Classification of Advertisements on the Internet and Optimal Network on Distinctiveness

Woojin Ra

Department of Computer Science, Australian National University, Canberra, Australia
u6058768@anu.edu.au

Abstract. Internet advertisements displayed around websites bring revenue to the host at the cost of visitors' bandwidth and can lessen the usability of the website. While numerous ad-blocking systems exist that rely on predefined rules, already implemented *AdEater*[1] system takes an inductive learning approach, automatically generating rules from training examples. Further experiments and modifications that build on *AdEater* demonstrate that a 3-layer neural network significantly outperforms *AdEater* with training time of less than one minute, classification time of 0.94 milliseconds and accuracy of 98% with a slightly modified dataset. This paper demonstrates that an evolution of a neural network based on distinctiveness resulted a space and time optimal network with 94% hidden layer reduction and 53% reduction of testing time.

1 Introduction

Internet advertisements offer a method of earning revenue for the website's host by typically displaying media with hyperlinks to third-party products. While these advertisements support the host and may even affect the livelihood of the host, the visitors must pay with bandwidth, usability of the website, and sometimes security of their device. Media, including images and videos, greatly increase loading time of the website thus wasting visitors' time if they have slow connections. Some visitors prefer not to view such advertisements. Other visitors agree that the Internet is public and should be free of advertisements. On a different note, some third-party companies are displaying malicious codes disguised as advertisements. These malicious codes exploit vulnerabilities present in browsers to survey activities without visitors' knowledge or to mine bitcoins while visitors stay on the website.

Previously implemented *AdEater* system was "a fully-implemented browsing assistant that automatically removes banner advertisements from Internet pages *before* the corresponding images are downloaded, so pages download faster" [4]. While *AdEater* focuses on gathering data and removing advertisements as an assistant, we focus on the improvements of *AdEater* as a neural network and the efficiency of that network. Section 2 describes the architecture of *AdEater* and dataset encoding, while section 3 describes the implementation of a 3-layer neural network featuring batch training with sigmoid activation functions. Furthermore, we test the neural network with the provided dataset [3] with slight modifications and compare the results with the original paper[2] (Sec. 3.2). Section 4 examines network reduction based on distinctiveness [2] and its optimization on the network. Section 5 extends the optimization of the network by applying dynamic reduction over iterations to teach a state of space and time optimality. Finally, section 6 discuss further improvements on the neural network, future plans and summarize conclusions.

[1] *AdEater* system was implemented by Nicholas Kushmerick at the University College Dublin on July 1997.
[2] *Learning to remove Internet advertisements* by Nicholas Kushmerick

2 The *AdEater* system and dataset encoding

AdEater is a fully-implemented browser assistant that gathers examples, creates rules by an inductive learning algorithm named C4.5 [6, 7], and removes advertisements before the browser downloads the media. In order to remove advertisements before the browser downloads the media, *AdEater* examines the anchor `<a>` tags[3] in raw HTML, then creates a fixed-width feature vector from the anchor tag for the trained system. These feature vectors have six categories: dimension, caption, alt, U_{base} , U_{target} and U_{img} , features mostly being boolean (binary) representation on one or two worded phrases that exist within the anchor. Phrases that do not meet the frequency requirement are ignored. Furthermore, certain keywords such as "https" or "www" are excluded from the feature vector as they contain little information on the anchors. The final dataset consisted of 3278 instances with 1558 features.

AdEater chose C4.5 algorithm based on the following requirements of the system: 1) the trained system when online, must execute quickly, 2) the learning algorithm must not be overly sensitive to missing features or classification noise, 3) learning algorithm must scale well with the number of features and be insensitive to irrelevant features and 4) the trained system needs to evolve over time [4].

