DEEP LEARNING CLASSIFICATION METHODS APPLIED TO TABULAR CYBERSECURITY BENCHMARKS

David A. Noever and Samantha E. Miller Noever

PeopleTec, Inc., Huntsville, Alabama, USA

ABSTRACT

This research recasts the network attack dataset from UNSW-NB15 as an intrusion detection problem in image space. Using one-hot-encodings, the resulting grayscale thumbnails provide a quarter-million examples for deep learning algorithms. Applying the MobileNetV2's convolutional neural network architecture, the work demonstrates a 97% accuracy in distinguishing normal and attack traffic. Further class refinements to 9 individual attack families (exploits, worms, shellcodes) show an overall 54% accuracy. Using feature importance rank, a random forest solution on subsets shows the most important source-destination factors and the least important ones as mainly obscure protocols. It further extends the image classification problem to other cybersecurity benchmarks such as malware signatures extracted from binary headers, with an 80% overall accuracy to detect computer viruses as portable executable files (headers only). Both novel image datasets are available to the research community on Kaggle.

KEYWORDS

Neural Networks, Computer Vision, Image Classification, Intrusion Detection, MNIST Benchmark.

1. INTRODUCTION

This work explores image-based classifiers for non-traditional tasks, either to recognize a pattern described previously only in numerical tables or to extract meaningful malware signatures as images. The interest in this approach arises from the success of applying deep convolutional neural networks (CNN) to similarly challenging but unconventional ways borrowed from computer vision. For example, audio classifiers and speech recognition have benefited from projecting the audio time-series into a spectrogram, which in turn a CNN can apply computer vision methods to identify words or sounds.

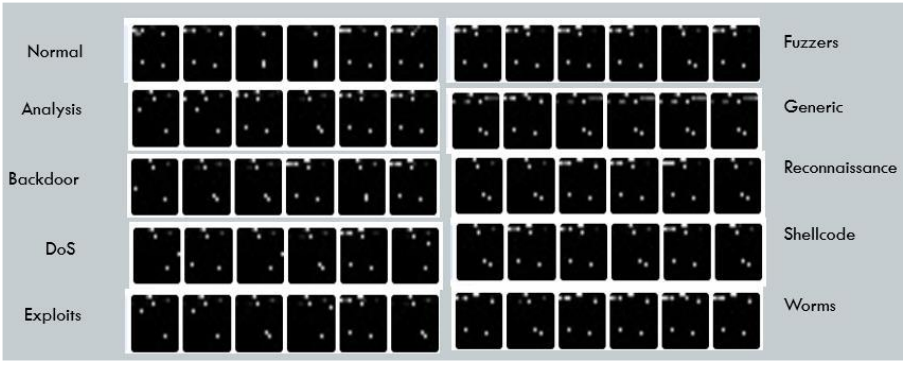

Figure 1. Nine attack types and one normal traffic dataset. We map the tabular features to grayscale thumbnails

DOI: 10.5121/ijnsa.2021.13301

For cybersecurity applications, this work updates the UNSW-NB15 attack dataset [1-4] and extends the popular intrusions detection system (IDS) originally inspired by the KDD-99/DARPA challenge [5-7]. The details of the UNSW-NB15 dataset are published in a series of previous papers [1-4] which described the raw network packet captures, generated features on labeled attacks, and scored statistical methods for identifying each attack family. As illustrated in Figure 1, the current approach aims to map scaled numerical features to images, a method likened to traditional spectrogram methods. These fingerprinting techniques have proven useful when image-based neural networks have solved similar but challenging time-dependent [8] or audio [9] problems. We test the capabilities for mapping tabular features to build fast image classifiers. One advantage of this hierarchical method arises from the unique power of transfer learning to high accuracy, even when the underlying patterns prove difficult for humans to understand or classify. The datasets are available on Kaggle [10].

2. METHODS

The new datasets to evaluate image classification as a network defense tool include the intrusion simulations from UNSW-NB15 [1-4] and a custom anti-virus example.

2.1. Network Intrusion Dataset

The use of convolutional neural networks for network attack classification depends on first converting all tabular feature sets into thumbnail images. We, therefore, recast the UNSW-NB15

	1	2	3	4	5	6	7	8
1	dur	spkts	dpkts	sbytes	dbytes	rate	sttl	dttl
2	swin	stcpb	dtcpb	dwin	tcprtt	synack	ackdat	smean
3	ct_dst_src_ltm	is_ftp_login	ct_ftp_cmd	ct_flw_http_mthd	ct_src_ltm	ct_srv_dst	is_sm_ips_ports	service_-
4	service_smtp	service_snmp	service_ssh	service_ssl	protocol_3pc	protocol_a	n	protocol_aes-sp3-d
5	protocol_cbt	protocol_cftp	protocol_chaos	protocol_compaq	protocol_cphb	protocol_cpnx	protocol_crtp	protocol_crudp
6	protocol_etherip	protocol_fc	protocol_fire	protocol_ggp	protocol_gmtp	protocol_gre	protocol_hmp	protocol_i-nlsp
7	protocol_il	protocol_ip	protocol_ipcomp	protocol_ipcv	protocol_ipip	protocol_iplt	protocol_ipnip	protocol_ippc
8	protocol_iso-ip	protocol_iso-tp4	protocol_kryptolan	protocol_l2tp	protocol_larp	protocol_leaf-1	protocol_leaf-2	protocol_merit-inp
9	protocol_nsfnet-igp	protocol_nvp	protocol_ospf	protocol_pgm	protocol_pim	protocol_pipe	protocol_pnni	protocol_pri-enc
10	protocol_sat-expak	protocol_sat-mon	protocol_sccopmce	protocol_scps	protocol_sctp	protocol_sdrp	protocol_secure-v	protocol_sep
11	protocol_stp	protocol_sun-nd	protocol_swipe	protocol_tcf	protocol_tcp	protocol_tlsp	protocol_tp++	protocol_trunk-1
12	protocol_vrrp	protocol_wb-expak	protocol_wb-mon	protocol_wsn	protocol_xnet	protocol_xns-idp	protocol_xtp	protocol_zero
13	pad	pad	pad	pad	pad	pad	pad	pad
14	pad	pad	pad	pad	pad	pad	pad	pad
15	pad	pad	pad	pad	pad	pad	pad	pad
16	pad	pad	pad	pad	pad	pad	pad	pad

