MGAN: PERPLEXING IMAGE RECOGNITION SYSTEMS USING MUTATION GENERATIVE ADVERSARIAL NETWORKS

Rohan Acharya
Horace Greeley High School
Chappaqua, NY
rohan.acharya@outlook.com

Xiaokui Shu
IBM T.J. Watson Research Center
Yorktown Heights, NY
Xiaokui.Shu@ibm.com

ABSTRACT

While deep neural networks (DNNs) have been successfully applied to solve problems in a variety of fields, they are vulnerable to carefully-crafted inputs known as adversarial samples. Several defense mechanisms have been proposed to detect adversarial samples at runtime by randomly mutating the target model and measuring the classification consistency among mutants. In this paper, we discover a vulnerability that can be exploited by adversaries and hope to encourage security researchers to patch DNNs for safety-critical image recognition applications, such as autonomous cars, ATMs, and medical imaging analysis systems. We propose MGAN, a novel framework that employs a generative adversarial network (GAN) to generate robust adversarial samples. We wire mutants as a third component in the original two-party architecture of GANs, which guides the generation of mutation-consistent adversarial samples. We evaluated our framework on the MNIST dataset of handwritten digits. Our results show that our approach allows for the creation of adversarial samples that can bypass mutation testing while maintaining a high accuracy.

1 Introduction

Deep learning has been widely successful in a range of applications such as image classification [1], speech recognition [2], and object detection [3]. As deep neural networks have pervaded our lives and made the transition from the lab to the real-world, security concerns pose a great threat. Szegedy et al. discovered that state-of-the-art deep neural networks were vulnerable to perturbed inputs that induce erroneous predictions [4]. Since then, many different methods have been proposed to generate such adversarial samples. These include the fast gradient sign method (FGSM) [5] and optimization methods such as the CW attack [6]. Xiao et al. were the first to apply generative adversarial networks (GAN) to the adversarial sample generation task [7]. Their AdvGAN consisted of (i) a feed-forward generator which generates perturbations to generate adversarial samples and (ii) a discriminator network which prevents the adversarial samples from being unrealistic. Numerous defense mechanisms have been proposed to detect adversarial samples, many of which take advantage of the sensitivity of adversarial samples to random mutations imposed on the target DNN [8].

In this paper, we apply model mutation testing to GANs at runtime in order to generate more robust adversarial samples. **Our purpose is to encourage security researchers to patch this vulnerability and improve the security of safety-critical image recognition systems.** Our contributions are listed as follows:

- We train MGAN to not only fool the target model, but also randomly mutated variations of the target. This results in adversarial samples that generalize better and cannot be detected by model mutation testing defense mechanisms.

- We show that MGAN maintains a similar accuracy to AdvGAN when evaluated on the MNIST dataset.

- We show that MGAN generates images that are more realistic than those generated by AdvGAN and less likely to be detected by model mutation testing defense mechanisms.

2 Background

I Neural Networks

Neural networks are functions *F(x) = y* that take an input x $\in R^n$ and produce an output $y \in R^m$. The model *F* is dependent upon model parameters θ; in the context of adversarial sample generation, the target neural network is fixed, so we will not go into the dependence on θ.

In this paper, we will focus on convolution neural networks (CNN), which are becoming increasingly popular for the task of *n*-class image classification [1]. The output of a CNN is calculated using the softmax function, defined as:

$$softmax(x_i) = \frac{exp(x_i)}{\sum_m exp(x_m))}.$$

This ensures that the output y satisfies $0 \leq y_i \leq 1$ and $y_1 + ... + y_m = 1$. As a result, output vector y is thus treated as a probability distribution; in other words, y_i is treated as the probability that input *x* has class *i* [6]. The classifier makes its prediction $P(x) = \text{argmax}\, F(x)$. A neural network typically consists of layers:

$$F = softmax \circ F_n \circ F_{n-1} \circ ... \circ F_1$$

where

$$F_i(x) = \sigma(\theta_i \cdot x) + \hat{\theta}_i$$

for some non-linear activation function σ, some matrix θ_i of model weights, and some vector $\hat{\theta}_i$ of model biases [6]. Together θ and $\hat{\theta}$ make up the model parameters. σ is usually sigmoid, tanh or ReLU [9]. In this paper we focus primarily on networks that use a ReLU activation function, as it currently is the most widely used activation function [6]. It is defined as $ReLU(x) = max(0, x)$.

