Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2015) 000–000

SAND2015-7908C

Procedia Computer Science

www.elsevier.com/locate/procedia

Complex Adaptive Systems, Publication 5
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2015-San Jose, CA

A signal processing approach for cyber data classification with deep neural networks

Jonathan A. Cox *, Conrad D. James, James B. Aimone

Sandia National Laboratories, 1515 Eubank Blvd. SE, Albuquerque, NM, 87123, USA

Abstract

Recent cyber security events have demonstrated the need for algorithms that adapt to the rapidly evolving threat landscape of complex network systems. In particular, human analysts often fail to identify data exfiltration when it is encrypted or disguised as innocuous data. Signature-based approaches for identifying data types are easily fooled and analysts can only investigate a small fraction of network events. However, neural networks can learn to identify subtle patterns in a suitably chosen input space. To this end, we have developed a signal processing approach for classifying data files which readily adapts to new data formats. We evaluate the performance for three input spaces consisting of the power spectral density, byte probability distribution and sliding-window entropy of the byte sequence in a file. By combining all three, we trained a deep neural network to discriminate amongst nine common data types found on the Internet with 97.4% accuracy.

Keywords: Neural networks; Classification; Cyber security; Signal processing; Deep learning

1. Introduction

Cyber security has become a significant concern for individuals, corporations and governments. Existing technology often relies on protocols and procedures for preventing intrusion and signature-based approaches for identifying malicious activity. Since cyber adversaries are continually adapting and improving their techniques in an effort to evade detection, identification is a complex and challenging task. For this reason, human analysts are often needed to identify sophisticated attacks. However, these approaches have failed to stem the rise in cyber attacks, in

* Corresponding author. Tel.: +1-505-284-6114; fax: +1-505-845-7065.
E-mail address: joncox@alum.mit.edu

part because they either cannot anticipate new threats or do not scale efficiently as the volume of data and attacks increases[1,2].

To combat this trend, we are investigating neuro-inspired algorithms for automating tasks that analysts must perform today. In particular, analysts are often presented with a large variety of data for further inspection. Often, it is not clear what type of data is being presented or if the existing signature-based tools accurately describe the true nature of the data.

Much of the prior work has been concerned with classifying malware or executables using entropy or other statistical features such as histograms of n-grams[3,4]. Other works have considered entropy of various files, although, we find it does not offer good performance for multi-class discrimination[5]. For instance, many compressed or encrypted file formats have similarly high entropy. Support Vector Machine classifiers, among others, require a large number of carefully hand engineered features to be developed[6]. In contrast, deep learning algorithms are able to learn suitable feature representations with considerably less hand engineering[7].

In this paper, we discuss a method for identifying data types which uses information theoretic, statistical and signal processing representations of the data. These representations are agnostic to the underlying data format and so are readily adapted to new data types in real-time. We compare the performance of a deep neural network classifier when trained on these representations and evaluate the performance for every combination of the three representations.

2. Method

In recent years, neural networks with multiple hidden layers have enjoyed considerable success in learning feature representations from the data for classification and compression[8]. Since these non-dynamical, feedforward networks have a fixed length input, additional pre-processing is often necessary for unstructured data. Therefore, arbitrary length files are transformed into three different fixed-length representations which capture some of the unique patterns of different types of data.

2.1. Representations

The first representation we consider is the Shannon entropy within a sliding window scanned through the data[5]. The x_i in (1) are the unique symbols, which range from 0 to 255 in this case. The byte probabilities are computed from a window with a length of 256 bytes, denoted by $P(x_i,t)$ where t is the window position. At each step through the data, the windows are overlapped by 50%. The entropy sequence resulting from the total number of window positions is cubically interpolated onto a uniform grid of 256 points to give a fixed length sequence. Therefore, (1) preserves location by giving a perspective of the entropy throughout the file. The two remaining methods are computed over the entire file.

$$H(X,t) = -\sum_{i=0}^{255} P(x_i,t)\log_2 P(x_i,t) \tag{1}$$

Another approach we consider is to treat the byte sequence as a signal (with amplitude ranging from 0 to 255), and transform it into frequency space. A similar technique has shown success characterizing sections of DNA in bioinformatics[9]. While the Fourier transform of the entropy has been considered for identifying malware[5], entropy alone is too low dimensional a representation to capture the full complexity of data, since it cannot be uniquely transformed to the original sequence. On the other hand, the complex Fourier transform is invertible and exactly captures the data sequence. Nevertheless, for simplicity and speed, we compute the power spectral density from a spectrogram with 256 positive frequency bins. The entire spectrogram is then summed over the time axis to compute the frequency marginal, as given by (2). Here, x is the sequence of byte values (0 to 255) in the data, and w is a rectangular window function. The summation over time collapses it into a single spectrum with a fixed number of frequency bins. Finally, the area of the spectrum is normalized to unit value, since the total power would otherwise be determined by the file length. If this normalization is not performed, the classifier can learn to discriminate based partially upon file length.