	9	10	11	12	13	14	15	16
1	sload	dload	sloss	dloss	sinpkt	dinpkt	sjit	djit
2	dmean	trans_depth	response_body_le	ct_srv_src	ct_state_ttl	ct_dst_ltm	ct_src_dport_ltm	ct_dst_sport_ltm
3	service_dhcp	service_dns	service_ftp	service_ftp-data	service_http	service_irc	service_pop3	service_radius
4	protocol_any	protocol_argus	protocol_aris	protocol_arp	protocol_ax.25	protocol_bbn-rcc	protocol_bna	protocol_br-sat-mon
5	protocol_dcn	protocol_ddp	protocol_ddx	protocol_dgp	protocol_egp	protocol_eigrp	protocol_emcon	protocol_encap
6	protocol_iatp	protocol_ib	protocol_idpr	protocol_idpr-cm	protocol_idrp	protocol_ifmp	protocol_igmp	protocol_igp
7	protocol_ipv6	protocol_ipv6-fra	protocol_ipv6-no	protocol_ipv6-opt	protocol_ipv6-rou	protocol_ipx-n-ip	protocol_irtp	protocol_isis
8	protocol_mfe-nsp	protocol_mhrp	protocol_micp	protocol_mobile	protocol_mtp	protocol_mux	protocol_narp	protocol_netblt
9	protocol_prm	protocol_ptp	protocol_pup	protocol_pvp	protocol_qnx	protocol_rdp	protocol_rsvp	protocol_rvd
10	protocol_skip	protocol_sm	protocol_smp	protocol_snp	protocol_sprite-rp	protocol_sps	protocol_srp	protocol_st2
11	protocol_trunk-2	protocol_ttp	protocol_udp	protocol_unas	protocol_uti	protocol_vines	protocol_visa	protocol_vmtp
12	state_ACC	state_CLO	state_CON	state_FIN	state_INT	state_REQ	state_RST	pad
13	pad	pad	pad	pad	pad	pad	pad	pad
14	pad	pad	pad	pad	pad	pad	pad	pad
15	pad	pad	pad	pad	pad	pad	pad	pad
16	pad	pad	pad	pad	pad	pad	pad	pad

Figure 2. Layout template for one-hot-encoded images.

tabular set of features as scaled image thumbnails [11] to solve for 9 families of attack types. This version of the dataset renders the corresponding UNSW-NB15 attack set as 256-pixel grayscale images (16 x16). We employ one-hot-encoding [12] for the categorical inputs and rescale all

numerical inputs as grayscale pixel values (0-255) between the training set's minimum and maximum values. The baseline UNSW-NB15 dataset [1-4] yields 194 values and the images are right padded with all black (255) values for any unused pixels (62) identically for all attack labels. This padding assists deep learning approaches [13] which have a stride length in powers of 2. The column labels are also included in the train and test sets as tabular formats (comma-separated value files) to compare image-based classification methods to more statistical approaches like decision trees, random forest, and support vector machines. The expectation is that all the legacy algorithms of both deep learning and statistical machine learning may assist in the new task after mapped to images of feature sets.

This approach shares many characteristics with the traditional MNIST dataset [14-20] and thus can build quickly on those findings for algorithmic comparisons. Several image-based problems to solve include simply binary classifiers for attack vs. normal traffic. Like MNIST digits [14], there are 10 categories shown (0=normal; 1-9 various attacks). As shown in Figure 2, the original 42 network features expand to 194 when one-hot-encoded [12]. This process converts all categorical data (services, protocols, and states) into individual columns with their presence marked by 1 and absence by 0. For instance, protocol_http becomes pixel value 255 at the appropriate grayscale image location (row=3, column = 13) if the attack used hypertext transfer protocol. Conversely, the same pixel maps to 0 if the protocol was not used. Each row of the UNSW-NB15 thus renders 256 features (of which 194 follow directly from the tabular set).

Training	
Attack	45332
Normal	37000
	82332
Testing	
Attack	119341
Normal	56000
	175341

Family	Count
Analysis	677
Backdoor	583
DoS	4089
Exploits	11132
Fuzzers	6062
Generic	18871
Normal	37000
Reconnaissance	3496
Shellcode	378
Worms	44
Grand Total	**82332**

Figure 3. Training and testing count per attack family count.

We leave unchanged the train/test split of the original UNSW-NB15 dataset at a 1:2 ratio [1-4]. The detailed counts for each class are shown in Figure 3. It is worth noting that the UNSW-NB15 dataset updates and statistically rebalances some of the KDD99 counts based on their analysis of duplicates and potential data leakage between training and test sites. The ratio of 1:2 for training and test presents a challenging amount of previously unseen data when an algorithm gets scored or deployed. In total, we created almost a quarter-million images as 256-pixel thumbnails using ImageMagick [11]. The largest training class (normal traffic: 37,000) outnumbers the smallest attack class (worms: 44) by nearly 1000:1 as a ratio of cases.

