# Wide ResNet Robustness with Adversarial Machine Learning

Johnny Li, E.J. Corpus, Nathan Cosendine, and Bin Hu

Old Dominion University

VICEROY Cybersecurity Research

September 13, 2023

Visual Supplement: **VICEROY Research**

**Abstract**

The world of artificial intelligence is expanding rapidly with the emergence of technologies like Llama 2, Chat-GPT 4, and Stable Diffusion XL, with many more expected by the end of this year. What do they all have in common? Well, they all rely on neural networks to extract patterns from data and produce results based on given prompts. However, the emphasis on security is often an afterthought, only considered when something goes wrong or when these models are used for malicious purposes. One innovative approach to address this problem is through adversarial machine learning. Adversarial machine learning involves training the model using carefully crafted inputs to help it adapt to perturbations—small and often intentional disturbances or changes introduced into the model. This allows the model to continue making reliable predictions without generating false results. Without adversarial machine learning, attackers can reduce the reliability of the model and manipulate its output predictions. In this paper, we will explore various neural network architectures, examine how perturbations affect neural networks, discuss the implementation of adversarial machine learning, and present our findings.

## Introduction

MobileNetV3, Transformers, DenseNet, Convolutional Neural Networks, Recurrent Neural Networks, and WideResNet represent some of the most well-known and widely used neural network architectures. Now, let's delve into each of them and explore their notable achievements.

MobileNetV3 is designed specifically for mobile and embedded devices, and its notable achievement lies in its remarkable efficiency. This architecture achieves this efficiency through a two-step process involving depthwise separable convolutions. The first step, known as depthwise convolution, applies separate filters for each input channel, thereby reducing the number of computations. The second step involves pointwise convolution, where a 1x1 convolution combines channel-wise information to capture relationships between channels. This design choice significantly reduces computational and memory requirements, making MobileNetV3 ideal for resource-constrained devices such as smartphones (Howard et al., 2019). Importantly, it strikes a delicate balance between accuracy and efficiency.

Transformers, primarily used in natural language processing tasks, have made significant strides in various language-related applications. Their architectural structure consists of an encoder-decoder framework, enriched by a mechanism called "self-attention." Self-attention allows the model to weigh the importance of different parts of the input text, similar to how humans pay attention to certain words based on context (Vaswani et al., 2017). Transformers often incorporate multiple "attention heads," working in parallel, to excel at capturing complex relationships within language. To account for the absence of inherent sequence order in

Transformers, they employ positional encoding, which provides crucial information about word positions within sentences.

DenseNet, short for Densely Connected Convolutional Networks, distinguishes itself through its dense connections between layers. This architecture organizes the network into dense blocks, wherein each layer not only feeds into the subsequent layer but also shares connections with all subsequent layers (Huang et al., 2017). This approach encourages feature reuse and facilitates gradient flow. Additionally, DenseNet often integrates bottleneck layers, represented by 1x1 convolutions, to control computational costs while preserving rich feature interactions.

Convolutional Neural Networks (CNNs) are tailored for visual data like images, structured in layers to extract meaningful features (Krizhevsky et al., 2012). CNNs employ convolution layers to detect features such as edges, shapes, and textures in input data. To enhance efficiency and invariance to small translations in the input, pooling layers are employed, reducing spatial dimensions in feature maps. Lastly, fully connected layers, typically used in the final stages of CNNs, enable predictions based on the extracted features, particularly in image classification tasks.

Recurrent Neural Networks (RNNs), employed for sequential data such as time series, text, or speech, are characterized by their hidden state. This hidden state carries information from the previous time step to the current one, allowing RNNs to capture dependencies in sequences. Nevertheless, RNNs can suffer from vanishing gradient issues when dealing with lengthy sequences, prompting the development of more advanced variants like LSTM and GRU, which mitigate this problem (Hochreiter & Schmidhuber, 1997).

WideResNet, a variation of ResNet known for its depth, focuses on widening the architecture rather than making it deeper. WideResNet achieves this by employing wider residual blocks with more channels than standard ResNets, enabling it to capture more diverse features from input data. This architectural choice effectively reduces overfitting, a common concern in very deep networks, by distributing the learning process across a broader range of features.

