Non-Negative Networks Against Adversarial Attacks

William Fleshman[1], Edward Raff[1,2], Jared Sylvester[1,2], Steven Forsyth[3], Mark McLean[1]
[1]Laboratory for Physical Sciences, [2]Booz Allen Hamilton, [3]Nvidia
{william.fleshman, edraff, jared, mrmclea}@lps.umd.edu, sforsyth@nvidia.com

Abstract

Adversarial attacks against Neural Networks are a problem of considerable importance, for which effective defenses are not yet readily available. We make progress toward this problem by showing that non-negative weight constraints can be used to improve resistance in specific scenarios. In particular, we show that they can provide an effective defense for binary classification problems with asymmetric cost, such as malware or spam detection. We also show how non-negativity can be leveraged to reduce an attacker's ability to perform targeted misclassification attacks in other domains such as image processing.

1 Introduction

Recently, there has been an increased research effort in exploring adversarial examples which fool machine learning based classifiers [Goodfellow et al., 2015; Kurakin et al., 2017b; Szegedy et al., 2014; Yuan et al., 2017]. The majority of this research focuses on the image domain, where an example is generated by making small perturbations to an image's pixels in order to make a large change in the distribution of predicted class probabilities.

We are particularly interested in adversarial attacks for *malware detection*, which is the task of determining if a file is benign or malicious. This involves a real-life adversary (the malware author) who is attempting to subvert detection tools, such as anti-viruses. With machine learning approaches to malware detection becoming more prevalent [Raff et al., 2018; Pascanu et al., 2015; Saxe and Berlin, 2015; Raff et al., 2016; Sahs and Khan, 2012], this is an area that requires solutions to the adversarial problem with greater urgency.

For example, Raff et al. [2018] trains a convolutional neural network to distinguish between benign and malicious Windows executable files. When working with images, any pixel can be arbitrarily altered, but this freedom does not carry over to the malware case. The executable format follows stricter rules which constrain the options available to the attacker [Kreuk et al., 2018; Russu et al., 2016; Grosse et al., 2016]. Perturbing an arbitrary byte of an executable file will most likely change the functionality of the file or prevent it from executing entirely. This property is useful for defending against an adversarial attack, as a malware author needs to evade detection with a *working* malicious file.

Kreuk et al. [2018] were able to get around these limitations by applying gradient-based attacks to create perturbations which were restricted to bytes located in unused sections of malicious executable files. The adversarial examples remained just as malicious, but the classifier was fooled by the introduction of overwhelmingly benign yet unused sections of the file. This is possible because the adversary controls the input, and the EXE format allows the existence of unused sections. Because of the complications and obfuscations that are available to malware authors, it is not necessarily possible to tell that a section is unused, even if its contents appear random.

An analogy to the image domain would be an attacker that could create new pixels which represent the desired class and put them outside of the cropping box of the image, such that they would be in the digital file, but never seen by a human observer. This contrasts with a standard adversarial attack

Preprint. Work in progress.

on images, since the attacker is typically limited to changing the values of pixels in the image rather than introducing new pixels entirely.

Given these unique characteristics and costs, we note that the malware case is one where we care only about targeted adversarial attacks. The adversary always wants to fool detectors in calling malicious files benign. As such, we introduce an approach to tackle targeted adversarial attacks by exploiting non-negative learning constraints. We will highlight the related work to this in section 2. In section 3 we will detail our motivation for non-negative learning for malware, as well as how we generalize its use to protect image classifiers from targeted attacks. The attack scenario and experiments will be detailed in section 4. In section 5 we will demonstrate how our approach reduces evasions to almost 0% for malware detection and provides robustness against confident adversarial attacks against images. Then we will end with our conclusions in section 6.

2 Related Work

The issues of targeted adversarial binary classification problems was first brought up by Dalvi et al. [2004], who noted its importance in a number of domains like fraud detection, counter terrorism, surveillance, and others. There have been several attempts at creating machine learning classifiers which can defend against such adversarial examples. Yuan et al. provide a thorough survey of both attacks and defenses specifically for deep learning systems [Yuan et al., 2017]. Some of these attacks will be used to compare the robustness of our technique to prior methods.

In our case we are learning against a real life adversary in a binary classification task, similar to the initial work in this space on evading spam filters [Lowd and Meek, 2005a; Dalvi et al., 2004; Lowd and Meek, 2005b]. Our malware case gives the defender a slight comparative advantage in constraining the attack to produce a working binary, where spam authors can insert more arbitrary content.

Prior works have looked at similar weight constraint based approaches to adversarial robustness. Kołcz and Teo uses a technique to keep the distribution of learned weights associated with features as even as possible during training [Kołcz and Teo, 2009]. By preventing any one feature from becoming overwhelmingly predictive, they force the adversary to manipulate many features in order to cause a misclassification. Similarly, Grosse et al. tested a suite of feature reduction methods specifically in the malware domain [Grosse et al., 2016]. First, they used the mutual information between features and the target class in order to limit the representation of each file to those features. Like Kołcz and Teo [2009], they created an alternative feature selection method to limit training to features which carried near equal importance. They found both of these techniques to be ineffective.

Our approach is also a feature reduction technique. The difference is that we train on all features, but only retain the capacity to distinguish a reduced number of features at test time — namely, only those indicative of the positive class. Training on all features allows the model to automatically determine which are important for the target class and utilizes the other features to accurately set a threshold, represented by a bias term in most models, for determining when a requisite amount of features are present for assigning samples to the target class.

Chorowski and Zurada used non-negative weight constraints in order to train more interpretable neural networks [Chorowski and Zurada, 2015]. They found that the constraints caused the neurons to isolate features in meaningful ways. We build on this technique in order to isolate features while also preventing our models from using the features predictive of the null class during inference.

Goodfellow et al. used RBF networks to show that low capacity models can be robust to adversarial perturbations but found they lack the ability to generalize[Goodfellow et al., 2015]. With our methods we find we are able to achieve generalization while also producing low confidence predictions during targeted attacks.

