Targeted DeepFool: A Simple and Effective Algorithm for Fooling Deep Neural Networks to Specific Classes

S. M. Fazle Rabby Labib
s.m.fazle.rabby.labib@g.bracu.ac.bd

Brac University,
Dhaka, Bangladesh

Abstract

Deep neural networks (DNNs) have significantly advanced various domains, but their vulnerability to adversarial attacks poses serious concerns. Understanding these vulnerabilities and developing effective defense mechanisms is crucial. DeepFool, an algorithm proposed by Moosavi-Dezfooli et al. (2016), finds minimal perturbations to misclassify input images. However, DeepFool lacks a targeted approach, making it less effective in specific attack scenarios. In this paper, we propose Targeted Deepfool, an augmented version of DeepFool that allows targeting specific classes for misclassification. We also introduce a minimum confidence score requirement parameter to enhance flexibility. Our experiments demonstrate the effectiveness and efficiency of the proposed method across different deep neural network architectures while preserving image integrity. The results highlight the importance of targeted attacks in evaluating DNN robustness against adversarial manipulation.

1 Introduction

Deep neural networks (DNNs) have revolutionized many fields including but not limited to speech recognition [4, 12], computer vision [8, 10], natural language processing [24], and even game playing [20]. However, their high accuracy and robustness can be compromised by adversaries who intentionally manipulate the input data to fool the model. Such attacks can have serious consequences in real-world applications such as autonomous driving, medical diagnosis, and security systems. Therefore, understanding the vulnerabilities of DNNs to adversarial attacks and developing effective defense mechanisms has become an important research area in machine learning and computer security. DeepFool is one of the algorithms, proposed by Moosavi-Dezfooli *et al.* [15], which iteratively finds the minimum amount of perturbations required to push a given input image to a misclassified region of the feature space. They use the following equation that defines an adversarial perturbation as the minimal perturbation r that is sufficient to change the estimated label $\hat{k}(x)$:

$$\Delta(x;\hat{k}) := \min_{r} ||r||_2 \text{ subject to } \hat{k}(x+r) \neq \hat{k}(x) \tag{1}$$

where, x is an image, and $\hat{k}(x)$ is the estimated label. With this, an image can be misclassified with a minimal amount of perturbations. But this approach is not focused on any specific

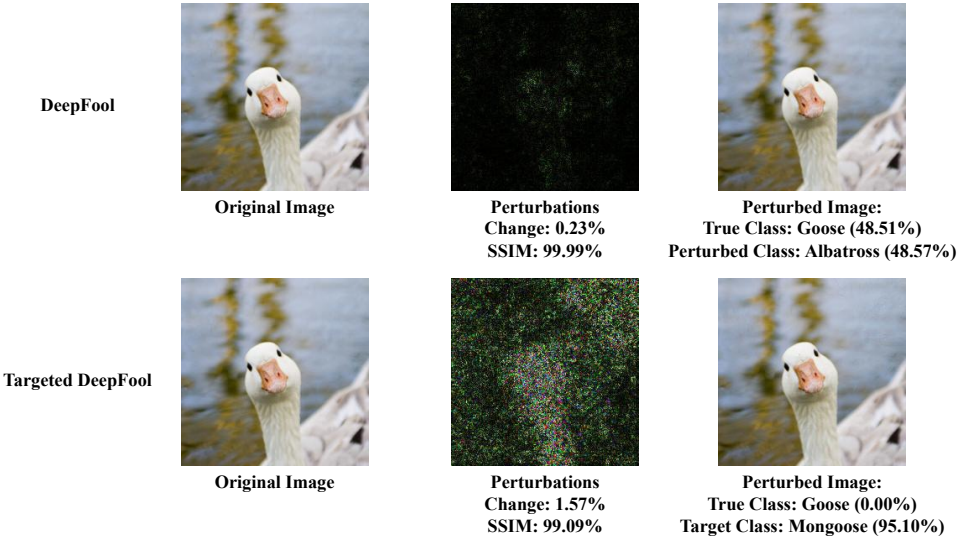

Figure 1: Comparison between original DeepFool and our proposed Targeted DeepFool. Perturbation image is scaled 20 times for visibility.

target. Instead, the images are classified as a different class with a minimal amount of perturbation. Thus, if an image x can be misclassified as some class A with less perturbation than some other class B, DeepFool will choose to use the perturbation that misclassifies x as class A. While small perturbations by untargeted attacks can fool a deep neural network by misclassifying data, targeted attacks may be more harmful as they aim to deceive the DNN into producing a specific output. An attacker may be able to deceive a self-driving car into misidentifying a stop sign as a green light. Or it can be used to fool security systems that use DNNs for face recognition. Therefore an accurate method of targeted attack to fool DNNs are necessary to make the models more robust against these type of attacks. While the DeepFool algorithm is a good approach to finding minimal perturbations to misclassify data to an unspecific target, a targeted approach is much needed.