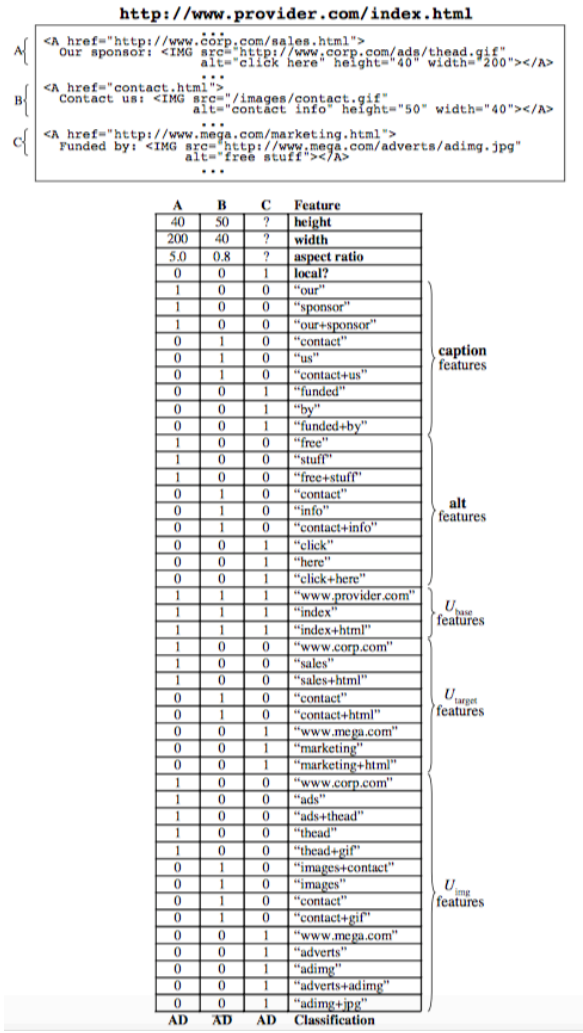


Fig. 1. An example Internet page and the encoding of its three instances. [4]

After training *AdEater* with C4.5 algorithm over a modest dataset, it was able to perform offline training in 5.8 CPU minutes and approximately 70 milliseconds to remove each image during the online classification phase with an accuracy of 97.1% [4]. The creator of *AdEater* determined that the architecture described above performed optimally after ten different experiments with different encodings and training methods as shown in Fig. 2 [4] and Fig. 3 [4].

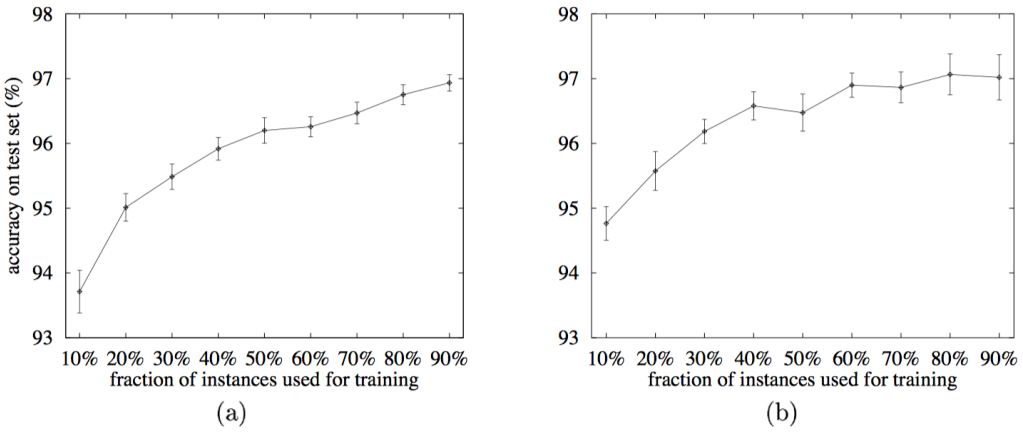


Fig. 2. Learning curves: (a) simple methodology and (b) realistic methodology. [4]