After evaluating various neural network architectures, we have selected WideResNet as the primary focus of our paper for several compelling reasons. WideResNet incorporates techniques such as dropout and weight decay, is known for its ability to enhance model generalization, and reduces susceptibility to overfitting. This proves particularly advantageous when dealing with limited training data. Additionally, WideResNet has exhibited greater resilience to certain types of noise and adversarial attacks when compared to other architectures. This robustness is particularly valuable in applications where input data may be noisy or corrupted. Furthermore, WideResNet can achieve competitive performance while demanding fewer computational resources than some alternative architectures. This is especially crucial in resource-constrained environments or on edge devices. These factors collectively make WideResNet an intriguing subject for adversarial machine learning research.

## Keywords

**Widths:**

The number of nodes/neurons or filters in each layer of a neural network. Wider layers increase the representational capacity and overall model complexity within each layer. Using more nodes/filters enables learning more complex features in each layer, but also increases computational cost and memory requirements.

**Depths:**

The number of layers in the overall neural network architecture. Deeper networks allow representing more hierarchical compositions of features, enabling learning of highly abstract concepts. However, deeper networks also increase the total number of trainable parameters, making overfitting during training more likely. The greater depth allows learning more complex feature combinations, but also increases training difficulty.

**Skip connection:**

It enables the gradient flow during training by creating a direct shortcut or path between layers in a neural network. Instead of passing the output of one layer directly to the next layer in a sequential manner, a skip connection creates a shortcut that directly connects an earlier layer to a later layer. This means that the output of a layer can bypass one or more intermediate layers and be added to the output of a later layer.

**Filters:**

Filters refer to the number of channels or feature maps used in the convolutional layers, and adjusting this number through the "widen factor" hyperparameter allows you to control the width and capacity of the network, which can impact its ability to capture complex features and patterns in data.

**Natural accuracy:**

The accuracy on clean, unperturbed examples from the original test set. This measures the model's standard performance on examples it was intended for.

**Robust accuracy:**

The accuracy on adversarially perturbed examples. This measures the model's robustness to adversarial attacks.

**Epsilon:**

The maximum L-norm distance of the adversarial perturbation from the original example. For example, epsilon of 0.01 for L-inf norm limits perturbations to -0.01 to 0.01 for each pixel in an image.

**Alpha:**

The step size used to update the adversarial perturbation in each step of an attack like PGD. Larger alpha allows finding stronger attacks faster.

**PGD steps:**

The number of gradient descent steps to take in the PGD adversarial attack algorithm. More steps find stronger adversarials but are slower to compute.

**Epochs:**

The number of complete passes through the training set during the training. More epochs improve natural and robust accuracy but take longer to train.

## Explaining the WideResNet architecture

The WideResNet architecture, proposed in 2016 by Sergey Zagoruyko and Nikos Komodakis, extends the ResNet by prioritizing width over depth to enhance accuracy (Zagoruyko & Komodakis, 2016). In a standard ResNet, residual blocks comprise convolutional layers with the same number of filters, connected by skip connections. However, WideResNet

widens these blocks by augmenting the number of filters in each convolutional layer, increasing model capacity and feature complexity. Customization of the architecture is achieved through control of depth (number of layers) and width (number of filters) via hyperparameters. The depth is determined by the number of stacked residual blocks, while the width is controlled by the "widen factor" denoted as "k." A higher "k" yields wider residual blocks. WideResNet often employs a bottleneck architecture in its residual blocks, involving three convolutional layers: 1x1, 3x3, and 1x1 convolutions. This design balances computational complexity with feature richness, with 1x1 convolutions handling dimensionality reduction and expansion, and the 3x3 convolution capturing spatial information.

For effective training, Batch Normalization precedes activation functions like ReLU in each convolutional layer, stabilizing input distributions and promoting faster convergence while mitigating the vanishing gradient problem. Skip connections (identity shortcuts) from ResNet are retained, facilitating gradient flow during training in very deep networks. Global Average Pooling (GAP) is the typical final pooling layer in WideResNet, replacing traditional fully connected layers. GAP computes spatial feature map averages, reducing model parameters and preventing overfitting. Weight initialization, such as He initialization, is critical in WideResNet to ensure effective convergence. Regularization techniques like dropout and weight decay are often employed to combat overfitting. Common optimization algorithms, including SGD with momentum and Adam, are used for training WideResNet models.

