Journal of Intelligent Information Systems (2023) 60:407–436
https://doi.org/10.1007/s10844-022-00747-z

Leveraging siamese networks for one-shot intrusion detection model

Hanan Hindy[1,2] · **Christos Tachtatzis**[3] · **Robert Atkinson**[3] · **David Brosset**[4] · **Miroslav Bures**[5] · **Ivan Andonovic**[3] · **Craig Michie**[3] · **Xavier Bellekens**[6]

Received: 7 April 2022 / Revised: 7 September 2022 / Accepted: 9 September 2022 /
Published online: 5 November 2022

Abstract

The use of supervised Machine Learning (ML) to enhance Intrusion Detection Systems (IDS) has been the subject of significant research. Supervised ML is based upon learning by example, demanding significant volumes of representative instances for effective training and the need to retrain the model for every unseen cyber-attack class. However, retraining the models in-situ renders the network susceptible to attacks owing to the time-window required to acquire a sufficient volume of data. Although anomaly detection systems provide a coarse-grained defence against unseen attacks, these approaches are significantly less accurate and suffer from high false-positive rates. Here, a complementary approach referred to as "One-Shot Learning", whereby a limited number of examples of a new attack-class is used to identify a new attack-class (out of many) is detailed. The model grants a new cyber-attack classification opportunity for classes that were not seen during training without retraining. A Siamese Network is trained to differentiate between classes based on pairs similarities, rather than features, allowing to identify new and previously unseen attacks. The performance of a pre-trained model to classify new attack-classes based only on one example is evaluated using three mainstream IDS datasets; CICIDS2017, NSL-KDD, and KDD Cup'99. The results confirm the adaptability of the model in classifying unseen attacks and the trade-off between performance and the need for distinctive class representations.

Keywords Continuous learning · Intrusion detection · Few-shot learning · Siamese Network · CICIDS2017 · NSL-KDD · Artificial Neural Network

✉ Hanan Hindy
hanan.hindy@cis.asu.edu.eg

✉ Xavier Bellekens
xavier@lupovis.io

Extended author information available on the last page of the article

1 Introduction

Intrusion Detection System (IDS) development has its roots in statistical models, and has recently evolved to the use of Machine Learning (ML) (Buczak & Guven, 2016) based on hybrid models and adaptive techniques (Hindy et al., 2020). Developments to date have highlighted two fundamental considerations in the design of effective supervised ML-based IDS; (a) availability of a large and representative historian of cyber-attacks consisting of many thousands of instances (Li et al., 2013) and (b) the time window resulting from the need to retrain models after the emergence of a new attack class has been recorded, renders the network open to damaging attacks. Supervised ML models are very accurate at identifying cyber-attacks previously been trained to recognise, but significantly under-perform for new unseen and “zero-day” attacks that emerge. Anomaly detection approaches have been explored to address the issue and whilst these schemes provide better performance against unseen attacks, their efficacy is inferior against known attacks when compared to supervised ML approaches. Further, anomaly-based approaches are also limited under multiple new attacks scenarios as they are simply classified into the same anomalous group, in so doing restricting the range of attack-specific countermeasures that can be employed.

Here, the development and evaluation of an ML-enabled approach that provides improved attack identification in the period between a range of previously unseen attacks at onset is reported and the deployment of a robust supervised ML model that informs on the most effective countermeasures. The methodology - referred to as One-Shot Learning - centres on the use of a Siamese Network, shown to be effective in identifying new classes based on one (or only a few) examples of a new class. An alternative approach is to create synthetic examples based on the domain knowledge of new attacks; however, this is challenging requiring a considerable amount of time to replicate a suitable representation of an environment with appropriate parameters, and is consequently subject to human error owing to cognitive biases.

One-Shot Learning was inspired by the generalisation learning ability of human beings. As discussed by Vinyals et al. (2016), “Humans learn new concepts with very little supervision, yet our best deep learning systems need hundreds or thousands of examples” (Vinyals et al., 2016). Therefore, One-Shot learning models aim at classifying previously unseen classes using one instance. The idea is to rely on previously seen classes and learn patterns and similarities instead of fitting the ML model to fixed classes. Few-Shot (N-Shot) learning is similar to One-Shot learning with a flexibility of using a few (N) instances to classify a class instead of one (Sun et al., 2019).

A Siamese Network is a network composed of two “twin” networks that are trained simultaneously to learn the similarity of two instances, called a pair. Leveraging this similarity-based learning, a previously unseen class could be added to the network without retraining. The initial stage of the development is the training phase. The Siamese Network is trained using similarities that discriminate between K classes; benign traffic and the $K - 1$ classes of *known* cyber-attacks. Any new traffic instance P is then compared against all *known* classes (used during training) plus an additional class ($K+1$ classes) where only a limited number of examples of class “$K+1$” are available, such as might be the case on the appearance of a *new* cyber-attack. This is achieved without any form of additional training.

The contributions of the paper are; (a) the use of a Siamese Network model to successfully classify new cyber attacks based on pair similarities solely, not reported for unknown attack classification usage to date. (b) evaluation of the proposed model performance to detect a new cyber-attack class based on one labelled instance without retraining. (c) evaluation of the proposed model performance to correctly classify two new cyber-attack classes without retraining. (d) comparison of the impact of a few labelled instances of the new attack class on detection performance. This paper paves the way researchers to start exploring the utilisation of One-Shot learning for IDS development.

The remainder of this paper is organised as follows; Section 2 details the main features of Siamese Networks; Section 3 outlines the related work; Section 4 depicts the Siamese Network architecture. Section 5 presents the methodology governing the training of the Siamese Network and its evaluation is explained showing the potential of the network to identify a new attack class based on a few (previously collected and labelled) examples of that attack class without retraining. Section 6 presents the properties of the datasets and their corresponding attack classes used in model development and performance evaluation; The performance of the model is assessed in Section 7; conclusions are drawn in Section 8.

2 Background

In supervised machine learning, a relationship exists between model complexity and the volume of training data; too few training examples and the model will over-fit, resulting in an unnecessarily complex model that produces poor results. Therefore, securing sufficient and representative data is a limiting factor in model development and performance (Jain, 2017). In practice, accessing and/or generating sufficiently large and representative training examples is a complex challenge and may involve significant manual effort and processing time (Roh et al., 2019). Nonetheless, there are publicly available datasets for training IDS systems, notably the KDD and CICIDS dataset families. These data are used to pre-train the Siamese Network, subsequently, in the evaluation of the performance of the model in identifying a new class of attack after a limited number of that class' samples has been recorded.

An alternative approach is to utilise "*Transfer Learning*" to mitigate the need for large volumes of training data (Pan et al., 2010). The premise of Transfer Learning to solve the target problem T (where data are limited), is to create a model M for a similar problem T' where large amounts of data are readily available. The initial model M is then "transferred" to the target problem T and partially re-trained on the small dataset. The rationale is that the initial training on T', yields training weights which discover features useful for the problem domain and hence applicable to the target problem T; hence after retraining, the model learns and generalises faster on the small dataset (Wang et al., 2017). Despite the potential of Transfer Learning as a viable solution, it does not eliminate the need for retraining.

Although transfer learning reduces training time, additional challenges are introduced; (a) identification of a suitable pre-trained model *"What to transfer?"* (Pan et al., 2010), (b) selection of the most appropriate tuning of the pre-trained model aligned to the new application domain *"How and When to transfer?"* (Pan et al., 2010) and (c) a reduction of the learning performance of the target domain known as "Negative Transfer" (Pan et al., 2010; Torrey and Shavlik, 2010). Transfer learning is a common approach in image processing where for example, models are trained on the ImageNet dataset (Nguyen et al., 2018). Unlike image processing, datasets are not, as yet, standardised in the cyber security domain which presents a significant additional challenge. Recent research on IDS proposed approaches in this respect (Singla et al., 2019).

One-Shot learning, first reported by Fei-Fei et al. (2006), is inspired by human generalisation learning and has been applied in multiple domains with the most prominent being image and video processing (Wang et al., 2018). One-shot learning has also been used in other domains, such as robotics (Bruce et al., 2017), language processing (Zhang and Zhao, 2018) and drug discovery (Altae-Tran et al., 2017). Considering the particular needs of the cybersecurity domain and the needs for IDS, the One-shot learning models that have been developed for other application domains are not directly applicable. In addition, the domain-specific data, features and requirements render the application of models from other domains directly invalid and thus adaptation of models is a necessity.

Based on the literature, the Siamese Network is the most frequently used. Various architectures have been proposed and assessed as the building block for the twin network (i.e., CNN (Chung et al., 2017; Chung & Weng, 2017), RNN (Tolosana et al., 2018) and GNN (Garcia & Bruna, 2017)). Matching Networks (Vinyals et al., 2016), Prototypical Networks (Snell et al., 2017) and Imitation Learning (Duan et al., 2017), particularly in the image processing domain, but amenable to be generalised to other domains.

3 Related work

Siamese Networks and Deep Metric Learning approaches have been proposed in the literature for IDS usage, however, they have not been proposed for One-Shot learning or for detecting attacks that are not included during the training phase. Moustakidis and Karlsson (Moustakidis & Karlsson, 2020) applied Siamese Networks for reducing dimensionality for a better preforming IDS. Andresini et al. (2021) proposed the use of Triplet Networks to learn the network feature embedding for better IDS performance. While Bedi et al. (2021, 2020) improves the IDS classification performance by using Siamese Networks to handle imbalanced classes problem by automatically detecting and handling majority and minority classes.

To the best of the authors' knowledge, the development reported here is the first proposing a One-Shot IDS model implementation. Although there are various manuscripts using ML and DL for IDS, comparing the proposed model in this paper with recent IDS models is not applicable. This is because the proposed model leverages One-Shot learning and aims to classify a class that was not used in the training phase. Therefore, it cannot be in comparison with classical classification models. However, an understanding of the classification performance is important to aid in the interpretation of the results discussed in Section 7.

Table 1 summarises the *classification* results of recent IDS studies that address multi-class attack classification and report explicit class metrics, not only the overall accuracy. Although a direct performance comparison is impractical, nevertheless these results assists in the appreciation of the performance of the different classes, captured when all classes are used during training. The results provide a reference with which to evaluate results reported when classes are excluded from training.

