Multi-task Learning for Intrusion Detection and Analysis of Computer Network Traffic

Reem Aljoufi[1,*] and *Aboubaker* Lasebae[1,**]

[1]Department of Computer Science, Middlesex University, London, UK

Abstract. Accurate identification of malicious computer network traffic is a challenging task for a number of reasons. This is especially highlighted when a new type of attack is launched because the amount of available data that belongs to this attack can be scarce. Having small amounts of such data makes understanding the behaviour of traffic and building models to accurately discover it more difficult. In this paper we present a novel classification method based on multi-task learning for the accurate identification of malicious network traffic even when little amount of training data is available. We show the effectiveness of our method by carrying out several experiments and comparisons with existing methods using open source data. Our results show that our method outperforms those methods especially when training data is scarce. Particularly, it achieves accuracy values of 98.51% and 99.76% on two computer network traffic dataset settings, whereas a start-of-the-art algorithm achieves accuracy values of 93.56% and 96.25% on the same settings.

1 Introduction

There are several purposes of Cyber-attacks. These usually include the access, change or destruction of sensitive information. Such attacks can also be launched to extort money from people or interrupt businesses. Hence, attacks on computer networks can be highly costly and the number of approaches that attempt to identify malicious network traffic by performing traffic analysis is increasing as time goes on. This is because the number of cyber-attacks keeps increasing. This means there is a need for better techniques to be invented to protect computer devices and networks [1].

Because it is possible to capture computer network traffic and transform it into a format suitable for machine learning, there exists some machine learning techniques for tackling this problem. This paper introduces a new machine learning method for the accurate detection of malicious traffic when existing training data is scarce. The method is a multi-task learning technique where datasets belonging to different network traffic types and attacks (i.e. tasks) are used together in the learning process instead of learning each task separately. The remainder of this paper is organised as follows: section 2 introduces the concept of multi-task learning and section 3 provides an overview of network traffic data, how it is captured, pre-processed and so on. Section 4 is focused on clearly listing the contributions of this work, and this is followed by a review of the main existing approaches in section 5. Sections 2 and 7 provide details of the proposed classification method and its evaluation respectively. The paper then ends with conclusions and future work in section 8.

*e-mail: ra1008@live.mdx.ac.uk
**e-mail: a.lasebae@mdx.ac.uk

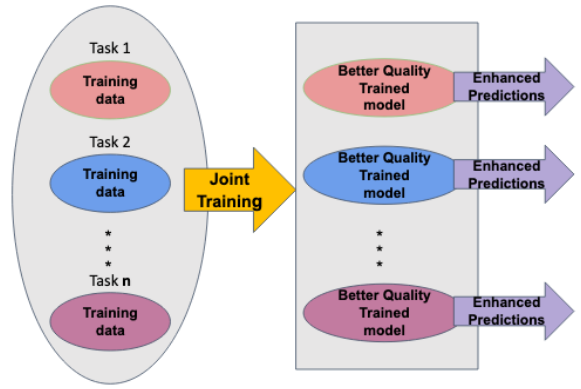
2 Overview of Multi-task Learning

According to Caruana [2], multi-task learning, or MTL for short, is a method for improving model generalization by means of learning related tasks together. The core idea is to use a form of representation that is shared by these related tasks so that the joint learning can occur. This has multiple benefits such as *inductive transfer*, which is the effect that happens internally when tasks learned together can benefit from each other and the ultimately resulting models are of better quality than when the tasks are learned individually [3] (which is known as single-task learning, or STL for short). A visual illustration of MTL and STL is shown in figure 1. Hence, MTL is by definition a branch of transfer learning [4], or TL for short, where some form of learning is performed on tasks where there are large amounts of data (such tasks are known as the *source tasks*) and this learning is exploited to enhance learning for tasks where data is scarce (such tasks are known as the *target tasks*).

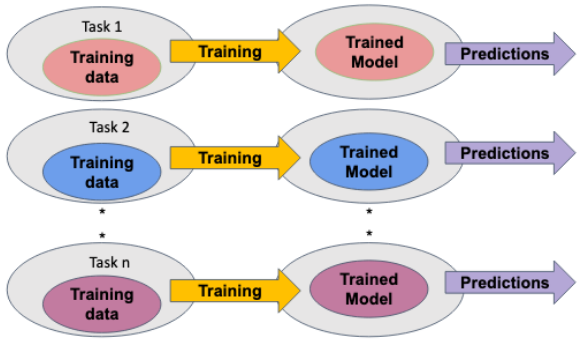
However, it is important to highlight that in MTL the tasks are learned together at the same time instead of learning from some tasks in one stage and then transferring the knowledge learned to other tasks in another stage. A good introduction to MTL with useful examples can be found at [5].