WideResNet's contributions include serving as a base model for ensemble learning, leading to performance boosts and robustness. Numerous architecture variants like WRN-SD and WRN-D introduce dropout and stochastic depth for improved training stability and generalization. Transfer learning with WideResNet on large datasets like ImageNet has led to

state-of-the-art results in various tasks. Its adaptability extends to applications like object detection, semantic segmentation, and generative modeling. Ongoing research explores the relationship between model width, depth, and performance, while the machine learning community has widely adopted WideResNet, with open-source implementations and pretrained models readily available for further development and application in diverse domains.

## Understanding Perturbations in Neural Networks

There are two types of adversarial attacks to increase loss in the model. The Projected Gradient Descent (PGD) attack was introduced by Madry et al. in their paper titled "Towards Deep Learning Models Resistant to Adversarial Attacks," which was presented at the International Conference on Learning Representations (ICLR) in 2018 (Madry et al., 2017). The authors include Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu.

The Projected Gradient Descent (PGD) attack is a popular adversarial attack method used to fool machine learning models, especially deep neural networks. It is a white-box attack, which means it assumes the attacker has access to the target model's architecture, parameters, and gradient information (Wang et al., 2021). The main goal of the PGD attack is to craft an adversarial example. An adversarial example is a slightly perturbed input data point that is very similar to a legitimate input but can cause the model to make incorrect predictions. The goal is to find a perturbed input x_adv that maximizes the loss $J(\theta, x_{adv}, y)$. The success of the PGD attack is measured by the model's incorrect prediction on the crafted adversarial example. If the model misclassifies the adversarial example, the attack is considered successful.

Notation:

Let x be the legitimate input to the model.

Let y be the true label associated with x.

Let θ represent the model's parameters.

Let J(θ, x, y) be the loss function used to train the model, where θ are the model parameters, x is the input, and y is the true label.

The PGD attack operates iteratively. In each iteration, the attacker:

- Computes the gradient of the loss with respect to the input: $\nabla_x J(\theta, x, y)$.

- Adversarially perturbs the input: $x = x + \alpha * \text{sign}(\nabla_x J(\theta, x, y))$, where α is a small step size.

- Clips the perturbed input to ensure it remains within the epsilon-ball around the original input: $x = \text{clip}(x, x - \varepsilon, x + \varepsilon)$.

- Repeats these steps for a predetermined number of iterations or until a stopping criterion is met (e.g., reaching a certain number of iterations or achieving a successful adversarial example).

The Fast Gradient Sign Method (FGSM) is a simple but effective white-box adversarial attack used to craft adversarial examples for machine learning models, particularly deep neural networks. This attack was introduced by Ian Goodfellow and his colleagues in their paper titled "Explaining and Harnessing Adversarial Examples" in 2015 (Goodfellow et al., 2014). The primary goal of the FGSM attack is to generate an adversarial example from a legitimate input that can cause the target model to make an incorrect prediction. The goal is to find a perturbed input x_adv that maximizes the loss J(θ, x_adv, y). The success of the FGSM attack is typically

measured by the model's incorrect prediction on the crafted adversarial example. If the model misclassifies the adversarial example, the attack is considered successful.

Notation:

Let x be the legitimate input to the model.

Let y be the true label associated with x.

Let θ represent the model's parameters.

Let J(θ, x, y) be the loss function used to train the model, where θ are the model parameters, x is the input, and y is the true label.

The FGSM attack operates in a single step. In this step, the attacker:

- Calculate the gradient of the loss with respect to the input: $\nabla_x J(\theta, x, y)$. This gradient tells us how sensitive the loss is to changes in each feature of the input.

- Compute the sign of the gradient: $\text{sign}(\nabla_x J(\theta, x, y))$. This gives us the direction in which the input should be perturbed to increase the loss.

- Craft the adversarial example by adding the scaled gradient to the original input:

$x_{adv} = x + \varepsilon * \text{sign}(\nabla_x J(\theta, x, y))$

These two attacks can lead to severe consequences and implications during exploitation and prevention events.

**Misclassification**

The primary goal of these attacks is to cause the model to misclassify input data. PGD and FGSM perturb the input data in such a way that the model's predictions are no longer accurate. In the world of art authentication, an adversarial attack could create convincing counterfeit paintings that fool art experts and collectors. This could lead to the circulation of fake artworks in the art market, devaluing genuine pieces and undermining the integrity of the art industry. Such attacks highlight the importance of robust authentication methods in the art world.

### Reduced Model Trustworthiness

Adversarial attacks undermine the trustworthiness of machine learning models. Users and developers may lose confidence in the model's predictions, which can limit its practical utility, especially in safety-critical applications like medical diagnosis or autonomous systems.

