Black-Box Attacks Using Data Augmentation Techniques

Po-Kai Chang (pc38) Renhao Lei (rl70) Yu Wang (yw88) Yuyu Qian (yq17)

Abstract

Deep neural networks (DNN) have been successful in classification tasks for different domains. However, none of these models can perfectly handle adversarial attacks so far. Out of those attack types, there is one type called "black-box." One approach to do black-box attacks is to train a substitute model to generate adversarial samples to fool the target model. We devised two black-box attacks variants by introducing two novel data augmentation techniques in generating the training data of the substitute model – Jacobian-based Dataset Augmentation and traditional image Dataset Augmentation. Though we found little difference in terms of transferability in our experiments between applying these two approaches and applying the original Jacobian-based method, our approaches have their respective strengths. Jacobian-based Dataset Augmentation has more intuitive geometric meaning. Traditional image Dataset Augmentation requires much fewer queries to the target model.

1 Introduction

Neural networks consistently misclassify adversarial examples, which exposes the vulnerability of classifiers. Adversarial examples are crafted inputs with small perturbations added to original inputs, remaining correctly classified by a human observer. As Ian[2] points out, the primary reason why neural networks are vulnerable to adversarial examples is their linear nature.

One classic example of adversary attack in real life is autonomous vehicles. Normally when an autonomous vehicle encounters stop signs, it is expected to recognize the sign and stop. However, if the stop sign was crafted, the model inside the vehicle would fail to classify it as the stop sign. Adversary attacks may cause severe accidents under this circumstance.

There are two types of adversary attack: white-box attack and black-box attack. Since a white box attack requires knowledge of model architecture, its application scenario is very limited. In this paper we mainly focused on black-box attacks.

In section 2, we introduce the threat model, including target model, adversarial capabilities and adversarial goal. In section 3, we explain our black-box attack strategy, which includes two steps: substitute model training with small input dataset and adversarial sample crafting. In order to obtain the substitute model with limited data, we design two approaches by introducing synthetic data generation techniques, the Jacobian-based Dataset Augmentation and traditional image Dataset Augmentation. Section 4 shows that one of our methods requires less computational power and much fewer queries to the target model while keeping similar performances compared with an approach designed by Papernot et al.[1]. Section 5 concludes our work.

2 Threat Model

In our work, the adversaries attempt to force the classifier to misclassify a data point to any category other than its correct category. In our scenario, we limit the access adversaries have to the output label of a target DNN only. Adversaries do not know the architecture of the target DNN, including the number, type and size of layers, and the training data used to train parameters. This kind of attack is called a black box attack, where adversaries can attack the system without knowing its internal details.

Target Model: The adversaries target a multi-class DNN classifier, which outputs a probability vector that indicates its belief of the given data point being categorized into predefined classes.

Adversarial capabilities: We name the targeted DNN oracle O, which produces labels for any given inputs. Adversaries can only access the label $\tilde{O}(\vec{x})$ for any input $\vec{x}$ by querying oracle O. The output label $\tilde{O}(\vec{x})$ is the index of the class assigned the largest probability by the DNN:

$$\tilde{O}(\vec{x}) = \arg \max_{j \in 0..N-1} O_j(\vec{x}) \quad (1)$$

where $O_j(\vec{x})$ is the j-th component of the probability vector $O(\vec{x})$ output by DNN O.

Adversarial Goal: We want to produce a minimally altered version of any input $\vec{x}$ (named adversarial example, denoted as $\overrightarrow{x^*}$) misclassified by oracle $O : \tilde{O}\left(\overrightarrow{x^*}\right) \neq \tilde{O}(\vec{x})$. An adversarial sample aims to solve the following optimization problem:

$$\overrightarrow{x^*} = \vec{x} + \arg\min\{\vec{z} : \tilde{O}(\vec{x} + \vec{z}) \neq \tilde{O}(\vec{x})\} = \vec{x} + \delta_{\vec{x}} \quad (2)$$

3 Black-box Attack Strategy

Our strategy is to train a substitute model that learns the decision boundary of the target model by feeding it a synthetic dataset generated by the adversary and labeled by the oracle. Then, the adversary uses the Fast Gradient Sign Method to craft adversarial examples based on the substitute model. The target DNN is expected to misclassify them due to transferability between architectures[2,4].