3 Network Traffic Data

Usually network traffic data is captured in a binary format known as the PCAP format (Packet CAPture). It is possible to extract numeric features from this data and transform it into tabular format suitable for machine learning



(a) Multi-task Learning (MTL)



(b) Single-task Learning (STL)

Figure 1: Visual Illustration of Single and Multi-task Learning

and further analysis. An existing open-source tool is ISCXFlowMeter [6]. After the data is ready for machine learning, various classification techniques can be used to classify traffic and identify its type (i.e. whether it is malicious or benign. And in case of malicious traffic, it is desirable to identify the attack type).

In our current time, there are several well known network attacks of which plenty of data is available. On the other hand, there are several attacks that are not well studied because of the scarcity of their captured data. An example source of such data is available at [7]. The available data is a reflection of a real-world situation where an Intrusion Detection Systems (IDSs), or Intrusion Prevention Systems (IPSs), can be used to spot network attacks in real-time. An intrusion detection system, or IDS for short, is a computer program that is used to monitor computer network traffic, attempt to identify malicious, or suspicious, activities and generate alerts when such activity is found [8]. Usually, malicious traffic is much less than normal (safe or benign) traffic, therefore, it is seen as an anomaly.

The data shows a significant difference between the number of available instances that belong to different traffic types (in total it contains 2830742 instances). For example, the number of instances available for benign traffic is as high as 2273096, the number of instances for the DDoS attack is as high as 128027. On the other hand, the number of instances available for an Infiltration attack is as low as 36 and the number of available instances for a Heartbleed attack is as low as 11. The full distribution of instance classes of this dataset is shown in figure 2. This is a typical situation as datasets of this type are usually highly imbalanced.

Traffic type	No of instances
BENIGN	2273096
DoS Hulk	231073
PortScan	158930
DDoS	128027
DoS GoldenEye	10293
FTP-Patator	7938
SSH-Patator	5897
DoS slowloris	5796
DoS Slowhttptest	5499
Bot	1966
Web Attack Brute Force	1507
Web Attack XSS	652
Infiltration	36
Web Attack Sql Injection	21
Heartbleed	11

Figure 2: Class Distribution of an Example IDS Dataset

In a scenario like this (where plenty of data is available for some tasks and very small amounts of data is available for some other tasks related to these tasks), usually poor predictive models are obtained when learning the tasks with small amounts of data. Please notice that in this paper, we refer to each network traffic category as a task (i.e. Benign is a task, DDoS attack is a task and so on). Because of the low performance of such models, some improvement is required. This paper introduces a multi-task learning approach to achieve such improvement. The idea is to learn tasks with large amounts of data together with tasks with small amounts of data in order to obtain better models for the latter tasks. More on multi-task learning in section 2

4 Contributions of This Work

The proposed MTL method in this work is inspired by a recent work that can be found at [3]. Although it was applied in the field of drug discovery, it is possible to adopt,

enhance and apply it in other fields such as the field of automatic analysis of computer network traffic. It is important to highlight the differences between the method introduced in this paper and the above-mentioned method. Our method differs in the following ways:

- The existing method measures the similarity between tasks using amino acid sequences whereas we use instance similarity
- The existing method is used in a regression context (i.e. to predict a real number) whereas our method is classification (i.e. to predict a category or class) and therefore the problem setting and evaluation metrics/procedures are different
- We have identified a weakness in the existing method. Because it attempts to address a regression problem, it is not straightforward to distinguish between instances from different tasks when the datasets are merged. Therefore, it adds a task ID column as a new feature (each task is given a task number) and this can mislead the classifier. This is because when an integer number (such as 1) is assigned to task *A* and another number (such as 2) to task *B*, mathematically 2 is greater than 1, but does this mean that task *B* is larger than task *A*? not necessarily .. a classifier does not know that and it will use the task ID as a number and not as a category (which introduces order between tasks and that does not really exist)
- This is a weakness in the existing technique that our work does not suffer from because the class label is used to distinguish between instances from different tasks

Based on the above, the contributions of our work can be summarised as follows:

1. Develop a multi-task learning (MTL) approach for the accurate identification of malicious network traffic (even in case of scarce data)
2. Instead of learning the similarity between tasks, we quantify it by measuring it using a well-known similarity metric (namely, the cosine similarity [9])
3. Identify a weakness in a recent existing method and demonstrate how our method avoids it
4. Experimentally show that our method outperforms existing widely used classifiers in different situations.

