AFEKA The Academic College of Engineering in Tel Aviv

School of Software Engineering: Intelligent Systems

Adversarial examples for social good

A project submitted toward the degree of
Master of Science in Intelligent Systems

Student Name: Michael Gudovsky

ID Number: 310845102

Supervisor Name: Dr. Chen Hajaj

Advisor Name: Dr. Yehudit Aperstein

Submission Date: 01/08/2023

Acknowledgments

I would like to express my deepest appreciation and gratitude to all those who have contributed to the completion of this project.

First and foremost, I am immensely grateful to my supervisors, Dr. Yehudit Aperstein and Dr. Chen Hajaj, for their guidance, support, and valuable insights throughout this research endeavor. Their expertise, patience, and continuous encouragement have been instrumental in shaping the direction of this project and enhancing its quality.

I would also like to extend my sincere thanks to the faculty members of software engineering at Afeka college of engineering, for providing me with a rich academic environment and the necessary resources to conduct this research. Their dedication to fostering an atmosphere of learning and intellectual growth has been truly inspiring.

I am indebted to my fellow classmate, Nathanael Guinzburg who has been a constant source of encouragement and motivation during this project. Their insightful discussions, constructive feedback, and unwavering support have significantly contributed to the development and refinement of my ideas.

Finally, I would like to acknowledge my wife, Ella Gudovsky, for her unwavering love, encouragement, and understanding throughout my academic journey. Her constant support and belief in my abilities have been a driving force behind my accomplishments.

Although it is not possible to name everyone individually, I am deeply thankful to all those who have directly or indirectly contributed to the completion of this project. Your collective efforts have played an integral role in shaping its outcomes.

In conclusion, this project would not have been possible without the guidance, support, and contributions of the aforementioned individuals and institutions. I am honored and grateful to have had the opportunity to work on this research, and I sincerely hope that its outcomes contribute positively to the field of study.

Thank you.

Abstract

As the use of the internet grows daily, keeping sensitive data away from prying eyes becomes a necessity. One way of doing so is asking users for a password when trying to log in to a service or a website that includes sensitive data (e.g., an online bank account). In this everlasting battle between hackers and online security measures, hackers try to bypass security by different means; some are sophisticated while others are as simple as brute force. Some websites use CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) to protect against such attacks and ensure that human beings are trying to log in and not a software bot [1]. Common to all CAPTCHAs is the fact that they present the user with a sample that software bots will fail to recognize but are easily identifiable by humans. In many cases, these samples are not simplistic ones but rather adversarial ones aimed to mislead the software bot. These adversarial samples are known as Adversarial Examples and can be distorted text, a perturbated image, a noisy audio clip, and other challenges [15].

Adversarial Examples are malicious inputs designed to fool machine learning models. They often transfer from one model to another, allowing attackers to mount black box attacks without knowing the target model's parameters. The goals of adversarial attacks are confidence reduction, misclassification, targeted misclassification, and source\target misclassification.

In this research, we focus on generalizing the Deep-Fool adversarial attack algorithm originally designed to deceive deep neural networks by crafting minimal perturbations to input samples [13]. Like many adversarial attack algorithms, Deep-Fool is typically considered in the white-box setting, where the attacker has complete knowledge of the model, including its architecture and parameters. This allows the attacker to compute the necessary gradients for constructing adversarial examples. Deep-Fool uses this knowledge to iteratively compute the minimal perturbation required to misclassify an input image by estimating the linear decision boundaries of the network and determining the closest boundary to the original image.

One of the main limitations of the Deep-Fool algorithm is its reduced effectiveness in a black-box attack scenario. In a black-box setting, the attacker does not have access to the internal details of the target model, such as its architecture or parameters. This lack of information is particularly problematic for Deep-Fool, as the algorithm relies heavily on gradient information to compute the minimal perturbations needed to cause misclassification. Without access to these gradients, the ability of Deep-Fool to generate effective adversarial examples is significantly hindered. Therefore, while Deep-Fool can be highly effective in a white-box setting where full model details are known, its performance can degrade substantially when such information is not available.