The domain in which we will be evaluating MGAN is image classification. The input is a *h x w* gray-scale image which is a two-dimension vector x $\in R^{hw}$, where x_i is the intensity of pixel *i* and is scaled to the range [0,1].

II Adversarial Attack Algorithms

A Fast Gradient Sign Method

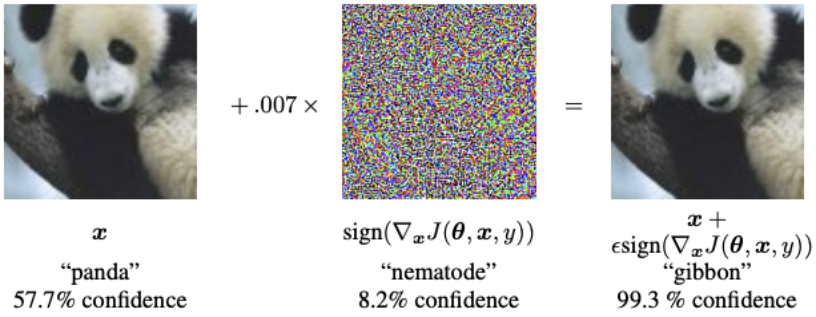

Figure 1: An example of the FGSM attack

Given an image x, FGSM finds adversarial example *x'* via

$$x' = x - \epsilon \cdot sign(\nabla loss_{F,t}(x)),$$

where ϵ is the attack step rate and t is the target label. For each pixel, FGSM calculates the gradient of the loss function to determine whether the pixel value should be increased or decreased and then shifts all of the pixels simultaneously [5].

B DeepFool

Moosavi-Dezfooli et al. proposed DeepFool to find the closest distance from the original input to the decision boundary of adversarial examples [10]. If f is a classifier, they used an iterative method to approximate the perturbation by considering f as linearized around x_i at each iteration. They computed the minimal perturbation as:

$$\underset{\eta_i}{\operatorname{argmin}} \|\eta_i\|$$
$$s.t. f(x_i) + \nabla f(x_i)^T \eta_i = 0$$

C AdvGAN

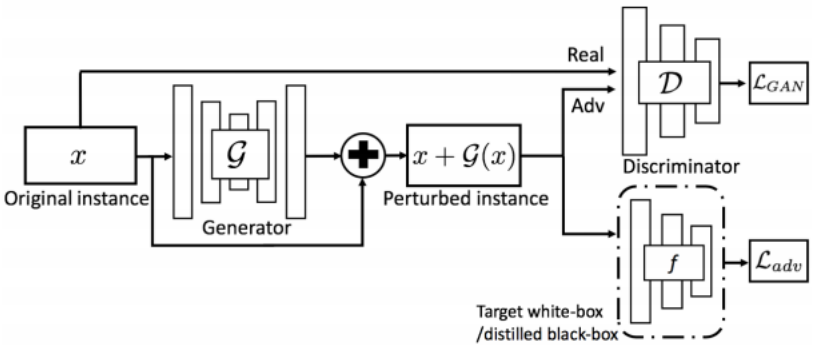

Figure 2: AdvGAN architecture

The AdvGAN framework consists of three components: a generator *G*, a discriminator *D*, and a target neural network *f*. As seen above, *G* takes in the original image x as input and generates perturbation *G(x)*. Then x + *G(x)* is fed into the discriminator in order to distinguish the generated image from x and encourage the generated image to be realistic [7]. The adversarial loss for a GAN is written as:

$$\mathcal{L}_{GAN} = \mathbb{E}_x \log D(x) + \mathbb{E}_x \log(1 - D(x + G(x)))$$

This loss function shows the discriminator's goal to distinguish the perturbed instance *x + G(x)* from the original instance *x*. Because real data is sampled from the true class, this also encourages generated instances to model data from the original class.