As shown in Table 1, the overall classification accuracy is higher than each class performance owing to class imbalance. For example, the TPR for the SSH and FTP attack classes in the CICIDS2017 dataset are 0% and 3.1% respectively, while the accuracy is 96% (Vinayakumar et al., 2019). Similarly, the TPR for the R2L and U2R in the KDD Cup'99 dataset are 24.3% and 15.5% respectively with an overall accuracy of 92.6%. Class imbalance is a common problem and is considered relative to the degree of imbalance, the

Table 1 Recent IDS studies for multi-class classification performance

Year/ Reference	ML Technique	Metric	Result
CICIDS2017			
2020/(Hossain et al., 2020)	Multi-layer Perceptron	Recall	SSH: 98%
			FTP: 77%
		Accuracy	Overall: 96%
2019/(Vinayakumar et al., 2019)	Deep Neural Network with 1 Layer	True Positive Rate	Normal: 64.6%
			SSH: 0%
			FTP:3.1%
			DDoS: 9.5%
KDD Cup’99			
		Accuracy	Overall: 92.6%
2019/(Vinayakumar et al., 2019)	Deep Neural Network with 1 Layer	True Positive Rate	Normal: 99.4%
			DoS: 93.9%
			Probe: 73.2%
			R2L: 24.3%
			U2R: 15.5%
NSL-KDD			
2021/(Bedi et al., 2021)	XGBoost & Siamese-NN	Recall	Normal:89.1%
			DoS:86.8%
			Probe:77.7%
			R2L:32.8%
			U2R:50.8%
2020/(Bedi et al., 2020)	Siamese-NN	Recall	Normal:91.22%
			DoS:85.37%
			Probe:48.66%
			R2L:33.25%
			U2R:56.72%
2020/(Li et al., 2020)	Multi-Convolutional Neural Network	Recall	KDDTest:$^{\pm}$
			Normal: 91.19%
			DoS: 86.63%
			Probe: 83.73%
			R2L: 35.15%
			U2R: 23.50%
			KDDTest:$^{-21}$
			Normal: 62.08%
			DoS: 77.04%
			Probe: 82.60%
			R2L: 35.15%
			U2R: 23.50%
		Accuracy	Overall: 77.8%
2019/(Vinayakumar et al., 2019)	Deep Neural Network with 1 Layer	True Positive Rate	Normal: 97.3%
			DoS: 77.7%
			Probe: 61%
			R2L: 43.3%
			U2R: 24.1%
2019/(Illy et al., 2019)	Ensemble model	Overall	KDDTest^{+}: 83.83%
			KDDTest^{-21}: 78.33%

overall dataset size, and the complexity of the data. Upsizing and downsizing are known techniques to handle class imbalance problem (Japkowicz and Stephen, 2002; Johnson & Khoshgoftaar, 2019). It is important to note that the class imbalance problem did not pose a problem for the method presented in this paper. This is due to the fact that equal number of pairs are randomly selected from a pool of instances, which ensures a balance in training and testing.

4 Siamese network architecture

Siamese Networks were first introduced by Bromley et al. (1994) in the 90s to solve the problem of matching hand-written signatures, subsequently adapted to other domains. Popular implementations of Siamese Networks for image and video processing are presented by Koch et al. (2015), Yao et al. (2018), and Varior et al. (2016). Moreover, it has been implemented for Natural Language Processing (NLP) tasks (Benajiba et al., 2019) and for the retrieval of similar questions (Das et al., 2016).

Figure 1 depicts the Siamese Network architecture composed of two identical subnetwork that share weights. The two networks are referred to as "Twin networks" and share a common architecture, i.e., two identical networks. The weights of the twin networks are initialised with random weights and pass their outputs to a similarity module, which in turn is responsible for calculating the distance defining "how alike" the two inputs are. The output of the latter is a comparison based on the similarity i.e., whether or not the pair are similar, the loss is then calculated and the weights are updated based on gradients.

Formally (Koch et al., 2015; Shaham and Lederman, 2018), given a pair of inputs (x_1,x_2) and a twin network (X,Y), such that x_1 is the input of X and x_2 is the input of Y, the similarity can be computed using Euclidean distance (1):

$$d_2 = \sqrt{\sum_{i=1}^{n}(f_1(x_1)_i - f_2(x_2)_i)^2} \tag{1}$$

such that f_1 and f_2 are the outputs of Networks X and Y respectively $f_1 \equiv f_2$ since X and Y are twin networks. Ultimately, the training goal is to minimise the overall loss l as defined in (2); for each given batch i of input pairs $(x_1,x_2)_i$ and label vector y_i, such that $y_i(x_1,x_2)_i = 1$ if x_1 and $x2$ belong to the same class and 0 otherwise.

$$l(x_1, x_2)_i = y(x_1, x_2)_i \log d_i + (1 - y(x_1, x_2)_i) \log(1 - d_i) + \lambda w^2 \tag{2}$$

such that λ is a l_2 regularisation parameter.

However, this loss function is sensitive to outliers (i.e. dissimilar pairs with large distances) which disproportionately affect the gradient estimation. An alternative loss function is the contrastive loss shown in (3) proposed by Chopra et al. (2005) and Hadsell et al. (2006). The contrastive loss caps the contribution of dissimilar pairs if the distance is within a specified margin m (Hadsell et al., 2006), hence limiting the effect of large distances.

$$l(x_1, x_2) = \sum_{n=1}^{B} y(x_1, x_2)_i * (d_i)^2 + (1 - y(x_1, x_2)_i) * (max(m - d_i, 0))^2 \tag{3}$$

such that $m > 0$ is a margin. In this study, the margin was set to $m = 1$ (Hadsell et al., 2006).

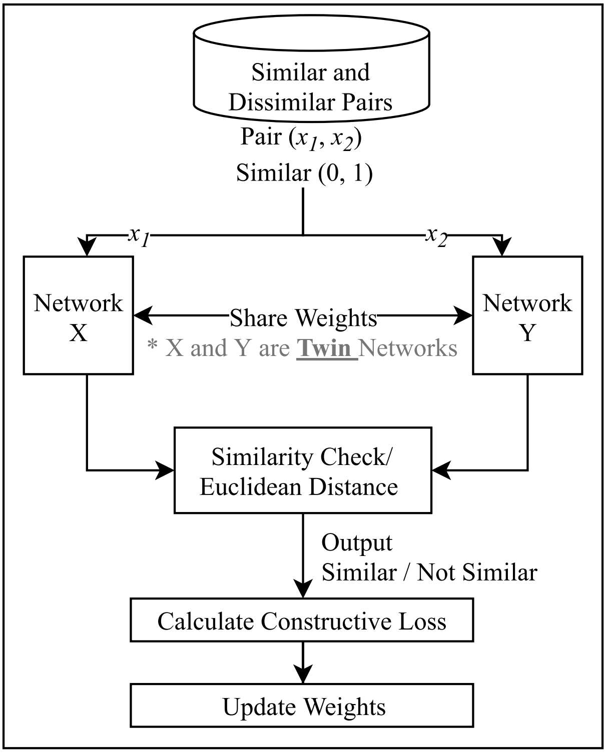


Fig. 1 Siamese Network Architecture

After training, given any two pairs, the network is capable of calculating their degree of similarity, $d_i \in [0,1]$, d_i mirror the degree of similarity for the pair; the lower the d_i, the closer the pair. Batches of pairs are used to train the network. Note, however, that an equal number of similar and dissimilar pairs are used in the batch.

The choice of the twin networks architecture is domain specific and based on the application context. Artificial Neural Network (ANN), Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) are commonly used architectures for establishing twin networks. CNNs are well-suited for image processing whilst LSTMs are routinely used with temporal data. In this context, ANNs are used as the building block of the twin network as their structure is aligned to the structure and format of the data used. Similar to a single ANN, the Siamese Network is trained in a back-propagation fashion. The twin networks are initialised with the same weights and during training, batches of similar and dissimilar pairs are used to calculate the loss, using the function given in (3). The weights are then updated based on the learning rate, gradient descent and optimisation function as shown in (4). Hyperparameter optimisation is performed to determine the model's set of optimal parameters.

$$\boldsymbol{W}_{t+1} = \boldsymbol{W}_t - \eta \frac{dE}{d\boldsymbol{W}_t} \tag{4}$$

such that η is the learning rate, and E is the error function.

The details of the optimised architecture (i.e., the number of layers, neurons, learning rate, etc.) are provided in Section 7.

5 Siamese network model

In this section, the proposed Siamese Network model is used as the One-Shot learning architecture. The performance of the network on classifying a new cyber-attack class without the need to retrain is evaluated with the new attack class represented by a limited number of labelled samples. This assess the capability of the Siamese Network to find similarity between pairs of classes that were not a part of the training.

Figure 2 shows the overall process of establishing the intrusion detection model based on one-shot learning and illustrates the methodology of assessing performance for new attack classes without retraining the model.

Given a dataset with N classes, first, an attack class e is chosen to act as the new cyber-attack; this class is excluded from the training process (Fig. 2-(1)). Second, for the remaining K classes after excluding e ($N-1$ classes), each class instances are split into two pools, as shown in Fig. 2-(2). Collectively, the first "half" is used as a pool of instances to generate the training set pairs both similar and dissimilar, as shown in Fig. 2-(4); the second "half" is used as the evaluation pool of instances.

Class e is used to mimic a real-life situation in which a new attack is detected with only a few labelled samples available. Therefore, the instances of e are split in two halves (Fig. 2-(3)), the first half representing a pool of labelled and the second half a pool of unlabelled (new) instances.

Since the model relies on random pair generation, pairs are drawn out randomly from the pools of instances. The rational for having pools of instances and to draw out pairs randomly is to hinder any selection bias either during training (i.e. selecting similar and dissimilar pairs) or during evaluation of the new class (i.e. selecting the labelled instances that best represent this class). Furthermore, the uniqueness of the pairs - no duplicates - is ensured. A "set" data structure is used. it is added to the batch of pairs unless that pair is already contained within the set. This is demonstrated in Algorithm 1. It is important to note that the choice of Siamese network training pairs is an open research question in the literature (Martin et al., 2018).

During evaluation, an instance i is paired with one random instance from each class. The instances are drawn out of the pool of testing instances only, resulting in N pairs. The similarity is then calculated for the N pairs. Instance i is classified (labelled) based on the pair with the highest similarity (i.e. least distance).

As discussed in Section 7, to determine the trade-off between the number of labelled instances of the new attack class and accuracy, the process is repeated j times for each instance i. Majority voting is then applied to deduce the instance label; the class with the highest votes is used as instance i label (Fig. 2-(7)).

