Artificial Intelligence Security: A systematic review of adversarial attacks in AI systems

Karan Goel

SCIT, EIS, University of Wollongong
kg956@uowmail.edu.au
Student Number: 7836685

1 Related works

In recent years, deep neural networks (DNNs) have been extensively employed across various domains, including computer vision [12], speech recognition [1], and natural language processing (NLP) [24], achieving significant advancements in both industry and academia. This success has spurred a widespread enthusiasm for artificial intelligence, particularly emphasizing the capabilities of deep learning.

Since 2004, there has been substantial growth in the development of adversarial techniques within machine learning despite these achievements, the security aspects of deep neural networks remain under explored and inadequately addressed [4]. Szegedy et al. [22] were among the first to highlight the susceptibility of neural networks to adversarial attacks. This revelation has catalyzed the study of adversarial methods in artificial intelligence into a burgeoning area of interest, with researchers continually proposing innovative attack and defense strategies.

1.1 Adversarial Attack

This section introduces analysis of different adversarial attack methods derived from consolidated research findings. The results are summarised in Table 1.

1.2 Common Terms

An adversarial attack targets deep neural networks through the use of adversarial examples. Depending on their attributes and effects, these attacks can be classified into various types, such as black-box attacks, white-box attacks, one-shot attacks, iterative attacks, targeted attacks, non-targeted attacks, specific perturbations, and universal perturbations. These categories are defined in the following manner:

Black-box attack: The attacker does not have access to the internals of the deep neural network model, including its structure and parameters. The attacker can only derive the output from the target model by inputting the original data.

White-box attack: In contrast, the attacker has complete access to the target model's structure and parameters, including training data, gradient information, and activation functions.

One-shot attack: This type of attack requires only a single computation to generate an adversarial example that has a high likelihood of misleading the target model.

Iterative attack: Unlike one-shot attacks, iterative attacks require multiple rounds of computation to produce adversarial examples. Although more time-consuming, they tend to be more effective.

Targeted attack: The adversarial examples are engineered to cause the target model to misclassify an input into a specific, erroneous category.

Non-targeted attack: Here, the goal is merely to induce an error in classification, regardless of the specific incorrect category.

Specific perturbation: This involves adding unique perturbations to each input, creating varied patterns of disturbance.

Universal perturbation: A single perturbation pattern is applied uniformly to multiple inputs, affecting each in the same way.

1.3 Adversarial Attacks

Currently, there is no consensus in the academic community regarding the underlying mechanisms of adversarial examples. Szegedy et al. [22] suggest that adversarial examples naturally occur in real data, albeit with low frequency, making it challenging for models to learn from these examples. Consequently, when adversarial examples emerge in a test set, classifiers often struggle to accurately identify them. On the other hand, Goodfellow et al. [9] argue that the susceptibility of neural networks to adversarial examples stems from the model's high-dimensional linear characteristics, particularly when using linear activation functions like Relu or Maxout, which heighten the model's vulnerability. Although the generation principles of adversarial examples are yet to be scientifically elucidated, recent efforts in developing adversarial example generation algorithms have laid a theoretical and practical foundation for enhancing the security of deep neural networks.

L-BFGS Szegedy et al. [22] noted that neural networks are susceptible to particular perturbations, which can cause significant deviations in model recognition outcomes. They developed the first adversarial attack algorithm for deep learning, named L-BFGS, which is expressed as:

$$\min c\|\delta\| + J_\theta(\mathbf{x}', \mathbf{l}') \quad (1)$$

with the condition that $\mathbf{x}' \in [0, 1]$. In this formulation, c is a positive constant, $\mathbf{x}'$ denotes the adversarial example generated by adding the perturbation δ to the original input, and J_θ is the loss function. The selection of the parameter c is crucial, as the effectiveness of the algorithm hinges on choosing a suitable c for this constrained optimization problem. The L-BFGS algorithm is renowned

for its ability to transfer across various datasets, thereby igniting widespread research interest in adversarial examples.

FGSM Goodfellow et al. [9] presented the Fast Gradient Sign Method (FGSM) to illustrate that the vulnerability of deep neural networks to adversarial examples is due to their high-dimensional linearity. FGSM creates adversarial perturbations by exploiting the direction of the steepest gradient in the deep learning model and then applying these perturbations to the input image to produce adversarial examples. The perturbation is calculated using the following formula:

$$\delta = \epsilon \cdot sign(\nabla_{\mathbf{x}} J_\theta(\theta, \mathbf{x}, \mathbf{y})) \quad (2)$$

where δ is the perturbation created; θ and $\mathbf{x}$ represent the model parameters and the input, respectively; $\mathbf{y}$ is the target associated with $\mathbf{x}$; J_θ is the loss function used in model training; ϵ is a scaling constant. At $\epsilon = 0.25$, FGSM attains a classification error rate of 99.9% and an average confidence rate of 79.3% on the MNIST dataset.