To understand the difficulty of implementing an attack under this threat model, recall that Equation 2 formalizes the adversarial goal, that is, to find a minimal disturbance that forces the target Oracle to misclassify. When the target is a non-convex ML model, a closed-form solution cannot be found: such as DNN. Most adversarial attacks [2,3,4] are based on the use of gradient-based optimization to approximate the solution according to the function defined by the DNN. Because the evaluation of these functions and their gradients requires an understanding of the DNN architecture and parameters, such an attack is impossible in our black box scenario. The results show that if the adversary can obtain independently collected labeled training sets from the same population distribution as Oracle, they can train a model with a different architecture and use it as a substitute model [4]. However, many modern machine learning systems require large and expensive training sets for training. For example, we consider a model trained with tens of thousands of labeled examples. This makes attacks based on this paradigm infeasible for adversaries who do not have large labeled data sets.

In this paper, we prove that the black-box attack can be done at low cost. In our method, the adversary only needs a small amount of input data as the initial dataset which is labeled by observing the Oracle. Then the adversary applies one of our data augmentation methods to expand the dataset. Using this synthetic data set, the adversary constructs a substitute F of the Oracle O. Then, the substitute network F is used to create adversarial samples, with sufficient knowledge of the substitute DNN parameters. The adversary can use one of the previously described attacks [2,3] to create adversarial samples. As long as there is transferability between F and O, adversarial samples made by the adversary for F should also be misclassified by O. This leads us to the following strategy:

Substitute Model Training: The attacker uses synthetic inputs to query oracle to build a model F close to the decision boundary of oracle model O.

Adversarial Sample Crafting: The attacker uses the substitute network F to make adversarial samples. Due to the transferability of adversarial samples, it is expected that Oracle O will misclassify these samples.

3.1 Substitute Model Training

Training a substitute model F approximating oracle O is challenging because we must: (1) select an architecture for our substitute without any prior knowledge of the targeted oracle's architecture, and (2) limit the number of queries made to the oracle in order to ensure that the approach is tractable. We design two approaches to overcome these challenges mainly by introducing synthetic data generation (i.e. data augmentation) techniques, the Jacobian-based Dataset Augmentation and traditional image Dataset Augmentation. Please note that this technique is not designed to maximize the substitute DNN's accuracy but rather ensure that it approximates the oracle's decision boundaries with few label queries.

Substitute Architecture: This factor is not the most limiting as the adversary must at least have some partial knowledge of the oracle input (e.g., images, text) and expected output. The adversary can thus use an architecture adapted to the input-output relation. For instance, a convolutional neural network is suitable for image classification. Furthermore, previous work shows[1] that the type, number, and size of layers used in the substitute DNN have relatively little impact on the success of the attacks. Adversaries can also consider performing an architecture exploration and train several substitute models before selecting the one yielding the highest attack success.

Generating a Synthetic Dataset: In theory, we can perform an unlimited number of queries to obtain oracle's output $\tilde{O}(\vec{x})$ for any input $\vec{x}$ belonging to the input domain. This will provide us with a copy of Oracle. In practice, however, this is not feasible: Consider a DNN with M input components, each component takes a discrete value in a set of K possible values, and the number of possible inputs to be queried is K^M. For input in the continuous domain, this difficulty is more obvious. In addition, a large number of queries make adversarial behaviors easy to detect. In order to solve this problem, we introduced two heuristic methods to study the input domain, as shown in Section 4, which greatly limits the number of Oracle queries. In addition, our technology also ensures that the substitute DNN is an approximation of the target DNN, that is, it learns similar decision boundaries.

The first heuristic used to generate synthetic training inputs is based on identifying directions in which the model's output is varying, around an initial set of training points. Such directions intuitively require more input-output pairs to capture the output variations of the target DNN O. Therefore, to get a substitute DNN accurately approximating the oracle's decision boundaries, the heuristic prioritizes these samples when querying the oracle for labels. These directions are identified with the substitute DNN's Jacobian matrix J_F, which is evaluated at several input points $\vec{x}$. Precisely, the adversary evaluates the Jacobian matrix dimension corresponding to the label assigned to input $\vec{x}$ by the oracle:

$J_F[\tilde{O}(\vec{x})]$. To obtain a new synthetic training point, a term $\lambda \cdot J_F[\tilde{O}(\vec{x})]$ is added to the original point $\vec{x}$. We name this technique Jacobian β, a Jacobian-based Dataset Augmentation. We base our substitute training algorithm on the idea of iteratively refining the model in directions identified using the Jacobian.