[3] Non-anchor images are rarely advertisements and are therefore ignored [4]

encoding e	K (max. phrase length)	M (min. phrase count)	use stop-list?	use **'local?'**?	use **'aspect ratio'**?	number of features	accuracy (%)	efficiency (E_e)	gain (E_e/E_{standard})
standard	2	10	yes	yes	yes	1558	97.2	0.06	1
just-words	1					1079	96.9	0.09	1.43
long-phrases	4					1979	97.1	0.05	0.79
most-phrases		2				9113	97.6	0.01	0.17
freq-phrases		100				35	96.6	2.8	44
no-phrases		∞				4	93.5	23	375
no-stoplist			no			1703	96.9	0.06	0.91
no-local				no		1557	97.1	0.06	1.0
no-aratio					no	1557	97.2	0.06	1.0

Fig. 3. A comparison of eight variations on the standard encoding. [4]

3 Neural network classification

A shift of focus from collection and classification of data to optimization of network allowed broader approaches moving forward from *AdEater* and introduced different methods of classification. The original paper *Learning to remove Internet advertisements* mentioned that *AdEater* must retrain once more data is collected and that one-sided errors are difficult to implement (due to the difficulty of implementing bias in C4.5). Thus, among these methods, a 3-layer neural network was chosen for its simplicity, flexibility and extensibility. A 3-layer neural network allows flexibility in dataset size and dataset encoding while it continually learns as more data is added to the dataset. Furthermore, neural networks easily allow for bias or weights on neurons which in turn allows one-sided errors (e.g., when in doubt, leave anchor intact) on advertisement predictions.

3.1 The *AdNetwork* implementation

The implemented neural network, now will be referred to as *AdNetwork*, consists of three layers: the input layer, the hidden layer and the output layer. Firstly, the input layer consists of 1555 binary neurons and 3 continuous (dimension) neurons, representing each feature in the dataset. Secondly, the output layer consists of 2 neurons that determine whether an instance is an advertisement or not. Finally, the hidden layer consists of 50 ($N_{\text{hidden}} = \lceil\sqrt{N_{\text{input}}}\rceil + 10$) neurons which was chosen from personal, prior knowledge of neural networks [5]. The number of input and output neurons are predetermined by the problem whereas the number of hidden neurons can vary wildly, thus requiring further examination (Sec. 5). On the process aspect, *AdNetwork* performs normalization on continuous features to improve the effectiveness of data [8]. A portion of preprocessed dataset then trains *AdNetwork* for 750 epochs through sigmoid $f(x) = 1/(1 + e^{-x})$ activation function and stochastic gradient descent optimization.

Fig. 4 shows the similarity of *AdNetwork* and *AdEater*'s accuracy over the offline training cycle, both being trained with 10% increments of the dataset.

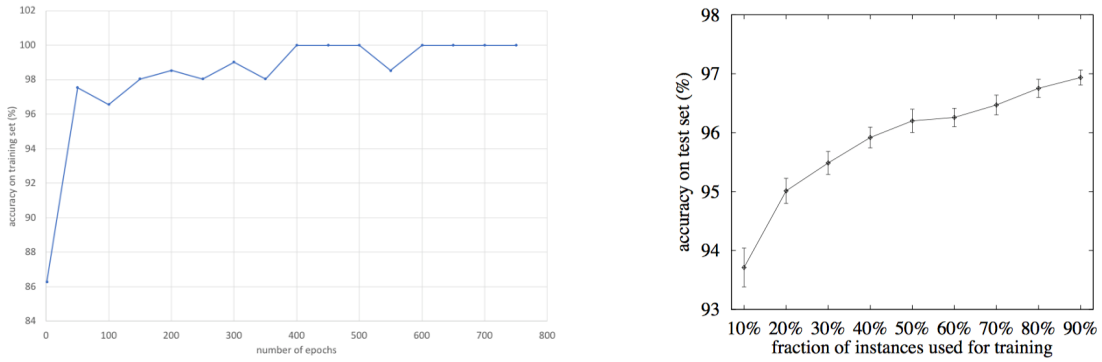


Fig. 4. Learning curves: left showing *AdNetwork* and right showing *AdEater*[4]

3.2 *AdNetwork* performance

Fig. 5 and 6 show the results of *AdNetwork* with the following specifications: 1558 input neurons, 50 hidden neurons, 2 output neurons, 750 epochs, 10% batch size and 0.03 learning rate.