FGSM is valued for its quick execution as it involves a single-step attack. However, this method can sometimes produce a low success rate of adversarial examples. To enhance this, Kurakin et al. [11] introduced an iterative version of FGSM (I-FGSM), which improves on the original by incrementally increasing the loss function in several smaller steps, facilitating the creation of more effective adversarial examples.

JSMA Papernot et al. [17] introduced the Jacobian-based Saliency Map Attack (JSMA), a novel approach that, instead of utilizing the gradient of the loss function, leverages the probability information from the model's output categories to perform back-propagation and generate gradient information for creating an adversarial saliency map. The forward derivative of the deep learning model relative to the input example $\mathbf{x}$ is given by:

$$\nabla F(\mathbf{x}) = \frac{\partial F(\mathbf{x})}{\partial \mathbf{x}} = \left[\frac{\partial F_j}{\partial x_i}\right]_{i,j} \quad (3)$$

allowing the evaluation of each pixel's impact on the model's classification decision through the forward gradient.

To assess how changes in pixel values influence the classifier's target, JSMA develops an adversarial significance graph using the Jacobian matrix, as shown below:

$$S(\mathbf{x}, t)[i] = \{\, 0 \; if \frac{\partial F_t(\mathbf{x})}{\partial x_i} < 0 or \sum_{j \neq t} \frac{\partial F_t(\mathbf{x})}{\partial x_i} > 0, \frac{\left|\frac{\partial F_t(\mathbf{x})}{\partial x_i}\right|}{\sum_{j \neq t} \left|\frac{\partial F_t(\mathbf{x})}{\partial x_i}\right|} otherwise \quad (4)$$

where i denotes an input feature. This calculation helps identify the pixel changes that can most significantly affect the target classification t. A larger i

indicates greater sensitivity of the target classifier to perturbations in that feature. Hence, JSMA chooses to perturb the pixel with the highest anti-significance value to create adversarial examples. The authors illustrate that altering merely 4.02% of the features in an example can lead to a JSMA attack success rate of 97% with minimal perturbations, making these alterations challenging to detect visually.

However, JSMA has several drawbacks. Firstly, its effectiveness is highly dependent on the accurate calculation of the Jacobian matrix, which can be computationally intensive, especially for large models with high-dimensional input spaces. Additionally, the method's focus on modifying specific pixels based on the Jacobian matrix may result in perturbations that are less effective against models not sensitive to these specific pixel changes.

C&W In response to the emergence of adversarial attack methods, Papernot et al. [18] introduced defensive distillation by leveraging the distillation algorithm [10] to transfer knowledge from a complex network to a simpler one. This technique conceals direct access to the original model, providing a defense against certain types of adversarial attacks. However, Carlini et al. [5] found that defensive distillation fails to withstand CW attacks, which adhere to constraints specified by ℓ_0, ℓ_2, and ℓ_∞ norms. The CW attack is generally expressed as:

$$\min \left(D(\mathbf{x}, \mathbf{x}+\delta) + c \cdot f(\mathbf{x}+\delta)\right) \quad \textit{subjectto} \quad \mathbf{x}+\delta \in [0,1]^n \quad (5)$$

In this formulation, D measures constraints like ℓ_0, ℓ_2, and ℓ_∞, where ℓ_0 limits the number of modified points in the original input, ℓ_2 constrains the total magnitude of the perturbation, and ℓ_∞ caps the maximum change allowed per pixel. The hyperparameter c and the function f, which involves different objective functions, play crucial roles in configuring the attack. Testing on datasets like MNIST and CIFAR showed that the CW attack can breach the distilled network with a 100% success rate and generate high-confidence adversarial examples through careful tuning of parameters.

This attack, while highly effective, suffers from high computational complexity, sensitivity to parameter tuning, and can be time-consuming and less practical for real-time applications.

