REDVET - Revistaelectrónica de Veterinaria- ISSN 1695-7504
Vol 25, No. 1 (2024)
http://www.veterinaria.org
Article Received: Revised: Accepted:

DETECTION OF ADVERSARIAL ATTACKS USING HYBRID GENERATIVE ADVERSARIAL NETWORKS (GANS)

K.Bala Bhaskar[1*], Dr.R.Satya Prasad[2]

[1*]Research Scholar, Dept of CSE, Acharya Nagarjuna University,
[2] Professor and Dean R & D, DEPT OF CSE, Dhanekula Institute of Engineering & Technology, Ganguru, Vijayawada.
profrsp@gmail.com

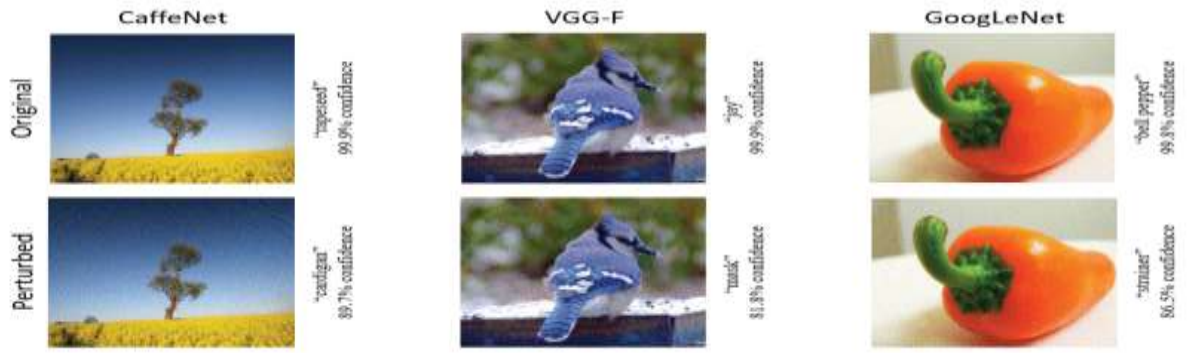
Abstract:
Adversarial attacks are a common technique used in machine learning to find vulnerabilities in deep learning models. In the context of hybridgenerative adversarial networks (GANs), adversarial attacks are used to perturb the input data so that the generated outputs are manipulated to produce unexpected or undesirable results. One example of an adversarial attack in GANs is the "Fast Gradient Sign Method" (FGSM), which involves adding a small perturbation to the input data to cause the GAN to generate an output that is significantly different from the desired result. This technique is often used to test the robustness of GANs against attacks and to identify potential weaknesses that malicious actors could exploit. Another type of adversarial attack in GANs is known as the "Boundary Attack," which involves finding the boundary between the decision regions of the generator and the discriminator in order to identify inputs that can be manipulated to produce a desired output. This paper introduced a hybrid DL model to overcome various issues in existing models. Experiments are conducted on two datasets such as MNIST dataset. Overall, adversarial attacks in hybrid GANs are an essential area of research as they help to identify potential vulnerabilities in the models and enable researchers to develop more robust and secure machine learning systems.

Keywords:generative adversarial networks (GANs), Adversarial attacks, Fast Gradient Sign Method (FGSM).

Introduction

Deep Learning (DL) [1] is a data-driven technology capable of accurately modelling difficult mathematical functions across enormous datasets. It has recently helped scientists make major discoveries in machine intelligence applications. Deep learning approaches are currently expanding our understanding for a wide range of cutting-edge scientific challenges, including analysing DNA mutations [2], reconstructing brain circuits [3], and studying cell data [4]. As a result, it is not surprising that many modern subfields of machine intelligence are rapidly adopting this technique as 'the tool' to solve long-standing challenges. Computer vision, like speech recognition [5] and natural language processing [6], is a discipline that now relies largely on deep learning.

Deep learning can help detect and protect against hostile threats. Adversarial assaults are a form of deep neural network attack in which input data is purposely perturbed such that the network misclassifies it. One prominent method for detecting adversarial attacks is to employ a group of deep neural networks (DNN). This entails training numerous deep neural networks with various topologies or initializations on the same dataset and then combining their results to make a final prediction. Adversarial examples generally cause various models to produce diverse outputs, making an ensemble more resistant to adversarial attacks than a single model.Another option is to utilise adversarial training, which entails creating antagonistic instances and including them into the training data. This can strengthen the network's resilience to adversarial attacks as it learns to recognise and defend against certain scenarios. Other approaches include utilising gradient regularisation to avoid adversarial perturbations, using a smaller feature space to decrease the possibility of adversarial attacks, and using generative models to generate synthetic data for adversarial training. Overall, the key to protecting against adversarial attacks is to combine these approaches to build a strong and resilient deep learning model capable of detecting and defending against a variety of attacks.



REDVET - Revistaelectrónica de Veterinaria- ISSN 1695-7504
Vol 25, No. 1 (2024)
http://www.veterinaria.org
Article Received: Revised: Accepted:

Figure 1: Example of attacks on deep learning models with 'universal adversarial perturbations': The attacks are demonstrated for CaffeNet [7], the VGG-F network [8], and GoogLeNet [9]. All of the networks accurately identified the original clean photos with high confidence. After modest perturbations were applied to the images, the networks predicted incorrect labels with similar high confidence. Note that the fluctuations are barely visible to the human vision system, yet their impact for deep learning models are devastating.