The second heuristic is a traditional image Dataset Augmentation method, called Data Generator. It takes in the initial dataset only once at the beginning and generates new images by adding small tweaks on the original images. This approach looks more intuitive, requires less computational power and much fewer queries to the target model but does the job pretty well. In Section 4 we see it produces comparable results during our experiment.

```
datagen = ImageDataGenerator(
    rotation_range=15,
    horizontal_flip=True,
    width_shift_range=0.1,
    height_shift_range=0.1
    )
```

In order to do the benchmarks, we also implemented a heuristic named Jacobian α mentioned in [1]. We show the comparison of their performances in Section 4.

Substitute DNN Training Algorithm: We now describe our training procedure as Algorithm 1 using Jacobian β and Algorithm 2 using Data Generator:

Algorithm 1: Substitute DNN Training: for oracle $\tilde{O}$, a maximum number $\max_\rho$ of substitute training epochs, a substitute architecture F, and an initial training set S_0.

input : $\tilde{O}, \max_\rho, S_0, \lambda$
Define architecture F;
for $\rho \in 0..\max_\rho - 1$ **do**
  // Label the substitute training set;
  $D \leftarrow \left\{(\vec{x}, \tilde{O}(\vec{x})) : \vec{x} \in S_\rho\right\}$;
  // Train F on D to evaluate parameters θ_F;
  $\theta_F \leftarrow \text{train}(F, D)$;
  // Perform Jacobian-based dataset augmentation;
  $S_{\rho+1} \leftarrow \left\{\vec{x} + \lambda \cdot \left(J_F[\tilde{O}(\vec{x})]\right) : \vec{x} \in S_\rho\right\} \cup S_\rho$;
end
return θ_F

Initial Collection (1): The adversary collects a very small set S_0 of inputs representative of the input domain. For instance, if the targeted oracle O classifies handwritten digits, the adversary collects 10 images of each digit 0 through 9. We show in Section 4 that this set does not necessarily have to come from the distribution from which the targeted oracle was trained.

Algorithm 2: Substitute DNN Training: for oracle $\tilde{O}$, a substitute architecture F, and an initial training set S_0.

input : $\tilde{O}, S_0, DataGenerator$
Define architecture F;
// Label the substitute training set;
$S_1 \leftarrow DataGenerator(S_0) \cup S_0$;
// Train F on S_1 to evaluate parameters θ_F;
return θ_F

Architecture Selection (2): The adversary selects an architecture to be trained as the substitute F.

Substitute Training: This is where the two algorithms differ from each other. The Data Generator does the data augmentation once and after that it's like a normal model training. On the other hand, the Jacobian β iteratively trains more accurate substitute DNNs F_ρ by repeating the following for $\rho \in 0..\rho_{\max}$:

Labeling (3): By querying for the labels $\tilde{O}(\vec{x})$ output by oracle O, the adversary labels each sample $\vec{x} \in S_\rho$ in its initial substitute training set S_ρ.

Training (4): The adversary trains the architecture chosen at step (2) using substitute training set S_ρ in conjunction with classical training techniques.

Augmentation (5): The adversary applies our augmentation technique on the initial substitute training set S_ρ to produce a larger substitute training set $S_{\rho+1}$ with more synthetic training points. This new training set better represents the model's decision boundaries. The adversary repeats steps (3) and (4) with the augmented set $S_{\rho+1}$.

Step (3) is repeated several times to increase the substitute DNN's accuracy and the similarity of its decision boundaries with the oracle. We introduce the term substitute training epoch, indexed with, to refer to each iteration performed. This leads to this formalization of the Jacobian-based Dataset Augmentation performed at step (5) of our substitute training algorithm to find more synthetic training points:

$$S_{\rho+1} = \left\{\vec{x} + \lambda \cdot \left(J_F[\tilde{O}(\vec{x})]\right) : \vec{x} \in S_\rho\right\} \cup S_\rho \qquad (3)$$

where λ is a parameter of the augmentation: it defines the size of the step taken in the sensitive direction identified by the Jacobian matrix to augment the set S_ρ into $S_{\rho+1}$.