To explore whether transfer learning from a convolutional neural network can identify network attacks, we tested the small (2 Mb) MobileNetV2 model [13] as pre-trained, then introduced both the binary and multi-class problems. The binary classifier determines whether a given image pattern represents normal or attack traffic. The multi-class problem identifies one of the 10 possible families (9 attacks in Figure 3 vs. normal).

The multi-class example shares an analogous data setup to the traditional MNIST handwriting dataset [14] and thus may benefit from the various state-of-the-art approaches developed to

handle those 10 classes. We solve both the binary and multi-class cases with a standard set of hyper-parameters (epochs:50, batch size:16, learning rate: 0.001). Slower learning rates disrupt the pre-trained layers of the neural network and preserve some of its beneficial weights for feature extraction in the images. We also explore the effects of smaller dataset size (<10,000 training examples vs. the full 250k) [22].

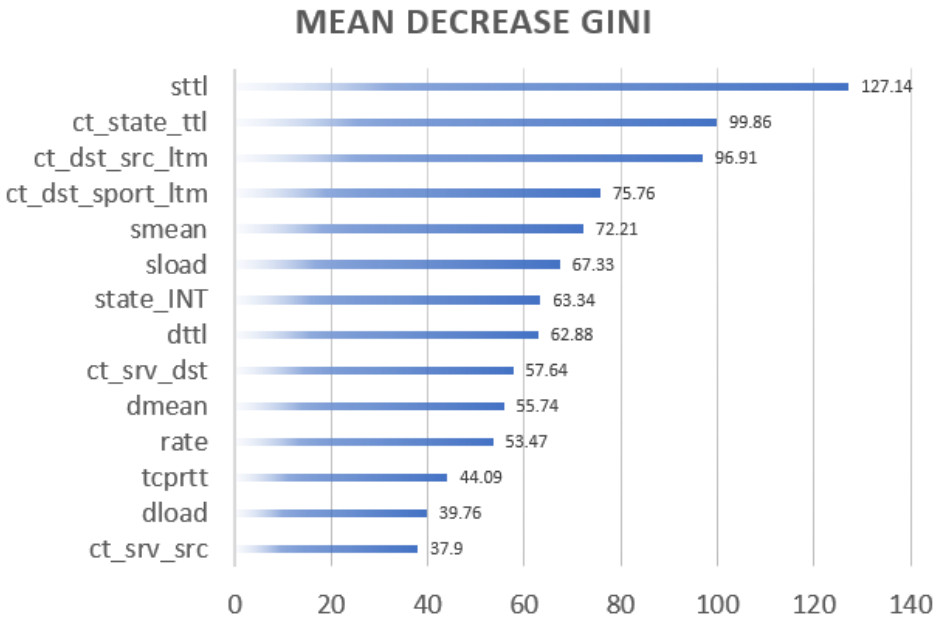

Figure 4. Feature information (GINI) Contribution to Attack Detection.

To rank the feature importance for detecting attacks, we applied a random forest algorithm (Figure 4) to the binary classifier [23]. The descending order for the top 14 contributors is shown using the Gini Index [23] (or impurity) which effectively gauges the factors contributing to a decision split between normal and attack. The highest contributors include 1) the "Source to destination Time To Live" value (sttl); 2) Number for each state (dependent protocol, e.g. ACC, CLO, CON) according to a specific range of values for source/destination Time To Live (ct_state_ttl); and 3) Number of connections of the same source and the destination address in 100 connections according to the last time (ct_dst_src_ltm). Not shown in Figure 4 are the least important which somewhat surprisingly include most of the one-hot-encoded protocol features that are more exotic than ordinary TCP (e.g. zero, XTP, XMN.IDP, WSN, etc.). In addition to providing a future path to reduce the intrusion detector's dimensionality, this feature importance rank defines what cannot be safely ignored in attack datasets like UNSW-NB15 [1-4].

Family	Accuracy	Test
Normal	98%	500
Attack	97%	542

Figure 5. Binary classifier results.

2.2. Anti-Virus Dataset

Recent interest in applying the same image-based classification techniques to anti-virus and malware detectors [24-30] motivates the present work to score a similar formatted problem and compare the algorithmic performance with existing methods. Intel Labs and Microsoft Threat Protection Intelligence Team recently launched their static malware collaboration called STAMINA: Scalable Deep Learning Approach for Malware Classification [31]. The present contribution is to reformulate the malware-image problem as a familiar MNIST variant, to generate the 9-virus clusters based on byte-similarities, and then to identify the virus family based on a grey-scale thumbnail image (32 x 32). Figure 6 shows the abstract images derived for each of the 10 classes, with "0" as the only one that is non-malicious.