Literature Survey

Szegedy et al. [10] have discovered some intriguing results involving adversarial attacks on deep learning in computer vision. In addition to picture-specific adversarial perturbations, Moosavi-Dezfooli et al. [11] demonstrated the possibility of 'universal perturbations' that can fool a network classifier on any image. Similarly, Athalye et al. [12] proved that real-world objects may be 3-D printed and mislead deep neural network classifiers. Given the importance of deep learning research in computer vision and its potential real-world applications, this article gives the first thorough assessment of adversarial assaults on deep learning in computer vision. Because the paper is meant for a broader audience than the Computer Vision community, it only presupposes a basic understanding of deep learning and image processing. Nonetheless, it provides technical specifics of significant contributions for interested readers.

Sabour et al. [13] demonstrated the ability to generate adversarial samples by modifying the internal layers of deep neural networks. The authors proved that it is possible to create internal network representations of hostile images that resemble representations of images from different categories. Papernot et al. [14] investigated the transferability of adversarial strategies for deep learning and other machine learning approaches, introducing further transferability attacks. Narodytska and Kasiviswanathan [15] also developed further black-box attacks that have been found effective in deceiving neural networks by modifying only a few pixel values in images. Liu et al. [16] developed the 'epsilon-neighborhood' attack, which has been demonstrated to mislead defensively distilled networks [17] with 100% success for white-box attacks. Oh et al. [18] utilized a 'Game Theory' approach to adversarial assaults and developed a way to defeat the countermeasures used against adversarial attacks on deep neural networks. Mpouri et al. [19] proposed a data-independent method for producing universal adversarial perturbations for deep network models. Hosseini et al. [20] developed the concept of 'semantic adversarial examples', which are input photos that represent semantically identical things to humans but are misclassified by deep neural networks. They employed negative visuals as semantic adversarial examples. Kanbak et al. [21] devised the 'ManiFool' algorithm in the wake of the DeepFool approach [22] to assess the robustness of deep neural networks against geometrically disturbed images. Dong et al. [23] developed an iterative approach to improving adversarial attacks in black-box scenarios. Carlini and Wagner [24] recently proved that 10 different perturbation defences can be beaten by new attacks based on new loss functions. Rozsa et al. [25] also presented a 'hot/cold' strategy for creating several hostile examples from a single image.

Tabacof et al. [26] explored adversarial assaults on auto encoders and suggested a technique for distorting the input image (to make it adversarial), which misleads the auto encoder into reconstructing an entirely other image. Their approach assaults a neural network's internal representation, causing the representation for the adversarial image to resemble that of the target image. However, it appears that auto encoders are far more resistant to adversarial attacks than ordinary classifier networks. Kos et al. [27] investigated approaches for creating adversarial examples for deep generative models, such as the variation auto encoder (VAE) and VAE-Generative Adversarial Networks (VAE-GAN). GANs, such as [28], are increasingly popular in computer vision applications due to their capacity to learn data distributions and generate realistic images from those distributions. The authors described three types of attacks for VAE and VAE-GANs. Based on the success of these attacks, it is inferred that deep generative models are likewise subject to adversaries who can persuade them to produce significantly diverse outputs. This study lends credence to the concept that "adversarial examples are a general phenomenon for current neural network architectures".

Various significant terms for adversarial attacks

- Adversarial examples (x') are malicious permutations of clean images (x) created by adding minor perturbations (e.g., noise) to trick the deep learning model. The extra perturbations are usually unnoticeable; however, there are a few exceptions when the perturbation is noticeable.
- Adversarial perturbation is the noise introduced to a clean instance image to generate an adversarial example. The Adversary is the attacker who creates and executes adversarial attacks against the ML model.
- Adversarial training is a type of machine learning training that use adversarial instances from the training set to improve the resilience of the deep learning model to adversarial attacks.
- Image-Detector reveals hostile images.
- The fooling rate/ratio measures an ML model's robustness and the strength of adversarial attacks. It is the proportion of adversarial samples that tricked the ML model. A high fooling ratio indicates that the assault is strong, and the ML model is not robust to adversarial cases.
- Targeted attacks cause the DNN model to misclassify hostile examples to a predetermined target label.

REDVET - Revistaelectrónica de Veterinaria- ISSN 1695-7504
Vol 25, No. 1 (2024)
http://www.veterinaria.org
Article Received: Revised: Accepted:

- The most prevalent sort of attack is the untargeted assault, which causes the DNN model to misclassify the adversarial case into an inaccurate label. Such attacks seek to compromise the integrity and availability of DNN models.