3.2 Adversarial Sample Crafting

Once the adversary trained a substitute DNN, it uses it to craft adversarial samples. This is performed by implementing the previously introduced approach described in [2]. We provide an overview of the approach, namely the Goodfellow et al. algorithm.

Goodfellow et al. algorithm: This algorithm is also known as the fast gradient sign method [2]. Given a model F with

an associated cost function $c(F, \vec{x}, y)$, the adversary crafts an adversarial sample $\overrightarrow{x^*} = \vec{x} + \delta_{\vec{x}}$ for a given legitimate sample $\vec{x}$ by computing the following perturbation:

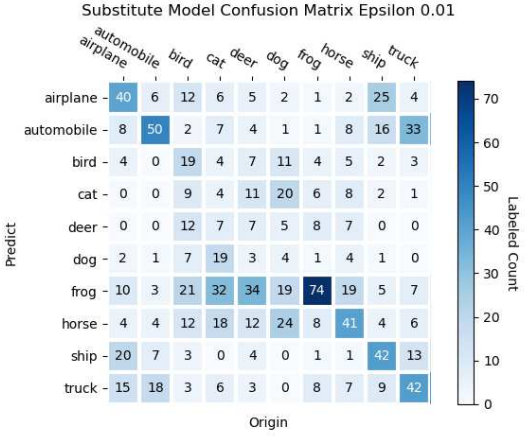
$$\delta_{\vec{x}} = \varepsilon \operatorname{sgn}\left(\nabla_{\vec{x}} c(F, \vec{x}, y)\right) \tag{4}$$

where perturbation $\operatorname{sgn}\left(\nabla_{\vec{x}} c(F, \vec{x}, y)\right)$ is the sign of the model's cost function 2 gradient. The cost gradient is computed with respect to $\vec{x}$ using sample $\vec{x}$ and label y as inputs. The value of the input variation parameter ε factoring the sign matrix controls the perturbation's amplitude. Increasing its value increases the likelihood of $\overrightarrow{x^*}$ being misclassified by model F but on the contrary makes adversarial samples easier to detect by humans. Feel free to check our repo here[1].

4 Validation of The Attack

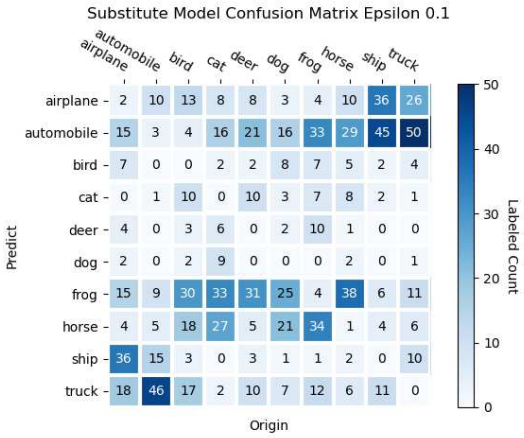
4.1 Confusion Matrix

As mentioned in the previous chapters that Epsilon is used to measure the extent our attacking will change the original image to. When the Epsilon equals to 0.01, we could tell from the diagram that most samples will be labeled correctly since most of the cases are in the diagonal.

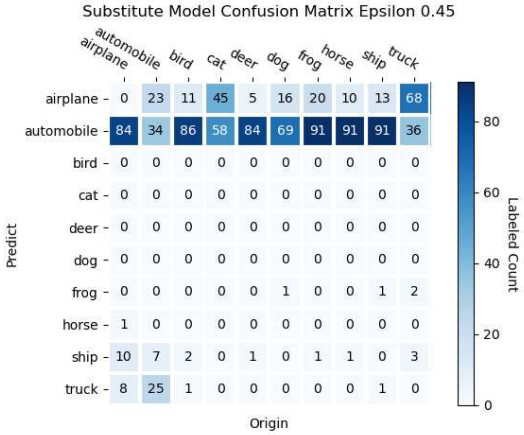


But with the increasing of the Epsilon, even at 0.1, the confusion matrix will show that predication will scatter throughout the whole matrix table and indicating the high mislabeling rate of the defender and attacking is being successful.