One-Pixel Su et al. [21] introduced the One-Pixel Attack (OPA), highlighting that modifying just a single pixel can create adversarial examples. This approach utilizes a differential evolution algorithm to find the most effective adversarial perturbations by changing one or a few pixels to mislead the model. The formula for the OPA attack is given by:

$$\max f_{adv}(\mathbf{x} + e(\mathbf{x})) \quad \textit{subjectto} \quad \|e(\mathbf{x})\| \leq d \quad (6)$$

where $d = 1$ indicates that only one pixel of the image is altered. The differential evolution algorithm optimizes the perturbation to enhance the adversarial

impact. In an n-dimensional image $\mathbf{x} = (x_1, \ldots, x_n)$, the perturbation affecting a single pixel is in line with the n-dimensional space. Each perturbation is described as a 5-tuple, including the x and y coordinates, along with the RGB value. The update mechanism within the differential evolution algorithm is specified as:

$$x_i(g+1) = x_{r1}(g) + F \cdot (x_{r2}(g) - x_{r3}(g)) \quad (7)$$

where x_i represents the candidate solution; x_{r1}, x_{r2}, and x_{r3} are randomly selected distinct indices from the current generation; F is the scaling factor, generally around 0.5; and g denotes the current generation. If a candidate solution demonstrates improvement over its predecessor in a particular iteration, it advances to the next generation. This iterative process continues until the most potent adversarial example is identified or until reaching a predetermined number of iterations.

The One-Pixel attack, despite its intriguing premise of altering just a single pixel, can be less effective against more complex or well-defended neural network models and often requires multiple iterations to find an effective perturbation, limiting its practicality in scenarios requiring immediate results.

DeepFool Moosavi-Dezfooli et al. [16] introduced DeepFool, a gradient iteration-based method for generating adversarial perturbations. DeepFool estimates perturbations by initially computing them and then adjusting pixels iteratively until the adversarial examples cross the decision boundary. In binary classification scenarios, where the classification function is $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$, the corresponding decision plane is defined as $\beta = \{\mathbf{x} : \mathbf{w}^T\mathbf{x} + b = 0\}$. The minimal perturbation δ affecting the classification of the original example $\mathbf{x}_0$ is the orthogonal projection of $\mathbf{x}_0$ onto β, with the calculation formula for δ as follows:

$$\delta^*(\mathbf{x}_0) := \arg\min_{\delta} \|\delta\|_2 \quad s.t. \quad sign(f(\mathbf{x}_0 + \delta)) \neq sign(f(\mathbf{x}_0)) = -\frac{f(\mathbf{x}_0)}{\|\mathbf{w}\|_2}\mathbf{w} \quad (8)$$

The objective function above iteratively determines the minimal adversarial perturbation δ. The optimization for each iteration i is defined as:

$$\arg\min_{\delta_i} \|\delta_i\|_2 \quad s.t. \quad f(\mathbf{x}_i) + \nabla f(\mathbf{x}_i)^T\delta_i = 0 \quad (9)$$

DeepFool evaluates the robustness of examples by assessing the minimum distance between the decision boundaries of normal and adversarial examples. While DeepFool provides more accurate perturbations in a shorter time compared to the single-step attack of FGSM, it is designed solely for non-targeted attacks.

ZOO Chen et al. [6] developed the Zeroth Order Optimization (ZOO) approach, a notable alternative to some existing black-box attack methods that depend on surrogate models and exploit attack transferability. Instead, ZOO approximates

first-order and second-order gradients and uses optimization algorithms such as Adam or Newton's method to iteratively determine the optimal adversarial example. The perturbation applied to an input $\mathbf{x}$ is described as: $\mathbf{x} = \mathbf{x} + h\mathbf{e}_i$, where h is a small constant and $\mathbf{e}_i$ is a unit vector with the i-th element set to 1 and all others to 0. The estimated first-order gradient is computed as follows:

$$\hat{\mathbf{g}}_i := \frac{\partial f(\mathbf{x})}{\partial x_i} \approx \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x} - h\mathbf{e}_i)}{2h} \quad (10)$$

The estimation of the second-order gradient is calculated as:

$$\hat{\mathbf{h}}_i := \frac{\partial^2 f(\mathbf{x})}{\partial x_i^2} \approx \frac{f(\mathbf{x} + h\mathbf{e}_i) - 2f(\mathbf{x}) + f(\mathbf{x} - h\mathbf{e}_i)}{h^2} \quad (11)$$

Experiments conducted by Chen et al. on the MNIST and CIFAR10 datasets showcased the ZOO attack's high success rate. However, compared to the white-box attack by Carlini Wagner (CW), the ZOO attack requires more time.