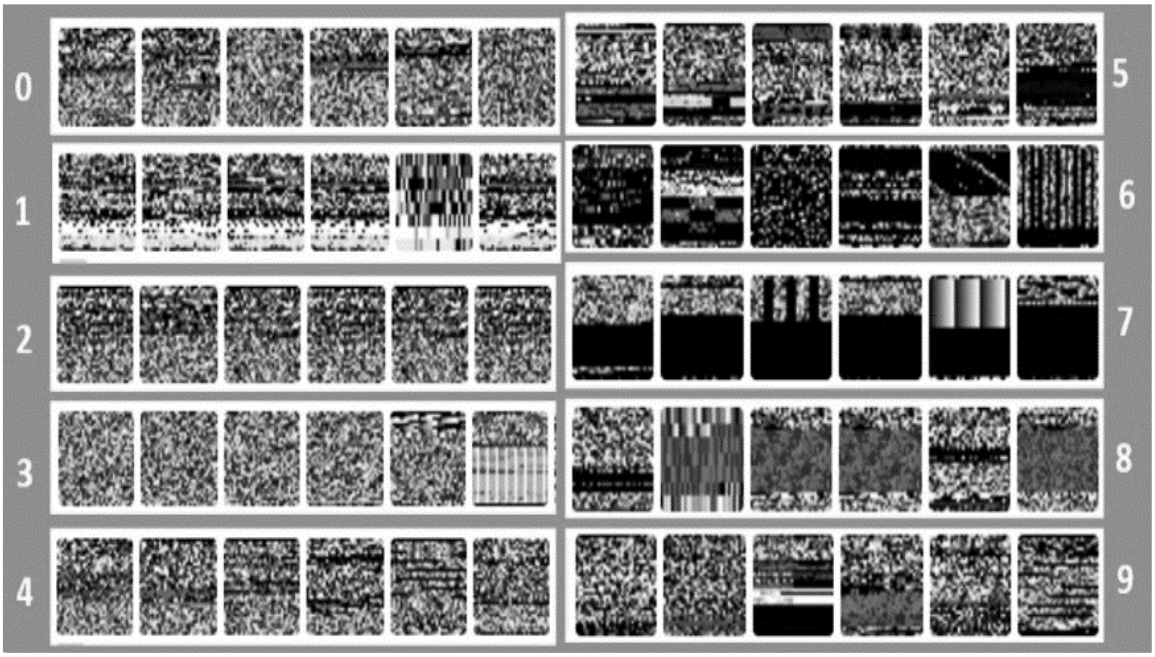

Figure 6. Virus MNIST showing 10 classes. The "0" class represents non malicious examples. The other 9 virus families were clustered using a K means method to match with the standard MNIST format and multi class solutions.

We explore the malware dataset first combined [27] as bulk virus downloads (from virusshare.com) and non-malicious examples (from portableapps.com). Microsoft documents the format header for Portable Executable (PE) files [32-33] and importable python libraries ("pefile", [34]) exist as convenient extraction tools from compiled and executable code. The Portable Executable dataset [27] contains 51,880 examples of the first 1024 bytes (32x32 pixel values) of the header. As comma-separated-values, the entire file's MD5 hash augments the information available for each example and provides enough identifying information to trace its operating features using community-supported repositories like VirusTotal.com.

Class	Count	Group	Type	Example
0	2516	Beneware	Good	putty.exe
1	7684	Malware	Adware	IESettings
2	3037	Malware	Trojan	Supreme.exe
3	2404	Malware	Trojan	myfile.exe
4	796	Malware	Installer	myfile.exe
5	6662	Malware	Backdoor	myfile.exe
6	15377	Malware	Crypto	Powershell
7	7494	Malware	Backdoor	BitTorrent.exe
8	2571	Malware	Downloader	myfile.exe
9	3339	Malware	Heuristic	myfile.exe

Figure 7. Class distributions and example types for malware and "beneware" PE File headers.

Previous solutions have designated a binary class label as either malware or not (e.g. benign programs or "beneware") [27]. If formulated as a two-class problem, the imbalanced ratio of malware to "beneware" equals approximately 20:1 (49,364:2516). Without rebalancing this ratio, a 95% correct solution would simply declare all cases as malware. To recast the image dataset as an MNIST-formatted alternative, we performed a standard cluster analysis using the KMeans algorithm [35]; we assigned cluster numbers equal to 9 based on the 1024 column byte vector. We exclude the identifying file hash. We attempted to assign dominant families to each derived cluster based on the known MD5 hashes but found multiple names and sample diversity when querying VirusTotal. To reduce the number of code changes comparing a larger or smaller multi-class problem with existing MNIST infrastructure, we opted to use the assigned 9-cluster result as a fully unsupervised example. Figure 7 summarizes the class distribution following KMeans clustering for the malware class only and the 9 resulting virus families. By testing 10 or more MD5 hash values against the VirusTotal.com database, we assigned the broad types and example executable names. These choices showed sufficient diversity and overlap that the designations provide only representative choices. The outcome proves more statistical and less operational for malware behavior.

We converted the CSV format [23] to greyscale images using the intermediate NetPBM text format (PGM) to create ASCII-raw images (Figure 6), then the ImageMagick [11] command-line tools for compressing the image to viewable JPEG files. We split the resulting 51,880 thumbnails into the same 85:15 ratio used by MNIST training and testing bins, such that the unseen test images help validate any algorithm's ability to generalize from the training images. The choice of 32x32 pixels to represent the PE header proves useful for later deep learning algorithms that depend on powers of 2 (in stride length) to form their convolutional layers. This differs slightly from the traditional (arbitrary) 28x28 pixel thumbnails in most MNIST variants [37]; the differences are cosmetic only for most algorithms other than deep learning ones. Three different formats are provided for download, including train and test sets as comma-separated values files, JPEG images sorted by class (10 total), and the original MNIST binary format (idx-ubyte) [14, 36]. These three formatting options should cover most all published MNIST solutions with only minor modifications.

3. RESULTS

The modeling of two separate cybersecurity-related datasets with image classifiers highlights future insertion points for including deep learning and more specifically pre-trained neural

networks like MobileNetV2 [13] for minimizing data requirements and transfer learning between unrelated tasks.

