Overcoming the Lack of Labeled Data: Training Intrusion Detection Models using Transfer Learning

Ankush Singla
Department of Computer Science
Purdue University
West Lafayette, USA
asingla@purdue.edu

Elisa Bertino
Department of Computer Science
Purdue University
West Lafayette, USA
bertino@purdue.edu

Dinesh Verma
Thomas J. Watson Research Center
IBM, Yorktown Heights
New York, USA
dverma@us.ibm.com

***Abstract*—Deep learning (DL) techniques have recently been proposed for enhancing the accuracy of network intrusion detection systems (NIDS). However, keeping the DL based detection models up to date requires large amounts of new labeled training data which is often expensive and time-consuming to collect. In this paper, we investigate the viability of transfer learning (TL), an approach that enables transferring learned features and knowledge from a trained source model to a target model with minimal new training data. We compare the performance of a NIDS model trained using TL with a NIDS model trained from scratch. We show that TL enables detection models to perform much better at identifying new attacks when there is relatively less training data available.**

***Index Terms*—Information Security, Neural Networks, Deep Learning, Intrusion Detection**

I. Introduction

Software systems are increasingly getting targeted by intelligent adversaries with sufficient resources leading to identification and exploitation of new vulnerabilities everyday. The McAfee Labs reports that the count of newly identified malware reached an all-time high of 57.3 million samples in Q3 of 2017 [1]. It is getting increasingly difficult for security researchers and analysts to identify and defend against the huge numbers of new exploits by manually analyzing them. We thus need intelligent techniques able to automatically perform such analyses. Deep Learning (DL) is a technology increasingly used to design such techniques. DL is a branch of machine learning (ML). It learns various characteristics from data and uses them for decision making on similar unseen data. Approaches have been proposed that use DL for various information security tasks like malware analysis [2] [3] [4], intrusion detection [5] [6] [7], and botnet detection [8] [9]. DL based techniques can learn from the patterns and features of training datasets containing packet traces of various attacks and can later look at current network traffic to detect ongoing attacks.

Problem: DL based NIDS generally require large amounts of labeled data to train the classification models in order to identify malware and attacks with high accuracy. However, there is often very little data related to new malware or attacks for reliably training these models. The new training data is often expensive and time-consuming to collect. Another issue is that whenever a new malware or attack is identified, the DL models have to be retrained on the new data from scratch, which requires a lot of computing resources and time.

Solution: The concept of *transfer learning* (TL) can be used to alleviate the problem of limited training data for new malware and attacks. TL enables transferring learned features and knowledge from a trained source model to a target model with minimal new training data. For example, an image classification model trained to detect 1000 different categories of objects can be re-purposed for a new specialized domain like identifying the various types of roses. Transferring the knowledge learned by the general image classification model to a rose classification model gives much better results than training on the rose image dataset from scratch. Another example of TL might be where a model trained to classify different models of cars is re-purposed to identify a new car model that was just released. To the best of our knowledge, the effectiveness of TL for information security applications has not been investigated.

Our contributions: In this paper, we experimentally assess the viability of TL for training DL models with minimal new labeled data for a NIDS. In our experiments, we consider a NIDS classifier initially trained to detect various malicious attacks from packet captures and then apply TL to the initial classifier to generate a model for detecting a new family of attacks. We compare the accuracy and precision of such a model with a DL model trained from scratch. We perform this comparison for various amounts of training data and show that TL based models are much more accurate than a DL model trained from scratch when there is very little training data available. To perform the experiments, we choose a NIDS dataset that contains multiple attack categories and divide it into two parts: a source dataset (containing various attack categories) and a target dataset (containing a new attack type not present in the source dataset).

II. Background

A. Neural networks

Machine learning (ML) is a branch of artificial intelligence (AI), that enables computers to learn and make decisions using

978-1-7281-1689-1/19/$31.00 ©2019 IEEE
DOI 10.1109/SMARTCOMP.2019.00031

statistical techniques without the need for manual programming [10]. ML involves training a model using input data and then making predictions on new data using the learned model. Some of the popular ML algorithms include decision trees [11], support vector machines (SVM) [12], Bayesian networks [13], and logistic regression [14].