$$S[\omega] \approx \sum_t \left\{ \left| \sum_n x[n] w[n-t] e^{-j\omega n} \right|^2 \right\} \tag{2}$$

Finally, we consider a histogram of the bytes in the file, which gives a distribution over byte value (3). This representation yields the probability distribution over all symbols, which is akin to $P(x_i,t)$ in (1). However, (3) is the distribution over the entire file—not within a sliding window. This is accomplished by counting the occurrences of each byte symbol (0 to 255) and normalizing by the total number of bytes, *N*. Normalization gives an estimate of the probability distribution of bytes and prevents the classifier from learning features based on the number of bytes in the file. Examples of each representation for each class of file considered are presented in the next section.

$$P(x_i) = \frac{\text{hist}(x_i)}{N} \tag{3}$$

2.2. Classifier

To classify the data into one of nine file types, we use a feedforward neural network with two hidden layers, as shown in Fig. 1. The hidden layer activation functions are hyperbolic tangent, while the output activation is a softmax, which can be interpreted as a probability distribution over classes. Training is conducted with mini-batches of 200 examples which are randomly chosen from 4500 files (500 per class) with replacement. We use ADADELTA gradient descent to optimize the network's parameters (weights and biases) since it has little sensitivity to hyperparameters such as learning rate[10]. This makes it simpler to compare different networks and input representations.

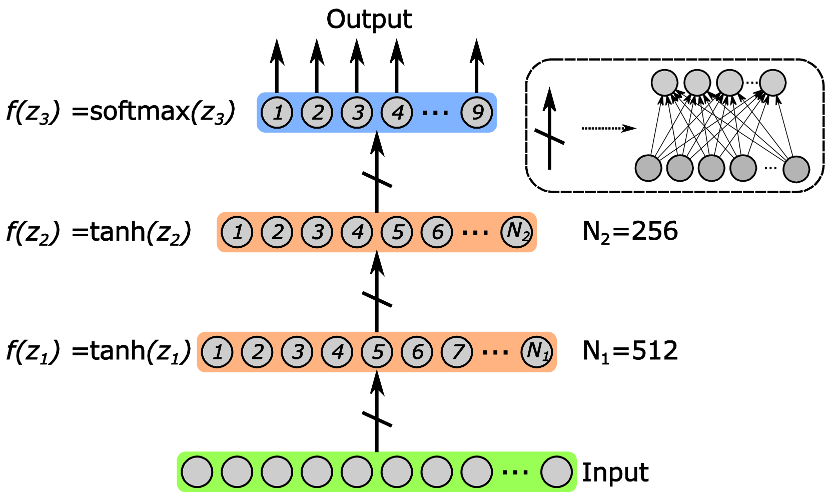

Fig. 1. A diagram of the feed forward neural network architecture used for classification. The hidden layers have hyperbolic tangent activation functions. The output layer has a softmax activation. All nodes are fully connected, as shown in the upper right corner.

To improve performance of the deep network, we first pre-train each hidden layer as a stacked denoising autoencoder, which is similar to pre-training a Deep Belief Network with contrastive divergence[7,11]. The final layer received supervised pre-training[8]. Pre-training alone is sufficient to achieve roughly 15-25% error on the test set.

During fine tuning of the entire network, we use denoising and dropout to prevent overfitting, which otherwise compromises performance. Nodes in the hidden layers are randomly dropped out (activation functions are zeroed) with 40% probability[12]. Denoising of the input layer randomly sets each input to zero with 25% probability. These techniques force the network to learn more general features, since they cannot count on any given feature being present at any layer. We developed our deep learning code in Matlab, which is also compatible with GNU Octave. It supports efficient GPU acceleration which we are exploring for online training and classification of streaming data. Presently, we are working to release this code under an open source license.