3.1. Network Intrusion Defense

As shown in Figures 8-10, the results for transfer learning with a deep convolutional neural network (MobileNetV2 [13]) demonstrate that the image-based binary classifier achieves greater than 97% accuracy in identifying whether an attack occurs (Figure 5).

Family	Accuracy	Test	Accuracy	Test	Family	Accuracy	Test	Accuracy	Test
Analysis	19%	102	21%	39	Generic	96%	226		0
Backdoor	69%	88	44%	41	Normal	98%	473	67%	36
DoS	34%	161	54%	41	Reconn	63%	176	5%	44
Exploits	66%	264	44%	50	Shellcode	21%	57	79%	42
Fuzzers	70%	204	82%	39	Worms	0%	7	14%	7

Figure 8. Multi-class results.

In Figure 8, the specific identification of an attack family averages 54% accuracy between the 10 classes, with large deviations between the best (normal and generic traffic: 96+%) and the worst (worms and analysis <19%). Figures 8-9 show the error matrix of which attack images confuse the neural network (worms misclassed as exploits). It is worth noting that the majority class (normal) loses no performance as more attack classes get added from the binary to the multi-classification example.

One contribution to this variance is the relative sparsity of UNSW-NB15 examples for the lower performing classes. To test this hypothesis, we performed the same experiment on a smaller subset (<4000) of images with a hold-out test and validation set that represents 20% of the training set (as opposed to 200% in the original UNSW-NB15 split). By better balancing, the dataset, undetectable worms, and other lesser represented classes could be detected in Figure 8 (second column). Mapping attacks to images shows the dependence of accuracy on both class size and imbalance [23].

One interesting outcome of using these image-based detection maps is their portability to small hardware appliances. The small MobileNetV2 architecture [13] is tailored to run on edge devices [37], such as mobile phones. Simple network detectors thus render a complex matrix of packet features into a rapid classifier capable of running in near real-time imagery (e.g. 30 frames -or attacks- per second). The reduction of the model to use tflite [38] (Tensorflow) as a set of stored weights represents a standard model [37] for deploying deep learning to edge devices.

	Normal	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Reconn	Shellcode	Worms
Normal	98%	0%	0%	0%	1%	1%	0%	0%	0%	0%
Analysis	0%	19%	51%	10%	19%	2%	0%	0%	0%	0%
Backdoor	0%	14%	69%	5%	9%	1%	0%	2%	0%	0%
DoS	0%	2%	0%	34%	59%	2%	0%	2%	1%	0%
Exploits	1%	0%	1%	19%	66%	3%	0%	9%	0%	0%
Fuzzers	0%	0%	3%	8%	9%	70%	0%	8%	0%	0%
Generic	0%	0%	0%	0%	2%	0%	96%	0%	0%	0%
Reconn	0%	1%	1%	14%	17%	3%	0%	63%	2%	0%
Shellcode	0%	0%	0%	0%	0%	5%	0%	74%	21%	0%
Worms	0%	0%	0%	0%	86%	14%	0%	0%	0%	0%

Figure 9. Confusion matrix by class for network intrusion data.

For the binary classifier distinguishing attacks from normal traffic, the current results (accuracy = 97%) compare to previous work using statistical machine learning [39], such as support vector machines (SVM, accuracy = 92.28%), Naïve Bayes (NB, accuracy = 74.19%), decision tree (simple, accuracy = 95.82%), and random forest (RF, accuracy = 97.49%). For a smaller UNSW-NB15 subset [40], deep learning approaches [41] reported 91.4% (normal) accuracy and 99.2% (attack). Using manual feature selection

Experiment	Classifier	Overall Accuracy	Analysis	Backdoor	DoS	Exploits	Fuzzers	Generic	Normal	Reconnaissance	Shellcode	Worms
1	K Nearest Neighbours	70%	14%	42%	17%	47%	26%	92%	81%	57%	56%	41%
2	Logistic Regression	63%	66%	18%	0%	38%	5%	85%	81%	9%	92%	82%
3	Random Forest	76%	15%	38%	12%	50%	15%	96%	92%	82%	86%	59%
4	XGBoost	76%	38%	45%	22%	48%	13%	95%	92%	82%	89%	86%
5	Decision Tree	76%	1%	31%	43%	44%	11%	95%	92%	82%	85%	73%
6	Neural Network	74%	0%	81%	8%	41%	16%	95%	91%	84%	69%	75%
7	OneVsRest - XGBoost	77%	43%	38%	24%	49%	13%	95%	92%	83%	90%	84%
8	OneVsRest - XGBoost (Balanced)	76%	22%	58%	11%	47%	13%	95%	92%	82%	89%	86%
9	OneVsRest - Neural Network	76%	36%	40%	14%	48%	12%	95%	92%	79%	82%	57%
10	OneVsRest - Neural Network (Balanced)	75%	14%	54%	17%	48%	12%	94%	91%	70%	77%	75%
11	OVR - XGBoost (With class-wise K best features)	76%	4%	63%	28%	45%	10%	94%	92%	78%	87%	75%
12	OVR - XGBoost (With num/binary features)	77%	43%	38%	24%	49%	13%	95%	92%	83%	90%	84%
13	XGBoost (With num/binary features)	76%	38%	45%	22%	48%	13%	95%	92%	82%	89%	86%
14	Current Results	54%	19%	69%	34%	66%	70%	96%	98%	63%	21%	0%

Figure 10. Comparison between previous statistic al solutions and the present result using image based classifiers and transfer learning with MobileNetV2 deep neural network.

[41], a larger deep learning approach n the tabular data reported 98% (normal) accuracy and 85% (attack). In summary, the original contribution of combining an image-based transfer learning method to traditional tabular data offers a promising detector for network defense, particularly for 97+% accuracy to distinguish normal and malicious traffic connections.