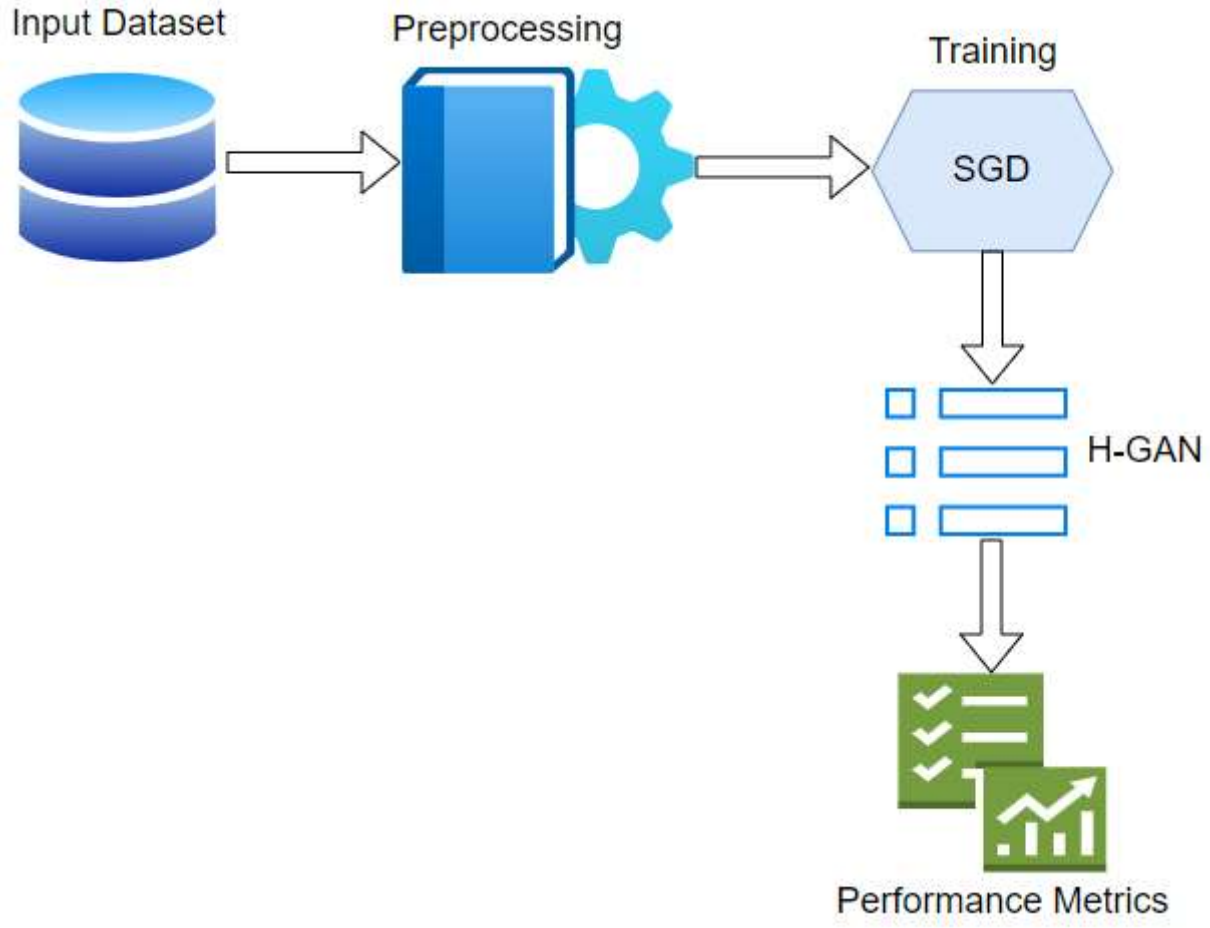


Figure 2: System Architecture

Preprocessing: The following techniques used for preprocessing for this dataset.
Normalization: MNIST images are grayscale and have pixel values in the range of 0 to 255. Normalizing the pixel values to be between 0 and 1 can improve model performance. This can be achieved by dividing each pixel value by 255.

Reshaping: The MNIST dataset contains images of size 28x28. Reshaping the images into a one-dimensional array of length 784 can simplify the input format for the model.
Filtering: Applying filters to the images can help remove noise and enhance features. Common filters include Gaussian blur, median blur, and edge detection filters.

Contrast normalization: Adjusting the contrast of the images can also improve model performance. This can be achieved by applying a contrast normalization function to each image.
These preprocessing techniques can be applied in combination or individually depending on the specific requirements of the model being trained on the MNIST dataset.

Training using Stochastic Gradient Descent (SGD)
SGD is a widely used optimization algorithm in deep learning that involves iteratively updating the weights of a neural network in the direction of the negative gradient of the loss function with respect to the weights. However, SGD can also be used for adversarial attacks, where the goal is to generate malicious inputs that can fool a neural network into making incorrect predictions. In adversarial attacks, the input to the neural network is perturbed in a way that is imperceptible to humans, but is sufficient to cause the network to make incorrect predictions. One way to generate such perturbations is to use SGD to optimize the input with respect to the loss function of the neural network. The basic idea of SGD-based adversarial attacks is to start with a benign input and then iteratively update it by computing the gradient of the loss function with respect to the input and adding a small perturbation in the opposite direction. This perturbation

is designed to maximize the loss function, which means that the resulting input is more likely to be misclassified by the neural network.

Because BGD takes a long time to calculate the gradient on large-scale training datasets, SGD [29] was proposed to address BGD's constraints. SGD is an iterative method for optimizing the cost function. As shown in Eq. 2, in each iteration, SGD randomly selects an example from the training data and calculates the gradient before performing a parameter update on the selected example. In contrast to BGD, which calculates the true gradient using the whole dataset, SGD aims to derive an estimate of the gradient using a limited number of cases. The cost of updating stochastic gradient descent is independent of training dataset size and can achieve linear convergence. In some circumstances, the cost of training some ML models using SGD can approach O(1) [30]. As a result, SGD has become the most extensively utilized optimization technique. The formulas for SGD for adversarial attacks are as follows:

Let X be the original input, Y be the true label, and F(x) be the output of the neural network for input x.