### Security Risks

In applications where machine learning models are used for security purposes, such as intrusion detection or spam filtering, PGD and FGSM attacks can be exploited by adversaries to evade detection or gain unauthorized access. For instance, an attacker might craft an adversarial example to bypass an email spam filter.

### Robustness Evaluation

PGD and FGSM attacks are often used to evaluate the robustness of machine learning models. By subjecting a model to these attacks, researchers can assess its vulnerability and identify areas for improvement. The consequences in this context can be positive, leading to more robust models when addressed appropriately.

### Overfitting Concerns

Defending against adversarial attacks can sometimes lead to overfitting. Models that are overly robust against adversarial examples may perform worse on clean, real-world data. Balancing robustness and generalization is a challenging trade-off in designing machine learning models.

### Increased Computational Complexity

Robustness against adversarial attacks often requires additional computational resources. Defensive techniques like adversarial training, which involves training the model on both clean and adversarial examples, can be computationally expensive and time-consuming.

### Transferability

Adversarial examples generated using PGD and FGSM on one model can sometimes transfer to other models, even those with different architectures. This means that an attack against one model can potentially be used to deceive multiple models, amplifying the consequences.

### Ethical Concerns

Adversarial attacks can raise ethical concerns when they are used maliciously or irresponsibly. For example, using adversarial techniques to manipulate content in deepfake videos, misinformation campaigns, or other malicious activities can have serious social and political consequences.

## Selecting PGD for Robustness Testing

For our experiment, we choose PGD for several reasons. PGD is an iterative attack, while FGSM is a single-step attack. PGD performs multiple iterations, gradually perturbing the input to maximize the loss. In contrast, FGSM computes the perturbation in a single step using the gradient which makes it less effective at finding optimal perturbations compared to iterative methods like PGD. The iterative nature of PGD allows it to explore a wider range of possible perturbations, making it more effective. Thus, PGD can generate more subtle and harder-to-detect adversarial examples. PGD is commonly used as a benchmark to evaluate the robustness of machine learning models. Models that can withstand PGD attacks are considered more robust. On the other hand, FGSM is often used as a baseline attack, and models that can defend against FGSM may still be vulnerable to more sophisticated attacks like PGD.

## Robustness Through Adversarial Machine Learning

To perform this experiment, we used xternalz's Wide-ResNet code and made changes to the code to adapt it for adversarial machine learning (Xternalz, 2017). Specifically, the code had many lines that were not suited for usage on a cluster, and it only provided standard training and clean testing. We removed the saving checkpoint and resume code during training because the code was not working properly, and we only had a time limit of 24 hours to run the code before the cluster was terminated. The tensorboard logging was removed because we only needed the final results, but we didn't need to visualize the process yet. After cleaning up the code, we added some code in to do adversarial machine learning. New arguments --epsilon, --pgd-steps, and --alpha were added to the argument parser. The pgd_attack function was added to perform adversarial attacks using Projected Gradient Descent (PGD). The train_adversarial and validate_adversarial functions were added to train and validate the model with adversarial

examples. Additional logic for adversarial training and validation was added to the main training loop.

Now, to compare the amount of lines in the code, we managed to reach from 283 lines in the original code to 218-220 lines in the adversarial code measuring natural accuracy and robust accuracy. This is about a 22% reduction, even though there were more new code added to conduct adversarial machine learning. Although the code was trimmed down, it did not offer faster training and testing because everything is reliant on the GPU, but having a smaller codebase allowed us to troubleshoot and fix bugs that were previously found. We also had some difficulties along the way. In the beginning, we used a 20 step PGD which caused a lot of training to go above 24 hours and ended up being canceled. Also, the parameters were updated during the generation of adversarial examples in the adversarial training. The model configurations were also large because we used higher depths and widths, making the model parameters massive. When generating adversarial examples, you want to perturb the input data to find the worst-case scenario for model misclassification. If you don't set the model to evaluation mode, it might track gradients, which is not necessary during adversarial example generation. Disabling gradient tracking reduces unnecessary computation and ensures that the perturbations you apply to the input data are not affected by the gradients. Setting the model to evaluation mode also ensures deterministic behavior. The same input will produce the same adversarial example across multiple runs, which is important for consistent evaluation.