Neural networks (NN) are systems inspired by the human brain and how the network of neurons processes information and performs computations [15]. The building blocks of these networks are called neurons that are grouped in different layers. Each layer of neurons receives inputs from the previous layer of neurons and passes its output to the next layer. Each neuron has its own activation functions, weights, and biases which it tunes and optimizes during the training phase depending on the cost function to minimize the total error between the predicted value and the provided value. A **deep neural network (DNN)** is an NN with multiple hidden layers in addition to the input and output layers. Any NN with less layers than that is called a shallow NN. **Deep learning (DL)** is a sub-field of ML, based on the use of DNNs.

B. Transfer learning

The original motivation behind TL is employing the knowledge and skills learned from performing a specific task towards performing a different but similar task. The first use of TL in the context of ML is attributed to Lorien Pratt and his algorithm called Discriminability-Based Transfer (DBT) [16]. Since then, there has been a lot of interest in TL in the context of different topics: learning to learn, life-long learning, knowledge transfer, inductive transfer, multi-task learning, knowledge consolidation, context-sensitive learning, knowledge-based inductive bias, meta learning, and incremental/cumulative learning [17]. TL has been shown to be successful in various domains like image classification [18] [19], text classification [20] [21], speech recognition [22] [23] etc. However, to the best of our knowledge there have been no efforts in using TL in the information security domain.

III. Dataset

A. Selecting a NIDS dataset

Several datasets have been created in the past for the evaluation of NIDS. For the longest time, researchers have used the KDD-CUP99 [24] benchmark dataset for evaluating their NIDS. However, such dataset suffers from various issues like redundant records, different probability distribution for testing and training datasets which skew the statistical detection algorithm results [25] [26]. The NSL-KDD [27] dataset was later created to address the shortcomings of the KDD-CUP99 dataset. However, it still does not represent the modern attack behaviors due to being roughly 20 years old.

The UNSW-NB15 dataset [28] does not suffer from those drawbacks and is thus a suitable dataset for our experiments. The raw network packets for UNSW-NB15 were generated by the IXIA PerfectStorm tool for normal traffic and for 9 types of attacks. The packets were captured using the Tcpdump tool and 49 features were extracted using various algorithms. These features include source IP and port, destination IP and port, transaction protocol, transaction bytes, TTL values etc. USNW-NB15 contains a total of 2,540,044 labeled records.

B. Splitting USNW-NB15 into sub-datasets

We use a subset of the full UNSW-NB15 dataset containing 175,341 records from the different categories, attack and normal (see Table I for the distribution of these categories).

TABLE I
DISTRIBUTION OF ATTACK CATEGORIES

Category	Percentage	Number of records
Normal	31.94 %	56000
Generic	22.81 %	40000
Exploits	19.04 %	33393
Fuzzers	10.37 %	18184
DoS	6.99 %	12264
Reconnaissance	5.98 %	10491
Analysis	1.14 %	2000
Backdoor	1.0 %	1746
Shellcode	0.65 %	1133
Worms	0.07 %	130

We use this dataset to create 9 different sub-datasets for each attack type. Each sub-dataset is the original UNSW-NB15 dataset split into two parts: **source dataset** and **target dataset** as described in Table II. For example, the generic attacks sub-dataset contains: (a) target dataset with 40,000 generic records and 30,000 normal records, and (b) source dataset containing the rest of the UNSW-NB15 dataset.

TABLE II
DESCRIPTION OF SUB-DATASETS

Category	Target Dataset	Source Dataset
Generic	Generic (40,000 records) + Normal (30,000 records)	Remaining 105,341 records
Exploits	Exploits (33,393 records) + Normal (30,000 records)	Remaining 111,948 records
Fuzzers	Fuzzers (18,184 records) + Normal (20,000 records)	Remaining 137,157 records
DoS	DoS (12,264 records) + Normal (15,000 records)	Remaining 148,077 records
Reconnaissance	Reconnaissance (10,491 records) + Normal (15,000 records)	Remaining 149,850 records
Analysis	Analysis (2,000 records) + Normal (4,000 records)	Remaining 169,341 records
Backdoor	Backdoor (1746 records) + Normal (2000 records)	Remaining 171,595 records
Shellcode	Shellcode (1133 records) + Normal (1500 records)	Remaining 172,708 records
Worms	Worms (130 records) + Normal (500 records)	Remaining 174,711 records

IV. Experiment Details

We provide below the details of the experiments, the NNs used, the metrics collected and the system setup.