5 Existing Work

The use of MTL dates back to the late 1990s where they were used in an artificial neural network configuration as detailed in [2]. Since then, researchers have paid attention to its usefulness and employed it to enhance learning in several machine learning contexts. One of the most recent approaches is the work in [3] where MTL was used to obtain better predictive models for drug discovery. The problem domain was focused on predicting the binding of small molecules (i.e. potential drugs) on human body proteins (each protein is considered a task). This binding is quantified using a real number, and therefore the problem setting was regression. As for task similarity, although there is existing work that focuses on learning that similarity [10, 11], the reported method exploits the amino-acid sequence of each protein and uses it to compute the similarity between proteins. This similarity is then assumed to be the task similarity and the approach combines datasets from various proteins into one large dataset and adds the similarity values as new features. Although the method exhibits significantly improved results over single task learning, it suffers from the weaknesses (and differs from our work in the ways) listed in Section 4.

Over the last few years, MTL was an active research area in deep learning (i.e. deep neural networks). An example configuration is known as *hard parameter sharing* where hidden layers are shared between multiple tasks while task-specific output layers are kept separate (this is the setting designed by Caruana [2]). This setting was shown to reduce overfitting as explained in [12]. Overfitting is the problem that occurs when a trained model performs well on the training data and poorly on unseen data. Another configuration is known as *soft parameter sharing* where each task keeps its own model (and parameters) and regularisation is used to ensure the parameters of separate models are similar [13, 14]. A recent survey of MTL in various deep learning configurations can be found in [15].

A recent approach that uses MTL for Network Traffic Classification is reported in [16]. It is a deep learning technique that is developed for network bandwidth and duration prediction tasks. Another recent deep learning MTL technique for the classification of network traffic can be found in [17]. The two methods are based on deep neural networks and they suffer from the usual deep learning drawbacks such as the need for large amounts of training data and the need for extensive compute power and resources. For example, in [17] a significant amount of data pre-processing is required as the data needs to be picturized before feeding it into the deep learning architecture (i.e. data must be transformed into a picture-like 2D representation to be suitable for convolutional neural networks). Also, in [16] the data is seen as a time-series and a 1D CNN architecture was used. The authors have selected only three features to use as inputs and no reason was provided for this. This is not the case with our approach as we simply concatenate datasets from different network traffic types and add the similarity values to obtain extra features (this is explained in detail in section 2). It is worth mentioning here that deep learning architectures for multi-task and transfer learning in the area of computer network traffic analysis are not difficult to exploit as they have been shown to contain security vulnerabilities [18].

AS stated in Section 1, MTL can be considered as a way of achieving transfer learning (TL). TL was used in the field of cyber-security for the identification of malicious network traffic. An example is the similarity-based instance transfer (SBIT) work in [19] and, its extension, the class-balanced similarity-based instance transfer (CB-SBIT) [20]. The reported approaches use a brute-force like method to copy instances from tasks with plenty of data

to similar tasks with small amounts of data. The copying process is performed only for similar instances. In other words, the similarity of instances in a small dataset and a large dataset is measured (assuming each dataset belongs to a different type of network attack) and whenever highly similar instances (to the instances in the small dataset) are found in the large dataset are found, they are copied to the small dataset so that its size increases (the copied instances are assigned the class of the dataset they are copied to). This is performed and then learning is performed using the newly increased in size dataset. Experiments reported by the authors show improvement in model predictions. Our work differs from this approach in more than one aspect. In more detail, our work employs MTL so that no instances are copied from large datasets to smaller datasets. This can give rise to a problem especially when no form of regularisation or weighting is used to control how much the copied instances should contribute to the target dataset (currently they are given the same weight as the original instances and this is not reliable). Our approach uses MTL to learn from the datasets as they are without changing them and this is advantageous. A survey of existing methods can be found in [21].