UAP Moosavi-Dezfooli et al. [15] introduced the Universal Adversarial Perturbations (UAP) technique, building on the principles of the DeepFool method. UAP employs adversarial perturbations to shift regular examples past the decision boundary, thereby producing adversarial examples. The goal is to achieve:

$$\hat{k}(\mathbf{x} + \delta) \neq \hat{k}(\mathbf{x}) for \text{“}most''\mathbf{x} \sim \mu \quad (12)$$

where δ is the universal adversarial perturbation that meets the following criteria:

$$\|\delta\|_p \leq \epsilon, \quad P_{\mathbf{x}\sim\mu}(\hat{k}(\mathbf{x} + \delta) \neq \hat{k}(\mathbf{x})) \geq 1 - \theta \quad (13)$$

In this context, $\hat{k}(\mathbf{x})$ represents the classification function. The magnitude of the perturbation δ is controlled by ϵ, while θ sets the attack's success rate on the original samples. During the iterative UAP process, the minimal perturbation required for each example is calculated using the DeepFool algorithm and is repeatedly refined until the optimal adversarial example is produced. Although initial experiments by Moosavi-Dezfooli et al. used ResNet to demonstrate the effectiveness of the universal perturbation, the UAP attack has been successfully adapted to other neural network architectures. However, a significant drawback of UAP is its potential to compromise the model's performance on legitimate inputs, particularly as the perturbation strength increases, reflecting a trade-off between effectiveness and discretion.

advGAN Xiao et al. [23] introduced AdvGAN, an adversarial attack method leveraging generative adversarial networks (GANs). AdvGAN comprises a generator G, a discriminator D, and a target network model C. The process begins by inputting an original example $\mathbf{x}$ into the generator to produce an adversarial

perturbation $g(\mathbf{x})$, which is then fed into both the discriminator and the target model. The discriminator D is responsible for categorizing the example.

The objective function of AdvGAN is defined as follows:

$$L = L_{adv} + \alpha L_{GAN} + \beta L_{hinge} \quad (14)$$

where the objective function is partitioned into three components:

- L_{adv} denotes the misclassification loss aimed at guiding the generator to generate optimal adversarial perturbations, given by:

$$L_{adv} = E_{\mathbf{x}}[\ell_C(\mathbf{x} + g(\mathbf{x}), \mathbf{t})] \quad (15)$$

where $\mathbf{t}$ represents the target class for misclassification.

- L_{GAN} represents the adversarial loss, which is the original loss function proposed by Goodfellow et al., formulated as:

$$L_{GAN} = E_{\mathbf{x}}[\log D(\mathbf{x})] + E_{\mathbf{x}}[\log(1 - D(\mathbf{x} + g(\mathbf{x}))] \quad (16)$$

This loss function aims to refine both the generator G and the discriminator D to enhance their capabilities.

- L_{hinge} is utilized for stabilizing the training of GANs, expressed as:

$$L_{hinge} = E_{\mathbf{x}}[\max(0, \|g(\mathbf{x})\|_2 - c)] \quad (17)$$

where c is a hyperparameter representing the optimized distance.

AdvGAN has undergone black-box attack experiments on the MNIST dataset, achieving a success rate of 92.76%.

This attack, while powerful in generating stealthy adversarial examples, can be computationally intensive due to the training requirements of generative adversarial networks, potentially leading to higher resource consumption and longer preparation times.

ATNs Baluja et al. [3] introduced Adversarial Transformation Networks (ATNs), which employ a generative model to craft adversarial examples. ATNs take an original example as input and produce an adversarial example by training a feed-forward neural network. This strategy aims to minimize perturbations while maintaining similarity between the adversarial and original examples, while simultaneously ensuring a high success rate for the adversarial attacks.

The objective function for ATNs is defined as:

$$\arg\min_{\theta} \sum_{\mathbf{x}_i \in \mathbf{X}} [\beta L_X(G(\mathbf{x}_i, \theta), \mathbf{x}_i) + L_Y(F(G(\mathbf{x}_i, \theta)), F(\mathbf{x}_i))] \quad (18)$$

Here, $G(\mathbf{x}_i, \theta)$ represents the generative model trained to produce the adversarial example from the original example. $F(\mathbf{x}_i)$ denotes the target model under attack. L_X and L_Y are loss functions for the input and output spaces, respectively. L_X ensures similarity between the adversarial and original examples by

constraining their differences, while L_Y measures and regulates the success rate of the adversarial attacks.