For the refined determination of the attack family, Figure 10 compares the present results with 13 other popular machine learning approaches [42] as shown by class type and accuracy. While the performance of the image-based approach to collecting features lags other methods in some under-represented attack families, the high accuracy for normal traffic (98%) vs generic attacks

(96%) offers a viable intrusion detection for the multi-class problem. Since Figure 8 demonstrates the accuracy dependence on sample size, the lower accuracy seen in under-represented classes like worms, analysis, and shellcode would benefit from supplemental or synthetic attack data.

3.2. Virus Detection

Figure 11 shows the accuracy per class in three different CNN models. The highest accuracy across all classes averages 80%.

Class	Count	Group	Type	Acc-1	Acc-2	Acc-3
0	2516	Beneware	Good	58.0%	39.0%	22.0%
1	7684	Malware	Adware	100.0%	99.0%	99.0%
2	3037	Malware	Trojan	81.0%	84.0%	91.0%
3	2404	Malware	Trojan	88.0%	90.0%	94.0%
4	796	Malware	Installer	100.0%	100.0%	100.0%
5	6662	Malware	Backdoor	75.0%	84.0%	83.0%
6	15377	Malware	Crypto	75.0%	64.0%	89.0%
7	7494	Malware	Backdoor	51.0%	72.0%	72.0%
8	2571	Malware	Downloader	68.0%	66.0%	76.0%
9	3339	Malware	Heuristic	70.0%	89.0%	74.0%

Figure 11. Accuracy per class. Acc-1 is 20% sampling, Acc-3 is 100% sampling and Acc-2 is slow learning rates.

To explore the effects of dataset size, we modeled a faster 20% sampling (Acc-1) with a learning rate of 0.001, batch size=16, and 50 epochs. A second model (Acc-2) featured a slower learning rate (0.0005) over longer training times (100 epochs). A slower learning rate can avoid disruptive steps when transfer learning from a pre-trained network like MobileNetV2. A final model (Acc-3) included 100% samples (the full 51,880 train and test sets) and a faster learning rate (0.001) and time (50 epochs). Across all 10 classes, the average accuracy varied less than 2% for the three cases but did peak at 80% for the larger dataset (Acc-3). The three models together show the highest false-negative rate for malware is class “0” or benign executables that get flagged as potentially malicious. This behavior may reflect the inexactness of the PE header as an indicator of malware, or the hijacking of benign header characteristics to disguise malware in the first 1024 bytes.

		Prediction									
		0	1	2	3	4	5	6	7	8	9
Actual Class	0	0.59	0.02	0.03	0.01	0.00	0.06	0.10	0.11	0.04	0.05
	1	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	2	0.11	0.00	0.81	0.02	0.00	0.04	0.00	0.02	0.01	0.00
	3	0.03	0.00	0.02	0.88	0.00	0.00	0.01	0.05	0.00	0.00
	4	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
	5	0.14	0.00	0.01	0.00	0.00	0.75	0.02	0.02	0.03	0.02
	6	0.16	0.00	0.00	0.00	0.00	0.02	0.75	0.01	0.02	0.03
	7	0.38	0.00	0.01	0.01	0.00	0.02	0.05	0.51	0.01	0.01
	8	0.18	0.00	0.01	0.00	0.00	0.02	0.05	0.03	0.68	0.05
	9	0.19	0.00	0.00	0.00	0.00	0.01	0.06	0.01	0.02	0.70

Figure 12. Error matrix highlighting the classes across the rows that each actual case most often gets mistaken for. The diagonal shows the correct proportion.

Figure 12 shows the error matrix for the first model (Acc-1), which highlights that the class of benign programs is most often mistake for backdoors (7), heuristic (9), and downloader (8). It's worth noting that particularly for a downloader that reaches out to a command and control malware site, this byte-code signature may prove malicious or benign since it depends on the website itself; a benign program may reach out to download an update from Microsoft on launch. The higher false-negative rates may also originate in the KMeans approach, given the clustering excluded the non-malicious cases to simplify the generation of 9 malware families as independent groups from good, benign, or normal software called here as "beneware". Further investigation may explore whether other clustering approaches benefit the class distinctions. For instance, density-based clustering (DBSCAN) optimizes the tight groups with minimal overlap. A further enhancement would use the MD5 hash to determine virus family and avoid clustering altogether as an alternative which also prevents the CNN from simply modeling the unsupervised (KMeans) algorithm itself rather than the natural distribution of malware images.

4. DISCUSSION AND CONCLUSIONS

The results demonstrate a viable path of converting tabular feature data to image thumbnails, then applying convolutional neural networks to classify attack families. One potential shortcoming in our approach is any dependencies on the parametric ordering in the table format. For instance, convolutional neural networks tend to highlight close neighbors as being related in the image [43], yet there is no obvious relationship in the generated images between protocols, services, or states that justify making them into a particular attack fingerprint. One could address this flaw quantitatively by shuffling the order and determining the change of accuracy (if any). Future work should compare alternative statistical methods borrowed from the extensive machine learning literature devoted to the MNIST (and its derivative [14-21]) dataset of handwriting recognition. One can anticipate that like MNIST solutions, there exist high accuracy decision trees (like extreme gradient boosted trees – XGBoost [44]) that generate both accuracy and inference speeds comparable to the deep learning approach here. Further work could also use the image dataset [45] to design new attacks (and defenses) based on the techniques of generative adversarial networks (GANs [46]). The network intrusion dataset is available on Kaggle [10].