To solve these problems, we had to lower the PGD to a step size of 10, reduce the model configurations to be a maximum of depth 40 and width 6, and add model.eval to ensure that the perturbations applied to the input data are not influenced by gradients and are based solely on the model's current parameter values. Model.eval() is used to prepare your model for inference or

evaluation tasks. It sets the model's behavior to be consistent with the assumptions of evaluation. If the model is not in evaluation mode, we might inadvertently exaggerate the model's vulnerability by introducing training-specific artifacts into the adversarial examples, which may not accurately reflect real-world adversarial attacks. So once the adversarial examples are generated, we can set the model back to training using model.train and repeat the process for each epoch. So, using the original code and modified code, we tested for standard training and natural accuracy, adversarial training and natural accuracy, and adversarial training and robust accuracy. Additionally, we also broke up the configurations into categories like same width and different depth, different depth and same width, and different depth and different width to measure accuracies across them.

## Identifying Optimal Configurations

Let’s talk about the main categories that we tracked and give some explanations. E.J had the greatest increment in model parameter size with the largest model having a total of 20,117,466 million parameters. In contrast, 13,144,794 and 8,949,210 were the highest model parameters for Nathan and Johnny, respectively. Models with a large number of parameters tend to be slower to train because they require more computation to update and optimize these parameters during the training process. As parameters increase, there is a correlation to longer training and testing times. Hence, this is reflected in the adversarial training and robust testing because it went over 24 hours. Hence, there is no result for it, due to the cluster expiring before it has finished entirely. Meanwhile, the goal of tracking parameters is to see if larger models can increase accuracy and remain competitive to smaller models within the constant of 24 hours. There is a delicate balance between parameters and training with limited resources, in our case, using one Tesla V100-SXM2-16GB for the GPU. The other categories include standard training

and clean testing, adversarial training and robust testing, and adversarial training and clean testing.

For regular training and testing, we got results in the high 90s. Looking at all of it, the accuracy increases across the board for the same widths and different depths, same depths and different widths, and different depths and widths. For example, WRN-10-1 and WRN-16-2 had the biggest jump of about 6.07% increase. WRN-34-5 and WRN-40-6 had the smallest increase of about 0.05%. Looking at these two extremes, we can tell that at smaller depths and smaller widths, the accuracies increase greatly. However, once the depths and widths become larger, they don't yield the same percentage increase, leading to a small improvement. This indicates that when choosing the model configuration of WRN-34-5 and WRN-40-6, the WRN-34-5 is the best one because it has fewer parameters and still maintains the accuracy at ~96%. The likewise is true for WRN-10-1 and WRN-16-2 because the WRN-16-2 has more parameters than WRN-10-1, but it still computes relatively fast, due to its small width, which is ideal for improving accuracy on smaller models. Once model configurations increase, there is a point when the increase is minimal and the training time is not reasonable while wasting valuable resources best suited for other tasks. I call this the gentle alteration because the accuracy hits a plateau where any improvement is little and stays relatively the same, like a serene sunset, slowly changing hues in the evening sky, with subtle shifts in colors that are barely perceptible. When optimizing the model, there are other options like fine-tuning, quantization, or pruning that can eke out accuracy while not resorting to retraining on larger and larger configurations until the data is overfitting.

For adversarial training and clean testing, the accuracies were still mostly in the 90s, but most of them had a 0.40-3.01% drop from the standard training and testing. This is expected

because the model is trained on adversarial examples, but it also is trained from clean data in the training dataset, so the accuracy is still high, but the model also is still evaluating on clean data in the validation set. Since the model is only evaluating on clean data, it is really similar to the standard training and clean testing results, but the drop in accuracy may show that the model may fail to evaluate some of the clean data, due to the tradeoff of having adversarial training evaluate on non-adversarial data. However, the slight decrease in accuracy proves that adversarial training can also have high accuracies when evaluating on clean examples, which since the model performs well on the clean data, it sets the stage for elevating the robustness and still works great by default without being evaluated on adversarial data. For WRN-10-4, it had the smallest decline of 0.40%, so it is both adversarial trained and evaluates almost as the standard testing and training. This is ideal because when both accuracies are relatively close, the configuration is at its peak in terms of classifying clean data and robustness. For WRN-40-4, as the model parameters were the greatest in Johnny's results, the accuracy actually dropped, so the model does not compete with WRN-40-3, even though the other categories did have a slight increase. Even though accuracy is a bit better in those categories, the decline in adversarial training and clean testing indicates that perhaps the WRN-40-3 is the better choice as it is competitive overall on all three accuracy categories. The lesson here is that in model selection, it's crucial to prioritize overall performance and balance between complexity and generalization rather than blindly chasing the model with the most parameters. As the saying goes, "a jack of all trades is a master of none, but oftentimes better than a master of one." Sometimes, compromising on the accuracy can make the model better in real-time usage which the close results between WRN-40-3 and WRN-40-4 suggests that favoring the smaller model can remain on par as WRN-40-4 and reduce the time to train the model.