This attack require extensive training to produce effective adversarial examples, which can lead to significant computational overhead and potentially limit their scalability across different tasks or larger datasets.

UPSET and ANGRI Sarkar et al. [20] introduced two black-box attack methods, UPSET and ANGRI. UPSET creates generic perturbations aimed at a specific target class, while ANGRI generates perturbations that are tailored to individual images. In UPSET, an adversarial perturbation R is produced through a residual generation network. If $\mathbf{t}$ represents the target category, then the perturbation is represented as $\mathbf{r}_t = R(t)$. The process for creating an adversarial example is formulated as:

$$\mathbf{x}' = U(\mathbf{x}, \mathbf{t}) = \max(\min(\mathbf{s} \times R(\mathbf{t}) + \mathbf{x}, 1), 0) \quad (19)$$

Here, U stands for the UPSET network, and $\mathbf{s}$ is a scaling factor used to adjust the intensity of the perturbation $\mathbf{r}_t$. The loss function for the UPSET network includes two components:

$$L(\mathbf{x}, \mathbf{x}', \mathbf{t}) = L_C(\mathbf{x}', \mathbf{t}) + L_F(\mathbf{x}, \mathbf{x}') \quad (20)$$

where L_C penalizes deviations from the target attack class, and L_F aims to maintain the visual similarity between the adversarial and original images. L_C could be defined as the log likelihood of the classifier's confidence in class $\mathbf{t}$ for the adversarial example $\mathbf{x}'$, and L_F generally involves a norm-based measure between $\mathbf{x}$ and $\mathbf{x}'$.

However, both UPSET and ANGRI have their drawbacks. UPSET's generic perturbations might not be as effective for every image, potentially leading to suboptimal results. On the other hand, while ANGRI produces image-specific perturbations, it requires access to input images, which may not always be feasible in practical scenarios. Additionally, ANGRI's reliance on image attributes might limit its applicability to domains where such attributes are not readily available or reliably extractable.

Houdini Cisse et al. [7] introduced the Houdini algorithm, specifically tailored to generate adversarial examples for intricate tasks that are challenging to address through gradient descent methods, such as speech recognition and semantic segmentation. These tasks often involve combinatorial and non-decomposable problems, posing obstacles for traditional adversarial example generation techniques. The loss function for the Houdini algorithm is outlined as:

$$L_H(\theta, \mathbf{x}, \mathbf{y}) = P_{\gamma \sim \mathcal{N}(0,1)} \left[g_\theta(\mathbf{x}, \mathbf{y}) - g_\theta(\mathbf{x}, \mathbf{y}') < \gamma \right] \cdot L(\mathbf{y}', \mathbf{y}) \quad (21)$$

Here, g_θ represents the target neural network with parameter θ. The expression $g_\theta(\mathbf{x}, \mathbf{y}) - g_\theta(\mathbf{x}, \mathbf{y}')$ signifies the difference between the actual score and the

predicted score. L denotes the original loss function employed in the network. Additionally, Cisse et al. effectively applied the Houdini algorithm to diverse tasks like speech recognition, language segmentation, and pose estimation, showcasing substantial performance enhancements.

BPDA To demonstrate the vulnerability of defenses based on gradient obfuscation, Athalye et al. [2] introduced the Backward Pass Differentiable Approximation (BPDA) technique. This approach involves constructing a pre-processor $g(x)$ for a given pre-trained classifier, where $g(x) \approx x$. The derivative in BPDA is approximated as:

$$\nabla_{\mathbf{x}} f(g(\mathbf{x}))\big|_{\mathbf{x}=\mathbf{x}'} \approx \nabla_{\mathbf{x}} f(\mathbf{x})\big|_{\mathbf{x}=g(\mathbf{x}')} \quad (22)$$

This equation helps to acquire the approximate gradient values, which are then used to generate adversarial examples through averaging over several iterations. Athalye et al. tested BPDA attacks against seven defense models that relied on obfuscating gradients, presented at ICLR 2018. BPDA was able to completely bypass six of these defenses and partially bypass one, thus demonstrating the inherent vulnerabilities of defense strategies based on gradient obfuscation.