This study, compares different methods, including Deep-Fool based models, Saliency Map based model, a naïve Gaussian noise model, and pre-trained adversarial models. The comparison is based on the structural similarity index (SSIM) and signal-to-noise ratio (SNR). We present a clear visualization of the original image, perturbation impact on the images, and classification success of SOTA classifiers to the perturbated images.

Importantly, we found the threshold that ensured that the perturbation is maximized for better misclassification rate, preserving the correct perception and classification of the image. Our research demonstrated that CCUAP affected the accuracy of the classifiers at higher SNR values compared to the naïve model. This indicates that CCUAP method introduces a more impactful perturbation at lower noise intensity.

The code for this research work [28] can be downloaded from GitHub.

Table of Contents

List of abbreviations

CAPTCHA	– Completely Automated Public Turing test to tell Computers and Humans Apart
SSIM	– Structural Similarity Index
SNR	– Signal to Noise Ratio
SOTA	– State of The Art
DL	– Deep Learning
BGD	– Background
DF	– Deep Fool
RT	– Random Tensor
BI	– Background Image
ADFT	– Averaging Deep-Fool Perturbation
ADFT_RT	– ADFT multiplied by Random Tensor
ADFT_RT_BI	– ADFT_RT with addition of Background Image
CCUAP	– Conditional Cumulative Universal Adversarial Perturbation

List of equations

List of figures

List of tables

Adversarial examples for social good

Michael Gudovsky

Dr. Yehudit Aperstein | Dr. Chen Hajaj | 01/08/2023

Chen Hajaj

1. Abstract

Adversarial Examples are malicious inputs designed to fool machine learning models. They often transfer from one model to another, allowing attackers to mount black box attacks without knowing the target model's parameters. The goals of adversarial attacks are confidence reduction, misclassification, targeted misclassification, and source\target misclassification. In this research, we apply adversarial examples to ImageNet dataset. We design and evaluate different universal adversarial perturbation techniques to find the most efficient universal perturbation. Here, efficiency refers to the success rate of the adversarial attack. A more efficient attack is one that causes misclassification for a larger fraction of inputs when the perturbation is applied. We then assess the performance of classifiers on adversarial examples derived from the universal perturbations and compare it with their performance on data modified using a naïve approach, such as the addition of Gaussian noise. We compare the performance for different levels of noise. Our contribution includes expanding the Deep-Fool algorithm for a universal perturbation, overcoming compression algorithm processing, and high transferability of the perturbation to other models.

2. Introduction

As the use of the internet grows daily, keeping sensitive data away from prying eyes becomes a necessity. One way of doing so is asking users for a password when trying to log in to a service or a website that includes sensitive data (e.g., an online bank account). In this everlasting battle between hackers and online security measures, hackers try to bypass security by different means; some are sophisticated while others are as simple as brute force. Some websites use CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) [1] to protect against such attacks and ensure that human beings are trying to log in and not a software bot. Common to all CAPTCHAs is the fact that they present the user with a sample that software bots will fail to recognize but are easily identifiable by humans. In many cases, these samples are not simplistic ones but rather adversarial ones aimed to mislead the software bot. These adversarial samples are known as Adversarial Examples and can be distorted text, a perturbated image, a noisy audio clip and other challenges.

A successive adversarial algorithm should meet several requirements and have a good tradeoff between those requirements. The added noise should be strong enough to fool the DL architecture, to be robust against preprocessing tools and yet, not affect humans' ability to recognize the object of interest in the image correctly.

In this study we search for an efficient universal perturbation which will be image and network agnostic and will make classification model misclassify an image with high probability. We construct different models based on deep fool algorithm and select the most efficient one based on misclassification of the models, SSIM and SNR and compare them to naïve perturbation based on Gaussian noise. We aim to maximize the tradeoff between a high misclassification of state-of-the-art models and invisibility to human eye perturbation. We show the transferability of the perturbation to different classification models and then focus on one specific model for model evaluation. We then investigate Saliency-Map-based perturbation and integrate it into one of our models. The dataset in our research is the ImageNet data provided to competitors of ImageNet Object Localization Challenge [3]. State-of-the-art pre-trained classification models from 'Pytorch'[5] are used to evaluate the adversarial examples' results.