To fill the gap, in this paper, we propose Targeted Deepfool, an approach to the DeepFool algorithm where we can target the class that we want to misclassify. We also want to augment DeepFool and make it more parametrized by giving the option to set minimum confidence score requirement. We show that the algorithm is simpler than the original, in terms of time complexity, and effective to fool different deep neural network architectures towards specific classes. Afterwards, we examine the performance of the proposed method. Our experiments show that the proposed system performs really efficiently on different devices, keeping the integrity of the image almost similar to the original one.

2 Related Works

In this section, we cover existing literature related to different adversarial attacks and data poisoning techniques against image classification models.

Adversarial attacks are done on data to perturb it to some extent so that it gets misclassified by an ML model. These attacks can be implemented in several ways in the form of black-box, white-box, and grey-box attacks. There are also data poisoning attacks which include label flipping, clean label, and backdoor attacks. Goodfellow *et al*. [9] present Fast Gradient Sign Method (FGSM), a type of adversarial attack for image classification that involves adding minimal noise to each pixel of an image, based on the gradient of the loss function with respect to the image. The required gradient is computed efficiently by using backpropagation. The algorithm proposed by Carlini and Wagner [2] finds the smallest noise to be added to an image to misclassify it. The Adversarial Patch attack, as proposed by Brown *et al*. [1], involves the creation of a small image patch that can be placed in the real world. When photographed and presented to an image classifier, the patch can cause the image to be misclassified as a chosen target class. The patch can be designed to be small and inconspicuous which makes it difficult to detect. The Universal adversarial perturbations by Moosavi-Dezfooli *et al*. [16] fools a deep neural network by adding the same perturbations to multiple images, causing it to misclassify all of the affected images. The paper by Shafahi *et al*. [19] applies a one-shot poisoning attack by injecting a single poison instance into a clean dataset, causing the model to misclassify a specific target instance without negatively impacting its performance on other examples. They also introduced a "watermarking" strategy that makes the poisoning more reliable. Huang *et al*. [13] proposed MetaPoison, a meta-learning approach for crafting poisons to fool neural networks using clean-label data poisoning. Their method outperforms previous approaches on various datasets and architectures, even for proprietary ML-as-a-service models that cannot be accessed or modified. This paper by Gao *et al*. [7] presents a black-box attack method called Patch-wise Iterative Fast Gradient Sign Method that outperforms pixel-wise methods in generating transferable adversarial examples against various mainstream models. The method can be used as a baseline for generating more transferable adversarial attacks. This paper by Zhao *et al*. [26] presents a GAN-based method for generating realistic adversarial examples for black-box classifiers in visual and textual tasks. Their approach involves training a generator network to produce perturbations that can be added to the original input to create an adversarial example. This paper by Di *et al*. [5] presents a camouflaging approach for targeted poisoning attacks based on the gradient-matching approach of Geiping et al [8]. The method generates a new set of points to undo the impact of the poison set, achieving better success rates than the poisoning approach on various models and datasets. Muñoz-González *et al*. [17] proposed pGAN, a scheme that generates poisoning points to maximize the error of a target classifier while minimizing detectability by a discriminator. The authors highlighted that pGAN can be used to study the tradeoffs between a system's performance and robustness as the number of poisoning points increases. Zhao *et al*. [27] proposed PerC-C&W and PerC-AL, two methods for creating adversarial images that use perceptual color distance to improve their imperceptibility.