DaST In real-world scenarios, acquiring pre-trained models can present considerable hurdles. Zhou et al. [25] proposed the Data-Free Substitute Training (DaST) method to tackle this challenge by developing substitute models for adversarial black-box attacks without relying on real data. DaST leverages specially designed Generative Adversarial Networks (GANs) for this purpose. Specifically, it employs a multi-branch architecture and a label-controlled loss mechanism to handle the uneven distribution of synthetic examples. The substitute model is trained using synthetic examples generated by the generative model, which are then labeled by the attacked model. The primary objective of the substitute model is to replicate the output of the target model, effectively transforming the process into a game where the target model acts as the referee. The loss function for this scenario is expressed as:

$$L_D = d(T(\hat{\mathbf{X}}), D(\hat{\mathbf{X}})) \quad (23)$$

where $d(T(\hat{\mathbf{X}}), D(\hat{\mathbf{X}}))$ represents the metric used to evaluate the output distance between the substitute model D and the target model T. To update the generative model, the loss is defined as:

$$L_G = e^{-d(T,D)} + \alpha L_C \quad (24)$$

where L_C denotes the label-controlled loss, and α is a weighting factor controlling the importance of L_C.

However, one drawback of DaST is its potential vulnerability to adversarial attacks. Since DaST relies on generating synthetic examples to train substitute models, adversaries could potentially exploit weaknesses in the synthetic

data generation process to craft adversarial examples that mislead the substitute model. Therefore, while DaST offers a promising approach for training substitute models without real data, its susceptibility to adversarial manipulation warrants careful consideration and additional defensive measures.

GAP++ In contrast to approaches that solely rely on input images to generate adversarial perturbations, Mao et al. [13], drawing inspiration from prior work [14], introduce a novel framework named GAP++, building upon the foundation of GAP [19]. GAP++ is engineered to infer targets using both input images and target labels, facilitating conditionally perturbed outputs. Unlike earlier models concentrating on single-target attacks, GAP++ conducts target-conditioned attacks by learning the correlation between attack targets and image semantics. This framework empowers the generation of various target perturbations utilizing a single trained model.

In the architecture of GAP++, each input image is paired with its corresponding target label as conditional information. For non-target attacks, where no specific target label is provided, a zero vector is utilized for off-target training. This strategy ensures that the absence of a target does not interfere with the learning of internal representations by concatenating zero tensors within the model.

Extensive experiments conducted on the MNIST and CIFAR10 datasets showcase GAP++'s superior performance compared to single-target attack models, achieving a higher deception rate with smaller perturbation norms. Despite incorporating the network architecture and normalization techniques from the original GAP, GAP++ is optimized for lighter performance, rendering it suitable for diverse attack tasks.

Morris II Cohen et al. [8] introduced the "Morris II" concept, a zero-click worm that exploits Generative Artificial Intelligence (GenAI) to autonomously spread across GenAI ecosystems. Morris II embeds malicious adversarial prompts into standard inputs, manipulating GenAI models to replicate these inputs, perform malicious actions, and propagate the worm to other systems. The attack mechanism leverages multi-modal inputs including text, images, and audio. The mathematical representation of the adversarial generation is as follows:

$$\hat{\mathbf{X}}' = G(\mathbf{X}, \theta) = embed(malicious\_prompt(\mathbf{X}, \theta)) \quad (25)$$

where G denotes the generative model embedding adversarial prompts into inputs, and θ represents the parameters controlling the embedding process. The effectiveness of this approach is validated through experiments on GenAI models such as Gemini Pro and ChatGPT 4.0, focusing on the propagation rate and success of malicious activities.

The loss function used to refine the adversarial prompt embedding is defined as:

$$L(\mathbf{X}, \hat{\mathbf{X}}') = L_{adv}(\hat{\mathbf{X}}', \mathbf{T}) + L_{prop}(\mathbf{X}, \hat{\mathbf{X}}') \quad (26)$$

where L_{adv} measures the effectiveness of the adversarial examples in performing malicious actions as intended, and L_{prop} ensures the worm's propagation capability by maintaining the functional integrity of the GenAI model's output.

Comparative analysis shows that while traditional adversarial attacks like DeepFool and C&W focus on efficiency and stealthiness, Morris II prioritizes autonomy and adaptability, enabling it to execute sustained campaigns without direct oversight. The generative model's ability to adaptively respond to defensive measures showcases a significant advancement over static adversarial techniques.

Extensive testing on MNIST and CIFAR10 confirms Morris II's high success rates and underlines the critical need for robust defense mechanisms against GenAI-exploiting attacks. This approach not only underscores the vulnerabilities in current GenAI applications but also sets a precedent for the development of autonomous, self-propagating cyber threats in AI-driven systems.