For the adversarial training and robust testing, the accuracies are in 50s to 70s, but most of them have a 22.50% -30.09% drop from the standard training and testing. Now, this time there is a steep drop in accuracy which is due to the use of evaluation on adversarial examples after undergoing adversarial training. The PGD attack, being the tenth step in the process, generates highly potent perturbations specifically engineered to induce substantial loss and trigger a higher rate of misclassifications within the dataset. As it is tested on real adversarial examples, it has gained recognition for its robustness, given that the model is subjected to evaluation with actual adversarial data. This comprehensive testing approach ensures that the model's performance under adverse conditions is thoroughly examined. For WRN-35-4, it had the smallest decrease in accuracy of 22.50%. In the real world, this model would be highly selected for robustness, due to its high robust accuracy of 73.52%. In terms of security, it is always better to choose the highest one because it will help stop many attempts to fool the model. Imagine picking a model with lower security, but high accuracy versus one with high security and high accuracy. Almost everyone will flock to the latter option because no one wants to have a bad experience when using the model. Now, the WRN-10-1 is obviously the biggest decrease in accuracy of 30.09%. Compared to the WRN-35-4, it is 7.59% lower which is a huge gap in terms of robustness. This model would not be used in the real world because 58.21% is really close to a 6 out of 10 chance to classify correctly. The remaining 4 out of 10 still leave a possible room for improvement. Although it trains super fast, it compromises on robust accuracy.

The key takeaway from these results is that using adequate depths and widths to boost robustness is better than using smaller models. As the parameters increased 14,741.45%, the robust accuracy also increased which highlights a clear distinction that higher configurations induce robustness. So, in this case, it is better to prioritize longer training times by upping model

configurations to ensure that the model can learn better from the adversarial training in order to create a more robust model, but keep in mind that more resources will be needed to train large models. This presents a new type of problem that future research will address which is to increase robustness without needing to increase model parameters.

## Navigating the AI Robustness Discussion

### Simplicity vs. Complexity

Contrary to the prevailing trend of using large, complex models, some argue that simplicity and feature engineering can lead to more robust models (Gómez-Carmona et al., 2019). They believe that overly complex models are inherently less interpretable and harder to make robust. In our experiment, we used simple techniques to implement adversarial machine learning and found that maintaining a compact codebase can help set up the basis for robustness. As well as setting limits to model configurations and time constraint, it can help focus less on increasing model capacity and more on using existing models to explore further options to boost robustness. As models increase in parameters, they can also introduce more vulnerabilities to the model, so reducing the size of the model can be the first step in fostering robustness. That being said, sometimes different applications require larger model sizes, so it is inevitable that model sizes will increase, but remember that if the code and model parameters increase, equal effort should be put on making them secure and efficient, so they can be as lean as possible.

### Overconfidence in AI Robustness

Some believe there is an overconfidence in the current state of AI robustness. They argue that AI systems are still far from being truly robust in the face of real-world challenges and should be treated with caution in critical applications (Franzoni et al., 2019). In our experiment,

we achieved very high robust accuracies that even we are shocked that we managed to beat out some benchmarks on RobustBench. As with all results, treat them with lots of skepticism and run experiments on your own to verify the findings. Even then, look to see if some bugs exist or if the code is missing some other important pieces. Our goal is to reveal all of our contributions to the public and hopefully gather some discussion to improve upon research. Like all experiments, even failed experiments can turn into blueprints for more ambitious projects. Although robustness in adversarial machine learning is one metric to measure a model's ability to correctly classify perturbations, there are also other methods out there that don't need perturbations to successfully trick or bypass model restrictions. Carnegie Mellon University's recent research reveals a critical vulnerability in large language models (Zou et al., 2023). These models can be manipulated to generate unintended harmful responses by adding innocuous text to a query, surpassing basic safety measures. Adversarial attacks on LLMs can now be more methodical, using character sequences to trick the AI model. This underscores the need for improved safeguards and responsible AI use. Continuous efforts are essential to ensure AI systems remain safe and beneficial.