3 Isolating Classes with Non-Negative Weight Constraints

We will start by building an intuition on how logistic regression with non-negative weight constraints assigns predictive power to only features indicative of the positive $(+)$ class while ignoring those associated with the negative $(-)$ class.

Let $\boldsymbol{C}(\cdot)$ be a trained logistic regression binary classifier of the form, $\boldsymbol{C}(\boldsymbol{x}) = \text{sign}\left(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x} + b\right)$, where $\boldsymbol{w}$ is the vector of non-negative learned coefficients of $\boldsymbol{C}(\cdot)$, $\boldsymbol{x}$ is a vector of boolean features for a given sample, and b is a scalar bias. The decision boundary of $\boldsymbol{C}(\cdot)$ exists where $\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x} + b = 0$, and because $\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x} \geq 0 \; \forall \; \boldsymbol{x}$, the bias b must be strictly negative in order for $\boldsymbol{C}(\cdot)$ to have the capacity to assign samples to both classes. The decision function can then be rewritten as:

$$\boldsymbol{C}(\boldsymbol{x}) = \begin{cases} (+) & \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x} \geq |b| \\ (-) & \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x} < |b| \end{cases} \tag{1}$$

Because $\boldsymbol{w}$ is non-negative, the presence of any feature $x_i \in \boldsymbol{x}$ can only increase the result of the dot product, thus pushing the classification toward $(+)$. Weights associated to features that are predictive of class $(-)$ will therefore be pushed toward 0 during training. When no features are present $(\boldsymbol{x} = \vec{0})$ the model defaults to a classification of $(-)$ due to the negative bias b. Unless a sufficient number of features predictive of class $(+)$ are present in the sample, the decision will remain unchanged. A classifier trained in this way will use features indicative of the $(-)$ class to set the bias term, but will not allow those features to participate in classification at test time. The same logic follows for logistic regression with non-boolean features if the features are also non-negative or scaled to be non-negative before training.

Given a problem with asymmetric misclassification goals, we can leverage this behavior to build a defense against adversarial attacks. For malware detection, the malware author wishes to avoid detection as malware $(+)$, and instead induce a false detection as benign $(-)$. However, there is no desire for the author of a benign program to make their applications be detected as malicious. Thus, if we model malware as the positive class with non-negative weights, *nothing can be added* to the file to make it seem more benign to the classifier $\boldsymbol{C}(\cdot)$. Because executable programs must maintain functionality, the malware author can not trivially remove content to reduce the malicious score either. This leaves the attacker with no recourse but to re-write their application, or perform more non-trivial acts such as packing to obscure information. Such obfuscations can then be remediated through existing approaches like dynamic analysis [Ugarte-Pedrero et al., 2016,?; Chistyakov et al., 2017].

Notably, this method also applies to neural networks with a sigmoid output neuron as long as the input to the final layer and the final layer's weights are constrained to be non-negative. The output layer of such a network is identical to our logistic regression example. The cumulative operation of the intermediate layers $\boldsymbol{\phi}(\cdot)$ can be interpreted as a re-representation of the features before applying the logistic regression such as $\boldsymbol{C}(\boldsymbol{x}) = \text{sign}\left(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b\right)$.

The ReLU function is a good choice for intermediate layers as it maintains the required non-negative representation and is already found in most modern neural networks.

For building intuition, in Figure 1 we provide an example of how this works for neural networks using MNIST. To fool the network into predicting the positive class (one) as the negative class (zero), the adversary must now make larger removals of content — to the point that the non-negative attack is no longer a realistic input.

It should be noted that constraining a model in this way does reduce the amount of information available for discriminating samples at inference time, and a drop in classification accuracy is likely to occur for most problems. The trade off between adversarial robustness and performance should be analyzed for the specific domain and use case. We will denote when a model is trained in a non-negative fashion by appending "+" to its name.

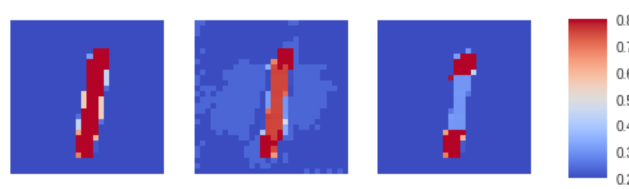


Figure 1: Left: Original Image; Middle: Gradient attack on LeNet; Right: Gradient attack on non-negative LeNet+. The attack on the standard model was able to add pixel intensity in a round, zero-shaped area to fool the classifier into thinking this was a zero. The attack on the constrained model was forced to remove pixel intensity from the one rather than adding in new values elsewhere.

3.1 Non-Negativity and Image Classification

While the primary focus of our work is on malware detection, we note that non-negativity can still be beneficial for multi-class problems such as image classification. However, we find it necessary to re-phrase how such tasks are handled. Normally, one would use the softmax, $\text{softmax}(\boldsymbol{v})_i = \exp(v_i)/\sum_{j=1}^{n}\exp(v_j)$, on the un-normalized probabilities $\boldsymbol{v}$ given by the final layer. The probability of a class i is then taken as $\text{softmax}(\boldsymbol{v})_i$. However we find that the softmax activation makes it easier to attack networks.

Take the non attacked activation pattern $\boldsymbol{v}$, where $v_i > v_j \; \forall \; j \neq i$. Now consider the new activation pattern $\hat{\boldsymbol{v}}$, which is produced by an adversarially perturbed input with the goal of inducing a prediction as class q instead of i. Then it is necessary to force $\hat{v}_q > \hat{v}_i$. Yet even if $\hat{v}_i \approx v_i$, the probability of class i can be made arbitrarily low by continuing to maximize the response of $\hat{\boldsymbol{v}}_q$. This means we are able to diminish the apparent probability of class i without having impacted the model's response to class i. Phrased analogously as an image classification problem, adversaries don't need to remove the amount of "cat" in a photo to induce a decision of "potato", but only increase the amount of "potato".