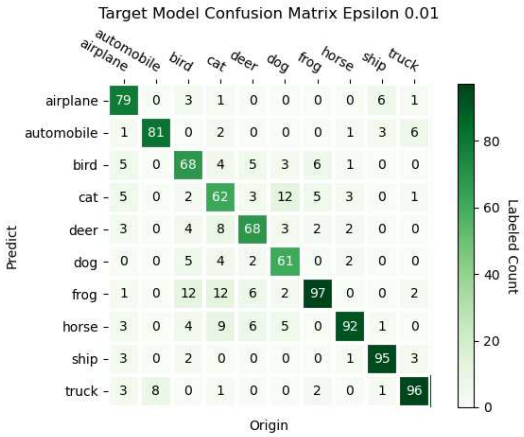
[1]https://github.com/RogerLei710/COMP-576-Final-Proj

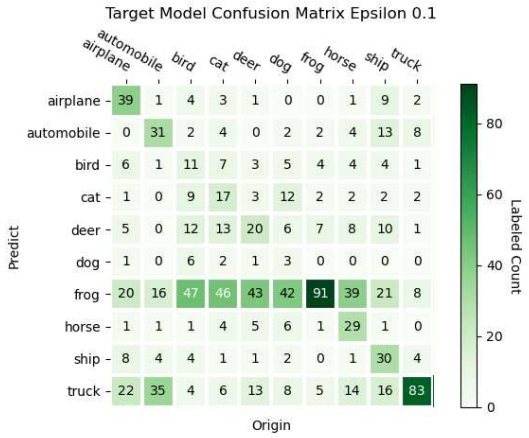


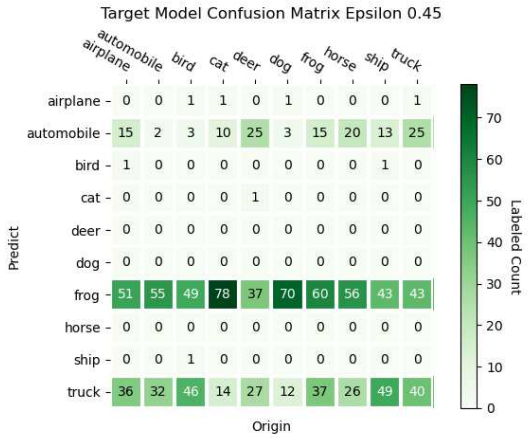
And when the Epsilon equals 0.45, all the samples will be labeled as either airplane or automobile in substitute model.

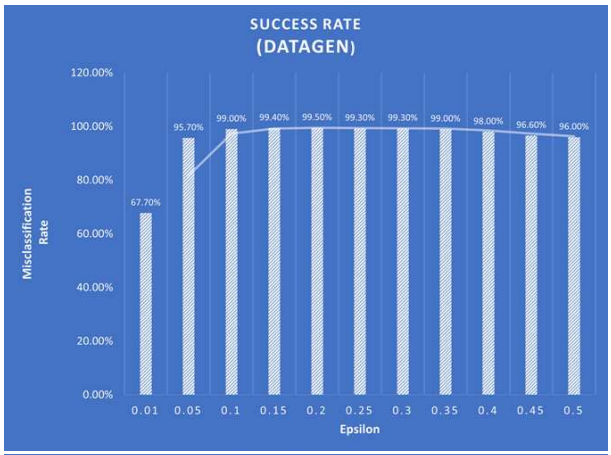


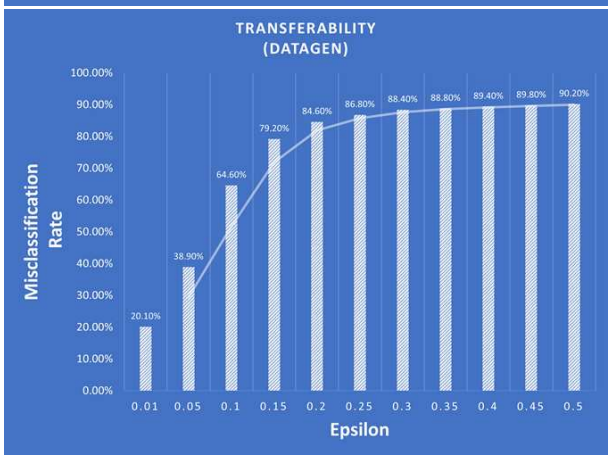
Yet, we could see though the tendency will be alike but the final label in the target model will be different from our substitute model and it is not unexpected since our substitute model is just a simple model mimicking the target model and the structure of the could be really different and would perform differently under the same circumstance.