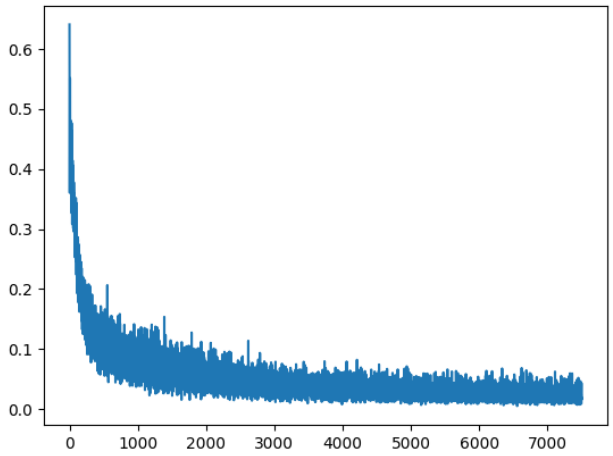


Fig. 5. Cross entropy loss of *AdNetwork*.

	Prediction: Not Ad	Prediction: Ad
Actual: Not Ad	2233	5
Actual: Ad	15	344
Accuracy	98.53%	
Training Time	81.5 seconds	
Testing Time	1.99 milliseconds	

Fig. 6. Results of *AdNetwork*.

Although *AdNetwork* performs with 98.53% accuracy, the confusion matrix should be further examined to properly judge the performance of *AdNetwork*. The saturation of non-advertisements in the dataset may lead to *AdNetwork* learning more about it thus leading to a lesser, 95.82% accuracy on advertisement classifications. Further experiments were conducted with a balanced (removal of non-advertisement instances) dataset; however, it performed poorly due to the smaller size of the dataset.

The need for better advertisement classification requires more instances of advertisements or the addition of bias towards advertisements at the cost of false-positive advertisement classification; and visa-versa for 100% accuracy of non-advertisement classification [5].

The results of *AdNetwork* show improvements on aspects of accuracy and time but it should be noted that *AdEater* utilized a CPU from 1999 while *AdNetwork* utilizes a modern CPU. The computational power at the time of *AdEater* may have rendered 3-layer neural networks unreasonable due to its training and testing time. The focus should be the fact that the flexibility and the extensibility of neural networks introduce greater room for improvements and customization.

4 Network reduction based on Distinctiveness

AdNetwork is now implemented from various experiments based on *AdEater*[4], and outputs reasonable and acceptable performance from the given dataset. However, a network reduction technique based on distinctiveness [2] can be applied to *AdNetwork* to further optimize the time and space usage. The distinctiveness of hidden neurons is determined from the neuron output activation vector over the instance presentation set [2] and represents the similarity or the difference (hence distinctiveness) of two hidden neurons. The distinctiveness of these sets of hidden neurons allow better understanding of the usefulness for each neuron since too similar neurons imply redundancy (can be merged) while too different neurons also imply redundancy (can be countered). After the calculation of vector angles, angles less than 15 degrees or greater 165 degrees are considered too similar or too different thus are removed from the network.

	Hidden units				
Pattern	1	2	3	Result	Target
p.0000	0.5996	0.4686	0.5046	1	1
p.0001	0.5349	0.4618	0.5059	1	1
p.0002	0.5473	0.4617	0.5039	1	1
…	…	…	…	…	…
p.2626	0.5176	0.4903	0.5195	0	0

Fig. 7. An example of hidden neurons vectors in *AdNetwork* described in Sec. 3.