In addition Chorowski and Zurada [2015] proved that a non-negative network trained with softmax activation can be transformed into an equivalent unconstrained network. This means there is little reason to expect our non-negative approach to provide benefit if we stay with the softmax activation, as it has an equivalent unconstrained form and should be equally susceptible to all adversarial attacks. As such we must move away from softmax to get the benefits of our non-negative approach in a multi-class scenario.

Instead we can look at the classification problem in a one-vs-all fashion by replacing the softmax activation over K classes with K independent classifications trained with the binary cross-entropy loss and using the sigmoid activation $\sigma(z) = 1/(1 + \exp(-z))$. Final probabilities after training are obtained by normalizing the sigmoid responses to sum to one. We find that this strategy combined with non-negative learning produces an effective defense against an adversary producing targeted high-confidence attacks (e.g., the network is 99% sure the cat is a potato). The one-vs-all component make it such that increasing the confidence of a new class eventually requires reducing the confidence of the original class. The non-negativity component increases the difficulty of this removal step, resulting in destructive changes to the image.

We make two important notes on how we apply non-negative training for image classification. First, we pre-train the network using the standard softmax activation, and then re-train the weights with our one-vs-all style and non-negative constraints on the final fully connected layers. Doing so we find only a small difference in accuracy between results, where training non-negative networks from scratch often has reduced accuracy. Second, we continue to use batch normalization without constraints. This is because batch normalization can be rolled into the bias term and as a re-scaling of the weights, and so does not break the non-negative constraint in any way. We find its positive impact on convergence greater when training with the non-negative constraints.

4 Experimental Methodology

Having defined the mechanism by which we will defend against targeted adversarial attacks, we will investigate its application to two malware detection models and four image classification tasks.

We will spend more time introducing the malware attacks, as readers may not have as much experience with this domain.

For malware, we will look at MalConv [Raff et al., 2018], a recently proposed neural network that learns from raw bytes. We will also consider an N-Gram based model [Raff et al., 2016]. Both of these techniques are applied to the raw bytes of a file. We use the same 2 million training datums and 80,000 testing set as used in [Raff and Nicholas, 2017].

Like Grosse et al. [2016], we only focus on the case where an adversary is attempting to cause a malware sample to evade detection by masquerading as a benign file. While it is possible to fool a classifier into thinking benign files are malicious, benign authors would not have such motivation.

4.1 Attacking MalConv

MalConv is the primary focus of our interest, as gradient based attacks can not naively be applied to it's architecture. Only recently have attacks been proposed [Kolosnjaji et al., 2018; Kreuk et al., 2018], and we will show that non-negativity allows us to subvert these adversaries. In MalConv, raw bytes of an executable are passed through a learned embedding layer which acts as a lookup table to transform each byte into an 8-dimensional vector of real values. This representation is then passed through a 1-dimensional gated convolution, global max pooling, and then a fully connected layer with sigmoid output. To handle varying file sizes, all sequences of bytes are padded to a fixed length of 2,100,000 using a special "End of File" value (256) from outside of the normal range of bytes (0–255).

The raw bytes are both discrete and non-ordinal, which prevents gradient based attacks from manipulating them directly. Kreuk et al. [2018] (and independently [Kolosnjaji et al., 2018]) devised a clever way of modifying gradient based attacks to work on EXEs, even with a non-differentiable embedding layer, and we will briefly recap their approach. This is done by performing the gradient search of an adversarial example in the 8-dimensional vector space produced by the embedding layer. A perturbed vector is then mapped to the byte which produces the nearest neighbor in the embedding space. Keeping with their notation[Kreuk et al., 2018], let $\boldsymbol{M} \in \mathbb{R}^{n \times d}$ be the lookup table from the embedding layer such that $\boldsymbol{M} : \boldsymbol{X} \to \boldsymbol{Z}$ where $\boldsymbol{X}$ is the set of n possible bytes and $\boldsymbol{Z} \subseteq \mathbb{R}^d$ is the embedding space. Then for some sequence of bytes $\boldsymbol{x} = (x_0, x_1, \ldots, x_L)$, we generate a sequence of vectors $\boldsymbol{z} = (\boldsymbol{M}[x_0], \boldsymbol{M}[x_1], \ldots, \boldsymbol{M}[x_L])$ were $\boldsymbol{M}[x_i]$ indicates row x_i of $\boldsymbol{M}$. Now we generate a new vector $\tilde{\boldsymbol{z}} = \boldsymbol{z} + \boldsymbol{\delta}$ where $\boldsymbol{\delta}$ is a perturbation generated from an adversarial attack. We map each element $\tilde{z}_i \in \tilde{\boldsymbol{z}}$ back to byte space by finding the nearest neighbor of $\tilde{z}_i$ among the rows of $\boldsymbol{M}$. By applying this technique to only specific safe regions of a binary, the execution of gradient based attacks against MalConv are possible without breaking the binary. To ensure that a "safe" area exists, they append an unused section to the binary. The larger this appended section is, the more space the adversary has to develop a strong enough signal of "benign-ness" to fool the algorithm.

We replicate the attack done by Kreuk et al. [2018] which uses the *fast gradient sign method* (FGSM) [Goodfellow et al., 2015] to generate an adversarial example in the embedding space. We find our $\tilde{\boldsymbol{z}}$ by solving: $\tilde{\boldsymbol{z}} = \boldsymbol{z} + \epsilon \cdot \mathrm{sign}\left(\nabla_{\boldsymbol{z}} \tilde{\ell}\left(\boldsymbol{z}, y; \boldsymbol{\theta}\right)\right)$, where $\tilde{\ell}(\cdot)$ is the loss function of our model parameterized by $\boldsymbol{\theta}$ and $\boldsymbol{z}$ is the embedded representation of some input with label y. The new $\tilde{\boldsymbol{z}}$ is then mapped back into byte space using the method previously discussed. We performed the attack on 1000 randomly selected malicious files, varying the size of the appended section used to generate the adversarial examples.