The deliverables of this project may be used as an attack on DL models (e.g., attacking a model of face recognition to classify unauthorized personnel as authorized personnel) as well as in defending mechanisms for BOT detection (e.g., CAPTCHAs [6].)

We evaluate the attacking performance of the generated adversarial perturbation by *Accuracy reduction*, *Transferability* [16], and *Robustness to image preprocessing tools***.**

3. Literature Review

Osadchy et al. [6] presented the DEEP-CAPTCHA, a secure new CAPTCHA mechanism based on immutable adversarial noise that deceives DL tools and cannot be removed using preprocessing. DeepCAPTCHA offers a playful and friendly interface for performing one of the most loathed

Internet-related tasks — solving CAPTCHAs. The Jacobian-based saliency map approach (JSMA) was introduced by Papernot et al. [8], this method iteratively perturbs input features with large adversarial saliency scores. Intuitively, this score reflects the adversarial goal of taking a sample away from its source class towards a chosen target class. JSMA allows us to select the most important pixel (maximum gradient) based on saliency map and then perturb the pixel to increase the likelihood of labeling the image as the target class.

Moosavi-Dezfooli et al. [13] introduced the Deep-Fool architecture, a simple and accurate method to fool deep neural networks; the authors propose an algorithm to efficiently compute perturbations that fool deep networks and thus reliably quantify the robustness of these classifiers.

Specifically, by adding such a quasi-imperceptible perturbation to natural images, the label estimated by the deep neural network is changed with high probability. Their fooling rates were very impressive and the perturbation was minimalistic and almost invisible to the human eye. In their research they found the universal perturbation in a method that is very similar to our work but they didn't put a weight on increasing the perturbation and finding a threshold between the perturbation visibility and fooling success of the networks. Furthermore, they also did not emphasize comparing their method to any other SOTA methods or even a naïve model. [14]. The Carlini-Wagner (C&W) attack was introduced by Carlini et al. [18], the authors formulate finding adversarial examples as an optimization problem; find some slight change δ that can be made to an input x that will change its classification, but so that the result is still in the valid range. The attack with L_0, L_2 and L_∞ distance norm can be targeted or nontargeted. In general, high confidence attacks have large perturbations and high transfer rates, and CW attack based on L_0, L_2 and L_∞ distance metric can defeat the defensive distillation [21]. The fast gradient sign method (FGSM) was introduced by Goodfellow et al. [11] the intuition behind the attack is to linearize the cost function, used to train a model around the neighborhood of the training point that the adversary wants to force the misclassification of. The generated images are misclassified by adding perturbations and linearizing the cost function in the gradient direction. Mopuri et al. [22] propose a novel data independent approach to generate image agnostic perturbation for a range of CNNs trained for object recognition. They show transferability across multiple network architectures trained either on the same or different data. Unlike the previous work mentioned, the perturbation is well visible and the fooling success with an emphasis to the perturbation transferability to another network is not as good. Phung et al. [23] introduces a novel adversarial attack method for image spam classifiers. The unique aspect of their approach is the creation of "universal" perturbations, which are image-agnostic and can be applied to any image spam. This property allows for efficient generation of adversarial examples, potentially enhancing their ability to evade detection by image spam classifiers. Hayes et al. [24] introduce universal adversarial networks, a generative network that is capable of fooling a target classifier when its generated output is added to a clean sample from a dataset. We show that this technique improves on known universal adversarial attacks. Mummadi et al. [25] presents a method to improve the robustness of image classifiers against universal adversarial perturbations. In this context, "universal" refers to perturbations that are input-agnostic, meaning the same perturbation can fool the classifier on many different inputs. The authors propose shared adversarial training, an extension of adversarial training that handles the trade-off between accuracy on clean examples and robustness against these universal perturbations more effectively. Metzen et al. [26] proposed attacks against semantic image seg- mentation: they present an approach for generating (universal) adversarial perturbations that make the network yield a desired target segmentation as output. they show empirically that there exist barely perceptible universal noise patterns which result in nearly the same predicted segmentation for arbitrary inputs. Chaubey et al. [27]. provide a detailed discussion on the various data driven and data-independent methods for generating universal perturbations, along with measures to defend against such perturbations. They also cover the applications of such universal perturbations in various deep learning tasks.