1. Define the loss function L(x, y) as the cross-entropy loss between the true label y and the predicted label f(x) for input x.

$$\mathrm{L(x,y)} = -\mathrm{y} * \log\big(\mathrm{F(x)}\big) - (10\mathrm{y}) * \big(1 - \mathrm{F(x)}\big) \quad (1)$$

2. Define the perturbation P as a small change to the input X.

$$\mathrm{X'} = \mathrm{X} + \mathrm{P} \quad (2)$$

3. Define the target label Y' as the label that we want the neural network to predict for the perturbed input X'.

$$\mathrm{Y'} = \mathrm{argmax(L(X', y))} \text{ where argmax is taken over all possible labels y.}$$

4. Define the loss function $\mathrm{L_{adv}(X, Y')}$ as the cross-entropy loss between the target label Y' and the predicted label F(X') for input X'.

$$\mathrm{L_{adv}(X, Y')} = -\mathrm{Y'} * \log\big(\mathrm{F(X')}\big) - \ (1 - \mathrm{Y'}) * \log\big(1 - \mathrm{F(X')}\big) \quad (3)$$

5. Use SGD to optimize the perturbation P to minimize the loss function $\mathrm{L_{adv}(X, Y')}$.

$$\mathrm{P} = \mathrm{P} - \mathrm{learning_{rate}} * \mathrm{gradient}(\mathrm{L_{adv}(X, Y')}, \mathrm{P}) \quad (4)$$

Where $\mathrm{gradient}(\mathrm{L_{adv}(X, Y')}, \mathrm{P})$ is the gradient of the loss function $\mathrm{L_{adv}(X, Y')}$ with respect to the perturbation P. The learning rate is a hyperparameter that controls the step size of the optimization process. The gradient can be computed using back-propagation through the neural network.

6. Clip the perturbation P to ensure that it stays within a small bounded range, to avoid generating a perturbation that is too large and obvious.

$$\mathrm{P} = \mathrm{clip(P, -epsilon, epsilon)} \quad (5)$$

Where epsilon is a small positive constant that controls the maximum magnitude of the perturbation.

7. Repeat steps 2-6 until the perturbation P causes misclassification by the neural network.

SGD-based adversarial attacks can be used to evaluate the robustness of neural networks against malicious inputs and to develop defenses against such attacks. However, it is important to note that the effectiveness of these attacks depends on several factors, such as the architecture of the neural network, the choice of loss function, and the level of perturbation that is added to the input. Once the model is trained, evaluate its performance on the validation set by using the H-GAN. Measure the model's accuracy, robustness, and ability to detect adversarial examples.After training, the patterns observed from the movement are helpful to the proposed model in finding exciting patterns from the input images. In the early stage of model training, the larger the learning rate, the longer the step length. Then the gradient can move down at a faster rate. However, gradually reducing the learning rate and the step length is adopted in the later stage of model training. It will help the algorithm converge and easily find the optimal solution.

Hybrid Generative Adversarial Networks (H-GANs)

Hybrid Generative Adversarial Networks (GANs) can be used to detect adversarial attacks in a number of ways. One approach is to train a hybrid GAN to generate both real and adversarial examples, and then use the generator to produce a large number of synthetic examples that are similar to the original dataset. These synthetic examples can then be used to train a detector that is capable of distinguishing between real and adversarial examples. Another approach is to use a hybrid GAN to generate adversarial examples that are specifically designed to fool a particular machine learning model. The generated adversarial examples can then be used to test the robustness of the model and to identify any weaknesses or vulnerabilities. In addition to these approaches, researchers have also explored the use of hybrid GANs in conjunction with other machine learning techniques, such as anomaly detection and clustering, to improve the accuracy and effectiveness of adversarial attack detection. By combining multiple techniques, hybrid GANs can provide a more comprehensive and robust approach to detecting adversarial attacks.

The hybrid GAN (Generative Adversarial Network) for adversarial attacks is a type of GAN that generates adversarial examples to fool a targeted machine learning model. The hybrid GAN combines the strengths of both the standard GAN and the conditional GAN.

REDVET - Revistaelectrónica de Veterinaria- ISSN 1695-7504
Vol 25, No. 1 (2024)
http://www.veterinaria.org
Article Received: Revised: Accepted:

The equations for the hybrid GAN for adversarial attacks can be represented as follows:
Discriminator loss function:

$$L_D = -E[\log(D(x))] - E\left[\log\left(1 - D(G(z))\right)\right] \quad (6)$$

Where:
D(x) is the discriminator's output for a real input x
G(z) is the generator's output for a random input z
log() is the natural logarithm
E[] denotes the expectation over the training data distribution and the generator's distribution.
The discriminator aims to differentiate between the real input x and the generator's output G(z), and its loss function is to minimize this difference.
Generator loss function:

$$L_G = -E\left[\log\left(D(G(z))\right)\right] \quad (7)$$

The generator's goal is to generate adversarial examples that can fool the targeted machine learning model. The generator aims to maximize the discriminator's output for its generated output G(z), and its loss function is to minimize this difference.

Conditional loss function:

$$L_c = ||x - G(z, y)\hat{2} \quad (8)$$

The conditional loss function measures the distance between the real input x and the generator's output G(z, y) for a given target class y. This loss function is used to ensure that the generated adversarial examples are close to the original input x in terms of the targeted class y.
By combining these loss functions, the hybrid GAN for adversarial attacks can generate high-quality adversarial examples that can fool targeted machine learning models.
The equation you provided represents the loss function for a generative adversarial network (GAN). In this equation, D_loss refers to the discriminator's loss, which is a measure of how well the discriminator is able to distinguish between real and generated samples. Let's break down the equation and explain each term using sample values:

D(x): This term represents the output of the discriminator when given a real sample x. The discriminator's goal is to correctly classify real samples as 1 (indicating real) and generated samples as 0 (indicating fake). The value of D(x) lies between 0 and 1, where higher values indicate the discriminator is confident that the input is real.
log(D(x)): Taking the logarithm of D(x) serves two purposes. First, it scales down the discriminator's loss when D(x) is close to 1, meaning it classified a real sample correctly. Second, it penalizes the discriminator more heavily when it incorrectly classifies a real sample as fake (i.e., when D(x) is close to 0).
G(z): This term represents the generator's output when given a random noise vector z. The generator tries to produce samples that resemble real data and can fool the discriminator. The generator's goal is to make D(G(z)) as close to 1 as possible, indicating that the generated sample is classified as real by the discriminator.
D(G(z)): Similar to D(x), this term represents the output of the discriminator when given a generated sample G(z). The discriminator aims to correctly classify generated samples as 0, indicating they are fake. Again, the value of D(G(z)) lies between 0 and 1.
log(1 - D(G(z))): This term is similar to log(D(x)), but it represents the loss when the discriminator misclassifies a generated sample as real. If D(G(z)) is close to 1, indicating the generator produced a convincing fake sample, the logarithm of 1 - D(G(z)) will be small. The overall D_loss is the sum of the two terms, -log(D(x)) and -log(1 - D(G(z))). This loss function encourages the discriminator to correctly classify both real and generated samples by minimizing the loss. By optimizing this loss function, the generator and discriminator can improve their respective abilities, leading to a more effective GAN training process.

Dataset Description
The MNIST dataset is a collection of handwritten digits that is widely used to train and test image processing and machine learning models. It comprises of 70,000 grayscale images, each measuring 28x28 pixels. The photos are separated into 60,000 training and 10,000 test samples. The photos contain handwritten digits ranging from 0 to 9 that are centered in the middle. Each image is labelled with the relevant numeral. The National Institute of Standards and Technology (NIST) developed the dataset, which was later changed to its current form by Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The MNIST dataset is frequently used as a benchmark dataset for image classification tasks, and several cutting-edge algorithms have been built with it as a foundation. It has also served as a standard dataset for teaching and practicing machine learning and computer vision techniques.

REDVET - Revistaelectrónica de Veterinaria- ISSN 1695-7504
Vol 25, No. 1 (2024)
http://www.veterinaria.org
Article Received: Revised: Accepted:

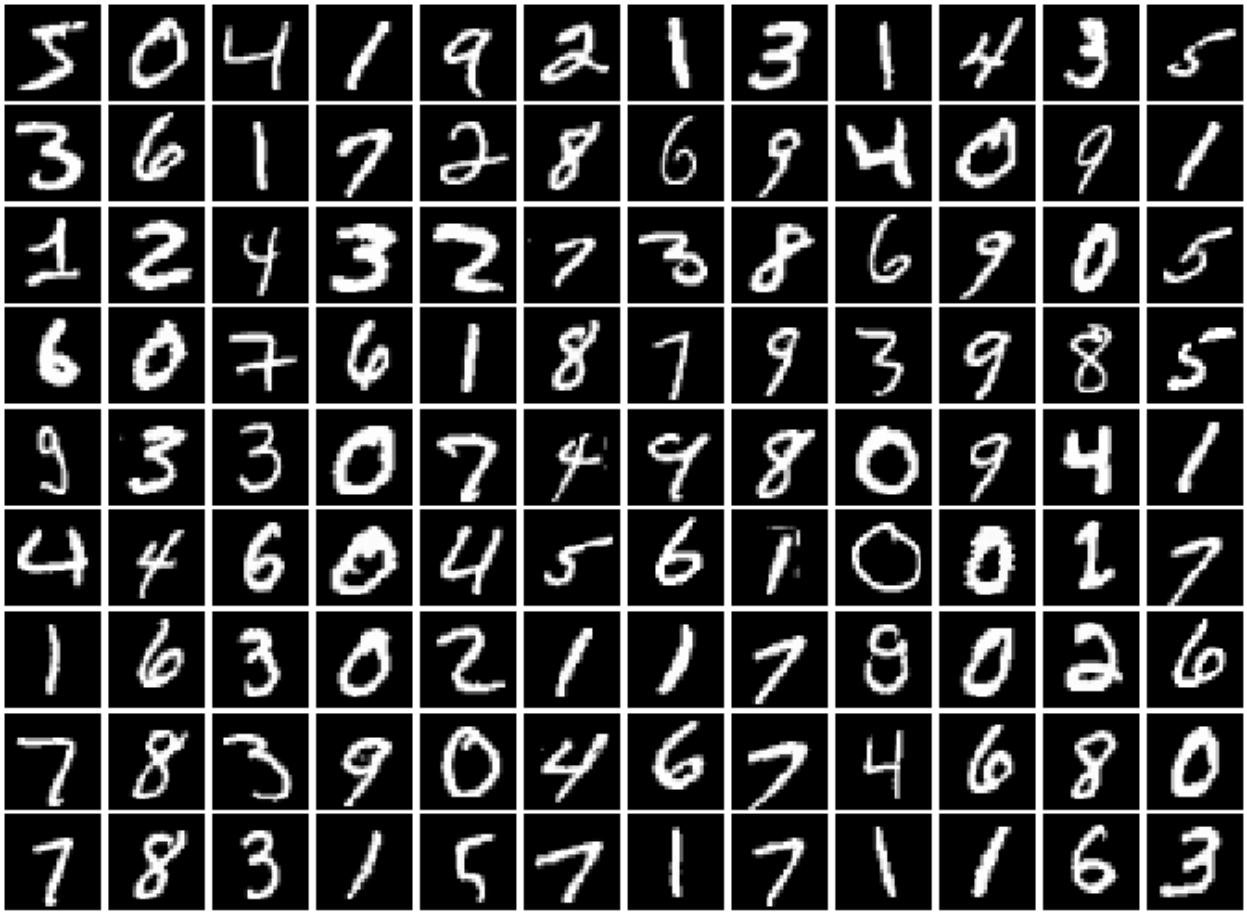


Figure 3: Sample Images in MNIST Dataset

Performance Metrics

Hybrid GANs are a type of generative adversarial network that incorporate both a generator and a discriminator, and they can be used for adversarial attacks by generating perturbations that are added to input data in order to deceive machine learning models.When evaluating the performance of hybrid GANs for adversarial attacks, the following metrics can be considered:

Mean Absolute Error (MAE)

Mean Absolute Error is the average of the difference between the ground truth and the predicted values. Mathematically, its represented as:

$$\mathrm{MAE} = \frac{1}{\mathrm{N}}\sum_{\mathrm{j}=1}^{\mathrm{N}} |\mathrm{y_j} - \check{\mathrm{y}}_\mathrm{j}| \qquad (5)$$

Where:
y_j: ground-truth value
y_{hat}: predicted value from the regression model
N: number of datums
To illustrate this with sample values, let's consider a simplified example:
Actual values: [2, 4, 6, 8]
Predicted values: [1.5, 4.2, 6.8, 8.9]
Using the MAE equation, we can calculate the mean absolute error as follows:
MAE = (1/4) * (|2 - 1.5| + |4 - 4.2| + |6 - 6.8| + |8 - 8.9|)
= (1/4) * (0.5 + 0.2 + 0.8 + 0.9)
= 0.6
Therefore, in this example, the mean absolute error (MAE) between the actual values and the predicted values is 0.6.

Root Mean Squared Error (RMSE)

RMSE is defined as the square root of the average squared difference between the target value and the regression model's projected value. Basically, sqrt(MSE). It can be mathematically represented as follows:

REDVET - Revistaelectrónica de Veterinaria- ISSN 1695-7504
Vol 25, No. 1 (2024)
http://www.veterinaria.org
Article Received: Revised: Accepted:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(y_j - \check{y}_j)^2} \qquad (6)$$

To calculate the Root Mean Squared Error (RMSE), you need a set of predicted values and the corresponding actual values. Let's say you have the following sample values:
Predicted values: [12, 15, 18, 21, 24]
Actual values: [10, 14, 17, 20, 23]
To calculate the RMSE, you can follow these steps:
Calculate the squared differences for each data point:
For the first data point: $(12 - 10)^2 = 4$
For the second data point: $(15 - 14)^2 = 1$
For the third data point: $(18 - 17)^2 = 1$
For the fourth data point: $(21 - 20)^2 = 1$
For the fifth data point: $(24 - 23)^2 = 1$
Compute the mean of squared differences:
Sum of squared differences = 4 + 1 + 1 + 1 + 1 = 8
Mean of squared differences = (8) / (5) = 1.6
Take the square root:
RMSE = sqrt(1.6) ≈ 1.2649
Therefore, the RMSE for the given sample values is approximately 1.2649.

Experimental Results

The experiments are conducted by using the python programming language. Several python libraries are used to develop and design the proposed algorithm. The dataset consists of 10k testing digit images and 60k training images. On the other side the confusion matrix is applied on overall testing samples.Performance is measured using criteria such as precision, sensitivity, specificity, and accuracy. The segmentation of brain imaging results is analyzed using a dice score. The confusion matrix assesses the proposed model's performance using the following variables: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) for a classifier.

$$\text{Specificity} = \frac{TN}{TN + FP}$$

$$\text{Sensitivity or Recall} = \frac{TP}{TP + FN}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

Table 1: Count Values of Confusion Matrix based on instances

Algorithms	True Positives (TP)	True Negatives (TN)	False Positives (FP)	False Negatives (FN)
DNN	6789	1234	878	1099
H-GNN	7897	1098	175	830

Table 2: List of Algorithms shows the performance metrics

Algorithms	Specificity	Sensitivity	Accuracy	Precision
DNN	0.4711	0.8855	0.7888	0.8462
H-GNN	0.4305	0.9783	0.8727	0.8779

REDVET - Revistaelectrónica de Veterinaria- ISSN 1695-7504
Vol 25, No. 1 (2024)
http://www.veterinaria.org
Article Received: Revised: Accepted:

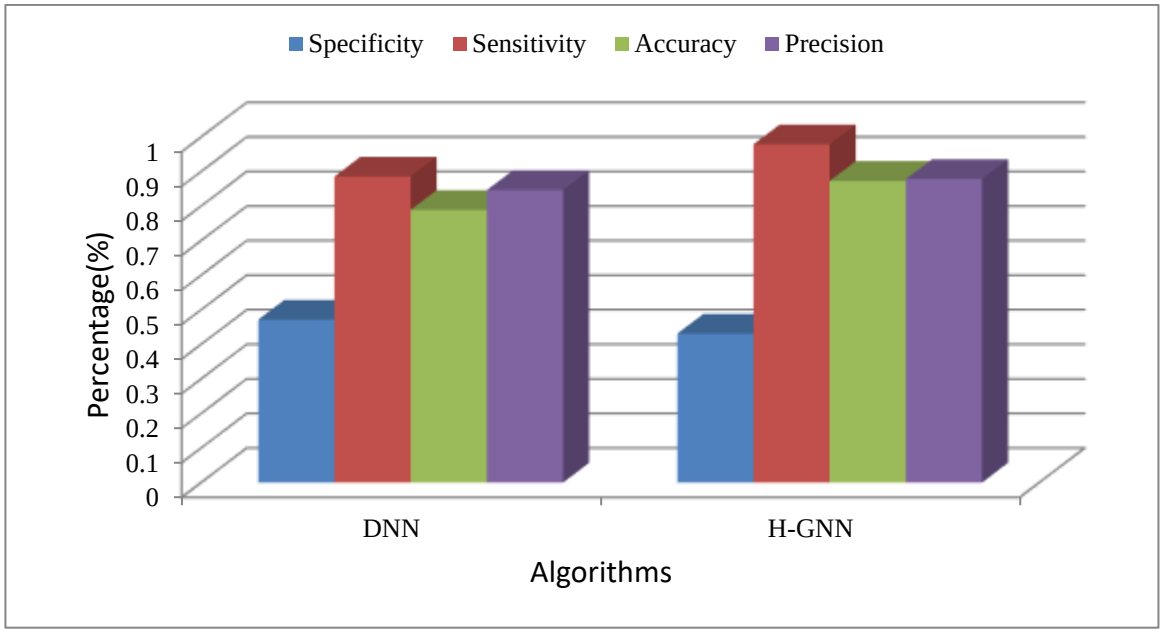


Figure 4: List of performances

Table 3: **Performance of DNN and H-GAN**

Algorithms	Mean Absolute Error (MAE)	Root Mean Squared Error (RMSE)
DNN	0.7876	0.8198
H-GNN	0.9234	0.9321

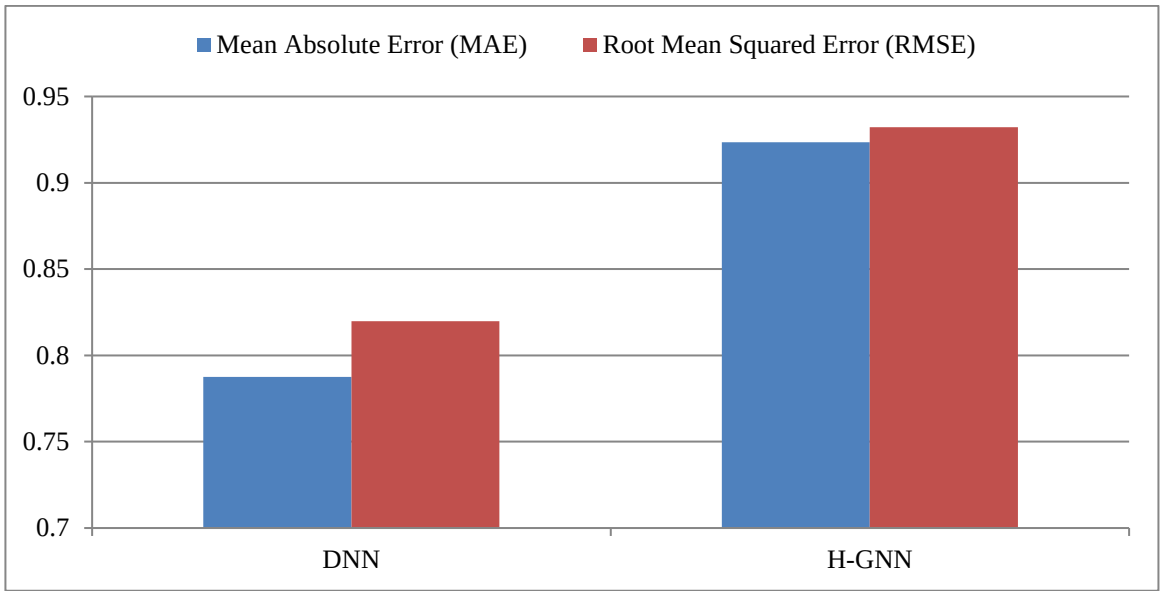


Figure 5: Comparative Performance of DNN and H-GNN

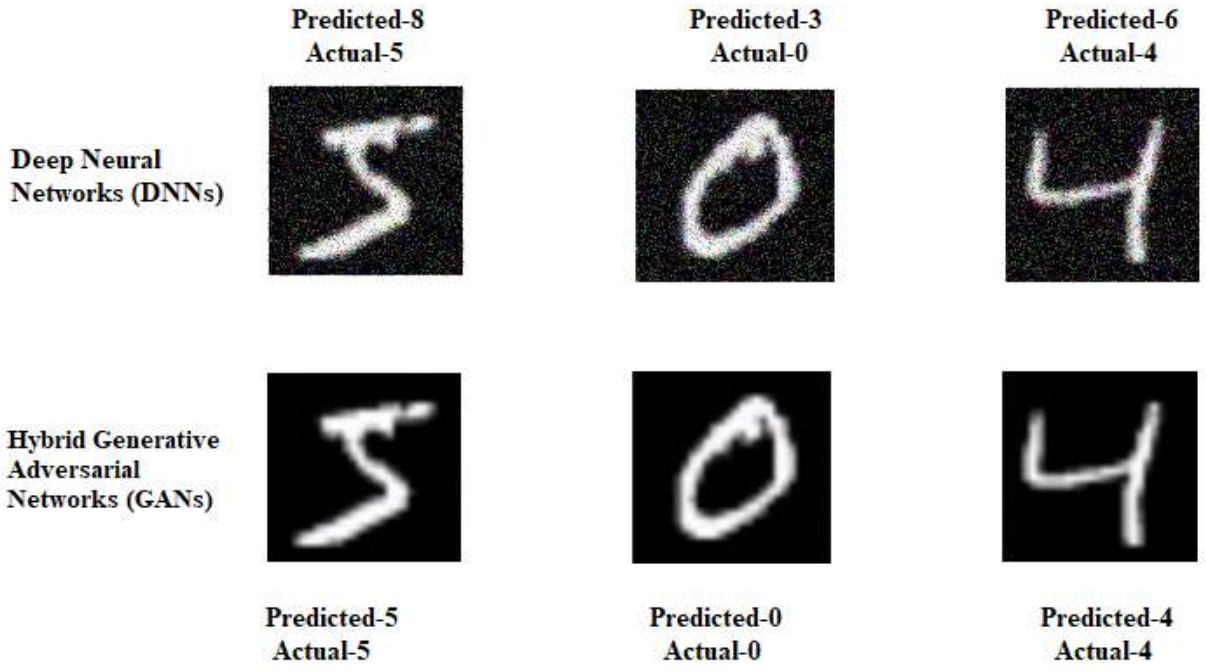


Figure 6: Performance of DNN and H-GAN

Conclusion

In this paper, A Hybrid Learning Model shows the accuracy of both the attack and defensive models of the fool state-of-the-art deep learning models attacks. This model is highly targeted to specific models or even specific instances, allowing for more precise and effective attacks to help us to the preventive and defensive mode of the attack and also secure future techniques and further studies of the boundary limit with the imbalanced data set classification by discovering the specialized optimization techniques data preparation techniques, learning algorithms, and performance metrics for One-Class data sets like MNIST Classification for the data and algorithms for the hybrid learning models Generative networks.