6 The Proposed Method

As stated previously, our MTL method in this work is a novel extension of the recent work in [3]. Our idea is to consider each network traffic type as a task and then learn a classifier in a multi-task learning setting (each task is represented by a corresponding dataset). As part of our algorithm, we compute similarity between all tasks (compute average instance similarity between instances of all datasets) as shown in algorithm 1.

Input : Two Datasets ($ds1$ and $ds2$)
Output: Vector of average similarity values

- Initialize empty similarity vector *vsim*;
for $i \in ds1$ **do**
  - Initialize temporary empty similarity vector *vsim_temp*;
  for $j \in ds2$ **do**
    - Compute s = cosine_similarity(i,j);
    - add s to *vsim_temp*;
  end
  - Compute avg_sim = *avg*(*vsim_temp*);
  - add *avg_sim* to *vsim*;
end
- Return *vsim*;

Algorithm 1: How to Compute Similarity Values between Instances

The result of the previous algorithm is a vector of the same length as $ds1$ that contains the similarity values of the instances of $ds1$ to instances in $ds2$. Using this algorithm, a vector of similarity values for each pair of datasets (i.e. tasks) can be easily computed.

After the pairwise similarity values of all tasks (i.e. using their corresponding datasets) have been computed, we concatenate these datasets and add similarity values as new features as shown in algorithm 2.

Input : n related datasets (each contains data from one network traffic type)
Output: A Trained Model that can be used for Future Predictions

1- Merge the n datasets along the columns (i.e. stack them);
2- Add n extra variables to the dataset resulting from step 1:
SimToDS_1, SimToDS_2, ..., SimToDS_n;
3- Populate the similarity values using the results from algorithm 1;
4- Train a model using the newly created dataset

Algorithm 2: How to Concatenate Datasets and add Similarity Values as new Features

What our method does is to join all datasets together into one large dataset and then add similarity values as new features to the resulting large dataset. After that a model can be trained using the large dataset and used for future predictions.

The dataset resulting after applying algorithm 2 should look like the example shown in figure 3. Notice the original features are on the left and the similarity values are on the right (this example has four new columns because this setting involves data from four tasks).

Task	f1	f2	.....	SimiToTask1	SimiToTask2	SimiToTask3	SimiToTask4
Task1				1	0.35	0.44	0.12
Task2				0.35	1	0.53	0.39
Task3				0.44	0.53	1	0.61
Task4				0.12	0.39	0.61	1

Figure 3: Contents of Dataset resulting after the Proposed MTL Method

7 Discussion of Experimental Results

In this section we are going to present our experiments, show and discuss their results. To demonstrate the effectiveness of our method, we have used the IDS dataset explained in section 3. We have randomly selected four tasks and some instances as shown in table 1.

Network Traffic Type (i.e. Task)	Number of Instances
Benign Traffic	41176
DDoS Traffic	58168
Bot Traffic	45
Portscan Traffic	57

Table 1: Data used for MTL Experimental Evaluation

Before starting our MTL experiments, we have run initial evaluation to select the best base-line classifier that can be trained on the dataset resulting after applying algorithm 2. Our results showed that RandomForest is the best

performer (and this is consistent with the work in [20]). Hence, we are going to use it as our base classifier for our MTL work.

An example dataset resulting after applying algorithm 2 is shown in figure 4. Notice the original features are on the left and the *label* column is the class column (each class is a task). The similarity values are on the right (this example has four new columns because this dataset contains four unique class labels as shown in table 1).

Fwd Packets/s	Bwd Packets/s	label	Sim2DDoS	Sim2Benign	Sim2PortScan	Sim2Bot
0.639	0.724	BENIGN	0.750253	1.000000	0.999135	0.552243
618.000	1236.090	DDoS	1.000000	0.825122	0.099968	0.489020
216.000	288.000	BENIGN	0.245391	1.000000	0.000055	0.571679
558.000	1116.150	DDoS	1.000000	0.694103	0.000006	0.583908
0.303	0.454	BENIGN	0.562439	1.000000	0.997833	0.564032
...	...	...	...	...	...	...
8.380	10.200	BENIGN	0.649460	1.000000	0.970705	0.598852
9.880	19.800	DDoS	1.000000	0.226170	0.465622	0.193312
8.340	19.500	DDoS	1.000000	0.202440	0.578942	0.370646
15.900	20.200	BENIGN	0.775083	1.000000	0.864287	0.451461
11.600	19.400	DDoS	1.000000	0.289061	0.345324	0.258006