For MalConv, adding an unused section allows an attacker to add benign features which overwhelm the classification. Our hypothesis is that MalConv^{+} should be immune to the attack since it only learns to look for maliciousness, defaulting to a decision of benign when no other evidence is present. We also note that this corresponds well with how Anti-Viruses prefer to have lower false positive rates to avoid interfering with user's applications.

4.2 Attacking N-Gram

The N-Gram model was trained using lasso regularized logistic regression on the top million most frequent 6-byte n-grams found in our 2 million file training set. The 6-byte grams are used as boolean features, where a 1 represents the n-gram's existence in a file. Lasso performed feature selection by assigning a weight of 0 to most of the n-grams. The resulting model had weights assigned to approximately 67,000 of the features.

We devise a white-box attack similar to the attack Kreuk et al. [2018] used against MalConv in that we inject benign bytes into an unused section appended to malicious files. Specifically, we take the most benign 6-grams by sorting them based on their learned logistic regression coefficients. We add benign 6-grams one at a time to the malicious file until a misclassification occurs. This ends up being the same kind of approach Lowd and Meek [2005a] used to perform "Good Word" attacks on spam filters, except we assume the adversary has perfect knowledge of the model. The simplicity of the N-Gram model allows us to do this targeted attack, and specifically look at the evasion rate as a function of the number of inserted features.

To prevent these attacks, we train N-Gram+ using non-negative weight constraints on the same data. This model is prevented from assigning negative weights to any of the features. We also remove the lasso regularization from N-Gram+ as the constraints are already performing feature selection by pushing the weights of benign features to zero. We perform a similar experiment with spam filters in the supplemental material.

4.3 Targeted Attacks on Image Classification

For our image classification experiments we follow the recommendations of Carlini and Wagner [2017] for evaluating an adversarial defense. In addition to the FGSM attack, we will also use a stronger iterated gradient attack. Specifically we use the Iterated Gradient Attack (IGA) introduced in [Kurakin et al., 2017a], using Keras for our models and Foolbox [Rauber et al., 2017] for the attack implementations. We evaluated the confidences at which such attacks can succeed against the standard and our non-negative models on MNIST, CIFAR 10 and 100, and Tiny ImageNet.

We are specifically interested in defending against an adversary creating a high confidence targeted attack (e.g., a label was previously classified as "cat", but now is classified as "potato" with a probability of 99%). As such we will look at the evasion rate for an adversary altering an image to other classes over a range of target probabilities p. The goal is to see the non-negative trained network have a lower evasion rate, especially for $p \geq 90\%$.

For MNIST and CIFAR 10, since there are only 10 classes, we calculate the evasion rate at a certain target probability p as the average rate at which an adversary can successfully alter the networks prediction to every other class and reach a minimum probability p. For CIFAR 100 and Tiny ImageNet, the larger number of classes prohibits this exhaustive pairwise comparison. Instead we evaluate the evasion rate against a randomly selected alternative class.

On MNIST, CIFAR 10, and CIFAR 100, due to their small image sizes ($\leq 32 \times 32$), we found that adversarial attacks would often "succeed" by changing the image to an unrecognizable degree. For this reason we set a threshold of 60 on the L1 distance between the original image and the adversarial modification. If the adversarial modified image exceeded this threshold, we counted the attack as a failure. This threshold was determined by examining several images; more information can be found in the appendix. For Tiny ImageNet this issue was not observed, therefore no threshold was used.

5 Results

Having reviewed the method by which we will fight targeted adversarial attacks, and how the malware attacks will be applied, we will now present the results of our non-negative networks. First we will review those related to malware detection, showing that non-negative learning effectively neutralizes evasion by a malware author. Then we will show how non-negative learning can improve robustness on several image classification benchmarks.

5.1 Malware Detection

Using the method outlined in section 4, Kreuk et al. [2018] reported a 100% evasion rate of their model. As shown in Figure 2, our replication of the attack yielded similar results for MalConv, which was evaded successfully for 95.4% of the files. The other 4.6% of files were all previously classified as malware with a sigmoid activation of 1.0 at machine precision. The sigmoid function cannot actually produce this value, but rounding errors for floating points near 1.0 can cause this to occur. The attack fails for these cases since there is no valid gradient for this output. A persistent adversary could still create a successful adversarial example by replacing the sigmoid output with a linear activation function before running the attack.

Our non-negative learning provides an effective defense, with only 0.6% of files able to evade MalConv+. Theoretically we would expect an evasion rate of 0.0%. Investigating these successful evasions uncovered a hidden weakness in the MalConv architecture. We found that both MalConv and MalConv+ learned to give a small amount of malicious predictive power to the special End of File (EOF) padding value. This is most likely a byproduct of the average malicious file size being less than the average benign file size in our training set, which causes the supposedly neutral EOF value itself to be seen as an indicator of maliciousness. The process of adding an unused file section

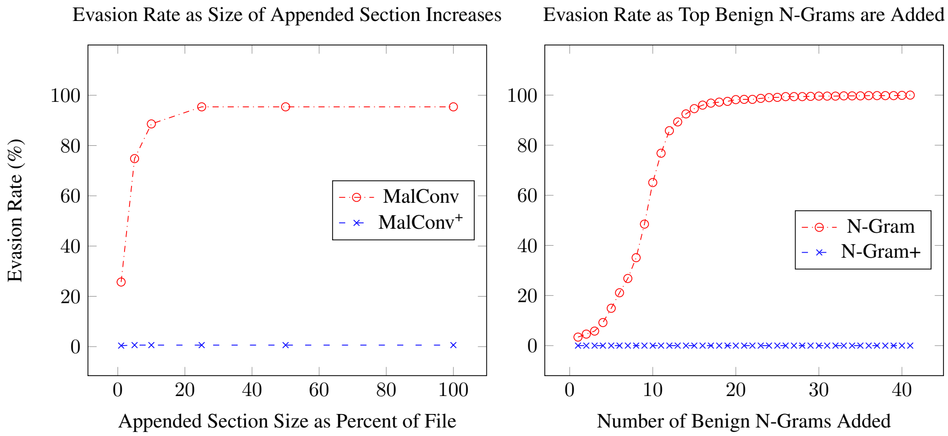


Figure 2: Evasion rate (y-axis for both figures) for MalConv and N-Gram based models. Left figure shows MalConv evasion as the appended section size increases, and right figure shows the N-Gram evasion as the number of benign n-grams are added.

necessarily reduces the amount of EOF padding tokens given to the network, as the file is increased in size (pushing it closer to the 2.1MB processing limit) the new section replaces the EOF tokens. Replacing the slightly malicious EOF tokens with benign content reduces the network's confidence in the file being malicious.