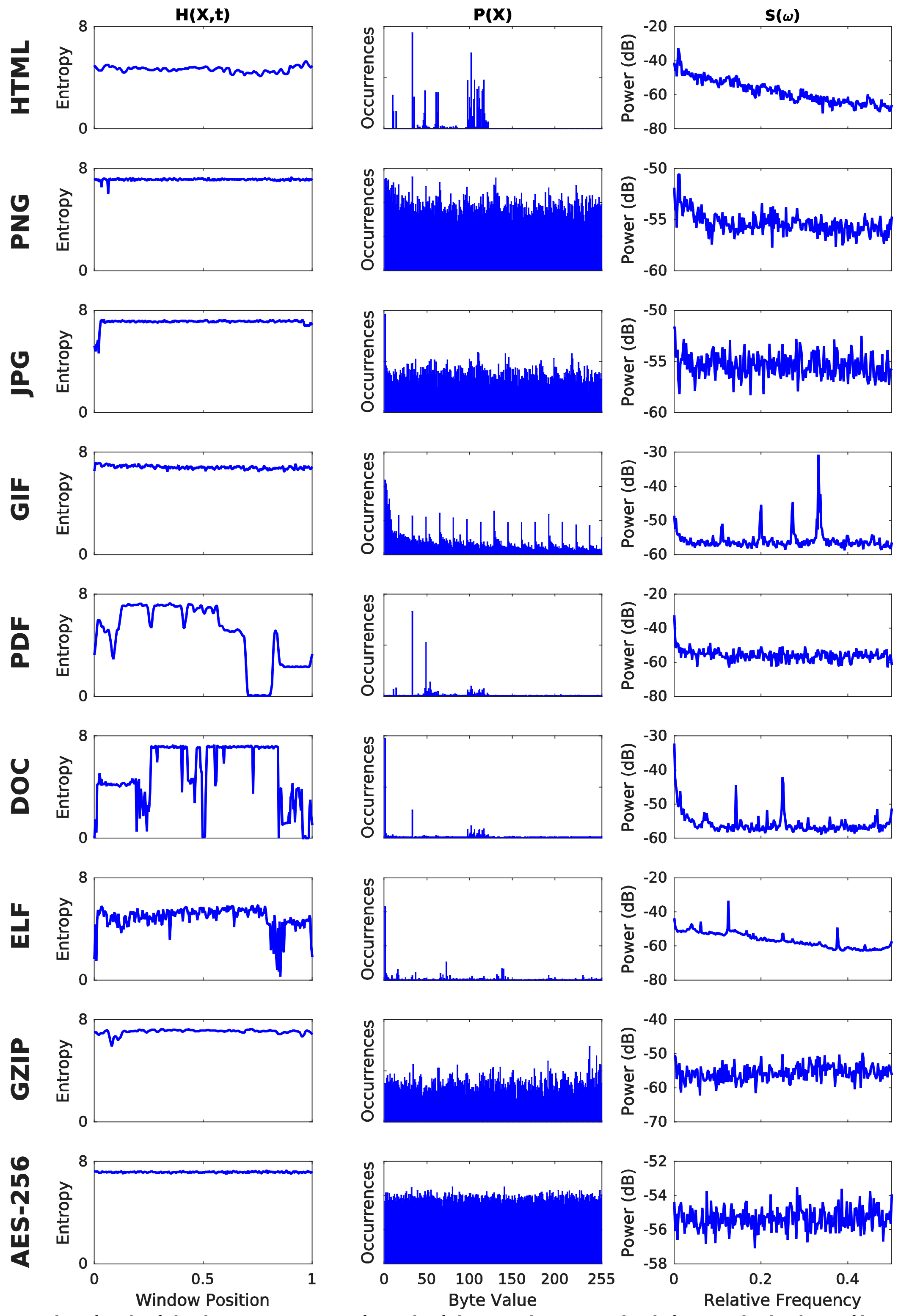

Fig. 2. Examples of each of the three representations for each of the nine classes considered, for a randomly chosen file. Each row represents a different data type. The left hand column shows the sliding window entropy, *H(X,t)*, the middle column shows the distribution of byte values, *P*(x_i), and the right hand column shows the power spectral density of the file, *S(ω)*.

3. Experiments

We investigated the performance of all three of the representations with the classifier. This is important for streaming processing, where the tradeoff between computational complexity and performance must be considered. For training, a database of 4500 files (500 from each class) was obtained from various Internet sites. Pre-training was conducted over 45 epochs per layer, while fine-tuning was performed for 267 epochs. Table 1 shows the accuracy achieved for all combinations of the three representations on a test set of 500 files (100 per class) averaged over three training runs. Interestingly, the byte distribution, P, performs the best of any of the methods alone. On the other hand, entropy performs the worst of the three. The best performance (97.44%) was achieved by combining all three techniques.

Table 1. Classifier performance as a function of input representation for the network shown in Fig. 1. (P, byte distribution; H, sliding-window entropy; S, power spectral density).