Figure 4: An Example Dataset resulting after the Proposed MTL Method

7.1 Comparison with RandomForest

In order to demonstrate the effectiveness of our method, we are going to compare its performance against the best performing classical classifier (which is RandomForest as stated above). As we have four separate datasets (one for each of the tasks shown in table 1), we have created several datasets by using various pairwise combinations of datasets (each two different tasks against each other). After obtaining these dataset combinations we have run a 10-fold cross-validation procedure using RandomForest. This procedure generates several train/test splits, trains a RandomForest model on the train split and uses it to predict the test split. After this we have filtered the predictions by label (i.e. task) and computed the average accuracy for each label (i.e. task). Figure **??** shows the results for each pair of tasks (read row vs column).

	Benign	DDoS	Bot	Portscan
Benign	1	(0.9997, 0.9997)	(0.9999, 0.8000)	(0.9999, 0.7543)
DDoS	(0.9996,0.9996)	1	(1.0, 0.8666)	(0.9999, 0.5263)
Bot	(0.7999, 0.9999)	(0.8888, 1.0)	1	(0.9555, 1.0)
Portscan	(0.7543, 0.9999)	(0.5087, 0.9999)	(1.0, 0.9555)	1

Figure 5: Results of RandomForest on Pairwise Task Combinations

As for the MTL evaluation procedure, we use the same datasets that have resulted after the pairwise dataset combination procedure but now we add the similarity values to them as extra features as explained in section 2. Our results are shown in figure 6.

	Benign	DDoS	Bot	Portscan
Benign	1	(1.0, 1.0)	(1.0, 1.0)	(1.0, 0.9824)
DDoS	(1.0, 1.0)	1	(1.0, 0.9777)	(1.0, 1.0)
Bot	(1.0, 1.0)	(0.9777, 1.0)	1	(0.9777, 1.0)
Portscan	(0.9824, 1.0)	(1.0, 1.0)	(1.0, 0.9777)	1

Figure 6: Results of our MTL Procedure on the same Pairwise Task Combinations

Notice the significant improvement in performance (compare corresponding cells). Please observe that it does not really matter which base classifier is used (the novel aspect or our work is adding the similarity values to improve the classification accuracy). You can focus on labels where the dataset size is small (i.e. notice the significant improvement when learning is performed for Bot and Portscan tasks).

7.2 Comparison with CB-SBIT

The implementation of the CB-SBIT algorithm [20] is freely available. Therefore, we have downloaded it and used it to perform evaluation and comparison with our method. In order to run the experiments, we have prepared the data so that it is compatible with how CB-SBIT works. We selected three attack types (Bot, Portscan and DDoS) and we added BENIGN data to them (making sure the resulting datasets are class balanced and there is no overlap in BENIGN instances). It is worth mentioning here that we are going to use accuracy as our performance evaluation metric because datasets are class balanced [22]. Notice that these datasets do not contain the similarity columns we add as part of our MTL procedure. Then we used the available CB-SBIT code as is in the following way:

1. Take the Bot data and randomly split it into two equally sized datasets (one for use as a Target data in the CB-SBIT algorithm and the other for use as test data)

2. We used the Portscan and DDoS datasets as source datasets in CB-SBIT

3. We ran CB-SBIT with its default parameters

The original Bot Target dataset size was 45 instances (21 Benign and 24 Bot) and the new Bot dataset size (after instance transfer by the CB-SBIT algorithm) was 118489 instances (59243 Benign and 59246 Bot). As the new Bot dataset is much larger than the original dataset one would expect a better model as we now have plenty of data. However, the results are not as expected. Accuracy before using CB-SBIT (i.e training the RF classifier on the original training Bot dataset) is 94.78% and accuracy after transfer (i.e training the RF classifier on the new dataset that contains the original instances of the small training Bot dataset and the instances transferred to it from the source datasets using the CB-SBIT algorithm) is 93.56%. This

is a surprising result as the CB-SBIT is performing what is known as *negative transfer* (i.e. results are worse after using the CB-SBIT algorithm).