## Robustness Trade-offs Are Unavoidable

Critics suggest that there's an inherent trade-off between robustness and performance. Pushing for extreme robustness often leads to sacrificing too much in terms of accuracy and efficiency (Raghunathan et al., 2020). They argue that it's better to find a balance tailored to specific applications. Finding a balance between robustness and performance is akin to steering a ship through turbulent waters. Pursuing extreme robustness, while admirable in its intent to fortify against potential threats, can lead to unintended consequences. For instance, in the realm of machine learning, an overemphasis on robustness may result in models that are overly

cautious, leading to false alarms or a decline in overall accuracy. This is particularly evident in adversarial settings, where models can become excessively conservative, sacrificing their primary purpose of accurate predictions. Moreover, dedicating excessive resources to fortifying against every conceivable threat can be inefficient and costly. By seeking a tailored balance, one can address specific vulnerabilities without compromising the overall effectiveness and efficiency of the system. This approach allows for a more nuanced and practical response to the challenges posed by adversarial scenarios, ultimately resulting in more adaptable and versatile solutions. While robustness is very important, the actual usage of the model is another aspect to consider when putting it to the test, so if both of them can be best case scenario, then the model will be able to be secure and produce useful results.

## The Imperative Role of Adversarial Machine Learning

### Real-World Relevance

Adversarial machine learning demonstrates its importance by addressing real-world security concerns. In a world where machine learning is increasingly used in critical applications like autonomous vehicles, healthcare, and finance, adversaries can exploit vulnerabilities for malicious purposes. Adversaries can subtly modify road signs by adding small stickers or graffiti-like alterations that may not be noticeable to the human eye but can confuse the vehicle's image recognition system. As a result, the vehicle might misinterpret a stop sign as a yield sign or vice versa, potentially leading to accidents or traffic violations. In medical imaging, adversaries can introduce imperceptible noise or alterations to medical images, such as X-rays or MRI scans. These manipulated images can mislead AI-driven diagnostic systems into detecting non-existent diseases or missing real medical conditions, endangering patients' lives due to

incorrect treatment or diagnosis. In high-frequency trading, adversaries can employ adversarial algorithms to manipulate stock prices by placing a series of strategically timed buy or sell orders. These manipulations can create false market trends, potentially causing market crashes or unfair financial gains. Showing how adversarial attacks can compromise these systems highlights the need for robust defenses.

## Regulatory and Ethical Concerns

Adversarial attacks can highlight the need for regulatory frameworks and ethical considerations in machine learning. This can be particularly compelling for audiences who are concerned about the responsible development and deployment of AI technologies. The absence of comprehensive regulatory practices on AI can be attributed to several factors. Firstly, the rapid pace of technological advancement often outpaces the ability of regulatory bodies to keep up. AI technologies evolve quickly, and crafting regulations that remain relevant and effective is a complex and ongoing challenge. Additionally, the international nature of AI development complicates matters, as regulations in one jurisdiction may not align with those in another, creating potential conflicts and ambiguities. Furthermore, AI is a broad field with diverse applications, making it challenging to create one-size-fits-all regulations. Balancing innovation and regulation is a delicate act, as overly restrictive measures could stifle progress, while inadequate oversight may lead to ethical and safety concerns. As a result, the absence of comprehensive regulatory practices on AI reflects the need for a nuanced, collaborative, and adaptive approach that considers the unique challenges posed by this rapidly evolving technology.

## Exposing Vulnerabilities

Adversarial attacks expose vulnerabilities in machine learning models. This helps convince the audience that machine learning is not infallible and that it's crucial to invest in research and development to strengthen these systems. Just like stress-testing a bridge reveals weaknesses, adversarial attacks reveal flaws in AI models. Investing in research and development (R&D) to counter adversarial attacks is not just important; it is absolutely critical in our technology-driven world. Adversarial attacks continually evolve, becoming more sophisticated and dangerous. Whether it's the potential manipulation of autonomous vehicles, financial fraud through compromised algorithms, or misinformation spread via AI-generated content, the consequences can be far-reaching and devastating. Without robust R&D efforts, we risk falling behind malicious actors who exploit vulnerabilities in AI systems. By prioritizing research, we empower ourselves to stay ahead in the ongoing arms race against adversarial attacks. This proactive approach enables us to develop cutting-edge defenses, resilient algorithms, and innovative strategies to protect critical systems, data, and privacy, ensuring the continued advancement and safe adoption of AI technologies across various sectors. In an era where technology's influence grows daily, research is not just important; it's our best defense against the looming threat of adversarial attacks.