The 0.6% of files that evaded MalConv$^+$ only did so when files were small, and the appended section ended up comprising 50% of the resulting binary. The slight maliciousness from the EOF was the needed feature to push the network into a decision of "malicious". However, the removal of EOFs by the unused section removed this slight signal, and pushed the decision back to "benign". If we instead replace the bytes of the unused section with random bytes from the uniform distribution, the files still evade detection. This means the evasion is not a function of the attack itself, but the modification of the binary that removes EOF tokens. A simple fix to this padding issue is to force the row of the embedding table corresponding to the special byte to be the zero vector during training. This would prevent the EOF token from providing any predictive power during inference.

We observed similar results for the N-Gram model. The evasion rate increases rapidly as benign features are added to the malicious files. We found that appending the top 41 most benign features resulted in a 100% evasion rate. This attack is completely mitigated by N-Gram$^+$ since none of its features have negative weights supporting the benign class. The only way to alter the classification would be to remove malicious n-grams from the files. Our results for both models are depicted in Figure 2.

Accuracy vs Defense

The only drawback of this approach is the possible reduction in overall accuracy. Limiting the available information at inference time will likely reduce performance for most classification tasks. Alas, many security related applications exist because adversaries are present in the domain. We have shown that under attack our normal classifiers completely fail — therefore a reduction in overall accuracy may be well worth the increase in model defensibility. Table 1 shows metrics from our models under normal conditions for comparison.

Table 1: Out of sample performance in the absence of attack.

Classifier	Accuracy %	Precision	Recall	AUC %
MalConv	94.1	0.913	0.972	98.1
MalConv$^+$	89.4	0.908	0.888	95.3
N-Gram	95.5	0.926	0.987	99.6
N-Gram$^+$	91.1	0.915	0.885	95.5

While our non-negative approach has paid a penalty in accuracy, we note that we can see this has predominately come from a reduction in recall. Because features can only indicate maliciousness, some malicious binaries are labeled as benign due to a lack of information. This scenario corresponds to the preferred deployment scenario of security products in general, which is to have a lower false positive rate (benign files marked malicious) at the expense of false negatives (malicious files marked benign) [Ferrand and Filiol, 2016; Masud et al., 2008; Alazab et al., 2011; Zhou and Inge, 2008; Yih et al., 2006]. As such the cost of non-negativity in this scenario is well aligned with its intended use case, making the cost palatable and the trade-off especially effective.

5.2 Image Classification

Having investigated the performance of non-negative learning for malware detection, we now look at its potential for image classification. In particular, we find it is possible to leverage non-negative learning as discussed in subsection 3.1 to provide robustness against confident targeted attacks. That is to say if the predicted class is y_i, the adversary wants to trick the model into predicting class $y_j, j \neq i$ and that the confidence of the prediction be $\geq p$.

For MNIST we will use LeNet. Our out of sample accuracy using a normal model is 99.2%, while the model with non-negative constrained dense layers achieves 98.6%. For CIFAR 10 and 100 we use a ResNet based architecture.[1] For CIFAR 10 we get 92.3% accuracy normally, and 91.6% with our non-negative approach. On CIFAR 100 the same architecture gets 72.2% accuracy normally, and 71.7% with our non-negative approach. For Tiny ImageNet, we also use a ResNet architecture with the weights of all but the final dense layers initialized pretrained from ImageNet.[2] The normal model has an accuracy of 56.6%, and the constrained model 56.3%. The results as a function of the target confidence p can be seen in Figure 3.

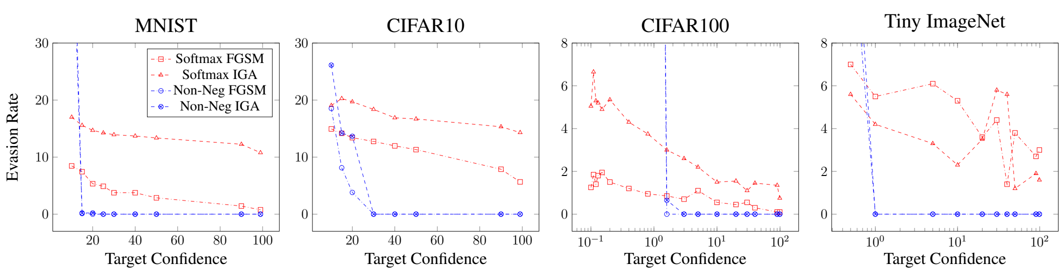


Figure 3: Targeted evasion rate (y-axis) as a function of the desired misclassification confidence p (x-axis) for four datasets. Due to the differing ranges of interest, right two figures shown in log scale for the x-axis.

An interesting artifact of our approach is that the non-negative networks are easier to fool for low-confidence errors. We posit this is due to the probability distribution over classes becoming near uniform under attack. On CIFAR100 the y-axis is truncated for legibility since the evasion rate of FGSM is 93% and IGA is 99%. Similarly for non-negative Tiny ImageNet, FGSM and IGA achieve 14% and 17% evasion rates when $p = 0.005$.

Despite these initial high evasion rates, we can see in all cases the success of targeted adversarial attacks reaches 0% as the desired probability p increases. For MNIST and CIFAR10, which have only 10 classes, this occurs at up to a target 30% confidence. As more classes are added, the difficulty of the attack increases. For Tiny ImageNet and CIFAR100, targeted attacks fail by $\leq 2\%$.