Algorithm 2 summarises the overall process of training and testing the model. Initially, a network architecture is determined, the number of input neurons being a function of the number of features with one neuron as the output layer. The number of hidden layers and number of neurons in each layer is then determined; each hidden layer has a number of neurons that are reduced by a fraction from the previous layer. The tuning of the architecture is performed using ANN parameter optimisation. During the

Algorithm 1 Generate training batch

Input: Dataset of $K(N-1)$ classes, Batch Size
Output: Batch of similar and dissimilar pairs and associated labels (0: dissimilar, 1: similar)

function GETTRAININGBATCH(batch_size)
  $num\_similar\_pairs = batch\_size/2$
  $num\_dissimilar\_pairs = batch\_size/2$
  $num\_similar\_pairs\_per\_class = num\_similar\_pairs/K$
  $all\_combinations = combinations(K)$
  $num\_dissimilar\_pairs\_per\_combination = num\_dissimilar\_pairs/len(all\_combinations)$
  $pairs\_set \leftarrow \{\}$
  for c in K **do**
    for $i = 0$ to $num\_similar\_pairs\_per\_class$ **do**
      $(ins_1, ins_2) \leftarrow$ 2 random instances $\in c_i$
      if $(ins_1, ins_2) \in pairs\_set$ **then**
        go to 10
      end if
      $pairs[i] \leftarrow \{ins_1, ins_2\}$
      $pairs\_set.add(\{ins_1, ins_2\})$
    end for
  end for
  for c_1, c_2 in $all\_combinations$ **do**
    for $i = 0$ to $num\_dissimilar\_pairs\_per\_combination$ **do**
      $ins_1 \leftarrow$ random instance $\in c_1$
      $ins_2 \leftarrow$ random instance $\in c_2$
      if $(ins_1, ins_2) \in pairs\_set$ **then**
        go to 20
      end if
      $pairs[i] \leftarrow \{ins_1, ins_2\}$
      $pairs\_set.add(\{ins_1, ins_2\})$
    end for
  end for
  $targets[0..batch\_size/2] \leftarrow 0$ ▷ Similar
  $targets[batch\_size/2..batch\_size] \leftarrow 1$ ▷ Dissimilar
  return $pairs, targets$
end function

training phase, both training and validation loss curves are monitored to ensure that the network converges, while avoiding overfitting. The parameters (the number of hidden layers, number of neurons in each layer, η - learning rate -, number of epochs, etc) are chosen based on the optimised state of the model.

Furthermore, it is important to note that regularisation of the network is carried out on the onset of unstable behaviour during training. Figure 3 shows an unstable network performance state.

As a result, the regularisation parameters of the network are reconsidered and dropout layers and kernel regularisation were added to obviate over-fitting and ensure network

Algorithm 2 Train and test siamese network

Input: Attacks Dataset
Output: Trained Siamese Network Evaluation
Ensure: $dataset = \{c_1, c_2, \ldots, c_n\}\ s.th.\ n \geq 3$
$train\_batch\_size, test\_batch\_size \leftarrow 30,000$
$n\_epochs \leftarrow 2000$
$excluded\_class =$ random class e s.th. $e \in dataset$
$training\_classes = dataset - e$
$training = 50\%\ c_i\ \forall c_i \in training\_classes$
$testing = dataset \cap \overline{training}$
$batch \leftarrow$ GetTrainingBatch($train\_batch\_size$)
Build Siamese Network with Random Weights
for $i = 0$ to $n\_iterations$ **do**
    Update Siamese Network Weights based on $batch$
end for
Evaluate(test_batch_size)

convergence. This is distinctly observed in Figs. 4 and 5. The models' architectures presented in Section 7 follow convergence validation.

Initially, the dataset is split as shown in Fig. 2. The model is trained for the optimal number of epochs with the generated batch of pairs as described in Algorithm 1. The *batch_size* =30,000 is based on the literature recommendation for the advisable Siamese Network training batch size (Pang et al., 2019; Koch et al., 2015). It is important to note that the classes are equally represented in both the training and testing batches. Note that the dataset should have at least 3 classes, otherwise, the model converges to a 50% similarity output and fails to train adequately. Algorithm 1 shows the training batch generation process.

An equal number of instances are used from each class for evaluation (Algorithm 3). For each new instance, a pair is selected with each class using the new instance and a random instance from each class. The similarity is calculated for each pair. The pair with the closest similarity contributes to the classification result. The process is performed j times and majority voting is used to collate the results ($j \in \{1,5,10, 15,20,25,30\}$). For class e (the attack class that is excluded from training), the first half acts as the pool of labelled and the second half act as the pool of new unlabelled instances.

6 Dataset

Three datasets are used to evaluate the proposed models. These datasets cover two benchmark IDS datasets, specifically, CICIDS2017 and NSL-KDD. Moreover, KDD Cup'99 is used in comparison to the NSL-KDD to demonstrate the effectiveness of having clean data when generating training pairs and also, when introducing new attacks to the trained model.

Each dataset contains N classes. K classes are used to train the network, such that $K = N - 1$. The K classes include normal/benign and $K - 1$ attack classes. The instances of each of the K class act as a pool used to generate similar and dissimilar pairs. Furthermore, one class is used to simulate a new attack, mimicking the situations in which

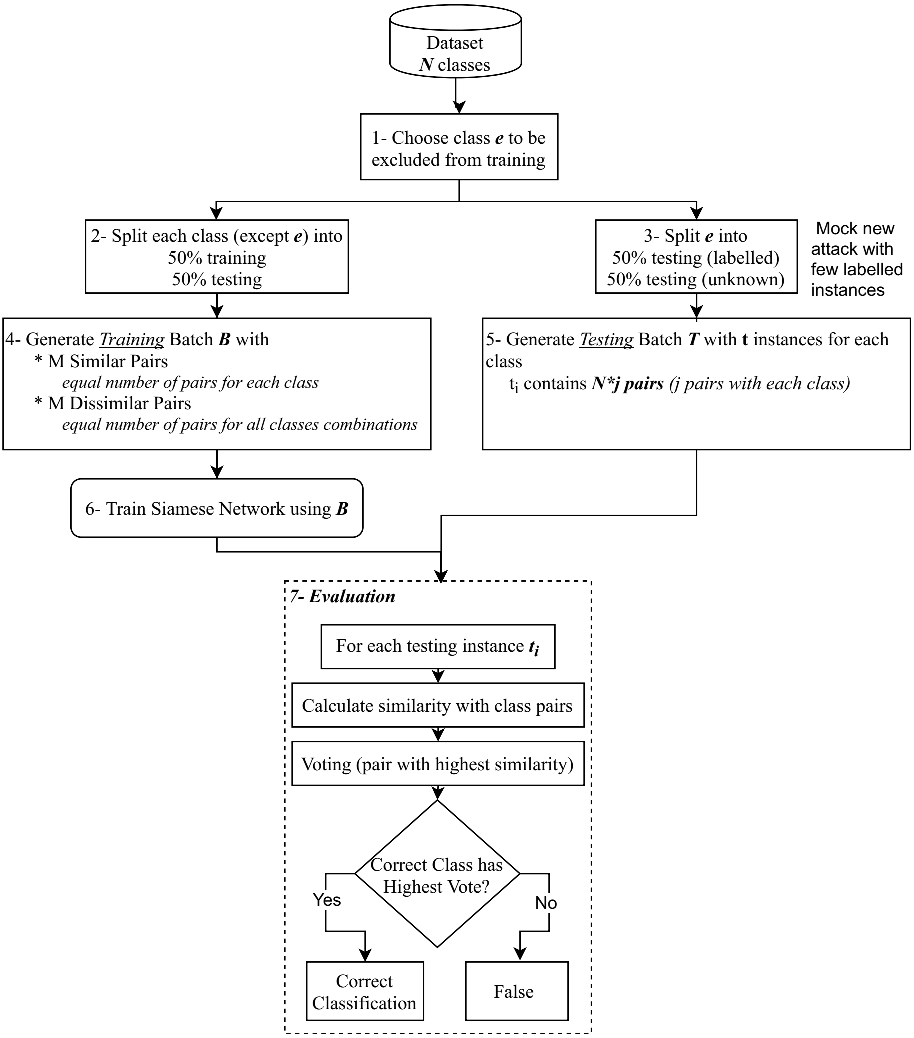


Fig. 2 Siamese Network for Intrusion Detection System (One-Shot)

little/limited data is available for a new attack. An overview of each dataset is presented in the following subsections.

6.1 CICIDSS2017

CICIDS2017 (Sharafaldin et al., 2018) is a recent dataset generated by the Canadian Institute for Cyber security (CIC) comprising up-to-date benign, insider, and outsider attacks. Bidirectional flow features are extracted from the raw ".pcap" files provided by the dataset. Then, the flows are labelled according to the published timestamps of the CICIDS2017 dataset. Table 2 lists the attacks used and the number of instances/flows for each.