The loss for fooling the target model f in a targeted attack is:

$$\mathcal{L}_{adv}=\mathbb{E}_x \ell_f(x + G(x), t)$$

where t is the target class and ℓ_f denotes the loss function used to train the target model f. The $\mathcal{L}_{adv}$ loss encourages the generated image to be misclassified as the target class t. Additionally, a soft hinge loss is added on the L2 norm to limit the magnitude of the perturbation:

$$\mathcal{L}_{hinge}=\mathbb{E}_x max(0,\|G(x)\|-c)$$

Finally, the full objective can be expressed as:

$$\mathcal{L} = \mathcal{L}_{adv} + \alpha\mathcal{L}_{GAN} + \beta\mathcal{L}_{hinge}$$

where α and β control the relative importance of each objective.

3 MGAN Framework

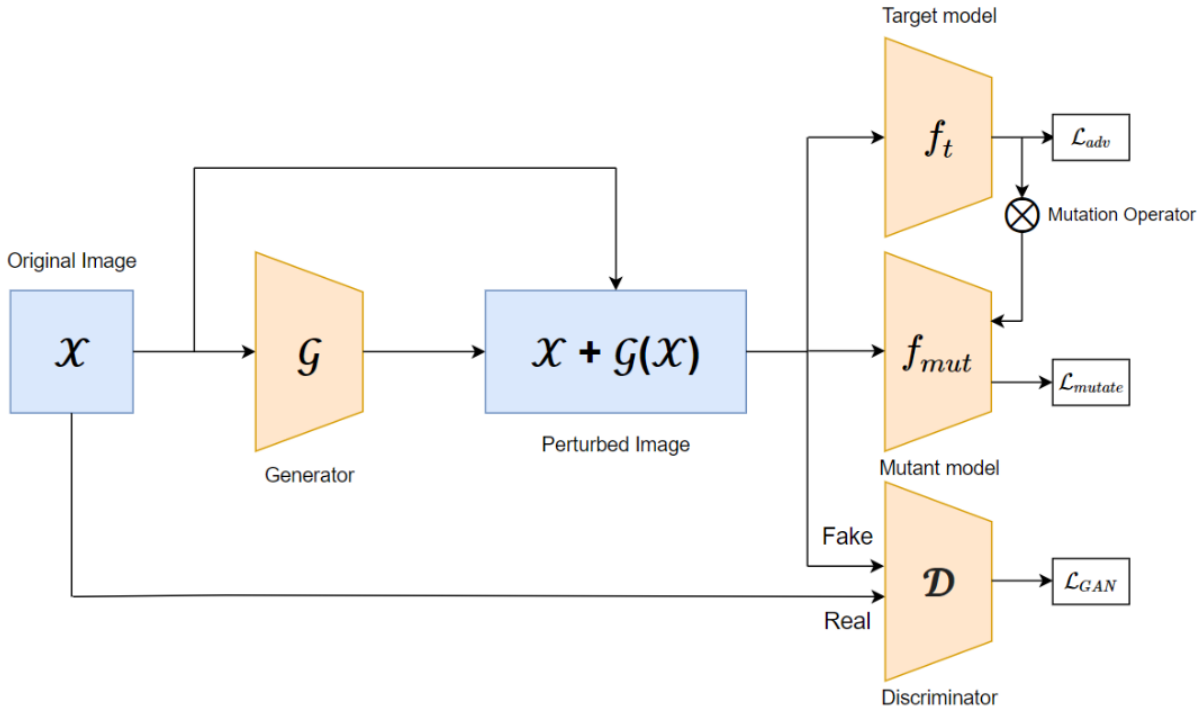

Figure 3: Architecture of MGAN

I Problem Definition

Let x $\in R^n$ be the input space, where n is the number of input features. If (x_i,y_i) is the ith instance within the training set, comprised of features $x_i \in \mathcal{X}$ which are generated according to unknown distribution x_i ~$\mathcal{P}_{data}$ with $y_i \in \mathcal{Y}$ denoting the correct labels, the target classifier attempts to learn f: $\mathcal{X} \rightarrow \mathcal{Y}$ from the domain of $\mathcal{X}$ (the training images) to $\mathcal{Y}$, the set of classification labels. Given a training instance x, MGAN's goal is to generate adversarial sample x' such

that $f(x') \neq y$. x' should also resemble x through the minimization of a mutation loss which we will later define. We show that this is a better alternative to the hinge loss used in AdvGAN.