3 Methodology

3.1 Background of Vanilla DeepFool

For the multiclass classifier, each class is on a hyperplane that divides one class from another, and $\boldsymbol{x}$ is the input that gets classified to whichever hyperplane it lies on. The original DeepFool algorithm finds the closest hyperplane and pushes $\boldsymbol{x}$ towards it and makes it mis-

classified with the minimum amount of perturbation. This is done iteratively until the image is misclassified. In the end, the algorithm returns the total perturbations $\hat{\boldsymbol{r}}$. The following equations are used to calculate the closest hyperplane, $\hat{l}$ where $\boldsymbol{w}'_k$ is a vector that points in the direction of the decision boundary between the predicted label and the k_{th} largest activation. This is done by subtracting the gradient of the predicted label from the gradient of the k_{th} largest activation. f'_k is the difference between the labels:

$$\boldsymbol{w}'_k \leftarrow \nabla f_k(\boldsymbol{x}_i) - \nabla f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i) \tag{2}$$

$$f'_k \leftarrow f_k(\boldsymbol{x}_i) - f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i) \tag{3}$$

After calculating $\boldsymbol{w}'_k$ and f'_k, the following equation calculates the closest hyperplane $\hat{l}$ and the minimum amount of perturbation for k_{th} iteration, r_i:

$$\hat{l} \leftarrow \arg\min_{k \neq \hat{k}(\boldsymbol{x}_0)} \frac{|f'_k|}{||\boldsymbol{w}'_k||_2} \tag{4}$$

$$\boldsymbol{r}_i \leftarrow \frac{|f'_{\hat{l}}|}{||\boldsymbol{w}'_{\hat{l}}||_2^2} \boldsymbol{w}'_{\hat{l}} \tag{5}$$

Whenever $\hat{k}(\boldsymbol{x}_i)$ changes into a different label, the loop is stopped and total perturbation, $\hat{r}$ is returned.

3.2 Targeted DeepFool

Now to turn the original DeepFool algorithm to misclassify an image into a specific target class we propose the algorithm shown in Algorithm 2 below.

Algorithm 1 DeepFool: multi-class case

Input: Image $\boldsymbol{x}$, classifier f.

Output: Perturbation $\hat{\boldsymbol{r}}$.
Initialize $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}, i \leftarrow 0$.
while $\hat{k}(\boldsymbol{x}_i) = \hat{k}(\boldsymbol{x}_0)$ **do**
  for $k \neq \hat{k}(\boldsymbol{x}_0)$ **do**
    $\boldsymbol{w}'_k \leftarrow \nabla f_k(\boldsymbol{x}_i) - \nabla f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
    $f'_k \leftarrow f_k(\boldsymbol{x}_i) - f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
  end for
  $\hat{l} \leftarrow \arg\min_{k \neq \hat{k}(\boldsymbol{x}_0)} \frac{|f'_k|}{||\boldsymbol{w}'_k||_2}$
  $\boldsymbol{r}_i \leftarrow \frac{|f'_{\hat{l}}|}{||\boldsymbol{w}'_{\hat{l}}||_2^2} \boldsymbol{w}'_{\hat{l}}$
  $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + \boldsymbol{r}_i$
  $i \leftarrow i + 1$
end while
return $\hat{\boldsymbol{r}} = \sum_i \boldsymbol{r}_i$

Algorithm 2 Proposed: Targeted DeepFool

Input: Image $\boldsymbol{x}$, classifier f,
target class t.
Output: Perturbation $\hat{\boldsymbol{r}}$.
Initialize $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}, i \leftarrow 0$.
while $\hat{k}(\boldsymbol{x}_i) \neq t$ **do**

  $\boldsymbol{w}'k \leftarrow \nabla f_t(\boldsymbol{x}_i) - \nabla f_{k(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
  $f'_k \leftarrow f_t(\boldsymbol{x}_i) - f_{k(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$

  $\hat{l} \leftarrow \frac{|f'_k|}{||\boldsymbol{w}'_k||_2}$
  $\boldsymbol{r}_i \leftarrow \frac{|f'_{\hat{l}}|}{||\boldsymbol{w}'_{\hat{l}}||_2^2} \boldsymbol{w}'_{\hat{l}}$
  $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + \boldsymbol{r}_i$
  $i \leftarrow i + 1$
end while
return $\hat{\boldsymbol{r}} = \sum_i \boldsymbol{r}_i$

Figure 2: Comparison between the original DeepFool and our proposed Targeted DeepFool algorithm.