Fig. 3 Siamese Network Loss Curve (Non-converging case)

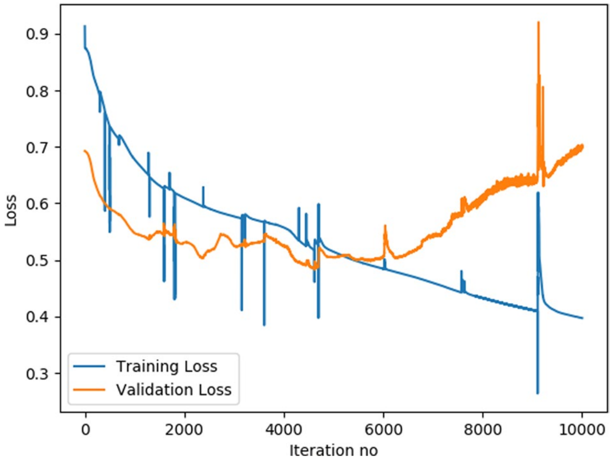


Fig. 4 Siamese Network Loss Curve (Converging case) - 1

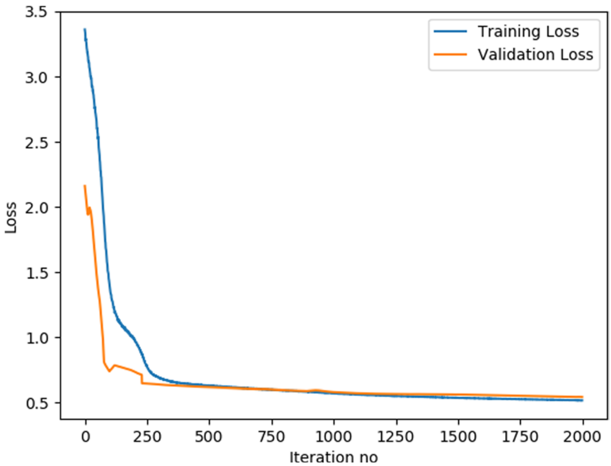
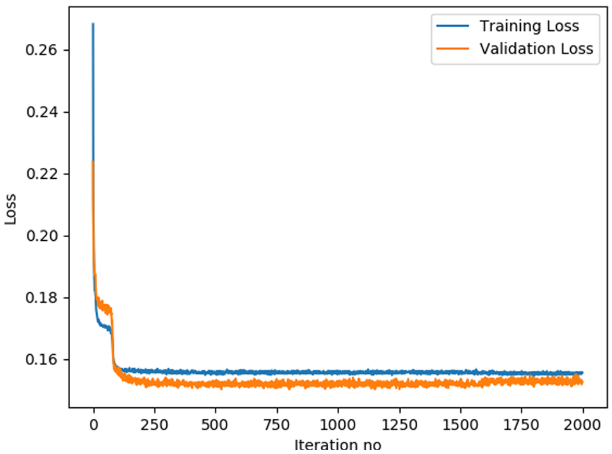


Fig. 5 Siamese Network Loss Curve (Converging case) - 2



Algorithm 3 Evaluate model

```
Input: Trained Siamese Network, Batch Size, Excluded Class (e)
Output: Accuracy
function EVALUATE(batch_size)
    n_correct ← 0
    num_per_class ← batch_size/N
    K ← N − e
    for c in N do
        for i = 0 to num_per_class do
            if c == e then
                ins_1 ← random instance ∈ e_unlabelled
            else
                ins_1 ← random instance ∈ c_testing
            end if
            for j = 0 to 5 do
                pairs ← (ins_1, random instance ∈ x∀x ∈ K)
                pairs.append(ins_1, random instance ∈ e_labelled
                similarities ← model.predict(pairs)
                votes[argmin(similarities)]+ = 1
            end for
            if argmax(votes) == c then
                n_correct+ = n_correct + 1
            end if
            confusion_matrix[c, argmax(votes)]+ = 1
        end for
    end for
    accuracy = n_correct ∗ 100/batch_size
    return accuracy, confusion_matrix
end function
```

6.2 KDD Cup'99

The KDD Cup'99 (Hettich and Bay, 1999a), although old, is still considered as the classic benchmark data set used in the evaluation of IDS performance. More than 60% of the research in the past decade (2008 - 2018) has been evaluated using KDD'99 (Hindy et al., 2020). KDD Cup'99 covers 4 attack classes alongside normal activity. The attacks contained in the data set are; Denial of Service (DoS), Remote to Local (R2L), User to Root (U2R) and probing.

The KDD Cup'99 data set is relatively large, however, the provider has made available a reduced subset of ˜10% (Hettich & Bay, 1999b). For the purposes of evaluation here, only the smaller subset is used. Table 3 shows the number of instances per class for the KDD Cup'99 data set.

Table 2 CICIDS2017 Classes and Corresponding Number of Occurrences

	Class	# of Occurrences
1	Normal	248607 (90.50%)
2	DoS (Hulk)	14427 (5.25%)
3	DoS (Slowloris)	2840 (1.03%)
4	FTP Brute Force	5228 (1.9%)
5	SSH Brute Force	3627 (1.32%)

Table 3 KDD Cup'99 Classes and Corresponding Number of Occurrences

	Class	# of Occurrences
1	Normal	97278 (19.70%)
2	DoS	391458 (79.24%)
3	Probe	4107 (0.82%)
4	U2R	1128 (0.23%)
5	R2L	52 (0.01%)

Table 4 NSL-KDD Classes and Corresponding Number of Occurrences

	Class	# of Occurrences
1	Normal	67343 (53.46%)
2	DoS	45927 (36.47%)
3	Probe	11656 (9.25%)
4	U2R	995 (0.78%)
5	R2L	52 (0.04%)

6.3 NSL-KDD

The NSL-KDD (for Cybersecurity, 2022) data set was proposed by the CIC to overcome the problems of the KDD Cup'99 set discussed by Tavallaee et al. (2009). Similar to KDD Cup'99, NSL-KDD covers 4 attack classes alongside normal activity. NSL-KDD is used for evaluating the effect of enhancing and filtering a data set on the similarity learning and performance. Table 4 shows the number of instances per class for the NSL-KDD data set.

NSL-KDD and KDD Cup'99 data sets have already been pre-processed and 42 features are available, a total of 118 features after encoding the categorical features. For the CICIDS2017, 31 bidirectional flow features are extracted. It is worth noting that no feature engineering or selection is performed to ensure that the excluded class from training does not indirectly influence the feature set.

7 One-shot evaluation

7.1 Evaluation metrics

This section discusses the metrics used to evaluate the model. The model evaluation (Algorithm 3) yields a Confusion Matrix (CM) that outlines the performance. A sample CM is presented in Table 5. Each row of the CM represents a class; True Positive (TP) is the number of attack instances correctly classified as attack; True Negative (TN) is the number of normal instances correctly classified as normal; False Positive (FP) is the number of normal instances wrongly classified as attack; False Negative (FN) is the number of attack instances wrongly classified as normal.

The overall accuracy is calculated as shown in (5). True Positive Rate (TPR) and False Negative Rate (FPR) for each class are shown in (6) and (7) respectively; finally, True Negative Rate (TNR) and False Positive Rate (FPR) are calculated using (8) and (9) respectively.

Table 5 Sample Confusion Matrix

	Predicted Class				
Correct	Normal	Attack_1	Attack_2	Attack_3	Attack_4
Normal	TN	FP_1	FP_2	FP_3	FP_4
Attack_1	FN_1	TP_{11}	TP_{12}	TP_{13}	TP_{14}
Attack_2	FN_2	TP_{21}	TP_{22}	TP_{23}	TP_{24}
Attack_3	FN_3	TP_{31}	TP_{32}	TP_{33}	TP_{34}
Attack_4	FN_4	TP_{41}	TP_{42}	TP_{43}	TP_{44}

$$OverallAccuracy = \frac{TN + \sum_{i=1}^{4} TP_{ii}}{TN + \sum_{i=1}^{4}\sum_{j=1}^{4} TP_{ij} + \sum_{i=1}^{4} FP_i + \sum_{i=1}^{4} FN_i} \tag{5}$$

$$TPR_i = \frac{TP_{ii}}{FN_i + \sum_{j=1}^{4} TP_{ij}} \tag{6}$$

$$FNR_i = \frac{FN_i}{FN_i + \sum_{j=1}^{4} TP_{ij}} \tag{7}$$

$$TNR = \frac{TN}{TN + \sum_{i=1}^{4} FP_i} \tag{8}$$

$$FPR = \frac{\sum_{i=1}^{4} FP_i}{TN + \sum_{i=1}^{4} FP_i} \tag{9}$$

7.2 Results

7.2.1 One excluded class

The number of hidden layers and neurons for the ANNs used as the building block for the twin networks and their optimised architecture are as follows (**bold** is used for the input layer, *italic* is used for the output layer of the Siamese Network before similarity calculation and Dr is a Dropout layer).

- CICIDS2017: **31**:25:Dr(0.1):20:Dr(0.05):*15*
- NSL-KDD and KDD Cup’99: **118**:98:Dr(0.1):79:Dr(0.1):59:Dr(0.1):39:Dr(0.1):*20*

The following lists the optimised hyper-parameters:

- Activation function: Relu
- L2: 0.001
- Optimiser: Adam

- Number of Epochs: 2000

The evaluation specifies how accurately the proposed network can classify both classes used in training and new attack classes without the need for retraining. The model leverages similarity-based learning. The new attack class is represented using one sample to mimic the labelling process of new attacks.

For each dataset evaluation, multiple experiments are conducted. Specifically, K $(N-1)$ experiments are evaluated, where N is the number of classes and K is the number of attack classes in order to evaluate the performance of the Siamese Network when using a different set of attack classes for training and evaluation. In each experiment, a separate attack class (e) is excluded, one at a time. The CM is presented alongside the overall model accuracy for each experiment.

The results of the evaluation of the performance impact of the number of labelled samples (j) of the new attack class e are presented in terms of overall accuracy, new attack True Positive Rate (TPR) and False Negative Rates (FNR), Normal True Negative Rate (TNR) and False Positive Rate (FPR), listed using j instances for majority voting, where $j \in \{1,5, 10,15,20,25,30\}$. The CMs use $j=5$.

The CMs of the CICIDS2017 One-Shot, excluding SSH class is presented in Tables 6 and excluding FTP in Table 7. The overall accuracy is 81.28% and 82.5% respectively. The results demonstrate the network capability to adapt to the emergence of a new cyber-attack after training. It is important to note that the new attack class performance is 73.03% and 70.03% for SSH and FTP respectively. Moreover, the added class demonstrates low FNRs, specifically 8% and 15% for FTP and SSH respectively.