II Model Mutation Operators

Ma et al. first proposed a mutation testing framework as a measure for the quality of test data used in deep learning systems [11]. By applying several mutation operators to a classifier, they injected faults into the model and evaluated the quality of test data based on the extent to which the injected faults could be detected. Wang et al. applied model mutation testing to the detection of adversarial samples and showed that adversarial samples had significantly higher label change rates than normal samples [8].

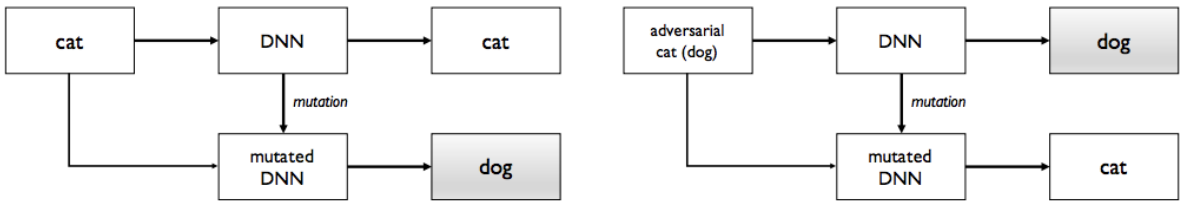

Figure 4: Label change rates of normal and adversarial samples against mutant DNN models (Wang et al.)

In this paper, we apply model mutation testing to improve a GAN's ability to generate robust adversarial samples with label change rates that are indistinguishable from those of normal samples. We will focus on four mutation operators [11]:

Neuron Activation Inverse (NAI): the non-linear behavior of DNNs is largely a result of the activation function used. Activation functions such as ReLU produce very different results depending on their activation status (input). The NAI operator changes the sign of the output of a neuron before it goes through the activation function. This not only results in drastically different outputs, but also allows for the creation of mutant neuron activation patterns which can reveal linear properties of DNNs.

Weight Shuffling (WS): neuron output is dependent upon neurons from the previous layers because of connections they share called weights. The WS operator selects a random neuron from a network and shuffles the weights of its connections to the previous layer. This often results in significantly different outputs, because the output of a DNN model is often determined by a few number of weights with extreme values.

Gaussian Fuzzing (GF): weights are often the parameter that most contributes to the decision logic of DNNs. The GF operator uses a Gaussian distribution $\mathcal{N}(\mu, \sigma)$ to mutate a weight w into a nearby value. For example, the weight will remain within 2σ with probability 95.46%.

Neuron Switch (NS): neurons in the layer of a DNN often have varying effects on the neurons they are connected to in the following layer. The NS switches two neurons within a layer to switch their influences on the next layer. This can be seen as a variation of Dropout, which punishes overfitted models.

III Training Procedure

The mechanism we use to train MGAN is very similar to that of AdvGAN. However, we wire mutants as a third component in the original two-party architecture of GANs, which guides the generation of mutation-consistent adversarial samples. Just like in all GAN architectures, we begin by calculating the adversarial loss:

$$\mathcal{L}_{GAN} = \mathbb{E}_x \log D(x) + \mathbb{E}_x \log(1 - D(x + G(x)))$$

Similar to the AdvGAN framework, we also calculate the adversarial loss, which encourages the generated samples to fool target neural network f:

$$\mathcal{L}_{adv} = \mathbb{E}_x \ell_f(x + G(x), t)$$

As a better alternative to the hinge loss on the L2 norm used in the AdvGAN framework, we propose a mutation loss which encourages the generator to generate adversarial samples that are consistent with random mutations imposed on f. We begin by generating a set of mutant models F and obtain the labels $f_i(x)$ for input samples $x \in \mathcal{X}$ on each mutated model $f_i \in F$. Wang et al. defined the label change rate (LCR) of sample x with respect to F as:

$$\varsigma(x) = \frac{|\{f_i | f_i \in F \text{ and } f_i(x) \neq f(x)\}|}{|F|}$$

where $\varsigma(x)$ measures how sensitive an input sample x is to random mutations of the target DNN model f.