	P	H	S	P, H	P, S	H, S	P,H,S
Accuracy (%)	95.30	77.11	81.26	96.74	96.63	94.44	97.44

The confusion matrix for the classifier trained with all three representations (P,H,S), Table 2, shows that PNG is the most difficult to classify, and can be confused with any of several other formats.

Table 2. Confusion matrix for the network provided with all three input representations (B,H,S). (ELF, x86-64 Linux executables; GZIP, gzip compression; AES, AES-256 encryption).

		Predicted Class								
		HTML	PNG	JPEG	GIF	PDF	DOC	ELF	GZIP	AES
Actual Class	HTML	100								
	PNG		91	1		1	1	1	1	4
	JPEG		1	99						
	GIF				100					
	PDF		1	3	1	95				
	DOC		1				99			
	ELF							100		
	GZIP								100	
	AES		5							95

To understand where entropy alone falls short, we also computed a confusion matrix (Table 3) for the classifier trained on only that representation. PNG and GIF images show the worst performance, being confused largely for AES-256 encrypted data. Examining Fig. 2, the entropy of both PNG and GIF files is very high and uniform throughout the file, much like for AES-256. However, the power spectral density and byte distribution spaces both show variation that entropy alone does not capture.

4. Conclusions

We have demonstrated that deep neural networks are a powerful means of identifying characteristic patterns in a variety of data types which are often transferred over the Internet. This method can identify a challenging variety of data formats without numerous hand engineered features, while automatically adapting to new formats. Moreover, we find that the power spectral density and byte distribution capture the most variation between file types, and are superior to sliding window entropy. However, combining these representations with sliding window entropy results in the most accurate and consistent classifier, with average 97.44% accuracy amongst nine file types.

For future work, we hope to thoroughly characterize the performance of the method when the “magic” file signatures[13] are removed from the files and replaced with false signatures. Preliminary studies indicate that our method is better able to identify such mutations, where signatures fail completely, because it relies on characteristic patterns present throughout the data. In addition, analysts are often presented with data that has no known signatures, for which this method may provide insight.

Table 3. Confusion matrix for the network provided only with the sliding window entropy, H.

		Predicted Class								
		HTML	PNG	JPEG	GIF	PDF	DOC	ELF	GZIP	AES
Actual Class	HTML	100								
	PNG		46	4	2	2	3			43
	JPEG		5	86	2	1	1		5	
	GIF		14	2	37				2	45
	PDF	3	7	4	2	83		1		
	DOC					1	98	1		
	ELF						5	95		
	GZIP				3				68	29
	AES									100

Acknowledgements

Funding for this work was provided by Sandia's Laboratory Directed Research and Development (LDRD) program. Sandia is a multi-program laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

References

1. Choo, K.-K. R. The cyber threat landscape: Challenges and future research directions. *Comput. Secur.* 2011; **30:** p. 719–731.
2. Dua, S., Du, X. *Data mining and machine learning in cybersecurity*. CRC Press; 2011. p 5-7.
3. Tabish, S. M., Shafiq, M. Z. & Farooq, M. Malware detection using statistical analysis of byte-level file content. *Proc. of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics*; 2009. p. 23-31
4. Ghezelbigloo, Z., VafaeiJahan, M. Role-opcode vs. opcode: The new method in computer malware detection. *Technology, Communication and Knowledge (ICTCK)*; 2014. p. 1-6.
5. Schmidt, T., Wählisch M., Gröning M. Context-adaptive Entropy Analysis as a Lightweight Detector of Zero-day Shellcode Intrusion for Mobiles. *Poster at the ACM WiSec*, New York; 2011.
6. Sportiello, L., Zanero, S. File Block Classification by Support Vector Machine. *Availability, Reliability and Security (ARES).* 2011. p. 307–312
7. Bengio, Y. Learning Deep Architectures for AI. *Found Trends Mach Learn* 2009; **2.** p. 1–127.
8. Hinton, G. E. Reducing the Dimensionality of Data with Neural Networks. *Science* 2006; **313**. p. 504–507.
9. Lyshevski, S. E. & Krueger, F. A. Nanoengineering bioinformatics: Fourier transform and entropy analysis. *Proc. of the 2004 American Control Conference*; *2004*. p. 17–322.
10. Zeiler, M. D. ADADELTA: An Adaptive Learning Rate Method. *ArXiv12125701 Cs* (2012). at <http://arxiv.org/abs/1212.5701>.
11. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y. & Manzagol, P.-A. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *J Mach Learn Res* 2010; **11.** p. 3371–3408.
12. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J Mach Learn Res* 2014; **15.** p. 1929–1958.
13. file (command). *Wikipedia, the free encyclopedia* (2015). at <http://en.wikipedia.org/w/index.php?title=File_(command)>.