To address the generality of this approach, we recast a second and consolidated dataset for scoring malware Portable Executable file headers as an abstract image recognition problem. The image-based approach may generalize better than other heuristic methods, particularly given how virus authors change single-bytes to fool hash-based signatures but CNN detection typically proves less sensitive to small image changes. The ability to read headers only and render a readable file fingerprint as a small image suggests a potential fast detection rate seen in other vision applications (near real-time if greater than 30 frames per second by convention). Further enhancements may improve accuracy with data augmentation [45] strategies. While the reduction of file headers to images may initially seem counterintuitive, the abstract representation of text or audio into images has benefited other machine learning applications. We anticipate that further work may render simple, standalone appliances that may operate offline for successful detection, particularly for the small, fast, and accurate models like MobileNetV2. The Virus-MNIST dataset is also available on Kaggle and Github [47].

ACKNOWLEDGMENTS

The author would like to thank the PeopleTec Technical Fellows program for encouragement and project assistance.

REFERENCES

[1] Moustafa, Nour, and Jill Slay. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)." Military Communications and Information Systems Conference (MilCIS), 2015. IEEE, 2015. See online https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/

[2] Moustafa, Nour, and Jill Slay. "The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 dataset and the comparison with the KDD99 dataset." Information Security Journal: A Global Perspective (2016): 1-14.

[3] Moustafa, Nour, et al. . "Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks." IEEE Transactions on Big Data (2017).

[4] Moustafa, Nour, et al. "Big data analytics for intrusion detection system: statistical decision-making using finite Dirichlet mixture models." Data Analytics and Decision Support for Cybersecurity. Springer, Cham, 2017. 127-156.

[5] Özgür, Atilla, and Hamit Erdem. "A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015." PeerJ Preprints 4 (2016): e1954v1.

[6] Olusola, A. A., Oladele, A. S., & Abosede, D. O. (2010, October). Analysis of KDD'99 intrusion detection dataset for selection of relevance features. In Proceedings of the world congress on engineering and computer science (Vol. 1, pp. 20-22). WCECS.

[7] Meena, Gaurav, and Ravi Raj Choudhary. "A review paper on IDS classification using KDD 99 and NSL KDD dataset in WEKA." In 2017 International Conference on Computer, Communications and Electronics (Comptelix), pp. 553-558. IEEE, 2017.

[8] Hatami, Nima, Yann Gavet, and Johan Debayle. "Classification of time-series images using deep convolutional neural networks." In Tenth international conference on machine vision (ICMV 2017), vol. 10696, p. 106960Y. International Society for Optics and Photonics, 2018.

[9] Hershey, Shawn, Sourish Chaudhuri, Daniel PW Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal et al. "CNN architectures for large-scale audio classification." In 2017 IEEE international conference on acoustics, speech and signal processing (ICASSP), pp. 131-135. IEEE, 2017.

[10] Noever, David "Intrusion Detection as an Image Classifier", Kaggle.com, (2021), https://www.kaggle.com/datamunge/intrusion-detection-as-an-image-classifier

[11] Salehi, Sohail. ImageMagick Tricks. Packt publishing ltd, 2006.

[12] Zhang, Weinan, Tianming Du, and Jun Wang. "Deep learning over multi-field categorical data." In European conference on information retrieval, pp. 45-57. Springer, Cham, 2016.

[13] Sandler, Mark, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. "Mobilenetv2: Inverted residuals and linear bottlenecks." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4510-4520. 2018.

[14] LeCun, Yann, Corinna Cortes, and C. J. Burges. "MNIST handwritten digit database." (2010): 18. http://yann.lecun.com/exdb/mnist/ and Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11):2278-2324, November 1998

[15] Cohen, Gregory, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. "EMNIST: Extending MNIST to handwritten letters." In 2017 International Joint Conference on Neural Networks (IJCNN), pp. 2921-2926. IEEE, 2017.

[16] Chen, Li, Song Wang, Wei Fan, Jun Sun, and Satoshi Naoi. "Beyond human recognition: A CNN-based framework for handwritten character recognition." In 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR), pp. 695-699. IEEE, 2015.

[17] Image Classification on MNIST, (accessed 01/2021), https://paperswithcode.com/sota/image-classification-on-mnist

[18] Grim, Jırı, and Petr Somol. "A Statistical Review of the MNIST Benchmark Data Problem." http://library.utia.cas.cz/separaty/2018/RO/grim-0497831.pdf

[19] Preda, Gabriel, Chinese MNIST: Chinese Numbers Handwritten Characters Images, (accessed 01/2021) https://www.kaggle.com/gpreda/chinese-mnist

[20] CoMNIST: Cyrillic-oriented MNIST, A Dataset of Latin and Cyrillic Letters, (accessed 01/2021) https://www.kaggle.com/gregvial/comnist