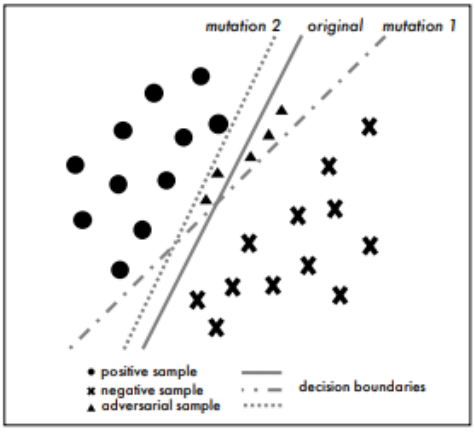

Figure 5: A visual representation of the model mutation testing effect (Wang et al.)

This observation shows us that given an input sample x, defense mechanisms can detect whether x is likely to be adversarial by calculating $\varsigma(x)$. Wang et al. showed that the LCR of adversarial samples is significantly higher than those of normal samples. MGAN attempts to find an adversarial sample x' such that $\varsigma(x') \approx \varsigma(x)$, where x is a normal

sample. MGAN's mutation objective can be summarized as:

$$\mathcal{L}_{mutate} = \mathcal{H}(f(x'), m(x'))$$

$$\underset{x'}{\operatorname{argmin}}\, \mathcal{L}_{mutate}$$

where $\mathcal{H}$ is the popular cross-entropy loss function, such that $\mathcal{H}(P, Q) = H(P) + KL(P\|Q)$ where $H(P)$ is the entropy of P and $KL(P\|Q)$ is the Kullback-Leibler Divergence between P and Q. Finally, we express out final objective as:

$$\mathcal{L} = \mathcal{L}_{adv} + \alpha\mathcal{L}_{GAN} + \beta\mathcal{L}_{mutate}$$

where α and β can be adjusted to trade off between accuracy and robustness.

4 Results and Discussion

In this section, we evaluate MGAN on the MNIST datatset, which consists of images of handwritten digits from 0-9, and compare its results to those of AdvGAN.

Implementation Parameters: We generate adversarial samples using the Adam optimizer with a batch size of 32 and learning rate 0.001. To make a fair comparison, both models are were trained identically and adversarial samples were generated using an L_∞ bound of 0.25. The target model being attacked was 5 hidden-layer convolutional neural network which achieved a 99.1% accuracy on the MNIST test dataset.

We evaluate MGAN's accuracy as well as its label change rate (LCR), or the percentage of adversarial samples that are classified differently by the target model after a mutation is applied. In Table 1, we show that MGAN's generated samples have label change rates nearly indistinguishable from those of normal images but still have a high accuracy. We also show that images generated by MGAN are more realistic than images generated by AdvGAN, despite AdvGAN's use of a hinge loss on the L2 norm of the perturbation.

Mutation Operator	MGAN Accuracy	AdvGAN Accuracy	MGAN LCR	AdvGAN LCR	Normal LCR
WS	96.83%	98.96%	12.21%	24.56%	11.52%
NAI	93.41%		44.83%	77.70%	41.50%
NS	98.21%		7.38%	18.39%	6.57%
GF	97.23%		16.21%	30.28%	15.67%

Table 1: Accuracy and label change rates of images generated by MGAN when trained using model mutation testing with several mutation operators. Accuracy for AdvGAN and label change rates for AdvGAN and normal samples are provided for a control benchmark.

The results show that adversarial samples generated by MGAN have a similar accuracy to those generated by AdvGAN, but have much lower label change rates. Additionally, the label change rates for MGAN's generated samples are nearly indistinguishable from those of normal samples taken from the original dataset.