Instead of running the loop till the image gets misclassified, we run it till the current label is not equal to target label. We also remove the for loop that is shown in line 6 of Algorithm 1, because we are not calculating the gradients of the best n classes that have the most probability to be classified after the original class. This also decreases the time complexity by $O(n)$. We change the equations 2 and 3 to the ones shown below where $\boldsymbol{w}'_k$ is now calculating the difference between the gradients for the target class and the true class. f'_k is calculating the perturbations needed with respect to target class and true class.

$$\boldsymbol{w}'k \leftarrow \nabla f_t(\boldsymbol{x}_i) - \nabla f_{k(\boldsymbol{x}_0)}(\boldsymbol{x}_i) \tag{6}$$

$$f'_k \leftarrow f_t(\boldsymbol{x}_i) - f_{k(\boldsymbol{x}_0)}(\boldsymbol{x}_i) \tag{7}$$

Since we are not comparing between the best n classes anymore we change the equation 4 to the one below:

$$\hat{l} \leftarrow \frac{|f'_k|}{||\boldsymbol{w}'_k||_2} \tag{8}$$

Only with these small changes, we are able to successfully misclassify an image to a specific class of our choosing.

4 Experimental Results

Here we apply our proposed method on multiple state of the art image classification models and show our findings.

4.1 Experiments

Dataset: We are using the validation images of the ILSVRC2012 [18] dataset for our experiments. It contains 50 thousand images with thousand different classes.
Models: We execute differerent pre-trained deep neural networks, such as, ResNet50 [11], AlexNet [14], EfficientNet_v2 [23], GoogLeNet [21], and Inception_v3 [22] to work on our proposed architecture. We also use one of the state-of-the-art architecture, Vision Transformer [6] image classification model to test our method. We use PyTorch 2.0 in different testbed systems, which include CUDA-enabled GPUs like RTX 3060 Ti, and RTX 3070 Ti.

For the tests, we use the validation images and generated a random target class that is not its true class. These images along with the target classes that were generated are fed into our function. We use several hyper-parameters such as overshoot which is set to 0.02, this is used as a termination criterion to prevent vanishing updates and oscillations. We set the minimum amount of confidence needed as 95% and the maximum iterations as 100. This is done because in most cases the confidence score of perturbed image is usually lower than expected (~60%), therefore we add another condition in the while loop to make the code run until desired confidence is reached. Although, this will lead to more perturbations. The code will run until these conditions are met or until maximum iterations is reached. These hyper-parameters can be tuned to one's needs.

We calculate the confidence score for the target class by passing the output tensor through softmax function. We find the change in image by calculating the L2 distance between

perturbed and original image and dividing with maximum L2 distance. We also calculate the Structural Similarity Index Measure (SSIM) [25] between the perturbed and original image and the number of iterations needed to perturb an image.