A. Description of experiments

We use two DNNs to perform our experiments. The first is a smaller DNN with just 2 intermediate layers with 10 neurons each and the second is a larger DNN with 5 intermediate layers (including 3 layers with 50 neurons and 2 layers with 10 neurons). These models are detailed in Section IV-C. We take the 9 sub-datasets mentioned in Section III and train the final NIDS models in the following two ways:

- **Base case:** We train the DNN from scratch using the target dataset.
- **Transfer learning case:** We reset the weights on the final softmax layer of the DNN pre-trained on the source dataset and use the target dataset to re-train the DNN while keeping the same weights on the other layers.

We sample the 9 sub-datasets with different numbers of records and compare their classification metrics. For example, starting with the target dataset in sub-dataset 1 (i.e., the generic sub-dataset), we use 70000, 35000, 17500, 8700, 4300, 2100, 1000, 500, 200, 100 training and testing samples for our experiments to compare the base and transfer learning cases. We report metrics from these experiments in Section V.

B. Data pre-processing

We drop the attack_category field from the dataset as in our experimental evaluation we focus on binary classification, that is, whether the record represents an attack or just benign traffic. We do not try to predict the specific attack category in our experiments. We encode the text fields and assign them numerical codes. For example, the protocol field can have possible values udp, tcp, icmp and arp; they will thus be converted to 1, 2, 3, 4, respectively, in the dataset. Similarly, the service and state fields are also assigned numerical values. For the other numerical fields, we standardize their values, and replace them with standard scores (or z-scores).

C. Neural network details

The basic layout and structure of the two DNNs used for our experiments are as follows.

Small DNN: We use a simple DNN with 2 regular densely-connected layers containing 10 neurons each with *relu* activation function, and a final layer containing 2 neurons and a *softmax* activation function. The model has 562 trainable parameters and has a size of 34.9 kB.

Large DNN: We use a simple DNN with 5 regular densely-connected layers containing 10, 50, 50, 50, 10 neurons respectively, with *relu* activation function, and a final layer containing 2 neurons and a *softmax* activation function. The model has 6,612 trainable parameters and has a size of 123 kB.

D. Training parameters

The training parameters used for our experiments are as follows. (a) *Train-test split:* For all the datasets, we randomly split the records into 75% training and 25% testing records. (b) *Epochs:* We use 20 epochs to allow our training algorithm to converge. (c) *Batch size:* We use a batch-size of 32 records for training all the NNs. (d) *Optimizer:* We use the Adam optimizer [29] with a starting learning rate of 0.001 to minimize the error function while training.

E. Metrics

For classification problems, we first calculate the following counts based on the comparison of actual versus the predicted results: (a) *True Positives (TP):* Number of records predicted to be attacks that are actually attacks. (b) *False Positives (FP):* Number of records predicted to be attacks that are actually normal. (c) *True Negatives (TN):* Number of records predicted to be normal that are actually normal. (d) *False Negatives (FN):* Number of records predicted to be normal that are actually attacks.

On the basis of the above counts, the following metrics are used to compare the classification performance.

$$Accuracy = \frac{TP}{TP + FP + TN + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F\text{-}score = \frac{2 * Precision * Recall}{Precision + Recall}$$

The above metrics vary from 0 to 1. *Accuracy* represents the fraction of correctly classified records. *Precision* represents the fraction of records classified as attacks that were actually attacks. *Recall* tells us the fraction of actual attacks that were correctly predicted as attacks. *F-score* represents both precision and recall with a single number. We ideally want to maximize all of these metrics for a good performing classification model.

F. System and libraries used

We use a standard PC with an Intel-Core i5 6300 HQ CPU (2.30 GHz) and 8 GB RAM for training and evaluating the NNs. We use Tensorflow [30], an open-source machine learning library, with the Keras wrapper [31] for defining and training the NN models. These libraries allow programmers to create powerful NN models with relative ease.

V. Results and Analysis

We now discuss the experimental results. Although, we performed experiments for all 9 sub-datasets as mentioned in Section III, due to space constraints we only present results from 3 sub-datasets (generic, reconnaissance and shell-code). The results from the other 6 sub-datasets are similar to the results reported here. We also captured metrics like precision, recall and f-score; however, we only report accuracy results. We observe similar trends for the other metrics.