Additionally, compared to the TPR of recent research, it is shown that when performing a multi-class classification using ANNs with all classes included in both training and testing, the SSH and FTP recall are 98% and 77% respectively (Hossain et al., 2020). In another study the TPRs are 0% and 3.1% respectively (Vinayakumar et al., 2019). One-to-one comparison is not practical, since in the proposed model, classes are excluded from training, but the multi-class classification results provide context and show that the proposed model results

Table 6 CICIDS2017 One-Shot Confusion Matrix (SSH not in Training)

Correct	Predicted Class					Overall
	Normal	DoS (Hulk)	DoS (Slowloris)	FTP	SSH	
Normal	4711	9	103	148	1029	81.28%
	(78.52%)	*(0.15%)*	*(1.72%)*	*(2.47%)*	*(17.15%)*	
DoS (Hulk)	93	5745	33	43	86	
	(1.55%)	**(95.75%)**	*(0.55%)*	*(0.72%)*	*(1.43%)*	
DoS (Slowloris)	507	0	4668	143	682	
	(8.45%)	*(0%)*	**(77.8%)**	*(2.38%)*	*(11.37%)*	
FTP	643	1	127	4879	350	
	(10.72%)	*(0.02%)*	*(2.12%)*	**(81.32%)**	*(5.83%)*	
SSH	924	34	310	350	4382	
	(15.4%)	*(0.57%)*	*(5.17%)*	*(5.83%)*	(73.03%)	

Bold represents the detection accuracy for both normal and attach classes

Blue represents the detection accuracy of the excluded class(es)

Table 7 CICIDS2017 One-Shot Confusion Matrix (FTP not in Training)

	Predicted Class					
Correct	Normal	DoS (Hulk)	DoS (Slowloris)	FTP	SSH	Overall
Normal	5231	3	152	189	425	82.5%
	(87.18%)	*(0.05%)*	*(2.53%)*	*(3.15%)*	*(7.08%)*	
DoS (Hulk)	70	5755	48	15	112	
	(1.17%)	**(95.92%)**	*(0.8%)*	*(0.25%)*	*(1.87%)*	
DoS (Slowloris)	424	1	4433	485	657	
	(7.07%)	*(0.02%)*	**(73.88%)**	*(8.08%)*	*(10.95%)*	
FTP	518	1	659	4202	620	
	(8.63%)	*(0.02%)*	*(10.98%)*	(70.03%)	*(10.33%)*	
SSH	546	3	198	124	5129	
	(9.1%)	*(0.05%)*	*(3.3%)*	*(2.07%)*	**(85.48%)**	

Bold represents the detection accuracy for both normal and attach classes

Blue represents the detection accuracy of the excluded class(es)

fall in line with the literature. Furthermore, the evaluation of the model is not subject to the class-imbalance issue. Classes are equally represented in both training and testing batches.

Furthermore, on inspection of Tables 8 and 9, it is evident that using five labelled instances of the new attack class results in an increase in both the overall accuracy and the TPR together with a drop in the FNR. Using only 1 labelled instance demonstrates a comparably poorer performance owing to the instance selection randomness, which could result in either a good or a bad class representative. However, using 5 random labelled instances boosts performance, reinforcing the importance of having distinctive class representatives.

The remainder of the CICIDS2017 results are characterised by similar behaviour. The full evaluation tables are listed in Appendix A for transparency and reproducibility. The results are listed as follows. DoS (Hulk) results are presented in Tables 16 and 17. The TPR rises from 50.97% when using one pair to 72.82% when using 30 pairs. DoS (Slowloris) results are presented in Tables 18 and 19, where the TPR rises from 91.07% when using one pair to 95.18% when using 30 pairs.

The CMs of the KDD Cup'99 and NSL-KDD data sets One-Shot, excluding the DoS attack from training are presented in Tables 10 and 11, respectively; the overall accuracies are 76.67% and 77.99%. It is important to note however, that the False Negative rates for the new class (i.e. DoS) are 26.38% for the KDD Cup'99 and 9.87% for the NSL-KDD. Additional to the

Table 8 CICIDS2017 One-Shot Accuracy (SSH not in Training) Using Different *j* Votes

No Votes	Overall	New Class (SSH)		Normal	
(*j*)	Accuracy	TPR	FNR	TNR	FPR
1	72.72%	64.10%	16.43%	63.35%	36.65%
5	81.28%	73.03%	15.40%	78.52%	21.48%
10	82.56%	77.82%	13.40%	79.95%	20.05%
15	82.58%	78.43%	13.03%	79.92%	20.08%
20	82.49%	78.33%	13.18%	79.97%	20.03%
25	82.43%	78.30%	13.25%	79.78%	20.22%
30	82.49%	78.45%	13.13%	79.97%	20.03%

Table 9 CICIDS2017 One-Shot Accuracy (FTP not in Training) Using Different *j* Votes

No Votes	Overall	New Class (FTP)		Normal	
(*j*)	Accuracy	TPR	FNR	TNR	FPR
1	72.91%	59.65%	8.03%	72.83%	27.17%
5	82.5%	70.03%	8.63%	87.18%	12.82%
10	84.57%	72.8%	8.32%	87.70%	12.30%
15	85.47%	76.72%	8.12%	87.40%	12.60%
20	85.78%	77.58%	8.10%	87.23%	12.77%
25	85.86%	78.27%	8.10%	86.92%	13.08%
30	85.94%	78.48%	8.00%	86.73%	13.27%

Table 10 KDD One-Shot Confusion Matrix (DoS Not in Training)

	Predicted Class					
Correct	Normal	DoS	Probe	R2L	U2R	Overall
Normal	4562	243	522	579	94	76.67%
	(76.03%)	*(4.05%)*	*(8.7%)*	*(9.65%)*	*(1.57%)*	
DoS	1583	2417	1831	168	1	
	(26.38%)	(40.28%)	*(30.52%)*	*(2.8%)*	*(0.02%)*	
Probe	159	214	5367	242	18	
	(2.65%)	*(3.57%)*	**(89.45%)**	*(4.03%)*	*(0.3%)*	
R2L	56	275	10	5571	88	
	(0.93%)	*(4.58%)*	*(0.17%)*	**(92.85%)**	*(1.47%)*	
U2R	17	205	655	40	5083	
	(0.28%)	*(3.42%)*	*(10.92%)*	*(0.67%)*	**(84.72%)**	

Bold represents the detection accuracy for both normal and attach classes

Blue represents the detection accuracy of the excluded class(es)

Table 11 NSL-KDD One-Shot Confusion Matrix (DoS Not in Training)

	Predicted Class					
Correct	Normal	DoS	Probe	R2L	U2R	Overall
Normal	5593	61	136	122	88	77.99%
	(93.22%)	*(1.02%)*	*(2.27%)*	*(2.03%)*	*(1.47%)*	
DoS	592	4732	653	12	11	
	(9.87%)	(78.87%)	*(10.88%)*	*(0.2%)*	*(0.18%)*	
Probe	67	3305	2595	19	14	
	(1.12%)	*(55.08%)*	**(43.25%)**	*(0.32%)*	*(0.23%)*	
R2L	212	7	27	5692	62	
	(3.53%)	*(0.12%)*	*(0.45%)*	**(94.87%)**	*(1.03%)*	
U2R	486	6	31	693	4784	
	(8.1%)	*(0.1%)*	*(0.52%)*	*(11.55%)*	**(79.73%)**	

Bold represents the detection accuracy for both normal and attach classes

Blue represents the detection accuracy of the excluded class(es)

Table 12 KDD One-Shot Accuracy (DoS not in Training) Using Different j Votes

No Votes	Overall	New Class (DoS)		Normal	
(j)	Accuracy	TPR	FNR	TNR	FPR
1	66.89%	41.67%	22.50%	66.35%	33.65%
5	76.67%	40.28%	26.38%	76.03%	23.97%
10	77.57%	40.07%	27.25%	76.10%	23.90%
15	77.67%	39.9%	27.32%	76.02%	23.98%
20	77.68%	39.93%	27.38%	76.02%	23.98%
25	77.68%	39.87%	27.40%	76.07%	23.93%
30	77.68%	39.88%	27.40%	76.03%	23.97%

observations arising from the CICIDS2017 evaluation, these results highlight two further elements; (a) the Siamese Network did not find a high similarity between the new attack and the normal instances; (b) the new attack class TPR in the NSL-KDD results is significantly higher than KDD Cup'99 (78.87% compared to 40.28%), because the NSL-KDD is an enhanced version of the KDD Cup'99 (filtered and duplicate instances removed). Knowing that the new class is not used in the training phase and the similarity is only calculated from a few instances, a better representation of instances improves performance (i.e. NSL-KDD instances). Results confirm that new labelled instances need to be appropriate representatives (Tables 12 and 13).

Likewise, In consideration of completeness, the remaining NSL-KDD and the KDD Cup'99 results - which demonstrate similar performance - are listed as follows; excluding Probe results are listed in Tables 20, 21, 26 and 27; 24, 25, 30 and 31 present the results when excluding R2L; Finally, excluding U2R are in Tables 22, 23, 28 and 29.

7.2.2 Two excluded classes

A second experiment is conducted to further assess the performance of the model. Unlike the results in Section 7.2.1, three classes are used to train the network and two classes excluded from the training. The experiment is aimed at evaluating the robustness of the trained network to discriminate more than one class without the need for re-training, in the scenario when a few instances of the new class are available and until sufficient instances are gathered. The goal is to correctly classify and label new attacks not just to discriminate from benign/normal behaviour. When attacks are correctly classified, effective attack-specific countermeasures can be deployed.

Table 13 NSL-KDD One-Shot Accuracy (DoS not in Training) Using Different j Votes

No Votes	Overall	New Class (DoS)		Normal	
(j)	Accuracy	TPR	FNR	TNR	FPR
1	72.75%	67.35%	9.05%	84.87%	15.13%
5	77.99%	78.87%	9.87%	93.22%	6.78%
10	77.7%	84.62%	9.87%	93.35%	6.65%
15	79.05%	83.78%	9.87%	93.32%	6.68%
20	78.63%	85.25%	9.87%	93.37%	6.63%
25	79.49%	84.62%	9.87%	93.35%	6.65%
30	79.12%	85.37%	9.87%	93.35%	6.65%

Table 14 CICIDS2017 One-Shot Confusion Matrix (DoS (Hulk) & FTP Not in Training)

Correct	Predicted Class					Overall
	Normal	DoS (Hulk)	DoS (Slow-loris)	FTP	SSH	
Normal	4895 **(81.58%)**	545 *(9.08%)*	14 *(0.23%)*	451 *(7.52%)*	95 *(1.58%)*	75.47%
DoS (Hulk)	716 *(11.93%)*	4148 (69.13%)	1012 *(16.87%)*	87 *(1.45%)*	37 *(0.62%)*	
DoS (Slowloris)	424 *(7.07%)*	1120 *(18.67%)*	3869 **(64.48%)**	405 *(6.75%)*	182 *(3.03%)*	
FTP	480 *(8%)*	51 *(0.85%)*	19 *(0.32%)*	5185 (86.42%)	265 *(4.42%)*	
SSH	520 *(8.67%)*	26 *(0.43%)*	19 *(0.32%)*	890 *(14.83%)*	4545 **(75.75%)**	

Bold represents the detection accuracy for both normal and attach classes

Blue represents the detection accuracy of the excluded class(es)

Table 14 presents the confusion matrix when DoS (Hulk) and FTP are excluded from the training. The detection accuracy is 69.13% and 86.42% for the Dos (Hulk) and FTP classes respectively; the FNR of the new classes is 11.93% and 8%. It is important to note that the TPR increases and the FNR decreases as more instances are used from each of class as evident in Table 15 reaching an FNR of 9.6% and 7.78% and a TPR of 72.85% and 83.58% for the DoS (Hulk) and FTP attacks respectively.