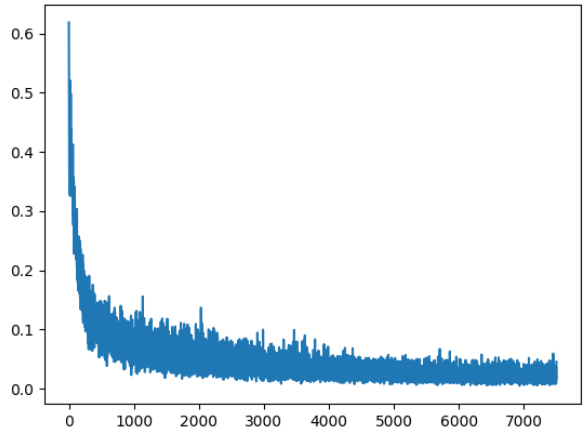
[4] Experiments and results found in *Learning to remove Internet advertisements*.

Although possible to discover pairs of similar neurons by inspection in some neural networks, *AdNetwork* features far too large number of patterns and is impossible to do this within a reasonable time. Hence the vector angles are calculated:

Pairs of neurons	Vector angles
1, 2	4.9
1, 3	3.7
2, 3	2.8

Fig. 8. An example of vector angles calculated from pairs of neurons in Fig. 7

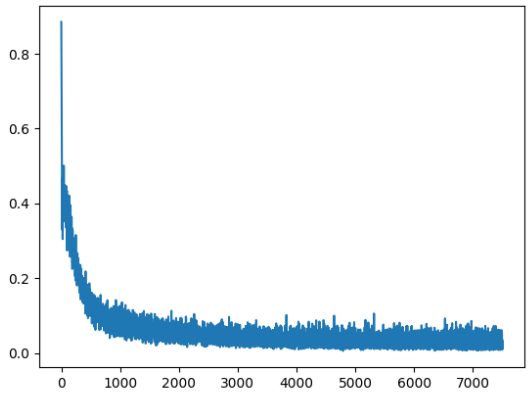
Following the example above, all three vector angles imply redundancy of some hidden neurons. Finally, this process was performed on *AdNetwork* and determined that out of 1225 pairs of hidden neurons, 960 of them were redundant. This led to removing half of the hidden neurons, shown in Fig. 9.



	Prediction: Not Ad	Prediction: Ad
Actual: Not Ad	2249	5
Actual: Ad	12	365
Accuracy	97.99%	
Training Time	59.4 seconds	
Testing Time	1.37 milliseconds	

Fig. 9. Left shows a 25 hidden neuron *AdNetwork* loss graph and right shows the results.

A 25 hidden neuron *AdNetwork* performed as well as the original *AdNetwork* but significantly reduced the training time and the testing time. This version of *AdNetwork* resulted in 188 redundant hidden neuron pairs out of 300 meaning further network reduction was required. Finally, Fig. 10 shows a repeated network reduction based on distinctiveness.



	Prediction: Not Ad	Prediction: Ad
Actual: Not Ad	2252	5
Actual: Ad	13	371
Accuracy	97.65%	
Training Time	55.7 seconds	
Testing Time	0.64 milliseconds	

Fig. 10. Left shows a 3 hidden neuron *AdNetwork* loss graph and right shows the results.

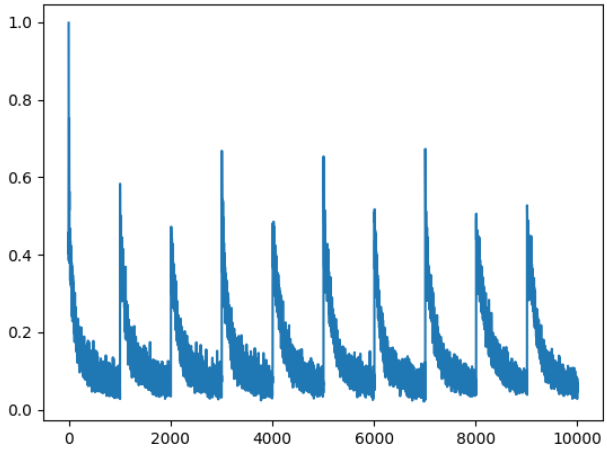
From Fig. 5, 9 and 10, the loss graph of *AdNetwork* are almost identical, implying that the neural network is learning similarly in all three experiments even though the hidden neurons were significantly decreased. The accuracy results of testing 50, 25 and 3 hidden neuron *AdNetwork* also support the previous claim while significantly decreasing the training time and the testing time of the dataset.