For the smaller DNN, we observe that using TL results in models with a much better classification accuracy than the base case, when we provide very few training samples. Both the TL case model and the base case model provide similar accuracy

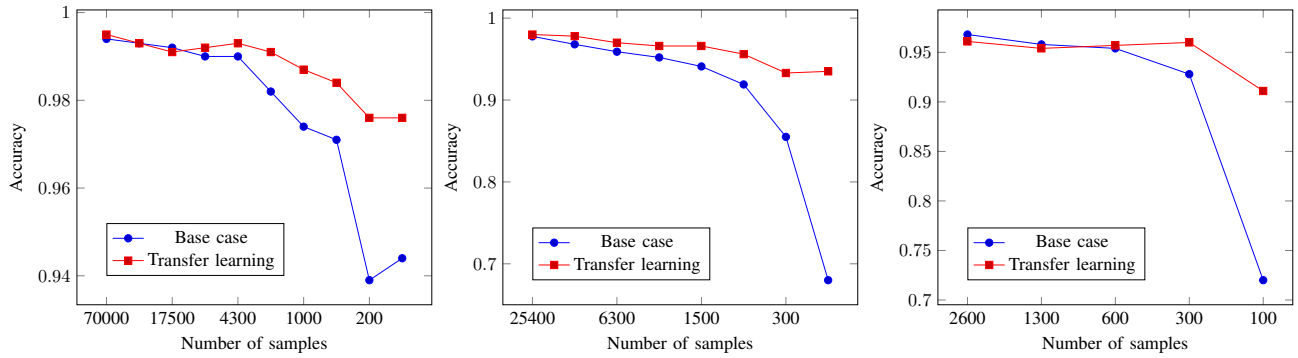

Fig. 1. Accuracy for smaller DNN (Generic, Reconnaissance and Shellcode attacks)

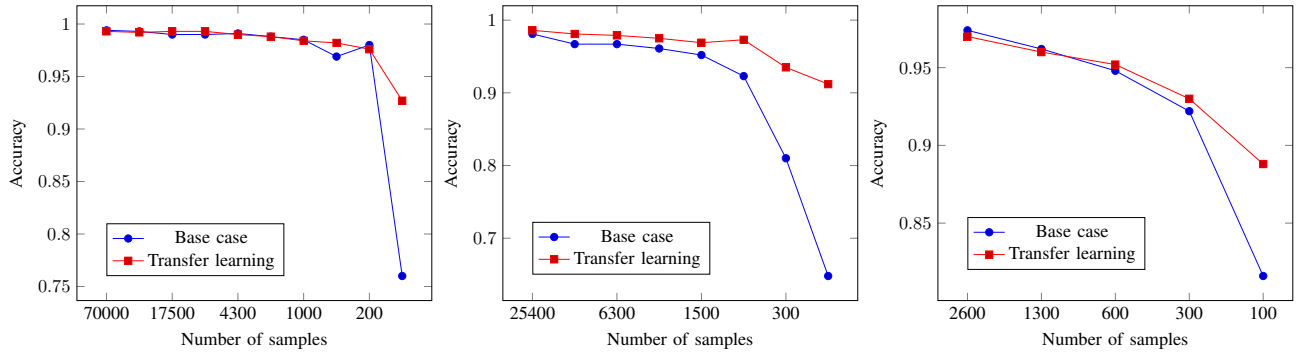

Fig. 2. Accuracy for larger DNN (Generic, Reconnaissance and Shellcode attacks)

when adequate training samples are provided (more than 300). However, the base case model accuracy gradually starts to decline with decreasing numbers of training samples. In the cases of generic, reconnaissance and shell-code attacks the TL model provides 3.2%, 25.5% and 19.1% better accuracy, respectively, than the base case model when trained with 100 training samples. We obtain similar results for the larger DNN. In the cases of generic, reconnaissance and shell-code attacks the TL model has 16.7%, 26.4% and 7.2% better accuracy, respectively, than the base case model when trained with 100 training samples.

The main reason for the TL models having equal or better accuracy than the base case models is that the optimizing algorithm can converge to the optimal weights quicker if the NN model already has weights initialized by training on a similar domain as opposed to starting from random weights or zero weights. Another major reason of base case models under-performing with less training data is the phenomenon of overfitting, which means the model is so tightly fitted to the provided training dataset distribution that it performs badly on any other unseen data. This phenomenon is more pronounced with less training data.