If *targeted* adversarial attacks were the only type of attack we needed to worry about, these results would also allow us to use the confidence as a method of detecting attacks. For example, CIFAR10 had the weakest results with needing a target confidence of 30% before targeted attacks failed. The average predicted confidence of the non-negative network on the test set was 93.8%. This means we can use the confidence itself as a measure of network robustness. If we default to a "no-answer" for everything with a confidence of 40% or less on CIFAR10, and assume anything below that level is an attack and error, the accuracy would have only gone down 1.2%.

[1] v1 model taken from `https://tinyurl.com/keras-cifar10-restnet`

[2] ResNet50 built-in application from `https://keras.io/applications/#resnet50`

6 Conclusion

We have shown that an increased robustness to adversarial examples can be achieved through non-negative weight constraints. Constrained binary classifiers can only identify features associated with the positive class during test time. Therefore, the only method for fooling the model is to remove features associated with that class. This method is particularly useful in security-centric domains like malware detection, which have well-known adversarial motivation. Forcing adversaries to remove maliciousness in these domains is the desired outcome. We have also described a technique to generalize this robustness to multi-class domains such as image classification. We showed a significant increase in robustness to targeted adversarial attacks while minimizing the amount of accuracy lost in doing so.

7 Supplemental Material

In this supplemental section we include additional details and results for interested readers. In particular, we show how our non-negative approach provides a better solution to attacks on spam filters, and further explain our selection of a threshold for the CIFAR results.

7.1 Preventing Good Word Attacks on Spam Filters

Lowd and Meek [2005a] created “Good Word” attacks to successfully evade spam filters without access to the model. These attacks append words commonly found in normal emails onto spam in order to overwhelm the model into thinking the email is legitimate.

In their seminal work, they noted that it was unrealistic to assume that an adversary would have access to the spam filter, and would thus need to somehow guess at which words are good words, or to somehow query the spam filter to steal information about which words are good. Others have simply used the most frequent words from the ham messages as a proxy to good word selection that an adversary could replicate [Jorgensen et al., 2008; Zhou et al., 2007]. We take the more pessimistic approach that the adversary has full access to our model, and can simply select the words that have the largest negative coefficients (the most good-looking words) for their attack.

By showing that our non-negative learning approach eliminates the possibility of good word attacks in this pessimistic case, we intrinsically cover all weaker cases of an adversary’s ability. We note as well that Lowd and Meek speculated the only effective solution to stop the good word attack would be to to periodically re-train the model.

We train two logistic regression models on the TREC 2006 and 2007 Spam Corpora.[3] The 2006 dataset contains 37,822 emails with 24,912 being spam. The 2007 dataset contains 75,419 messages with 50,199 of them being spam. We performed very little text preprocessing and represented each email as a vector of boolean features corresponding to the top 10,000 most common words in the corpus. The first model is trained with lasso regularization in a traditional manner. The second model is trained with non-negative constraints on the coefficients in order to isolate only the features predictive of spam during inference.

The accuracies for our traditional models were high on both datasets, but both were susceptible to our version of “Good Word” attack. Both classifiers were evaded 100% of the time by appending only 7 words to each message in the 2006 case and only 4 words in the 2007 case. These words correspond to the features with the lowest regression coefficients (i.e., negative values with high magnitude) for each model.

Using the non-negative approach lowers our accuracy for both datasets, but completely eliminates these attacks as all “Good Words” have coefficients of 0. The spam author would only be able to evade detection by removing words indicative of spam from their message. A comparison of performance is shown in Table 2

Despite the drops in accuracy imposed by our non-negative constraint, the results are better than prior works in defending against weaker versions of the “Good Word” attack. For example, Jorgensen et al. [2008] developed a defense based on multiple instance learning. Their approach when attacked with all of their selected good words had a precision of 0.772 and a recall of 0.743 on the 2006 TREC

[3]Acquired at `https://trec.nist.gov/data/spam.html`

Table 2: Out of sample performance in the absence of attack.

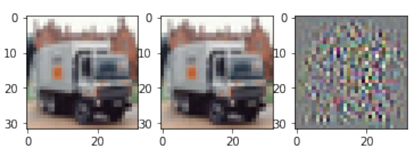
Classifier	Accuracy %	Precision	Recall	AUC %	F1 Score
2006 Lasso	96.5	0.974	0.993	97.1	0.983
2006 Non-Negative	82.6	0.912	0.820	83.5	0.864
2007 Lasso	99.7	0.999	0.999	99.7	0.999
2007 Non-Negative	93.6	0.962	0.940	93.0	0.951

corpus. This was the best result of all their tested methods, but our non-negative approach achieves a superior 0.912 and 0.820 precision and recall respectively.

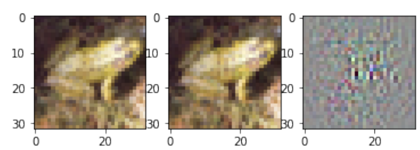
7.2 Threshold for CIFAR Adversaries

When creating an adversarial attack, it is necessary that some portion of the image must be changed as an intrinsic part of the attack. There is currently considerable debate about the nature of how large that change ought to be, how it should be measured, and how much we should care about the nature of changes to the original image. All of these could be topics of research in their own right, and we do not claim to solve them in this work. We use the L_1 distance as a measure simply because it has been used by prior works, even if it is not ideal.

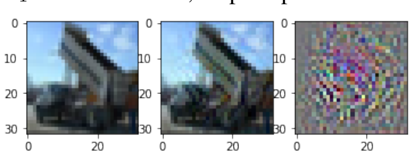
We also take the stance that it is important that no perceptible difference between the input and attacked result is important, while recognizing that is not an agreed upon procedure by everyone in the community. Intuitively we find the lack of perceptible difference important because it leaves no ambiguity about the ground truth label. If the input is noticeably perturbed by an attack, the true label of the new attacked image may come into question. Our CIFAR 10 & 100 results against the non-negative networks often produced large differences, brining us to a need to impose a threshold at which we will consider an attack a "failure."