5 Evolution of *AdNetwork* based on Distinctiveness

Evolution of neural networks allow them to iteratively search for a set of parameters with the best results. To implement evolution, there must be a set of neural networks with random parameters which then produce new networks with a mixture of two fittest neural networks [5]. Although the traditional sense of evolution requires a set of neural networks, *AdNetwork* only needs to evolve a single parameter, the number of hidden neurons, and with the goal of space and time optimality, a set of *AdNetworks* are not required. Now, the evolution for *AdNetwork* will be determined by the number of redundant neurons described in section 4.

Section 4 shows that a network reduction by distinctiveness also reduces the overall use of space and time. However, the previous experiments only focus on 50, 25 and 3 hidden neurons and while it showed evidence of improvements, it is difficult to deem it optimal. Therefore, *AdNetwork* should start with a sufficiently large number of hidden neurons which is then iteratively and dynamically reduced to optimality. To achieve a dynamically reduced *AdNetwork*, it updates its parameters with a certain interval and a removal rate. The interval was chosen to be 100 epochs to let *AdNetwork* learn before being reduced to output accurate angles between the hidden neurons and the survival rate was chosen to be 60% to emulate random survival of evolution.

Ten experiments were conducted with 50 initial hidden neurons and found that *AdNetwork* reduced its hidden layer to 4 neurons on average while keeping its accuracy and testing time. On the contrary, the training time was increased significantly to 131.67 seconds on average which is an acceptable time compared to finding optimality by brute force[5]. Fig. 11 shows a cross entropy loss of *AdNetwork* with evolution and its results.



	Prediction: Not Ad	Prediction: Ad
Actual: Not Ad	2241	9
Actual: Ad	48	321
Accuracy	97.72%	
Training Time	131.67 seconds	
Testing Time	0.91 milliseconds	

Fig. 11. Left shows a 50 hidden neuron evolving *AdNetwork* loss graph and right shows the average results over 10 trials.

The above findings show that with each evolution, *AdNetwork* continues to feature a decrease in loss, meaning *AdNetwork* continues to learn. One interesting point to note is *AdNetwork*, on average, reduced to 4 hidden neurons instead of 3 discovered in section 4 thus the original *AdNetwork* was further tested with 4 hidden neurons instead of 3. The results showed better testing accuracy than training accuracy. This incremental addition of a hidden neuron consistently improved the testing accuracy of *AdNetwork* over several experiments meaning the network also learned to classify Internet advertisements with abstract patterns rather than discrete patterns.

[5] Brute force may take any time between 2785 (training of 1 hidden neuron × 50) to 4075 (training of 50 hidden neuron × 50) seconds

	Prediction: Not Ad	Prediction: Ad
Actual: Not Ad	2259	6
Actual: Ad	16	353
Accuracy	98.14%	
Training Time	59.67 seconds	
Testing Time	0.69 milliseconds	

	Prediction: Not Ad	Prediction: Ad
Actual: Not Ad	546	6
Actual: Ad	3	85
Accuracy	98.59%	
Training Time	59.67 seconds	
Testing Time	0.69 milliseconds	

Fig. 12. Left shows training accuracy of 4 hidden neuron *AdNetwork* and right shows testing accuracy.

AdNetwork can now be deemed space and time optimized based on distinctiveness and four hidden neurons can also be deemed optimal with the current parameters of *AdNetwork*. While *AdNetwork* with evolution shows further improvements from the original *AdNetwork*, the findings raise new questions regarding its accuracy. *AdNetwork* with 4 hidden neurons consistently showed better results in its testing accuracy than its training accuracy which may mean that *AdNetwork* with 4 hidden neurons is correctly learning abstract patterns, but this should be further examined with larger testing set.