4.2 Results

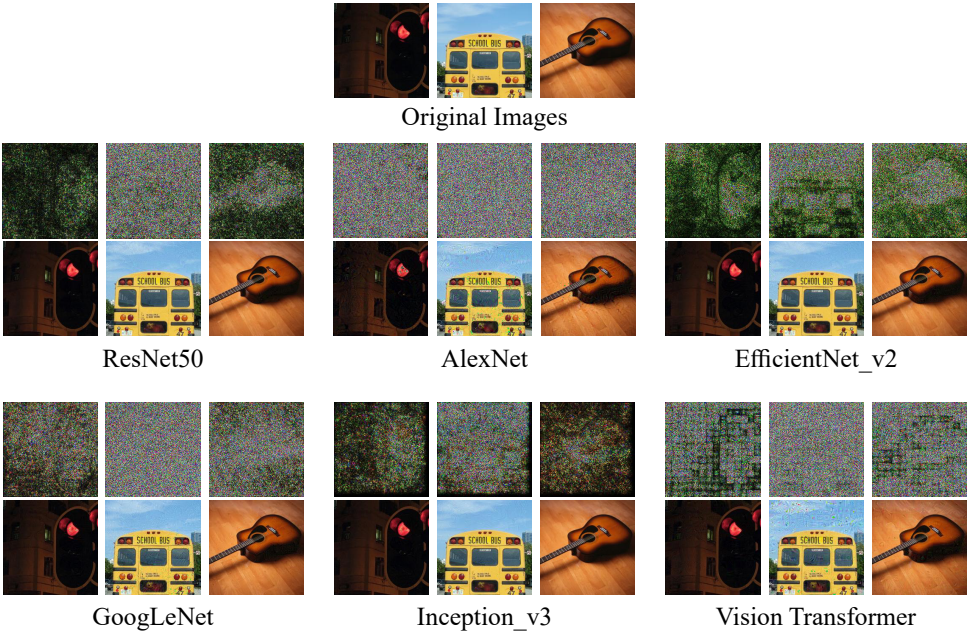

Figure 3: Some sample images from our experiments. The perturbed classes are as follows: Traffic light as Manhole cover, School bus as Ambulance, Acoustic guitar as Assault Rifle. Perturbations shown in second row are scaled 20 times for visibility

Classifier	Confidence	Perturbations	SSIM	Iterations	Success	Time
ResNet50*	0.97	2.14%	0.99	29	0.97	0.37 s
AlexNet*	0.97	9.08%	0.92	25	0.94	0.52 s
EfficientNet_v2**	0.97	3.37%	0.98	33	0.97	0.31 s
GoogLeNet**	0.97	3.45%	0.97	33	0.97	1.48 s
Inception_v3*	0.97	2.35%	0.99	38	0.97	1.14 s
Vision Transformer*	0.96	11.27%	0.89	67	0.89	2.36 s

Table 1: Comparative analysis of our method's performance on various classifiers. These are the mean values from our experiment results. Here, * beside the name of the classifier means the model is run on an RTX 3070 Ti, and ** means RTX 3060 Ti is used to run the classifier. Also, time means the average time it takes to run on a single image.

In Table 1, we see the results obtained from our comprehensive evaluation of various popular image classification models.

The confidence score reflects the classifier's level of confidence in its predictions for the perturbed images. Our method has generated perturbed images with a mean confidence score of 0.97, indicating that the classifiers have a 97% confidence level in their prediction of the target class for the image. This is made possible by setting the minimum confidence hyperparameter at 0.95.

The magnitude of perturbations added to the images, referred to as "Perturbations," quantifies the level of changes required to deceive the classifier. Notably, the classifiers ResNet50, EfficientNet_v2, GoogLeNet and Inception_v3 revealed a considerable vulnerability to our approach, with perturbation rates ranging from 2.14% to 3.37%. In contrast, AlexNet and Vision Transformer require the most amount of perturbations to fool, with rates of 9.08% and 11.27% respectively.

ResNet50, EfficientNet_v2, GoogLeNet, and Inception_v3 consistently exhibit high mean SSIM scores, ranging from 97 to 99. On the other hand, AlexNet and Vision Transformer have the lowest mean SSIM scores, with 0.92 and 0.89, respectively.

The "Iterations" metric indicates the number of iterations required to achieve a successful misclassification. The attack against classifiers EfficientNet_v2, GoogLeNet, and Inception_v3 perform consistently well, with iteration counts ranging from 33 to 38. However, Vision Transformer requires a significantly higher number of iterations to fool it, with an average of 67 iterations per image.

The success metric shows the percentage of images being successfully misclassified as the randomly selected target image. The attack consistently succeeds against 97% of the dataset when applied against ResNet50, EfficientNet_v2, GoogLeNet and Inception_v3. Keeping up with the trend of other metrics, the attack has the lowest success rate against AlexNet and Vision Transformer, achieving success rate of 94% and 89% respectively.

Finally, the computational time needed to execute the attack against a single image is denoted as "Time." Notably, the attack against EfficientNet_v2 and Inception_v3 exhibited faster execution times, requiring approximately 0.31 seconds and 1.14 seconds per image respectively. On the other hand, Vision Transformer requires the highest computational overhead, with an average execution time of 2.36 seconds per image.

Overall, we find our method to be effective in various degrees against most of these classifiers. The results provide insights into the comparative strengths and weaknesses of the given image classifiers under adversarial conditions, which can aid in developing improved defense mechanisms and enhancing the overall robustness of image classifiers.

5 Discussion

During the development of the targeted DeepFool method, we observe an interesting phenomenon. While the original method exhibits the ability to misclassify an image with a minimal amount of perturbation, we notice that the confidence score associated with the perturbed image is often very low. This can also be seen in figure 1. This discovery prompts us to further investigate this issue and devise a solution to address it.