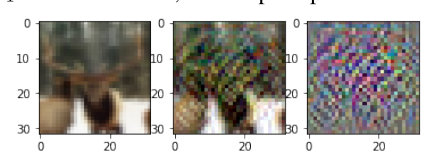
(a) L_1 difference of 10, no perceptible difference.



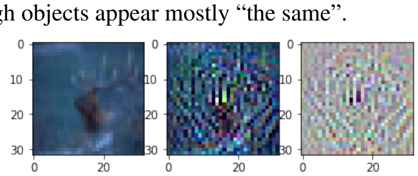
(b) L_1 difference of 30, minute perceptible difference.



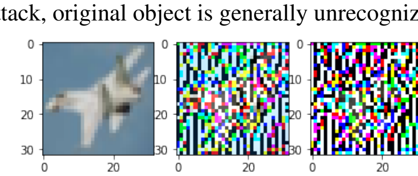
(c) L_1 difference of 60, obvious perceptible difference, though objects appear mostly "the same".



(d) L_1 difference of 150, significant artifacts emerging in attack, original object is generally unrecognizable.



(e) L_1 difference of 400, original object is no longer recognizable.



(f) L_1 difference of 1000, the image has been completely destroyed.

Figure 4: Examples of IGA attacks against CIFAR 10 images with our non-negative network. In each sub figure, the left most image is the original image, the middle is the attacked result, and the right shows the difference. Moving from sub-figure (a) to (f), the L_1 difference between the original and adversarial image increases.

Figure 4 highlights the need to choose a threshold by providing examples of the spectrum of L_1 distances we observed between the original and the adversarial-generated images. It starts with small distances, such as Figure 4a, which has only an L_1 distance of 10 and is clearly still a truck. At the extreme end, we also had results like Figure 4f which had an L_1 distance of 1000, and is wholly unrecognizable. We argue that such an attack must be a failure, as the input does not even resemble the distribution of images from CIFAR.

The question is, where does one draw the line? While we argue that an imperceptible difference is important to avoid label ambiguity, we have attempted to give deference in allowing large magnitude attacks while also recognizing that L_1 is not the ideal method to measure visual perceptual difference. As such we have experimentally selected a threshold of 60 as one that allows for perceptible differences, and at the edge of no longer being recognizable as its original class.

Our decision to use a threshold of 60 is best shown in Figure 4c where the L_1 difference starts to demonstrate an obvious perceptible difference. We feel this image represents a balance between the subjective ability of being able to still tell that it is a type of car / truck, and having difficulty recognizing what is in the image (or if it is still valid) without the context of the original image next to it.

Allowing larger thresholds for the CIFAR attacks begins to enter a territory where it is not clear to us that the true label of the image has been retained. Figure 4d shows one such example with a deer, where the adversarial image has the same colors but is unclear to us what the adversarial image should be labeled as.

References

Mamoun Alazab, Sitalakshmi Venkatraman, Paul Watters, and Moutaz Alazab. Zero-day Malware Detection Based on Supervised Learning Algorithms of API Call Signatures. In *Proceedings of the Ninth Australasian Data Mining Conference - Volume 121*, AusDM '11, pages 171–182, Darlinghurst, Australia, Australia, 2011. Australian Computer Society, Inc. ISBN 978-1-921770-02-9. URL `http://dl.acm.org/citation.cfm?id=2483628.2483648`.

Nicholas Carlini and David Wagner. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, AISec '17, pages 3–14, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5202-4. doi: 10.1145/3128572.3140444. URL `http://doi.acm.org/10.1145/3128572.3140444`.

Alexander Chistyakov, Ekaterina Lobacheva, Arseny Kuznetsov, and Alexey Romanenko. Semantic Embeddings for Program behavior Patterns. In *ICLR Workshop*, 2017.

Jan Chorowski and Jacek M Zurada. Learning Understandable Neural Networks With Nonnegative Weight Constraints. *IEEE Transactions on Neural Networks and Learning Systems*, 26(1):62–69, 1 2015. ISSN 2162-237X. doi: 10.1109/TNNLS.2014.2310059. URL `http://ieeexplore.ieee.org/document/6783731/`.

Nilesh Dalvi, Pedro Domingos, Mausam, Sumit Sanghai, and Deepak Verma. Adversarial Classification. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 99–108, New York, NY, USA, 2004. ACM. ISBN 1-58113-888-1. doi: 10.1145/1014052.1014066. URL `http://doi.acm.org/10.1145/1014052.1014066`.

Olivier Ferrand and Eric Filiol. Combinatorial detection of malware by IAT discrimination. *Journal of Computer Virology and Hacking Techniques*, 12(3):131–136, 2016. ISSN 2263-8733. doi: 10.1007/s11416-015-0257-8. URL `http://dx.doi.org/10.1007/s11416-015-0257-8`.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations*, 2015.

Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick D. McDaniel. Adversarial perturbations against deep neural networks for malware classification. *CoRR*, abs/1606.04435, 2016. URL `http://arxiv.org/abs/1606.04435`.

Zach Jorgensen, Yan Zhou, and Meador Inge. A Multiple Instance Learning Strategy for Combating Good Word Attacks on Spam Filters. *J. Mach. Learn. Res.*, 9:1115–1146, 6 2008. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=1390681.1390719`.

Aleksander Kołcz and Choon H Teo. Feature Weighting for Improved Classifier Robustness. In *6th Conference on Email and Anti-Spam (CEAS'09)*, 7 2009. URL `http://ceas.cc/2009/papers/ceas2009-paper-25.pdf`.

Bojan Kolosnjaji, Ambra Demontis, Battista Biggio, Davide Maiorca, Giorgio Giacinto, Claudia Eckert, and Fabio Roli. Adversarial Malware Binaries: Evading Deep Learning for Malware Detection in Executables. 2018. URL `https://arxiv.org/pdf/1803.04173.pdf`.