8 Conclusion and future work

The paper presents an Intrusion Detection Siamese Network framework capable of classifying new cyber-attacks based on a limited number of labelled instances (One-Shot). The evaluation of the model was performed on three different data sets; CICIDS2017, KDD Cup'99, and the NSL-KDD, an enhancement of the KDD Cup'99.

The results of the evaluation reconfirm that particular consideration must be given on creating the training set, ensuring an equal number of training pairs for every class

Table 15 CICIDS2017 One-Shot Accuracy (DoS (Hulk) & FTP not in Training) Using Different *j* Votes

No Votes	Overall	New Class (DoS (Hulk))		New Class (FTP)		Normal	
(*j*)	Accuracy	TPR	FNR	TPR	FNR	TNR	FPR
1	65.17%	59.95%	13.42%	76.88%	9.6%	64.05%	35.95%
5	75.47%	69.13%	11.93%	86.42%	8%	81.58%	18.42%
10	77.43%	73.25%	10.4%	87.55%	7.68%	83.6%	16.4%
15	77.78%	72.03%	10.13%	87.67%	7.72%	83.68%	16.32%
20	78.05%	73.4%	9.6%	87.73%	7.67%	83.48%	16.52%
25	78.04%	72.65%	9.53%	87.85%	7.77%	83.7%	16.3%
30	78.04%	72.85%	9.6%	87.83%	7.78%	83.58%	16.42%

combination. The core requirement, in turn, presents a challenge of an exploding number of combinations between all instances. Thus, distinct pairs are chosen to create large batches in the region of 30,000 pairs to mitigate the growth. During evaluation, similarity comparison using a single point for each class resulted in noisy predictions due to randomness obviated through the selection of multiple (*j*) random instances from each class and aggregation using majority voting.

The results demonstrate the ability of the proposed architecture to classify cyber-attacks based on learning from similarity. Moreover, the results highlighted the need for representative instances for the new attack class. Furthermore, evidence is provided to confirm the ability of One-Shot learning methodologies to adapt to new cyber-attacks without retraining when only a few instances are available for a new attack. An overall accuracy of between 80% - 85% for the CICIDS2017 data-set was evaluated, demonstrating acceptable accuracy in detecting previously unseen attacks. Further and also important to the application is that the overall accuracy was achieved at a low FNR for the new attack classes. The overall accuracy reached above 75% for the KDD Cup'99 and NSL-KDD data sets. Further and also important to the application is that the overall accuracy was achieved at a low FNR for the new attack classes.

Appendix A: Full results tables

As aforementioned, in the One-Shot evaluation multiple experiments are conducted. In each experiment, a different class of the dataset is excluded. For transparency and due to the page limit, the full confusion matrices are presented in this section.

Tables 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, and 31 list the CMs and the One-Shot accuracy of the remaining classes.

Table 16 CICIDS2017 One-Shot Confusion Matrix (DoS(Hulk) Not in Training)

	Predicted Class					
Correct	Normal	DoS (Hulk)	DoS (Slowloris)	FTP	SSH	Overall
Normal	4314	1095	174	113	304	80.81%
	(71.9%)	*(18.25%)*	*(2.9%)*	*(1.88%)*	*(5.07%)*	
DoS (Hulk)	78	5708	60	58	96	
	(1.3%)	(95.13%)	*(1%)*	*(0.97%)*	*(1.6%)*	
DoS (Slowloris)	451	51	4767	111	620	
	(7.52%)	*(0.85%)*	**(79.45%)**	*(1.85%)*	*(10.33%)*	
FTP	624	171	138	4521	546	
	(10.4%)	*(2.85%)*	*(2.3%)*	**(75.35%)**	*(9.1%)*	
SSH	597	26	245	198	4934	
	(9.95%)	*(0.43%)*	*(4.08%)*	*(3.3%)*	**(82.23%)**	

Bold represents the detection accuracy for both normal and attach classes

Blue represents the detection accuracy of the excluded class(es)

Table 17 CICIDS2017 One-Shot Accuracy (DoS (Hulk) not in Training) Using Different *j* Votes

No Votes	Overall	New Class (Hulk)		Normal	
(*j*)	Accuracy	TPR	FNR	TNR	FPR
1	72.28%	91.07%	4.90%	58.05%	41.95%
5	80.81%	95.13%	1.30%	71.90%	28.10%
10	82.59%	95.22%	1.22%	75.58%	24.42%
15	82.54%	95.23%	1.20%	74.67%	25.33%
20	82.86%	95.2%	1.20%	76.02%	23.98%
25	82.76%	95.2%	1.15%	75.50%	24.50%
30	82.93%	95.18%	1.22%	76.15%	23.85%

Table 18 CICIDS2017 One-Shot Confusion Matrix (Dos(Slowloris) Not in Training)

	Predicted Class					
Correct	Normal	DoS (Hulk)	DoS (Slowloris)	FTP	SSH	Overall
Normal	5307	6	459	64	164	81.07%
	(88.45%)	*(0.1%)*	*(7.65%)*	*(1.07%)*	*(2.73%)*	
DoS (Hulk)	37	5794	65	53	51	
	(0.62%)	**(96.57%)**	*(1.08%)*	*(0.88%)*	*(0.85%)*	
DoS (Slowloris)	574	26	4024	582	794	
	(9.57%)	*(0.43%)*	(67.07%)	*(9.7%)*	*(13.23%)*	
FTP	482	1	598	4639	280	
	(8.03%)	*(0.02%)*	*(9.97%)*	**(77.32%)**	*(4.67%)*	
SSH	446	0	817	181	4556	
	(7.43%)	*(0%)*	*(13.62%)*	*(3.02%)*	**(75.93%)**	

Bold represents the detection accuracy for both normal and attach classes

Blue represents the detection accuracy of the excluded class(es)

Table 19 CICIDS2017 One-Shot Accuracy (DoS (Slowloris) not in Training) Using Different *j* Votes

No Votes	Overall	New Class (Slowloris)		Normal	
(*j*)	Accuracy	TPR	FNR	TNR	FPR
1	70.89%	50.97%	11.50%	72.65%	27.35%
5	81.07%	67.07%	9.57%	88.45%	11.55%
10	82.67%	71.38%	7.38%	89.48%	10.52%
15	82.85%	72.20%	7.18%	89.37%	10.63%
20	83.01%	72.77%	6.85%	89.67%	10.33%
25	82.98%	72.93%	6.58%	89.65%	10.35%
30	82.94%	72.82%	6.68%	89.70%	10.30%

Table 20 NSL-KDD One-Shot Confusion Matrix (Probe Not in Training)

	Predicted Class					
Correct	Normal	DoS	Probe	R2L	U2R	Overall
Normal	5389	89	195	245	82	75.31%
	(89.82%)	*(1.48%)*	*(3.25%)*	*(4.08%)*	*(1.37%)*	
DoS	37	5842	95	21	5	
	(0.62%)	**(97.37%)**	*(1.58%)*	*(0.35%)*	*(0.08%)*	
Probe	1697	2571	565	948	219	
	(28.28%)	*(42.85%)*	*(9.42%)*	*(15.8%)*	*(3.65%)*	
R2L	54	0	55	5800	91	
	(0.9%)	*(0%)*	*(0.92%)*	**(96.67%)**	*(1.52%)*	
U2R	263	0	21	720	4996	
	(4.38%)	*(0%)*	*(0.35%)*	*(12%)*	**(83.27%)**	

Bold represents the detection accuracy for both normal and attach classes

Blue represents the detection accuracy of the excluded class(es)

Table 21 NSL-KDD One-Shot Accuracy (Probe not in Training) Using Different *j* Votes

No Votes	Overall	New Class (Probe)		Normal	
(*j*)	Accuracy	TPR	FNR	TNR	FPR
1	70.62%	18.80%	24.78%	77.53%	22.47%
5	75.31%	9.42%	28.28%	89.82%	10.18%
10	75.2%	4.83%	28.82%	91.08%	8.92%
15	75.12%	4.05%	29.08%	91.18%	8.82%
20	75.11%	3.47%	29.20%	91.45%	8.55%
25	75%	3.02%	29.55%	91.35%	8.65%
30	74.94%	2.68%	29.68%	91.33%	8.67%

Table 22 NSL- KDD One-Shot Confusion Matrix (R2L Not in Training)

	Predicted Class					
Correct	Normal	DoS	Probe	R2L	U2R	Overall
Normal	5199	24	148	530	99	80.16%
	(86.65%)	*(0.4%)*	*(2.47%)*	*(8.83%)*	*(1.65%)*	
DoS	15	5799	36	26	124	
	(0.25%)	**(96.65%)**	*(0.6%)*	*(0.43%)*	*(2.07%)*	
Probe	90	242	5416	236	16	
	(1.5%)	*(4.03%)*	**(90.27%)**	*(3.93%)*	*(0.27%)*	
R2L	2526	1	142	2759	572	
	(42.1%)	*(0.02%)*	*(2.37%)*	*(45.98%)*	*(9.53%)*	
U2R	852	3	0	270	4875	
	(14.2%)	*(0.05%)*	*(0%)*	*(4.5%)*	**(81.25%)**	

Bold represents the detection accuracy for both normal and attach classes

Blue represents the detection accuracy of the excluded class(es)

Table 23 NSL-KDD One-Shot Accuracy (R2L not in Training) Using Different *j* Votes

No Votes	Overall	New Class (R2L)		Normal	
(*j*)	Accuracy	TPR	FNR	TNR	FPR
1	74.5%	46.05%	38.13%	74.73%	25.27%
5	80.16%	45.98%	42.10%	86.65%	13.35%
10	80.79%	46.82%	41.58%	88.07%	11.93%
15	81.09%	49.02%	39.88%	87.72%	12.28%
20	81%	48.62%	40.38%	87.90%	12.10%
25	80.95%	48.37%	40.63%	87.88%	12.12%
30	80.91%	48.2%	40.93%	87.93%	12.07%

Table 24 NSL-KDD One-Shot Confusion Matrix (U2R Not in Training)