VI. Related Work

Various researchers have used NNs for intrusion detection. Cannady et al. [5] were one of the first to show the use of NNs for misuse-based intrusion detection. Palagiri [6] introduced another approach using NNs for misuse-based intrusion detection. They explored network based intrusion detection using a perceptron-based, feed-forward NN and a system based on classifying, self-organizing maps. Hodo et al. [7] focus on IoT networks and suggest the use of NNs as an offline IDS to analyze information from the network and identify DoS/DDoS attacks. Tuor et al. [32] propose an online unsupervised DL approach for real-time detection anomalous network activity from system logs. Katherios et al. [33] propose a real-time network anomaly detection system that reduces the manual workload by coupling two learning stages.

However, none of those research efforts have analyzed the impact of the amount of training data on the performance of the generated models. All experiments carried out in such previous works, always assume NNs trained with adequate data amounts. No experiments have been reported assessing the models' performance when there is very little training data available. To the best of our knowledge, we are the first to assess the performance of NN models for NIDS when training data is limited, and assess the use of TL in these scenarios.

VII. Conclusion and Future Work

In this paper, we report experimental results showing that TL is effective in reducing the amount of labeled data required for training NN models for NIDS. This allows for quick, easy

and efficient training of NN models to detect newer attacks and malware.

Our work has a lot of scope for further research. NIDS datasets other than the UNSW-NB15, for specific network types (for example: IoT networks, mobile networks or vehicular networks) need to be explored. In this work we look at transferring knowledge from a source dataset to a target dataset belonging to the same type of network. It would be interesting to see how effective TL is when source and target datasets belong to different kinds of network. For example, can a model trained to identify attacks and malware for mobile networks be used to identify malwares for IoT devices, by using minimal labeled data and TL? The attacks on these specific networks might have common characteristics in terms of the traffic patterns generated by these attacks which can make TL feasible. To be able to successfully use TL for different network types, we need to handle input feature discrepancies in the source and target datasets. Possible approaches to handle such issue include: (a) removing the input layer of the NN trained on the source dataset and replace it with an input layer with the feature size of the target dataset, (b) assigning a zero value to the input features from the source dataset not present in the target dataset when performing TL, (c) assigning random values to input features from the source dataset not present in the target dataset when performing TL, (d) training different NNs, one using the features common to the source and the target datasets and another using the new features from the target dataset, and then combining them using ensemble learning [34].

It would be interesting to explore other types of specialized NNs like convolutional neural networks (CNN) and recurrent neural networks (RNN). These specific NNs may be able to provide better classification accuracy than general NNs. Experimenting with different number or ordering of layers with different neurons and activation functions can be explored to improve the performance of those NNs. Varying hyperparameters, like the choice of optimizer, number of epochs for training and loss function, can also result in interesting results.

In this work, we looked at the classification accuracy and precision metrics for comparing and evaluating the resulting NN models. However, for the deployment of these models in real-world devices, efficient and low-latency NN models are crucial for real-time intrusion detection, especially for smaller IoT and mobile devices. Thus, optimization techniques are needed to reduce latency, and memory and computation costs.

In addition, the applicability of TL must be investigated for security tasks other than NIDS. Security tasks, like malware analysis and botnet detection, are tasks with constantly changing data distribution, due to new attacks and malware and often have limited labeled data for training.

VIII. Acknowledgements

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

We would also like to thank Wei-Jen Lee for the suggestions concerning the TL techniques to use when the source and target datasets have different input features.

References

[1] McAfee, "Mcafee labs threats report," Dec 2017. [Online]. Available: https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-dec-2017.pdf

[2] J. O. Kephart, G. B. Sorkin, W. C. Arnold, D. M. Chess, G. J. Tesauro, S. R. White, and T. Watson, "Biologically inspired defenses against computer viruses," in *IJCAI (1)*, 1995, pp. 985–996.

[3] G. E. Dahl, J. W. Stokes, L. Deng, and D. Yu, "Large-scale malware classification using random projections and neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3422–3426.

[4] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *Malicious and Unwanted Software (MALWARE), 2015 10th International Conference on*. IEEE, 2015, pp. 11–20.