1.4 Adversarial Attacks Comparison

L-BFGS, an early adversarial attack algorithm, has served as a foundation for subsequent methods, inspiring their development. The adversarial examples produced by L-BFGS exhibit high transferability, seamlessly adapting across diverse neural network architectures. However, the effectiveness of the JSMA is constrained by its reliance on the Jacobian matrix, which exhibits significant variation across input examples, limiting its transferability. Conversely, the Fast Gradient Sign Method (FGSM) offers swift perturbation generation with just one iteration, ensuring efficiency but often resulting in lower success rates compared to iterative techniques like Projected Gradient Descent (PGD).

In contrast to FGSM, JSMA, and other methods, DeepFool generates relatively subtle perturbations, albeit lacking the capability for targeted attacks. Universal Adversarial Perturbations (UAP) extend the concept of DeepFool to enhance generalization, enabling widespread attacks across various models and datasets, thereby meeting real-world application demands. The One-Pixel attack achieves deception by modifying a single pixel; however, it necessitates multiple iterations for optimal solutions, hence sacrificing efficiency.

The C&W attack is notably aggressive and adept at overcoming defenses like defensive distillation, a feat unattainable by L-BFGS, FGSM, and DeepFool, albeit at the expense of efficiency. UPSET and ANGRI, introduced concurrently, demonstrate distinct approaches: UPSET's independence from input data properties facilitates generalized attacks, whereas ANGRI's reliance on such properties during training constrains its ability to execute generalized attacks effectively.

AdvGAN, DaST, GAP++, and Morris leverage Generative Adversarial Networks (GANs) in their attack methodologies, yielding robust attack effects. The adversarial examples generated by these methods closely resemble the original examples due to the competitive dynamics between the generator and discriminator, thereby augmenting their efficacy in adversarial scenarios.

Efficiency and Speed

- **FGSM:** Known for rapid execution as a single-step attack method, providing high efficiency in time-sensitive scenarios.
- **DeepFool and ZOO:** DeepFool offers more precise perturbations and is faster than ZOO, which requires more time due to its complex gradient estimation process.

Attack Success Rate and Reliability

- **C&W and JSMA:** Both have high success rates. C&W aggressively overcomes defensive measures like defensive distillation. JSMA's effectiveness is occasionally limited due to its dependency on the Jacobian matrix.
- **One-Pixel and UAP:** Demonstrate that minimal perturbations can significantly impact, with One-Pixel requiring multiple iterations and UAP showing broad effectiveness across models.

Table 1. Enhanced Comparison of Adversarial Attack Methods

Method	Efficiency	Success Rate	Ref	Attack Type	Attack Target	Perturbation	Stealthiness
L-BFGS	Low	High	[22]	White-box	Targeted	ℓ_∞	Moderate
FGSM	High	Moderate	[9]	White-box	Targeted	ℓ_∞	Low
JSMA	Moderate	High	[17]	White-box	Targeted	ℓ_2	High
C&W	Moderate	Very High	[5]	White-box	Targeted	$\ell_0, \ell_2, \ell_\infty$	Low
One-Pixel	Low	Moderate	[21]	Black-box	Non-targeted	ℓ_0	High
DeepFool	Moderate	High	[16]	White-box	Non-targeted	$\ell_0, \ell_2, \ell_\infty$	High
ZOO	Low	Moderate	[6]	Black-box	Targeted	ℓ_2	Moderate
UAP	Moderate	High	[15]	White-box	Non-targeted	ℓ_2, ℓ_∞	High
AdvGAN	High	High	[23]	White-box	Targeted	ℓ_2	High
ATNs	Moderate	Moderate	[3]	White-box	Targeted	ℓ_∞	High
UPSET	High	Low	[20]	Black-box	Targeted	ℓ_∞	Moderate
Houdini	Moderate	Moderate	[7]	Black-box	Targeted	ℓ_2, ℓ_∞	High
BPDA	Moderate	Moderate	[2]	Black-box	Targeted	ℓ_2, ℓ_∞	Moderate
DaST	Low	Moderate	[25]	Black-box	Targeted	ℓ_∞	Moderate
GAP++	High	Moderate	[13]	White-box	Targeted	$\ell_0, \ell_2, \ell_\infty$	Moderate
Morris II	Moderate	High	[8]	Black-box	Targeted	$\ell_0, \ell_2, \ell_\infty$	Low

Stealth and Detectability

- **AdvGAN:** Produces perturbations that are difficult to detect by both machines and humans, thus enhancing the stealth of attacks.
- **DeepFool:** Generates the smallest necessary perturbations for misclassification, making it less detectable.