	Predicted Class					
Correct	Normal	DoS	Probe	R2L	U2R	Overall
Normal	4530	127	76	237	1030	77.04%
	(75.5%)	*(2.12%)*	*(1.27%)*	*(3.95%)*	*(17.17%)*	
DoS	120	5771	49	16	44	
	(2%)	**(96.18%)**	*(0.82%)*	*(0.27%)*	*(0.73%)*	
Probe	43	304	5574	69	10	
	(0.72%)	*(5.07%)*	**(92.9%)**	*(1.15%)*	*(0.17%)*	
R2L	403	1	27	5238	331	
	(6.72%)	*(0.02%)*	*(0.45%)*	**(87.3%)**	*(5.52%)*	
U2R	2191	0	221	1589	1999	
	(36.52%)	*(0%)*	*(3.68%)*	*(26.48%)*	(33.32%)	

Bold represents the detection accuracy for both normal and attach classes

Blue represents the detection accuracy of the excluded class(es)

Table 25 NSL-KDD One-Shot Accuracy (U2R not in Training) Using Different *j* Votes

No Votes	Overall	New Class (U2R)		Normal	
(*j*)	Accuracy	TPR	FNR	TNR	FPR
1	72.42%	34.37%	35.55%	66.58%	33.42%
5	77.04%	33.32%	36.52%	75.50%	24.50%
10	77.08%	30.42%	36.95%	77.85%	22.15%
15	77.19%	30.2%	36.70%	78.22%	21.78%
20	77.12%	29.37%	36.67%	78.52%	21.48%
25	77.14%	28.85%	36.72%	78.87%	21.13%
30	77.12%	28.3%	37.10%	79.25%	20.75%

Table 26 KDD One-Shot Confusion Matrix (Probe Not in Training)

	Predicted Class					
Correct	Normal	DoS	Probe	R2L	U2R	Overall
Normal	4515	16	383	1016	70	72.23%
	(75.25%)	*(0.27%)*	*(6.38%)*	*(16.93%)*	*(1.17%)*	
DoS	18	5896	81	4	1	
	(0.3%)	**(98.27%)**	*(1.35%)*	*(0.07%)*	*(0.02%)*	
Probe	719	3707	612	941	21	
	(11.98%)	*(61.78%)*	(10.2%)	*(15.68%)*	*(0.35%)*	
R2L	26	0	16	5946	12	
	(0.43%)	*(0%)*	*(0.27%)*	**(99.1%)**	*(0.2%)*	
U2R	55	37	264	943	4701	
	(0.92%)	*(0.62%)*	*(4.4%)*	*(15.72%)*	**(78.35%)**	

Bold represents the detection accuracy for both normal and attach classes

Blue represents the detection accuracy of the excluded class(es)

Table 27 KDD One-Shot Accuracy (Probe not in Training) Using Different *j* Votes

No Votes	Overall	New Class (Probe)		Normal	
(*j*)	Accuracy	TPR	FNR	TNR	FPR
1	66.72%	15.72%	11.77%	65.72%	34.28%
5	72.23%	10.2%	11.98%	75.25%	24.75%
10	72.59%	5.9%	13.30%	78.65%	21.35%
15	72.35%	4.82%	13.08%	78.57%	21.43%
20	72.26%	3.58%	13.50%	79.20%	20.80%
25	72.17%	3.05%	13.55%	79.23%	20.77%
30	72.07%	2.17%	13.98%	79.62%	20.38%

Table 28 KDD One-Shot Confusion Matrix (R2L Not in Training)

	Predicted Class					
Correct	Normal	DoS	Probe	R2L	U2R	Overall
Normal	4288	1	400	730	581	74.2%
	(71.47%)	*(0.02%)*	*(6.67%)*	*(12.17%)*	*(9.68%)*	
DoS	10	5909	72	9	0	
	(0.17%)	**(98.48%)**	*(1.2%)*	*(0.15%)*	*(0%)*	
Probe	90	160	5338	165	247	
	(1.5%)	*(2.67%)*	**(88.97%)**	*(2.75%)*	*(4.12%)*	
R2L	1702	2	1344	2148	804	
	(28.37%)	*(0.03%)*	*(22.4%)*	(35.8%)	*(13.4%)*	
U2R	527	1	682	213	4577	
	(8.78%)	*(0.02%)*	*(11.37%)*	*(3.55%)*	**(76.28%)**	

Bold represents the detection accuracy for both normal and attach classes

Blue represents the detection accuracy of the excluded class(es)

Table 29 KDD One-Shot Accuracy (R2L not in Training) Using Different *j* Votes

No Votes	Overall	New Class (R2L)		Normal	
(*j*)	Accuracy	TPR	FNR	TNR	FPR
1	67.75%	38.48%	25.95%	59.65%	40.35%
5	74.2%	35.8%	28.37%	71.47%	28.53%
10	77.27%	42.22%	23.85%	74.38%	25.62%
15	78.34%	46.65%	22.05%	74.50%	25.50%
20	78.94%	49.18%	21.45%	74.62%	25.38%
25	79.44%	51.32%	20.72%	74.65%	25.35%
30	79.87%	53.35%	20.65%	74.55%	25.45%

Table 30 KDD One-Shot Confusion Matrix (U2R Not in Training)

	Predicted Class					
Correct	Normal	DoS	Probe	R2L	U2R	Overall
Normal	4146 **(69.1%)**	5 *(0.08%)*	440 *(7.33%)*	796 *(13.27%)*	613 *(10.22%)*	75.72%
DoS	7 *(0.12%)*	5921 **(98.68%)**	59 *(0.98%)*	6 *(0.1%)*	7 *(0.12%)*	
Probe	53 *(0.88%)*	384 *(6.4%)*	5449 **(90.82%)**	59 *(0.98%)*	55 *(0.92%)*	
R2L	35 *(0.58%)*	0 *(0%)*	13 *(0.22%)*	5849 **(97.48%)**	103 *(1.72%)*	
U2R	958 *(15.97%)*	1 *(0.02%)*	669 *(11.15%)*	3022 *(50.37%)*	1350 (22.5%)	

Bold represents the detection accuracy for both normal and attach classes

Blue represents the detection accuracy of the excluded class(es)

Table 31 KDD One-Shot Accuracy (U2R not in Training) Using Different *j* Votes

No Votes	Overall	New Class (U2R)		Normal	
(*j*)	Accuracy	TPR	FNR	TNR	FPR
1	70.69%	21.40%	17.28%	59.27%	40.73%
5	75.72%	22.5%	15.97%	69.10%	30.90%
10	76.26%	21.82%	17.17%	72.18%	27.82%
15	76.33%	21.83%	17.15%	72.52%	27.48%
20	76.31%	21.48%	17.52%	72.72%	27.28%
25	76.34%	21.45%	17.55%	72.77%	27.23%
30	76.33%	21.27%	17.73%	72.90%	27.10%

Author contributions Conceptualization: Hanan Hindy Methodology: Hanan Hindy, Christos Tachtatzis, Robert Atkinson, Ivan Andonovic Software: Hanan Hindy Validation: Christos Tachtatzis, Robert Atkinson, Xavier Bellekens Formal analysis: Ivan Andonovic, Craig Michie Investigation: Hanan Hindy Resources: David Brosset, Miroslav Bures Writing - Original Draft: Hanan Hindy Writing - Review & Editing: Christos Tachtatzis, Robert Atkinson, David Brosset, Miroslav Bures, Ivan Andonovic, Craig Michie, Xavier Bellekens Supervision: Xavier Bellekens Project administration: Xavier Bellekens

Funding Not applicable.

Data availability The 'CICIDS2017' dataset supporting the conclusions of this article is available in the Canadian Institute for Cybersecurity (CIC) repository, http://www.unb.ca/cic/datasets/ids-2017.html

Code availability The code will be available through a GitHub repository.

Declarations

Ethics approval Not applicable

Consent to participate Not applicable

Competing interests The authors declare that they have no competing interests.

References

Altae-Tran, H., Ramsundar, B., Pappu, A.S., & Pande, V. (2017). Low data drug discovery with One-Shot learning. *ACS Central Science*, *3*(4), 283–293. https://doi.org/10.1021/acscentsci.6b00367.

Andresini, G., Appice, A., & Malerba, D. (2021). Autoencoder-based deep metric learning for network intrusion detection. *Information Sciences*, *569*, 706–727. https://doi.org/10.1016/j.ins.2021.05.016.

Bedi, P., Gupta, N., & Jindal, V. (2020). Siam-IDS: Handling class imbalance problem in intrusion detection systems using siamese neural network. *Procedia Computer Science*, *171*, 780–789.

Bedi, P., Gupta, N., & Jindal, V. (2021). I-SiamIDS: an improved Siam-IDS for handling class imbalance in network-based intrusion detection systems. *Applied Intelligence*, *51*(2), 1133–1151.

Benajiba, Y., Sun, J., Zhang, Y., Jiang, L., Weng, Z., & Biran, O. (2019). Siamese Networks for semantic pattern similarity. In *2019 IEEE 13th international conference on semantic computing (ICSC)* (pp. 191–194). IEEE. https://doi.org/10.1109/ICOSC.2019.8665512

Buczak, A.L., & Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, *18*(2), 1153–1176. https://doi.org/10.1109/COMST.2015.2494502.

Bruce, J., Sünderhauf, N., Mirowski, P., Hadsell, R., & Milford, M. (2017). One-Shot reinforcement learning for robot navigation with interactive replay. arXiv:1711.10137.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., & Shah, R. (1994). Signature verification using a "Siamese" time delay neural network. In *Advances in neural information processing systems* (pp. 737–744). http://papers.nips.cc/paper/769-signature-verification-using-a.pdf

Chung, D., Tahboub, K., & Delp, E.J. (2017). A two stream Siamese convolutional neural network for person re-identification. In *Proceedings of the IEEE international conference on computer vision* (pp. 1983–1991). https://doi.org/10.1109/ICCV.2017.218

Chung, Y.-A., & Weng, W.-H. (2017). Learning deep representations of medical images using Siamese CNNs with application to content-based image retrieval. arXiv:1711.08490.

Chopra, S., Hadsell, R., & LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. In *CVPR* (1) (pp. 539–546). https://doi.org/10.1109/CVPR.2005.202.

