv preprint arXiv: 1412.6980, 2014.

[25] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. arXiv preprint arXiv: 1312.6114, 2013.

[26] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In Advances in neural information processing systems, pages 3581–3589, 2014.

[27] Solomon Kullback. Information Theory and Statistics. John Riley and Sons. Inc. New York, 1959.

[28] Manuel Lopez-Martin, Belen Carro, and Antonio Sanchez-Esguevillas. Variational data generative model for intrusion detection. Knowledge and Information Systems, 60(1): 569–590, 2019.

[29] Manuel Lopez-Martin, Belen Carro, Antonio Sanchez-Esguevillas, and Jaime Lloret. Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in iot. Sensors, 17(9): 1967, 2017.

[30] Simone A Ludwig. Applying a neural network ensemble to intrusion detection. Journal of Artificial Intelligence and Soft Computing Research, 9, 2019.

[31] Borja Molina-Coronado, Usue Mori, Alexander Mendiburu, and José Miguel-Alonso. Survey of Network Intrusion Detection Methods from the Perspective of the Knowledge Discovery in Databases Process. arXiv preprint arXiv: 2001.09697, 2020.

[32] N Moustafa and J Slay. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In 2015 Military Communications and Information Systems Conference (MilCIS), pages 1–6, 2015.

[33] Srinivas Mukkamala, Andrew H. Sung, and Ajith Abraham. Intrusion detection using an ensemble of intelligent paradigms. Journal of Network and Computer Applications, 28(2): 167–182, 2005.

[34] Hien M Nguyen, Eric W Cooper, and Katsuari Kamei. Borderline over-sampling for imbalanced data classification. In Proceedings: Fifth International Workshop on Computational Intelligence & Applications, volume 2009, pages 24–29. IEEE SMC Hiroshima Chapter, 2009.

[35] Genki Osada, Kazumasa Omote, and Takashi Nishide. Network intrusion detection based on semi-supervised variational auto-encoder. In European Symposium on Research in Computer Security, pages 344–361. Springer, 2017.

[36] Nikunj C Oza and Kagan Tumer. Classifier ensembles: Select real-world applications. Information Fusion, 9(1): 4–20, 2008.

[37] Yoshihiko Ozaki, Yuki Tanigaki, Shuhei Watanabe, and Masaki Onishi. Multiobjective tree-structured parzen estimator for computationally expensive optimization problems. In Proceedings of the 2020 Genetic and Evolutionary Computation Conference, pages 533–541, 2020.

[38] Sandhya Peddabachigari, Ajith Abraham, and Johnson Thomas. Intrusion detection systems using decision trees and support vector machines. International Journal of Applied Science and Computations, 11(3): 118–134, 2004.

[39] Karen Scarfone and Peter Mell. Guide to Intrusion Detection and Prevention Systems (IDPS) Recommendations of the National Institute of Standards and Technology. Nist Special Publication, 800-94: 127, 2007.

[40] Benedetto Marco Serinelli, Anastasija Collen, and Niels Alexander Nijdam. Training guidance with kdd cup 1999 and nsl-kdd data sets of anidinr: Anomaly-based network intrusion detection system. Procedia Computer Science, 175: 560–565, 2020.

[41] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of Bayesian optimization. Proceedings of the IEEE, 104(1): 148–175, 2015.

[42] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In ICISSP, pages 108–116, 2018.

[43] Rahul Sharma, Chien Aun Chan, and Christopher Leckie. Evaluation of centralised vs distributed collaborative intrusion detection systems in multi-access edge computing. In 2020 IFIP Networking Conference (Networking), pages 343–351. IEEE, 2020.

[44] Vadim Smolyakov. Ensemble Learning to Improve Machine Learning Results, 2017.

[45] Steven R Snapp, James Brentano, Gihan Dias, Terrance L Goan, L Todd Heberlein, Che-Lin Ho, and Karl N Levitt. DIDS (distributed intrusion detection system)-motivation, architecture, and an early prototype. 2017.

[46] Salvatore J Stolfo, Wei Fan, Wenke Lee, Andreas Prodromidis, and Philip K Chan. Cost-based modeling for fraud and intrusion detection: Results from the jam project. In Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00, volume 2, pages 130–144. IEEE, 2000.

[47] Jiayu Sun, Xinzhou Wang, Naixue Xiong, and Jie Shao. Learning sparse representation with variational auto-encoder for anomaly detection. IEEE Access, 6: 33353–33361, 2018.

[48] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A Detailed Analysis of the KDD CUP 99 Data Set. In IEEE Symposium on Computational Intelligence for Security and Defense Applications - CISDA, pages 1–6. IEEE, IEEE, 2009.

[49] Daxin Tian, Yanheng Liu, and Yang Xiang. Large-scale network intrusion detection based on distributed learning algorithm. International Journal of Information Security, 8(1): 25–35, 2009.

[50] Michal Tkáč and Robert Verner. Artificial neural networks in business: Two decades of research. Applied Soft Computing, 38: 788–804, 2016.

[51] Chih Fong Tsai, Yu Feng Hsu, Chia Ying Lin, and Wei Yang Lin. Intrusion detection by machine learning: A review. Expert Systems with Applications, 36(10): 11994–12000, 2009.

[52] Bo K Wong, Thomas A Bodnovich, and Yakup Selvi. Neural network applications in business: A review and analysis of the literature (1988–1995). Decision Support Systems, 19(4): 301–320, 1997.

[53] Yanqing Yang, Kangfeng Zheng, Chunhua Wu, and Yixian Yang. Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network. Sensors, 19(11): 2528, 2019.

[54] Chuanlong Yin, Yuefei Zhu, Jinlong Fei, and Xinzheng He. A deep learning approach for intrusion detection using recurrent neural networks. Ieee Access, 5: 21954–21961, 2017.

[55] Anazida Zainal, Mohd Aizaini Maarof, and Siti Mariyam Shamsuddin. Ensemble classifiers for network intrusion detection system. Journal of Information Assurance and Security, 4(3): 217–225, 2009.

Csaba Brunner received both his M.Sc. Degree with distinction and Ph.D. Degree at the Corvinus University of Budapest in 2016 and 2020, respectively. During his Ph.D. studies, and following his degree he worked for EPAM Systems inc., a company working primarily in software development, as a Data Scientist developing machine learning models. Dr. Brunner's research interests are autoencoder networks and their various application areas, particularly in the fields of computer security and anomaly detection. Furthermore Dr. Brunner is also interested in the fields of computer vision and natural language processing.

Andrea Kő, Ph.D., Habil, CISA is a Professor and Director of the Institute of Information Technology of Corvinus University of Budapest. She has a University Doctoral degree in Computer Science (1992) from Corvinus University of Budapest, Hungary and a PhD degree in Management and Business Administration (2005) from Corvinus University of Budapest, Hungary. She participated in several international and national research projects and published more than 140 papers in international scientific journals and conferences (https://scholar.google.com/citations?hl=en&user=EXE1k5H_QBAC&view_op=list_works&sortby=pubdate). Her research interests include digital transformation, intelligent systems, business analytics, and semantic technologies.

Szabina Fodor, Ph.D., Habil is an associate professor at Corvinus University of Budapest, Hungary. She has a Ph.D. in Computer Sciences from Eötvös Loránd University, Budapest, Hungary. She published more than 20 papers in international scientific journals (https://scholar.google.com/citations?hl=en&user=wZ-_3jYAAAAJ&view_op=list_works&&sortby=pubdate). Her main research interests include NPL technologies, numerical analysis of ABS-based mathematical algorithms, gamification and data-driven research in higher education.