Felix Kreuk, Assi Barak, Shir Aviv-Reuven, Moran Baruch, Benny Pinkas, and Joseph Keshet. Adversarial Examples on Discrete Sequences for Beating Whole-Binary Malware Detection. 2018. URL `http://arxiv.org/abs/1802.04528`.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial Machine Learning at Scale. In *International Conference on Learning Representations*, 2017a. URL `http://arxiv.org/abs/1611.01236`.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *International Conference on Learning Representations (ICLR)*, 2017b. URL `http://arxiv.org/abs/1607.02533`.

Daniel Lowd and Christopher Meek. Good Word Attacks on Statistical Spam Filters. In *Conference on email and anti-spam (CEAS)*, pages 125–132, 2005a. URL `http://www.utdallas.edu/~muratk/courses/dmsec_files/125.pdf`.

Daniel Lowd and Christopher Meek. Adversarial Learning. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 641–647, New York, NY, USA, 2005b. ACM. ISBN 1-59593-135-X. doi: 10.1145/1081870.1081950. URL `http://doi.acm.org/10.1145/1081870.1081950`.

Mohammad M. Masud, Latifur Khan, and Bhavani Thuraisingham. A scalable multi-level feature extraction technique to detect malicious executables. *Information Systems Frontiers*, 10(1):33–45, 3 2008. ISSN 1387-3326. doi: 10.1007/s10796-007-9054-3. URL `http://link.springer.com/10.1007/s10796-007-9054-3`.

Razvan Pascanu, Jack W Stokes, Hermineh Sanossian, Mady Marinescu, and Anil Thomas. Malware Classification With Recurrent Networks. IEEE - Institute of Electrical and Electronics Engineers, 4 2015. URL `http://research.microsoft.com/apps/pubs/default.aspx?id=249072`.

Edward Raff and Charles Nicholas. Malware Classification and Class Imbalance via Stochastic Hashed LZJD. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, AISec '17, pages 111–120, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5202-4. doi: 10.1145/3128572.3140446. URL `http://doi.acm.org/10.1145/3128572.3140446`.

Edward Raff, Richard Zak, Russell Cox, Jared Sylvester, Paul Yacci, Rebecca Ward, Anna Tracy, Mark McLean, and Charles Nicholas. An investigation of byte n-gram features for malware classification. *Journal of Computer Virology and Hacking Techniques*, 9 2016. ISSN 2263-8733. doi: 10.1007/s11416-016-0283-1. URL `http://link.springer.com/10.1007/s11416-016-0283-1`.

Edward Raff, Jon Barker, Jared Sylvester, Robert Brandon, Bryan Catanzaro, and Charles Nicholas. Malware Detection by Eating a Whole EXE. In *AAAI Workshop on Artificial Intelligence for Cyber Security*, 10 2018. URL `http://arxiv.org/abs/1710.09435`.

Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox: A Python toolbox to benchmark the robustness of machine learning models. 2017. URL `http://arxiv.org/abs/1707.04131`.

Paolo Russu, Ambra Demontis, Battista Biggio, Giorgio Fumera, and Fabio Roli. Secure Kernel Machines Against Evasion Attacks. In *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*, AISec '16, pages 59–69, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4573-6. doi: 10.1145/2996758.2996771. URL `http://doi.acm.org/10.1145/2996758.2996771`.

Justin Sahs and Latifur Khan. A Machine Learning Approach to Android Malware Detection. In *2012 European Intelligence and Security Informatics Conference*, pages 141–147. IEEE, 8 2012. ISBN 978-1-4673-2358-1. doi: 10.1109/EISIC.2012.34. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6735264`.

Joshua Saxe and Konstantin Berlin. Deep Neural Network Based Malware Detection Using Two Dimensional Binary Program Features. *CoRR abs/1508.03096*, 8 2015. URL `http://arxiv.org/abs/1508.03096`.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014. ISBN 1549-9618. doi: 10.1021/ct2009208. URL `http://arxiv.org/abs/1312.6199`.

Xabier Ugarte-Pedrero, Davide Balzarotti, Igor Santos, and Pablo G Bringas. RAMBO: Run-Time Packer Analysis with Multiple Branch Observation. In *Proceedings of the 13th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment - Volume 9721*, DIMVA 2016, pages 186–206, New York, NY, USA, 2016. Springer-Verlag New York, Inc. ISBN 978-3-319-40666-4. doi: 10.1007/978-3-319-40667-1{_}10. URL http://dx.doi.org/10.1007/978-3-319-40667-1_10.

Scott Wen-tau Yih, Joshua Goodman, and Geoff Hulten. Learning at Low False Positive Rates. In *Proceedings of the 3rd Conference on Email and Anti-Spam*. CEAS, 7 2006. URL https://www.microsoft.com/en-us/research/publication/learning-at-low-false-positive-rates/.

Xiaoyong Yuan, Pan He, Qile Zhu, Rajendra Rana Bhat, and Xiaolin Li. Adversarial Examples: Attacks and Defenses for Deep Learning. *arXiv*, 2017. URL http://arxiv.org/abs/1712.07107.

Yan Zhou and W Meador Inge. Malware Detection Using Adaptive Data Compression. In *Proceedings of the 1st ACM Workshop on Workshop on AISec*, AISec '08, pages 53–60, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-291-7. doi: 10.1145/1456377.1456393. URL http://doi.acm.org/10.1145/1456377.1456393.

Yan Zhou, Zach Jorgensen, and Meador Inge. Combating Good Word Attacks on Statistical Spam Filters with Multiple Instance Learning. In *19th IEEE International Conference on Tools with Artificial Intelligence(ICTAI 2007)*, pages 298–305. IEEE, 10 2007. ISBN 0-7695-3015-X. doi: 10.1109/ICTAI.2007.120. URL http://ieeexplore.ieee.org/document/4410395/.