Das, A., Yenala, H., Chinnakotla, M., & Shrivastava, M. (2016). Together we stand: Siamese networks for similar question retrieval. In *Proceedings of the 54th annual meeting of the association for computational linguistics (Volume 1: Long Papers)*, (Vol. 1 pp. 378–387). https://www.aclweb.org/anthology/P16-1036.pdf

Duan, Y., Andrychowicz, M., Stadie, B., Ho, O.J., Schneider, J., Sutskever, I., Abbeel, P., & Zaremba, W. (2017). One-shot imitation learning. In *Advances in neural information processing systems* (pp. 1087–1098). http://papers.nips.cc/paper/6709-one-shot-imitation-learning.pdf

Fei-Fei, L., Fergus, R., & Perona, P. (2006). One-Shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(4), 594–611. https://doi.org/10.1109/TPAMI.2006.79.

Canadian Institute for Cybersecurity. (2009). NSL-KDD dataset. http://www.unb.ca/cic/datasets/nsl.html.

Garcia, V., & Bruna, J. (2017). Few-shot learning with graph neural networks. arXiv:1711.04043.

Hadsell, R., Chopra, S., & LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, (Vol. 2 pp. 1735–1742). IEEE. https://doi.org/10.1109/CVPR.2006.100

Hettich, S., & Bay, S.D. (1999). The UCI KDD Archive. http://kdd.ics.uci.edu. Accessed 15 June 2018.

Hettich, S., & Bay, S. D. (1999). KDDCup 1999 Data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. Accessed 07 Dec 2018.

Hindy, H., Brosset, D., Bayne, E., Seeam, A.K., Tachtatzis, C., Atkinson, R., & Bellekens, X. (2020). A taxonomy of network threats and the effect of current datasets on intrusion detection systems. *IEEE Access*, *8*, 104650–104675. https://doi.org/10.1109/ACCESS.2020.3000179.

Hossain, M.D., Ochiai, H., Doudou, F., & Kadobayashi, Y. (2020). Ssh and ftp brute-force attacks detection in computer networks: Lstm and machine learning approaches. In *2020 5th international conference on computer and communication systems (ICCCS)* (pp. 491–497). https://doi.org/10.1109/ICCCS49078.2020.9118459

Illy, P., Kaddoum, G., Miranda Moreira, C., Kaur, K., & Garg, S. (2019). Securing fog-to-things environment using intrusion detection system based on ensemble learning. In *2019 IEEE wireless communications and networking conference (WCNC)* (pp. 1–7). https://doi.org/10.1109/WCNC.2019.8885534

Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent Data Analysis*, *6*(5), 429–449. https://doi.org/10.3233/IDA-2002-6504.

Jain, S. (2017). NanoNets: How to use deep learning when you have limited data. https://medium.com/nanonets/nanonets-how-to-use-deep-learning-when-you-have-limited-data-f68c0b512cab.

Johnson, J.M., & Khoshgoftaar, T.M. (2019). Survey on deep learning with class imbalance. *Journal of Big Data*, *6*(1), 27. https://doi.org/10.1186/s40537-019-0192-5.

Koch, G., Zemel, R., & Salakhutdinov, R. (2015). Siamese neural networks for One-Shot image recognition. In *ICML deep learning workshop*, Vol. 2. https://www.cs.cmu.edu/rsalakhu/papers/oneshot1.pdf

Li, B., Springer, J., Bebis, G., & Gunes, M.H. (2013). A survey of network flow applications. *Journal of Network and Computer Applications*, *36*(2), 567–581. https://doi.org/10.1016/j.jnca.2012.12.020.

Li, Y., Xu, Y., Liu, Z., Hou, H., Zheng, Y., Xin, Y., Zhao, Y., & Cui, L. (2020). Robust detection for network intrusion of industrial iot based on multi-cnn fusion. *Measurement*, *154*, 107450. https://doi.org/10.1016/j.measurement.2019.107450.

Martin, K., Wiratunga, N., Massie, S., & Clos, J. (2018). Informed pair selection for self-paced metric learning in siamese neural networks. In M. Bramer M. Petridis (Eds.) *Artificial Intelligence XXXV* (pp. 34–49). Springer. https://doi.org/10.1007/978-3-030-04191-5_3

Moustakidis, S., & Karlsson, P. (2020). A novel feature extraction methodology using siamese convolutional neural networks for intrusion detection. *Cybersecurity*, *3*(1), 1–13.

Nguyen, L.D., Lin, D., Lin, Z., & Cao, J. (2018). Deep CNNs for microscopic image classification by exploiting transfer learning and feature concatenation. In *2018 IEEE international symposium on circuits and systems (ISCAS)* (pp. 1–5). IEEE. https://doi.org/10.1109/ISCAS.2018.8351550

Pan, S.J., Yang, Q., & et al. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22* (10), 1345–1359. https://doi.org/10.1109/TKDE.2009.191.

Pang, S., Qiao, S., Song, T., Zhao, J., & Zheng, P. (2019). An improved convolutional network architecture based on residual modeling for person re-identification in edge computing. *IEEE Access*, *7*, 106748–106759. https://doi.org/10.1109/ACCESS.2019.2933364.

Roh, Y., Heo, G., & Whang, S.E. (2019). A survey on data collection for machine learning: A Big Data-AI integration perspective. *IEEE Transactions on Knowledge and Data Engineering*, *33*(4), 1328–1347. https://doi.org/10.1109/TKDE.2019.2946162.

Singla, A., Bertino, E., & Verma, D. (2019). Overcoming the lack of labeled data: Training intrusion detection models using transfer learning. In *IEEE international conference on smart computing (SMARTCOMP)* (pp. 69–74). IEEE. https://doi.org/10.1109/SMARTCOMP.2019.00031

Sun, Q., Liu, Y., Chua, T.-S., & Schiele, B. (2019). Meta-transfer learning for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 403–412). https://openaccess.thecvf.com/content_CVPR_2019/papers/Sun_Meta-Transfer_Learning_for_Few-Shot_Learning_CVPR_2019_paper.pdf

Snell, J., Swersky, K., & Zemel, R. (2017). Prototypical networks for few-shot learning. In *Advances in neural information processing systems* (pp. 4077–4087). http://papers.nips.cc/paper/6996-prototypical-networks-for-few-shot-learning.pdf

Shaham, U., & Lederman, R.R. (2018). Learning by coincidence: Siamese networks and common variable learning. *Pattern Recognition*, *74*, 52–63. https://doi.org/10.1016/j.patcog.2017.09.015.

Sharafaldin, I., Lashkari, A.H., & Ghorbani, A.A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *ICISSP* (pp. 108–116). https://doi.org/10.5220/0006639801080116

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A.A. (2009). A detailed analysis of the KDD CUP 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications* (pp. 1–6). IEEE. https://doi.org/10.1109/CISDA.2009.5356528

Torrey, L., & Shavlik, J. (2010). Transfer learning. In *Handbook of research on machine learning applications and trends: Algorithms, methods, and techniques* (pp. 242–264). IGI Global. https://doi.org/10.4018/978-1-60566-766-9.ch011

Tolosana, R., Vera-Rodriguez, R., Fierrez, J., & Ortega-Garcia, J. (2018). Exploring recurrent neural networks for on-line handwritten signature biometrics. *IEEE Access*, *6*, 5128–5138. https://doi.org/10.1109/ACCESS.2018.2793966.

Varior, R.R., Haloi, M., & Wang, G. (2016). Gated on Computer Vision and Pattern Recognitiontecture for human re-identification. In *European conference on computer vision* (pp. 791–808). Springer. https://doi.org/10.1007/978-3-319-46484-8_48

Vinyals, O., Blundell, C., Lillicrap, T., & Wierstra, D. (2016). Matching networks for One Shot learning. In *Advances in neural information processing systems* (pp. 3630–3638). http://papers.nips.cc/paper/6385-matching-networks-for-one-shot-learning.pdf

Vinayakumar, R., Alazab, M., Soman, K.P., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep learning approach for intelligent intrusion detection system. *IEEE Access*, *7*, 41525–41550. https://doi.org/10.1109/ACCESS.2019.2895334.

Wang, Q., Zhao, X., Huang, J., Feng, Y., Liu, Z., Su, J., Luo, Z., & Cheng, G. (2017). Addressing complexities of machine learning in big data: Principles trends and challenges from systematical perspectives. https://doi.org/10.20944/preprints201710.0076.v2.

Wang, L., Li, Y., & Wang, S. (2018). Feature learning for One-Shot face recognition. In *2018 25th IEEE international conference on image processing (ICIP)* (pp. 2386–2390). IEEE. https://doi.org/10.1109/ICIP.2018.8451464

Yao, Y., Wu, X., Zuo, W., & Zhang, D. (2018). Learning Siamese network with top-down modulation for visual tracking. In *International conference on intelligent science and big data engineering* (pp. 378–388). Springer. https://doi.org/10.1007/978-3-030-02698-1_33

Zhang, Z., & Zhao, H. (2018). One-Shot learning for question-answering in Gaokao history challenge. In *Proceedings of the 27th international conference on computational linguistics* (pp. 449–461). arXiv:1806.09105

Authors and Affiliations

Hanan Hindy[1,2] · **Christos Tachtatzis**[3] · **Robert Atkinson**[3] · **David Brosset**[4] · **Miroslav Bures**[5] · **Ivan Andonovic**[3] · **Craig Michie**[3] · **Xavier Bellekens**[6]

Christos Tachtatzis
christos.tachtatzis@strath.ac.uk

Robert Atkinson
robert.atkinson@strath.ac.uk

David Brosset
david.brosset@ecole-navale.fr

Miroslav Bures
miroslav.bures@fel.cvut.cz

Ivan Andonovic
i.andonovic@strath.ac.uk

Craig Michie
c.michie@strath.ac.uk

1 Division of Cybersecurity, Abertay University, Dundee DD1 1HG, UK

2 Computer Science Department, Faculty of Computer and Information Sciences, Ain Shams University, 11711 Cairo, Egypt

3 Department of Electronic and Electrical Engineering, University of Strathclyde, Glasgow G1 1XW, UK

4 Naval Academy Research Institute, Arts et Métiers Institute of Technology, BCRM Brest, École Navale, CC 600, CEDEX 9, 29240 Brest, France

5 Department of Computer Science, Faculty of Electrical Engineering, Czech Technical University in Prague, Karlovo namesti 13, 121 35 Praha 2, Czechia

6 Lupovis Limited, Glasgow, UK