We have repeated the previous steps but now with using the Portscan data as Target and Test, and adding the Bot dataset to the source datasets.

1. Take the Portscan data and randomly split it into two equally sized datasets (one for use as a Target data in the CB-SBIT algorithm and the other for use as test data)
2. We used the Bot and DDoS datasets as source datasets in CB-SBIT
3. We ran CB-SBIT with its default parameters

The original Portscan Target dataset size was 57 instances (26 Benign and 31 Portscan) and the new Portscan dataset size (after instance transfer by the CB-SBIT algorithm) was 203877 instances (101936 Benign and 101941 Portscan). Accuracy before transfer (i.e training the RF classifier on the original small training Portscan dataset) is 82.45% and accuracy after transfer (i.e training the RF classifier on the new dataset that contains the original instances of the small training Portscan dataset and the instances transferred to it from the source datasets using the CB-SBIT algorithm) is 96.25%). This is positive transfer (i.e. results are better after using the CB-SBIT algorithm)

We have applied our MTL method using the same datasets we have used to evaluate the performance of the CB-SBIT algorithm. Remember in our MTL we compute the similarity and add new columns as features (as explained in section 2). Therefore, these datasets contain the similarity columns that are computed as part of our MTL approach. The procedure was as follows:

1. Concatenate the DDoS dataset with the training datasets of both Bot and Portscan
2. Train a RandomForest classifier on the resulting dataset
3. Predict the Bot and Portscan test datasets and compute the accuracy for each of them

After MTL, the accuracy was 98.51% on the Bot test dataset and 99.76% on the Portscan test dataset. This shows that our MTL approach outperforms the CB-SBIT algorithms on both datasets. The results of this evaluation are shown in figure 7. In summary, for the Portscan data, CB-SBIT shows an improvement in accuracy when compared with using the RF classifier without instance transfer (i.e. the single task learning), but MTL shows a better improvement. On the other hand, for the Bot data, CB-SBIT shows a dis-improvement in accuracy when compared with using the RF classifier without instance transfer (i.e. the single task learning), because accuracy goes down, whereas MTL shows a significant improvement.

Our implementation will soon be open sourced and available on this Github repository[1].

[1]https://github.com/reem2021/multi-task-ids

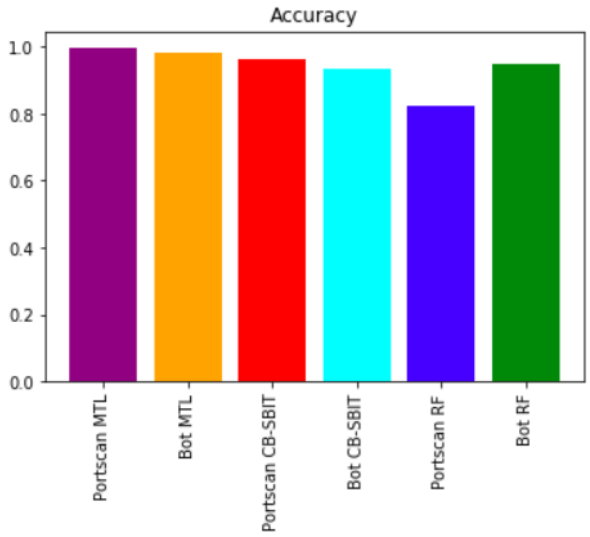


Figure 7: Results of Comparing Performance of CB-SBIT and our MTL Approach

8 Conclusions and Future Work

. This paper introduces a novel classification method that is based on multi-task learning and shows its effectiveness in accurately classifying computer network traffic especially when training data is scarce. The method is inspired by an existing regression method that suffers from drawbacks as was explained in previous sections. The performance of several classical classifiers was evaluated and it was concluded that RandomForest is the winner (this is consistent with existing research) and therefore it was selected for comparison. In order to run fair evaluation, and because four separate datasets were available (each one represents a task), several datasets were formed by using pairwise combinations (each two different tasks against each other). After that the resulting datasets were used as a basis for binary classification.

The method proposed in this paper was evaluated against RandomForest (while using the same data) and the results demonstrated significant improvement in performance when using our method as opposed to RandomForest.