6 Future work

Unlike the original *AdEater*, *AdNetwork* takes a neural network approach to resolve two of the future work discussed in the original paper by Nicholas Kushmerick:

- Some users might prefer one-sided errors (e.g., when in doubt, leave images intact). We know of no easy way to bias C4.5 rules in this manner but extending the learning algorithm to do so would be interesting [4].
- We already mentioned that our task is ideal for exploring "incremental" learning, in which a classifier is modified based on an update to the training instances, rather than being relearned from scratch. As described above, nearest-neighbour and other lazy learning algorithms are incremental but are undesirable for other reasons. Incorporation an incremental decision tree or rule learning algorithm would improve overall efficiency [4].

The above two discussions are still valid on *AdNetwork* as it focuses on the shift in architecture with improvements on its time and space efficiency. However, as mentioned in Sec. 3, *AdNetwork* allows for much easier manipulation of bias and for continuous learning.

AdEater was published in 1999 when websites mainly incorporated images for Internet advertisements. However, moving forward from *AdEater* introduces new types of advertisements such as videos, interactive media, malicious codes and pop-ups, to name a few [1]. The evolution in Internet advertisements call for a broader acceptance of HTML elements in the dataset to accommodate new formats of Internet advertisements (e.g., new Google advertisements are written in `<div>` blocks with titles which clearly indicate advertisements). Furthermore, third-party companies that specialize in Internet advertisements feature gigantic libraries of media which require hashed URLs that contain little to no information, rendering U_{img} features useless.

Fig. 13. Left shows a Google advertisement and right shows the corresponding HTML `<div>` tag with hashed image URL.

7 Conclusion

The shift of focus and the key findings from the original paper[6] describing *AdEater* inspired a neural network based classifier named *AdNetwork*. The initial implementation of *AdNetwork* showed slight improvements on the accuracy of 98.53% while the confusion matrix showed room for further improvements with its consequences, depending on the users' need. In addition, the nature of neural networks enabled future works (described in Sec. 5) to be easily implemented. However, *AdNetwork* focused on space and time optimization by implementing a network reduction technique based on distinctiveness. Neural network reduction on the hidden layer allowed for 94% reduction of hidden neurons, 32% reduction of training time and 53% reduction of testing time while keeping the accuracy at $\pm$1%. This neural network reduction technique was further examined by dynamically reducing *AdNetwork*'s hidden layer with non-traditional evolutionary methods. Evolving *AdNetwork* resulted a 4-neuron hidden layer that is both space and time optimal. *AdNetwork* with 4 hidden neurons consistently showed better testing accuracy with approximately 7% increase in training and testing time. *AdNetwork* also showed signs of learning abstract patterns rather than absolute patterns which may eliminate suspiciousness of overfitting. Finally, future works consist of: a) putting weight bias for advertisements (or non-advertisements) to approach 100% accuracy on advertisement classification (or non-advertisement classification) and b) allowing broader acceptance of HTML tags within the dataset encoding to adapt for contemporary Internet advertisements.

[6] *Learning to remove Internet advertisements* by Nicholas Kushmerick

References

[1] Evans, D. S. (2009). The Online Advertising Industry: Economics, Evolution, and Privacy. *Jounral of Economic Perspectives*, 23(3), pp. 37-60.

[2] Gedeon, TD & Harris, D. (1991). Network Reduction Techniques. In *Proceedings International Conference on Neural Networks Methodologies and Applications*. AMSE.

[3] Kushmerick, N. (1999). *Internet advertisements*, UCI, http://archive.ics.uci.edu/ml/machine-learning-databases/internet_ads/.

[4] Kushmerick, N. (1999). Learning to remove Internet advertisements. In *3rd Int. Conf. on Autonomous Agents.*

[5] MacLeod, C. (NA). *An Introduction to Practical Neural Networks and Genetic Algorithms For Engineers and Scientists*. NA.

[6] Mitchell, T. (1997). *Machine Learning*. McGraw Hill.

[7] Ross Quinlan, J. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

[8] Van den Bout, D. E. & Miller III, T. K. (1989). Improving the performance of the Hopfield-Tank neural network through normalization and annealing. In *Biological Cybernetics*, 62(2), pp. 129-139.