4.2 Success Rate And Transferability

Let us discuss the success rate and transferability. The success rate is the proportion of adversarial samples misclassified by the substitute DNN. And the transferability of adversarial samples refers to the oracle misclassification rate of adversarial samples. We see that the success rate will not grow forever with Epsilon ranging from 0.01 to 0.5 and after the epsilon reaches 0.1 the success rate does not grow significantly and after it reaches 0.2, the success rate will drop. And the transferability will converge to 90%. The noticeable part is the dropping of the success rate and it is also explainable since with the modification rate, which is represented by Epsilon, becomes higher, the attacking samples will lose too much information and become random pixel generated images.

4.3 Oracle Query Number

This following equation is used to calculate the Oracle query number needed for the attacking.

$$\begin{aligned} OracleQueryNum = \{&(Dataset.size * trainRatio) \\ &if Heuristic == DataGenerator \\ &(Dataset.size * trainRatio) * 2^{\epsilon} \\ &if Heuristic == DataGenerator\} \end{aligned} \tag{5}$$

According to the equation, with the epoch number set 2, we could get the query number 2000, 8000, 8000. Normally, the less the queries, the less possibility the defender will detect the attack. According to the paper The two types of defense strategies are: 1. reactive where one seeks to detect adversarial examples, and 2. proactive where one makes the model itself more robust. And the more the oracle query the more likely the defender could detect the attack. Our attack requires a small amount of queries and may be distributed among a set of colluding users, but practically speaking, the defender could still increase the higher input dimensionality or modeling complexity to increase the query number.

5 Conclusion

In this work, we present two data augmentation methods to create train data for training a substitute network. Jacobian-based Dataset Augmentation has its strength in its simplicity – it is more interpretable than the original Jacobian-based method in the direction of perturbing an input data point. That is, going with the normalized direction of the labeled class in the Jacobian matrix should have higher probability to still make the image stay in the same class after perturbations. Traditional image Dataset Augmentation approach has its advantage in far fewer queries compared with any Jacobian-based methods so far. Since a Jacobian-based method would need to query oracle O for labeling whenever creating an augmented data point, the number of queries is the same as the additional data size augmented by this method. However, Traditional image Dataset Augmentation approach, in the scope of image classification, does not need any labeling of the additionally added data points. That is, since we are confident that an image should still be classified as what it is after rotations, moderate shifts, etc, the created data points do not require additional labeling queries to oracle O.

In our experiments of comparing different augmentation approaches' effects on the target model over different datasets, we found out that there is little difference between them in terms of transferability. Other than that, we can find the trend of each method in transferability when increasing the perturbation size is very similar. Also, we find that in Cifar10's dataset, there is an interesting phenomenon that a lot of perturbed images are mis-classified as the "frog" class. The root causes are not clear to us. The perturbed images do not intuitively seem to be more possible to be misclassified into the "frog" class, either.

In the future, we can explore more about other augmentation techniques to help approximate a target model's decision boundary. The discussion and exploration of the aforementioned misclassification phenomenon can also be done in the future. Since our experiments are only run on Cifar10 and Mnist, we also expect to have other experiments on other datasets – not restricted to image's domain.

References

[1] Nicolas Papernot, et al. Practical Black-Box Attacks against Machine Learning. In Proceedings of ACM Asia Conference on Computer and Communications Security, 2017

[2] Ian J Goodfellow, et al. Explaining and harnessing adversarial examples. In Proceedings of the International Conference on Learning Representations, 2015.

[3] Nicolas Papernot, et al. The limitations of deep learning in adversarial settings. In Proceedings of the 1st IEEE European Symposium on Security and Privacy, 2016.

[4] Christian Szegedy, et al. Intriguing properties of neural networks. In Proceedings of the International Conference on Learning Representations, 2014.

[5] K. Eykholt, et al. Robust physical-world attacks on deep learning models. ArXiv: 1707.08945, 2017.

[6] M. Sharif, et al. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, ACM, Vienna, Austria, pp. 1528–1540, 2016.

[7] Zhixuan Zhou, et al. Fake News Detection via NLP is Vulnerable to Adversarial Attacks

[8] Seyed-Mohsen, et al, Universal adversarial perturbations, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017