In addition, a recent open-source transfer learning approach was used in the evaluation experiments so that it is possible to compare the performance of our method against it (namely this is the successful CB-SBIT approach). Our experiments reveal that, not only the fact that our method outperforms CB-SBIT when data is scarce, but also CB-SBIT can result in negative transfer which leads to worse results (when compared with results obtained without applying the transfer learning technique of CB-SBIT).

In the near future we plan to test our approach on data from other domains and experiment with other similarity measures. In addition, we plan to explore the possibility of finding a method to learn the similarity between tasks instead of computing it.

References

[1] D. Dasgupta, Z. Akhtar, S. Sen, The Journal of Defense Modeling and Simulation **0**, 1548512920951275 (0), https://doi.org/10.1177/1548512920951275

[2] R. Caruana, Machine Learning **28**, 41 (1997)

[3] N. Sadawi, I. Olier, J. Vanschoren, J.N. van Rijn, J. Besnard, R. Bickerton, C. Grosan, L. Soldatova, R.D. King, Journal of Cheminformatics **11**, 68 (2019)

[4] S.J. Pan, Q. Yang, IEEE Trans. on Knowl. and Data Eng. **22**, 1345 (2010)

[5] J. Zhou1, J. Chen, J. Ye, *Multi-task learning: Theory, algorithms, and applications*, https://archive.siam.org/meetings/sdm12/zhou_chen_ye.pdf

[6] G. Draper-Gil, A.H. Lashkari, M.S.I. Mamun, A.A. Ghorbani, *Characterization of Encrypted and VPN Traffic using Time-related Features*, in *ICISSP* (2016)

[7] C.I. for Cybersecurity, *Intrusion detection evaluation dataset (cic-ids2017)*, https://www.unb.ca/cic/datasets/ids-2017.html

[8] R. Di Pietro, L.V. Mancini, *Intrusion Detection Systems*, 1st edn. (Springer Publishing Company, Incorporated, 2008), ISBN 0387772650

[9] M.M. Deza, E. Deza, *Encyclopedia of Distances* (Springer Berlin Heidelberg, 2009)

[10] C. Shui, M. Abbasi, L. Robitaille, B. Wang, C. Gagné, CoRR **abs/1903.09109** (2019), 1903.09109

[11] S. Ben-David, R.S. Borbely, Mach. Learn. **73**, 273–287 (2008)

[12] J. Baxter, Mach. Learn. **28**, 7–39 (1997)

[13] L. Duong, T. Cohn, S. Bird, P. Cook, *Low Resource Dependency Parsing: Cross-lingual Parameter Sharing in a Neural Network Parser*, in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)* (Association for Computational Linguistics, Beijing, China, 2015), pp. 845–850, https://www.aclweb.org/anthology/P15-2139

[14] Y. Yang, T.M. Hospedales, CoRR **abs/1606.04038** (2016), 1606.04038

[15] S. Vandenhende, S. Georgoulis, W.V. Gansbeke, M. Proesmans, D. Dai, L.V. Gool, *Multi-task learning for dense prediction tasks: A survey* (2020), 2004.13379

[16] S. Rezaei, X. Liu, *Multitask Learning for Network Traffic Classification*, in *2020 29th International Conference on Computer Communications and Networks (ICCCN)* (2020), pp. 1–9

[17] H. Huang, H. Deng, J. Chen, L. Han, W. Wang, International Journal of Emerging Technologies in Learning (iJET) **13**, 4 (2018)

[18] S. Rezaei, X. Liu, *A Target-Agnostic Attack on Deep Models: Exploiting Security Vulnerabilities of Transfer Learning*, in *International Conference on Learning Representations* (2020), https://openreview.net/forum?id=BylVcTNtDS

[19] B. Alothman, International Journal of Intelligent Computing Research (IJICR) **9**, 880– (2018)

[20] B. Alothman, H. Janicke, S.Y. Yerima, *Class Balanced Similarity-Based Instance Transfer Learning for Botnet Family Classification*, in *Discovery Science*, edited by L. Soldatova, J. Vanschoren, G. Papadopoulos, M. Ceci (Springer International Publishing, Cham, 2018), pp. 99–113, ISBN 978-3-030-01771-2

[21] Y. Zhang, Q. Yang, CoRR **abs/1707.08114** (2017), 1707.08114

[22] G. Santafe, I.n. Inza, J.A. Lozano, Artif. Intell. Rev. **44**, 467 (2015)