[21] Prabhu, Vinay Uday. "Kannada-MNIST: A new handwritten digits dataset for the Kannada language." arXiv preprint arXiv:1908.01242 (2019). https://www.kaggle.com/higgstachyon/kannada-mnist
[22] Warden, P. "How many images do you need to train a neural network?" (2017). https://petewarden.com/2017/12/14/how-many-images-do-you-need-to-train-a-neural-network/
[23] Han, Hong, Xiaoling Guo, and Hua Yu. "Variable selection using mean decrease accuracy and mean decrease Gini based on random forest." In 2016 7th IEEE International Conference On Software Engineering And Service Science (ICSESS), pp. 219-224. IEEE, 2016.
[24] Anderson, H. S., & Roth, P. (2018). Ember: an open dataset for training static PE malware machine learning models. arXiv preprint arXiv:1804.04637.
[25] Manavi, F., & Hamzeh, A. (2020, September). A New Method for Ransomware Detection Based on PE Header Using Convolutional Neural Networks. In 2020 17th International ISC Conference on Information Security and Cryptology (ISCISC) (pp. 82-87). IEEE.
[26] Vasan, D., Alazab, M., Wassan, S., Safaei, B., & Zheng, Q. (2020). Image-Based malware classification using an ensemble of CNN architectures (IMCEC). Computers & Security, 92, 101748.
[27] Oliveira, Angelo (2019). Malware Analysis Datasets: Raw PE as Image. IEEE Dataport. https://dx.doi.org/10.21227/8brp-j220, https://ieee-dataport.org/open-access/malware-analysis-datasets-raw-pe-image and Kaggle, https://www.kaggle.com/ang3loliveira/malware-analysis-datasets-pe-section-headers
[28] Oliveira, Angelo (2019). Malware Analysis Datasets: PE Section Headers. IEEE Dataport. https://dx.doi.org/10.21227/2czh-es14 , https://ieee-dataport.org/open-access/malware-analysis-datasets-pe-section-headers and Kaggle, https://www.kaggle.com/ang3loliveira/malware-analysis-datasets-raw-pe-as-image
[29] Oliveira, Angelo (2019). Malware Analysis Datasets: Top-1000 PE Imports. IEEE Dataport. https://dx.doi.org/10.21227/004e-v304 , https://ieee-dataport.org/open-access/malware-analysis-datasets-top-1000-pe-imports and Kaggle, https://www.kaggle.com/ang3loliveira/malware-analysis-datasets-top1000-pe-imports
[30] Freitas, S., Duggal, R., & Chau, D. H. (2021). MalNet: A Large-Scale Cybersecurity Image Database of Malicious Software. arXiv preprint arXiv:2102.01072.
[31] Chen, L., Sahita, R., Parikh, J., Marino, M. (2020), "STAMINA: Scalable Deep Learning Approach for Malware Classification," Intel Labs Whitepaper, https://www.intel.com/content/www/us/en/artificial-intelligence/documents/stamina-deep-learning-for-malware-protection-whitepaper.html
[32] Microsoft, "PE Format", https://docs.microsoft.com/en-us/windows/win32/debug/pe-format , accessed online (Jan 2021)
[33] InfoSec Institute, "Demystifying PE File", https://resources.infosecinstitute.com/topic/2-malware-researchers-handbook-demystifying-pe-file/ , accessed online (Jan 2021)
[34] pefile, "Python PE parsing module", https://pypi.org/project/pefile/ accessed online (Jan 2021) and description of uses, https://malwology.com/2018/08/24/python-for-malware-analysis-getting-started/
[35] Hartigan, J. A. (1985). Statistical theory in clustering. Journal of classification, 2(1), 63-76.
[36] Lu, Arlen, "Convert-own-data-to-MNIST-format" (accessed 01/2021) https://github.com/Arlen0615/Convert-own-data-to-MNIST-format
[37] Lee, Juhyun, Nikolay Chirkov, Ekaterina Ignasheva, Yury Pisarchyk, Mogan Shieh, Fabio Riccardi, Raman Sarokin, Andrei Kulik, and Matthias Grundmann. "On-Device Augmented Reality with Mobile GPUs."
[38] Shah, Vishal, and Neha Sajnani. "Multi-Class Image Classification using CNN and Tflite." International Journal of Research in Engineering, Science and Management 3, no. 11 (2020): 65-68.
[39] Belouch, Mustapha, Salah El Hadaj, and Mohamed Idhammad. "Performance evaluation of intrusion detection based on machine learning using Apache Spark." Procedia Computer Science 127 (2018): 1-6.
[40] Choudhary, Sarika, and Nishtha Kesswani. "Analysis of KDD-Cup'99, NSL-KDD and UNSW-NB15 Datasets using Deep Learning in IoT." Procedia Computer Science 167 (2020): 1561-1573.
[41] Kanimozhi, V., and Prem Jacob. "UNSW-NB15 dataset feature selection and network intrusion detection using deep learning." International Journal of Recent Technology and Engineering 7: 443-446.
[42] Pujari, Rakshit, "Network Attack Detection and Classification Using Machine Learning Models Based on UNSW-NB15 Data-Set", Medium, (10/2020). https://i-rakshitpujari.medium.com/network-

attack-detection-and-classification-using-machine-learning-models-based-on-unsw-nb15-a645bba73987

[43] Liu, Li, Jie Chen, Paul Fieguth, Guoying Zhao, Rama Chellappa, and Matti Pietikäinen. "From BoW to CNN: Two decades of texture representation for texture classification." International Journal of Computer Vision 127, no. 1 (2019): 74-109.

[44] Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." In Proceedings of the 22nd ACM Sigkdd International Conference On Knowledge Discovery And Data Mining, pp. 785-794. 2016.

[45] Shorten, Connor, and Taghi M. Khoshgoftaar. "A survey on image data augmentation for deep learning." Journal of Big Data 6, no. 1 (2019): 1-48.

[46] Samangouei, Pouya, Maya Kabkab, and Rama Chellappa. "Defense-GAN: Protecting classifiers against adversarial attacks using generative models." arXiv preprint arXiv:1805.06605 (2018).

[47] Noever, D. and Noever, Samantha E. Miller, “Virus-MNIST: Portable Executable Files as Images for Malware Detection”, https://www.kaggle.com/datamunge/virusmnist and https://github.com/reveondivad/virus-mnist