## Public Perception

Public perception plays a crucial role in the adoption of AI technologies. If the audience understands that adversarial attacks can erode trust in AI systems, they may be more inclined to support measures to improve security and reliability. Education and awareness campaigns are instrumental in shaping this perception. When the general public is well-informed about the potential vulnerabilities and consequences of adversarial attacks on AI systems, they are more likely to become advocates for robust security measures. This awareness not only empowers

individuals and organizations to make informed decisions about AI adoption but also instills a sense of responsibility among developers and users. They are less likely to fall victim to misinformation or fearmongering regarding AI. A well-informed public can discern between genuine concerns and unfounded fears, which can lead to a more balanced and constructive discourse surrounding AI adoption. They are more likely to engage in conversations and debates on AI ethics and safety. These discussions can drive innovation in security and reliability, fostering the development of new technologies and strategies to combat threats. Awareness of adversarial attacks can promote international cooperation and collaboration in addressing AI security challenges. As AI is a global technology, shared understanding and efforts to enhance security can lead to a more secure and reliable AI ecosystem on a global scale. In conclusion, public perception is not merely a passive factor in AI adoption; it is a dynamic force that can shape the industry's future and ensure that AI technologies meet the highest standards of security and reliability.

## References

Franzoni, Valentina & Vallverdu, Jordi & Milani, Alfredo. (2019). Errors, Biases and Overconfidence in Artificial Emotional Modeling. 86-90. 10.1145/3358695.3361749. https://doi.org/10.1145/3358695.3361749

Gómez-Carmona, O., Casado-Mansilla, D., López-de-Ipiña, D., & García-Zubía, J. (2019). Simplicity is Best: Addressing the Computational Cost of Machine Learning Classifiers in Constrained Edge Devices. *Proceedings of the 9th International Conference on the Internet of Things*. https://dl.acm.org/doi/10.1145/3365871.3365889

Goodfellow, I.J., Shlens, J., & Szegedy, C. (2014). Explaining and Harnessing Adversarial Examples. *CoRR, abs/1412.6572*. https://arxiv.org/abs/1412.6572

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735-1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735

Howard, A.G., Sandler, M., Chu, G., Chen, L., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q.V., & Adam, H. (2019). Searching for MobileNetV3. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 1314-1324. https://arxiv.org/abs/1905.02244

Huang, Gao & Liu, Zhuang & van der Maaten, Laurens & Weinberger, Kilian. (2017). Densely Connected Convolutional Networks. 10.1109/CVPR.2017.243. https://openaccess.thecvf.com/content_cvpr_2017/papers/Huang_Densely_Connected_Convolutional_CVPR_2017_paper.pdf

Krizhevsky, Alex & Sutskever, Ilya & Hinton, Geoffrey. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems*. 25. 10.1145/3065386. https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2017). Towards Deep Learning Models Resistant to Adversarial Attacks. *ArXiv, abs/1706.06083*. https://arxiv.org/abs/1706.06083

Raghunathan, A., Xie, S.M., Yang, F., Duchi, J.C., & Liang, P. (2020). Understanding and Mitigating the Tradeoff Between Robustness and Accuracy. *ArXiv, abs/2002.10716*. https://arxiv.org/abs/2002.10716

Vaswani, A., Shazeer, N.M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., & Polosukhin, I. (2017). Attention is All you Need. *NIPS*. https://arxiv.org/abs/1706.03762

Wang, Y., Ma, X., Bailey, J., Yi, J., Zhou, B., & Gu, Q. (2021). On the Convergence and Robustness of Adversarial Training. *International Conference on Machine Learning*. https://arxiv.org/abs/2112.08304

Xternalz. (2017). WideResNet-pytorch. GitHub. https://github.com/xternalz/WideResNet-pytorch

Zagoruyko, S., & Komodakis, N. (2016). Wide Residual Networks. *ArXiv, abs/1605.07146*. https://arxiv.org/abs/1605.07146

Zou, A., Wang, Z., Kolter, J. Z., & Fredrikson, M. (2023). Universal and transferable adversarial attacks on aligned language models. arXiv preprint arXiv:2307.15043. https://arxiv.org/abs/2307.15043