MGAN was also able to produce very realistic adversarial samples, which we suspect is due to its objective of minimizing the mutate loss and encouraging the generation of mutation-consistent adversarial samples. Examples are shown below:

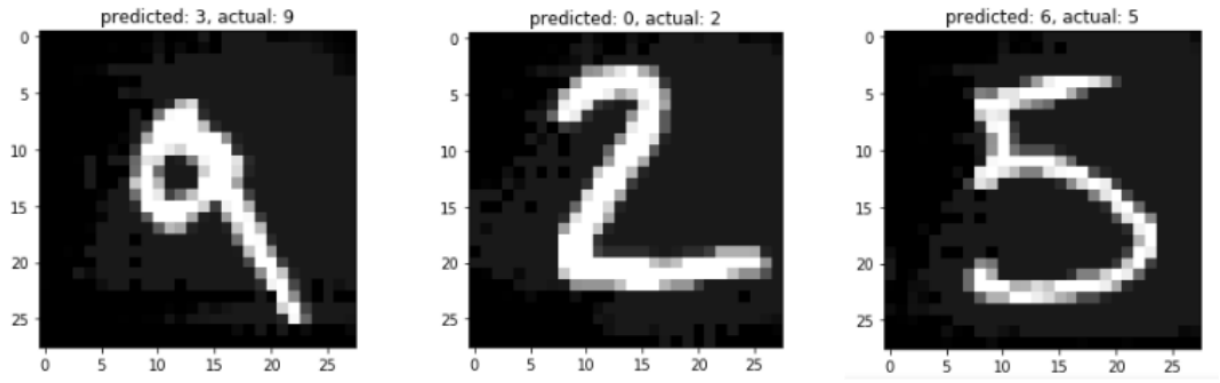

Figure 6: Adversarial samples generated by MGAN, along with the target model's predicted labels and actual labels

5 Conclusion

In this paper, we propose MGAN, a framework that utilizes generative adversarial networks to generate robust adversarial samples that are immune to model mutation testing. We demonstrate that adversarial samples generated by MGAN are 1) robust and mutation-consistent and 2) effective and have a high success rate. Our goal is to encourage security researchers to address this vulnerability. We suggest that they look into ways of creating classifiers that focus on robust features and are not affected by inconspicuous non-robust features [12]. For a classifier that aims to detect a dog, this would mean identifying features such as eyes and a snout as opposed to specific patterns in the pixel values. These non-robust features are what adversarial samples take advantage of. If we want our models to ignore them, we must introduce priors into the training process in order to steer them towards relying on robust features. In an increasingly AI-dependent world, this problem must be immediately addressed by the AI security community to prevent adversaries from hacking into safety-critical image recognition systems.

Acknowledgments

I would like to thank my mentor, Dr. Xiaokui Shu, for his unconditional support. He guided me through the initial literature search and provided me with crucial assistance in the brainstorming phase. I would also like to thank my high school science research teacher, Ms. Mary Rose Joseph, for giving me advice throughout the research process.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017.

[2] George Saon, Hong-Kwang J. Kuo, Steven Rennie, and Michael Picheny. The IBM 2015 English Conversational Telephone Speech Recognition System. *arXiv e-prints*, page arXiv:1505.05899, May 2015.

[3] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv e-prints*, page arXiv:1506.01497, Jun 2015.

[4] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv e-prints*, page arXiv:1312.6199, Dec 2013.

[5] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv e-prints*, page arXiv:1412.6572, Dec 2014.

[6] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. *arXiv e-prints*, page arXiv:1608.04644, Aug 2016.

[7] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating Adversarial Examples with Adversarial Networks. *arXiv e-prints*, page arXiv:1801.02610, Jan 2018.

[8] Jingyi Wang, Guoliang Dong, Jun Sun, Xinyu Wang, and Peixin Zhang. Adversarial Sample Detection for Deep Neural Network through Model Mutation Testing. *arXiv e-prints*, page arXiv:1812.05793, Dec 2018.

[9] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation Functions: Comparison of trends in Practice and Research for Deep Learning. *arXiv e-prints*, page arXiv:1811.03378, Nov 2018.

[10] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. DeepFool: a simple and accurate method to fool deep neural networks. *arXiv e-prints*, page arXiv:1511.04599, Nov 2015.

[11] Lei Ma, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Felix Juefei-Xu, Chao Xie, Li Li, Yang Liu, Jianjun Zhao, and Yadong Wang. DeepMutation: Mutation Testing of Deep Learning Systems. *arXiv e-prints*, page arXiv:1805.05206, May 2018.

[12] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial Examples Are Not Bugs, They Are Features. *arXiv e-prints*, page arXiv:1905.02175, May 2019.