To tackle this issue, we introduce an important hyperparameter that allows us to specify a minimum confidence threshold as required. We aim to enhance the overall confidence of

the perturbed images while maintaining their effectiveness in inducing misclassification to specific classes by incorporating this hyperparameter into our targeted DeepFool approach. One consequence of introducing this hyperparameter is an increase in the number of perturbations added to the original image. However, we find that the additional perturbations are negligible in magnitude. Moreover, we conduct a comprehensive evaluation by comparing the SSIM scores between the perturbed images and the original ones. Encouragingly, we consistently observe high SSIM scores ranging from 0.97 to 0.99 across various classifiers, as seen in table 1, indicating that the perturbations introduced by our modified method preserve the visual similarity to the original images to a remarkable extent.

6 Conclusion

In this paper, we propose an algorithm, Targeted DeepFool, which improves on the original DeepFool algorithm and makes it not only able to misclassify an image to a specific target class but also achieve the desired amount of confidence needed to fool a classifier. We show that our algorithm is simple and requires less time complexity. We demonstrate that the algorithm performs well against various state-of-the-art image classifiers. We believe that by training and fine-tuning the classifiers on the images generated by the algorithm, they will become more robust to future attacks. However, it is important to note that this work was limited to only one dataset, and future research could implement it on various datasets for a more comprehensive evaluation. Furthermore, another area of potential future research lies in devising an approach that minimizes computational requirements. Although the proposed algorithm already demonstrates improved time complexity, there is still room for optimizing its computational demands to ensure broader practical applicability.
In conclusion, by addressing the limitations of the original DeepFool algorithm, our enhanced targeted DeepFool method exhibits promising potential for improving the effectiveness and robustness of adversarial attacks. We hope, these findings contribute to the advancement of adversarial machine learning and provide a foundation for further exploration and refinement of targeted attack methods.

References

[1] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch, 2018.

[2] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks, 2017.

[3] Junyi Chai, Hao Zeng, Anming Li, and Eric W.T. Ngai. Deep learning in computer vision: A critical review of emerging techniques and application scenarios. *Machine Learning with Applications*, 6:100134, 2021. ISSN 2666-8270. doi: https://doi.org/10.1016/j.mlwa.2021.100134. URL https://www.sciencedirect.com/science/article/pii/S2666827021000670.

[4] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964, 2016. doi: 10.1109/ICASSP.2016.7472621.

[5] Jimmy Z. Di, Jack Douglas, Jayadev Acharya, Gautam Kamath, and Ayush Sekhari. Hidden poison: Machine unlearning enables camouflaged poisoning attacks, 2022.

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[7] Lianli Gao, Qilong Zhang, Jingkuan Song, Xianglong Liu, and Heng Tao Shen. Patch-wise attack for fooling deep neural network, 2020.

[8] Jonas Geiping, Liam Fowl, W. Ronny Huang, Wojciech Czaja, Gavin Taylor, Michael Moeller, and Tom Goldstein. Witches' brew: Industrial scale data poisoning via gradient matching, 2021.

[9] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.

[10] Khurram Azeem Hashmi, Didier Stricker, and Muhammamd Zeshan Afzal. Spatio-temporal learnable proposals for end-to-end video object detection. 2022.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

[12] Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29 (6):82–97, 2012. doi: 10.1109/MSP.2012.2205597.

[13] W. Ronny Huang, Jonas Geiping, Liam Fowl, Gavin Taylor, and Tom Goldstein. Metapoison: Practical general-purpose clean-label data poisoning, 2021.

[14] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks, 2014.

[15] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks, 2016.

[16] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations, 2017.

[17] Luis Muñoz-González, Bjarne Pfitzner, Matteo Russo, Javier Carnerero-Cano, and Emil C. Lupu. Poisoning attacks with generative adversarial nets, 2019.

[18] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[19] Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks, 2018.

[20] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[21] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[22] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

[23] Mingxing Tan and Quoc V. Le. Efficientnetv2: Smaller models and faster training, 2021.

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[25] Z. Wang, E.P. Simoncelli, and A.C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402 Vol.2, 2003. doi: 10.1109/ACSSC.2003.1292216.

[26] Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples, 2018.

[27] Zhengyu Zhao, Zhuoran Liu, and Martha Larson. Towards large yet imperceptible adversarial image perturbations with perceptual color distance, 2020.