General Applicability and Transferability

- **L-BFGS:** Provides a foundation with high transferability across different neural network architectures.
- **Universal Adversarial Perturbations (UAP):** Excel in general applicability, successfully attacking multiple models without specific adjustments.

Novelty and Technological Innovation

- **ZOO:** Novel approach to black-box attacks, eliminating the need for training substitute models using zeroth order optimization.
- **AdvGAN and GAP++:** Represent the cutting edge by incorporating complex generative models that adjust perturbations dynamically.
- **Morris II:** A groundbreaking zero-click worm utilizing generative AI to autonomously spread across AI ecosystems. Its autonomy and adaptability mark significant advancements.

References

1. Abdel-Hamid, O., Mohamed, A.r., Jiang, H., Penn, G.: Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition. In: 2012 IEEE international conference on Acoustics, speech and signal processing (ICASSP). pp. 4277–4280. IEEE (2012)
2. Athalye, A., Carlini, N., Wagner, D.: Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In: International conference on machine learning. pp. 274–283. PMLR (2018)
3. Baluja, S., Fischer, I.: Adversarial transformation networks: Learning to generate adversarial examples. arXiv preprint arXiv:1703.09387 (2017)
4. Biggio, B., Roli, F.: Wild patterns: Ten years after the rise of adversarial machine learning. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 2154–2156 (2018)
5. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 ieee symposium on security and privacy (sp). pp. 39–57. Ieee (2017)
6. Chen, P.Y., Zhang, H., Sharma, Y., Yi, J., Hsieh, C.J.: Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In: Proceedings of the 10th ACM workshop on artificial intelligence and security. pp. 15–26 (2017)
7. Cisse, M., Adi, Y., Neverova, N., Keshet, J.: Houdini: Fooling deep structured prediction models. arXiv preprint arXiv:1707.05373 (2017)
8. Cohen, S., Bitton, R., Nassi, B.: Here comes the ai worm: Unleashing zero-click worms that target genai-powered applications. arXiv preprint arXiv:2403.02817 (2024)
9. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
10. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
11. Kurakin, A., Goodfellow, I.J., Bengio, S.: Adversarial examples in the physical world. In: Artificial intelligence safety and security, pp. 99–112. Chapman and Hall/CRC (2018)

12. Lu, J., Xiong, C., Parikh, D., Socher, R.: Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 375–383 (2017)
13. Mao, X., Chen, Y., Li, Y., He, Y., Xue, H.: Gap++: Learning to generate target-conditioned adversarial examples. arXiv preprint arXiv:2006.05097 (2020)
14. Mao, X., Chen, Y., Li, Y., Xiong, T., He, Y., Xue, H.: Bilinear representation for language-based image editing using conditional generative adversarial networks. In: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 2047–2051. IEEE (2019)
15. Moosavi-Dezfooli, S.M., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1765–1773 (2017)
16. Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: Deepfool: a simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2574–2582 (2016)
17. Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A.: The limitations of deep learning in adversarial settings. In: 2016 IEEE European symposium on security and privacy (EuroS&P). pp. 372–387. IEEE (2016)
18. Papernot, N., McDaniel, P., Wu, X., Jha, S., Swami, A.: Distillation as a defense to adversarial perturbations against deep neural networks. In: 2016 IEEE symposium on security and privacy (SP). pp. 582–597. IEEE (2016)
19. Poursaeed, O., Katsman, I., Gao, B., Belongie, S.: Generative adversarial perturbations. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4422–4431 (2018)
20. Sarkar, S., Bansal, A., Mahbub, U., Chellappa, R.: Upset and angri: Breaking high performance image classifiers. arXiv preprint arXiv:1707.01159 (2017)
21. Su, J., Vargas, D.V., Sakurai, K.: One pixel attack for fooling deep neural networks. IEEE Transactions on Evolutionary Computation **23**(5), 828–841 (2019)
22. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
23. Xiao, C., Li, B., Zhu, J.Y., He, W., Liu, M., Song, D.: Generating adversarial examples with adversarial networks. arXiv preprint arXiv:1801.02610 (2018)
24. Yu, A.W., Dohan, D., Luong, M.T., Zhao, R., Chen, K., Norouzi, M., Le, Q.V.: Qanet: Combining local convolution with global self-attention for reading comprehension. arXiv preprint arXiv:1804.09541 (2018)
25. Zhou, M., Wu, J., Liu, Y., Liu, S., Zhu, C.: Dast: Data-free substitute training for adversarial attacks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 234–243 (2020)