[5] J. Cannady, "Artificial neural networks for misuse detection," in *National information systems security conference*, vol. 26. Baltimore, 1998.

[6] C. Palagiri, "Network-based intrusion detection using neural networks," *Department of Computer Science Rensselaer Polytechnic Institute Troy, New York*, pp. 12 180–3590, 2002.

[7] E. Hodo, X. Bellekens, A. Hamilton, P.-L. Dubouilh, E. Iorkyase, C. Tachtatzis, and R. Atkinson, "Threat analysis of iot networks using artificial neural network intrusion detection system," in *Networks, Computers and Communications (ISNCC), 2016 International Symposium on*. IEEE, 2016, pp. 1–6.

[8] A. Karim, R. Salleh, and M. K. Khan, "Smartbot: A behavioral analysis framework augmented with machine learning to identify mobile botnet applications," *PloS one*, vol. 11, no. 3, p. e0150077, 2016.

[9] I. Arnaldo, A. Cuesta-Infante, A. Arun, M. Lam, C. Bassias, and K. Veeramachaneni, "Learning representations for log data in cybersecurity," in *International Conference on Cyber Security Cryptography and Machine Learning*. Springer, 2017, pp. 250–268.

[10] J. R. Koza, F. H. Bennett, D. Andre, and M. A. Keane, "Automated design of both the topology and sizing of analog electrical circuits using genetic programming," in *Artificial Intelligence in Design96*. Springer, 1996, pp. 151–170.

[11] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[12] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their applications*, vol. 13, no. 4, pp. 18–28, 1998.

[13] T. D. Nielsen and F. V. Jensen, *Bayesian networks and decision graphs*. Springer Science & Business Media, 2009.

[14] N. M. Nasrabadi, "Pattern recognition and machine learning," *Journal of electronic imaging*, vol. 16, no. 4, p. 049901, 2007.

[15] M. van Gerven and S. M. Bohte, "Editorial: Artificial neural networks as models of neural information processing," *Front. Comput. Neurosci.*, vol. 2017, 2017. [Online]. Available: https://doi.org/10.3389/fncom.2017.00114

[16] L. Y. Pratt, "Discriminability-based transfer between neural networks," in *Advances in neural information processing systems*, 1993, pp. 204–211.

[17] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[18] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1717–1724.

[19] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught learning: transfer learning from unlabeled data," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 759–766.
[20] C. B. Do and A. Y. Ng, "Transfer learning for text classification," in *Advances in Neural Information Processing Systems*, 2006, pp. 299–306.
[21] W. Dai, G.-R. Xue, Q. Yang, and Y. Yu, "Transferring naive bayes classifiers for text classification," in *AAAI*, vol. 7, 2007, pp. 540–545.
[22] J. Deng, Z. Zhang, E. Marchi, and B. Schuller, "Sparse autoencoder-based feature transfer learning for speech emotion recognition," in *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*. IEEE, 2013, pp. 511–516.
[23] J.-T. Huang, J. Li, D. Yu, L. Deng, and Y. Gong, "Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 7304–7308.
[24] M. L. Labs, "Kdd cup 1999 data," 1999. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/task.html
[25] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," *ACM Transactions on Information and System Security (TISSEC)*, vol. 3, no. 4, pp. 262–294, 2000.
[26] M. V. Mahoney and P. K. Chan, "An analysis of the 1999 darpa/lincoln laboratory evaluation data for network anomaly detection," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2003, pp. 220–237.
[27] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*. IEEE, 2009, pp. 1–6.
[28] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *Military Communications and Information Systems Conference (MilCIS), 2015*. IEEE, 2015, pp. 1–6.
[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980
[30] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.
[31] F. Chollet, "keras," *GitHub*, 2015, https://github.com/fchollet/keras.
[32] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," *CoRR*, vol. abs/1710.00811, 2017. [Online]. Available: http://arxiv.org/abs/1710.00811
[33] G. Kathareios, A. Anghel, A. Mate, R. Clauberg, and M. Gusat, "Catch it if you can: Real-time network anomaly detection with low false alarm rates," in *Machine Learning and Applications (ICMLA), 2017 16th IEEE International Conference on*. IEEE, 2017, pp. 924–929.
[34] T. G. Dietterich *et al.*, "Ensemble learning," *The handbook of brain theory and neural networks*, vol. 2, pp. 110–125, 2002.