Therefore, researchers and practitioners must consider the potential implications and use these techniques responsibly. Overall, hybrid learning model GANs are a promising approach for generating adversarial attacks, and further research in this area can lead to improved understanding and defense mechanisms against such invasions of ethical concerns, as these attacks can be used maliciously to cause harm or manipulate the outcome of machine learning systems. The overall performance of the Hybrid Learning Model GAN achieves better with the results of foolproof techniques for the attacks by knowing the threshold values for the Boundary limit and boundary attacks of the GAN.

References

[1] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning", Nature, vol. 521, pp. 436-444, May 2015.
[2] H. Y. Xiong, B. Alipanahi, L. J. Lee, H. Bretschneider, D. Merico, R. K. C. Yuen, et al., "The human splicing code reveals new insights into the genetic determinants of disease", Science, vol. 347, no. 6218, Jan. 2015.
[3] M. Helmstaedter, K. L. Briggman, S. C. Turaga, V. Jain, H. S. Seung and W. Denk, "Connectomic reconstruction of the inner plexiform layer in the mouse retina", Nature, vol. 500, pp. 168-174, Aug. 2013.
[4] M. Amodio, D. Van Dijk, K. Srinivasan, W. S. Chen, H. Mohsen, K. R. Moon, et al., "Exploring single-cell data with deep multitasking neural networks", Nature Methods, vol. 16, no. 11, pp. 1-7, 2019.
[5] G. Hickok and D. Poeppel, "The cortical organization of speech processing", Nature Rev. Neurosci., vol. 8, no. 5, pp. 393-402, 2007.
[6] C. Manning and H. Schutze, Foundations of Statistical Natural Language Processing, Cambridge, MA, USA:MIT Press, 1999.
[9] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet classification with deep convolutional neural networks", Proc. Adv. Neural Inf. Process. Syst., pp. 1097-1105, 2012.
[10] K. Chatfield, K. Simonyan, A. Vedaldi and A. Zisserman, Return of the devil in the details: Delving deep into convolutional nets, 2014.
[11] C. Szegedy et al., "Going deeper with convolutions", Proc. IEEE Conf. Comput. Vis. Pattern Recognit., pp. 1-9, Jun. 2015.
[12] S. M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi and P. Frossard, "Universal adversarial perturbations", Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), pp. 86-94, Jul. 2017.

[13] S. Sabour, Y. Cao, F. Faghri and D. J. Fleet, Adversarial manipulation of deep representations, 2015.
[14] N. Papernot, P. McDaniel and I. Goodfellow, Transferability in machine learning: From phenomena to black-box attacks using adversarial samples, 2016.
[15] N. Narodytska and S. P. Kasiviswanathan, Simple black-box adversarial perturbations for deep networks, 2016.
[16] Y. Liu, W. Zhang, S. Li and N. Yu, Enhanced attacks on defensively distilled deep neural networks, 2017.
[17] N. Papernot and P. McDaniel, On the effectiveness of defensive distillation, 2016.
[18] S. J. Oh, M. Fritz and B. Schiele, Adversarial image perturbation for privacy protection—A game theory perspective, 2017S.
[19] K. R. Mopuri, U. Garg and R. V. Babu, Fast feature fool: A data independent approach to universal adversarial perturbations, 2017.
[20] H. Hosseini, B. Xiao, M. Jaiswal and R. Poovendran, On the limitation of convolutional neural networks in recognizing negative images, 2017
[21] C. Kanbak, S.-M. Moosavi-Dezfooli and P. Frossard, Geometric robustness of deep networks: Analysis and improvement, 2017.
[22] S. Moosavi-Dezfooli, A. Fawzi and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks", Proc. IEEE Conf. Comput. Vis. Pattern Recognit., pp. 2574-2582, Jun. 2016.
[23] Y. Dong et al., Boosting adversarial attacks with momentum, 2017.
[24] N. Carlini and D. Wagner, Adversarial examples are not easily detected: Bypassing ten detection methods, 2017.
[25] A. Rozsa, E. M. Rudd and T. E. Boult, Adversarial diversity and hard positive generation, 2016.
[26] P. Tabacof, J. Tavares and E. Valle, Adversarial images for variational autoencoders, 2016.
[27] J. Kos, I. Fischer and D. Song, Adversarial examples for generative models, 2017.
[28] D. P. Kingma and M. Welling, Auto-encoding variational Bayes, 2013.
[29] H. Robbins and S. Monro, "A stochastic approximation method", Ann. Math. Statist., vol. 22, no. 3, pp. 400-407, 1951.
[30] J. Heaton, Ian Goodfellow Yoshua Bengio and Aaron Courville: Deep Learning, Cambridge, MA, USA:MIT Press, 2018.
[31] S. Ruder, "An overview of gradient descent optimization algorithms", arXiv:1609.04747, 2016.