4. Our contribution

- We expend the Deep-Fool algorithm for a universal perturbation; we no longer need to receive an image and iterate over the Deep-Fool algorithm to perturbate it in real life but add a specific perturbation learned in the averaging process.
- Our perturbation overcomes the compression algorithm preprocessing tools: in the existing method, if one saves the perturbated image, the compression algorithm removes the perturbation, and if one tries to classify the image again, the classifier will classify it correctly.
- We show the transferability of the perturbation to other classifiers in Table 4. Specifically, we show that our perturbation affects different classifiers; despite learning using a specific architecture, the perturbated image fools' other classifiers with reasonable success.
- We compare different variation of perturbation averaging based on Deep-Fool algorithm and a naïve model based on Gaussian noise in a manner of SNR and similarity to the original image.
- We find the threshold that allows us to increase perturbation and the misclassification rate yet allow people to classify the image correctly.

5. Methodology

We research for the ultimate perturbation, which is designed step by step with different tradeoffs, subjected to the two most important demands, robustness to image processing tools and the lack of effect on the human ability for correct classification of the image. This perturbation should be independent of the attacked DL architecture and the image agnostic; this is achieved by learning the minimal perturbation. We construct different models of universal perturbation using the Deep-Fool algorithm; we start with a baseline averaging over the whole training set, where the minimal perturbation for each image is summed and averaged; then we search for improvements to this model to gain a higher misclassification success with smaller perturbation. Finally, we teach a model based on already learned perturbation over previous iterations which provides us with a lower computational costs and avoidance of summing unnecessary perturbation.

Additionally, we evaluate Saliency-map based perturbation and integrate it to Deep-Fool algorithm.

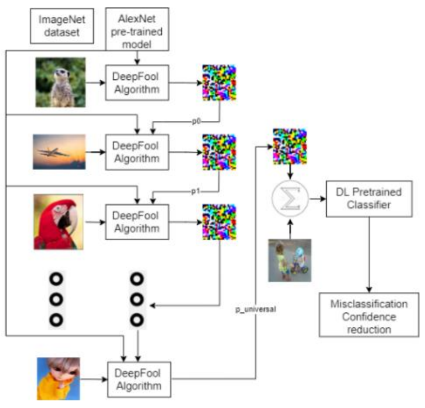


Figure 1 – High-level architecture

In Figure 1 we show an illustration of universal perturbation learning. We research for a universal perturbation to make state-of-the-art classifiers misclassify a perturbated image with a high misclassification rate.

We enter Deep-Fool model with one image and the surrogate classifier, after the first iteration we acquire the initial perturbation. From the second iteration we enter the algorithm with new image, same surrogate classifier and the perturbation acquired from previous iterations. If the previous perturbation is sufficient for changing the classification label, the model continues to the next iteration with a new image, if the previous perturbation is not sufficient, we add additional perturbation until the classification label is changed. After the learning is done, we acquire the final perturbation which is added to every image on the test set.

5.1. Data Preparation.

The ImageNet dataset is a widely used large-scale image database that has been instrumental in the development and evaluation of computer vision algorithms. The dataset covers a wide range of object categories, including animals, vehicles, household items, and natural scenes. Several benchmark models and architectures, such as AlexNet, VGGNet, and ResNet, have been trained and evaluated on the ImageNet dataset. The performance of these models on the ImageNet classification task has become a standard metric for evaluating the progress of computer vision algorithms over the years.

While sampling the images, we noticed a small amount of grayscale images; we know that the handling of those images is different from 'RGB' images. We examined the amount of grayscale images in the training and the test sets. The ratio of the distribution was approximately 98% to RGB images. We learned that for our research, we can neglect the grayscale images since their quantity is insignificant compared to the 'RGB' images. We use PyTorch transforms to preprocess the input images; first by resizing the image to size of 256, center crop the image to 224×224, and normalize it with mean and std of: mean = [0.485,0.456,0.406], std = [0.229,0.224,0.225], finally we convert the image to a multi-dimensional matrix containing elements of a single data type, a PyTorch tensor. After creating the perturbated tensor, we use the transforms again to convert the tensor back to an image type. For the visualization of the perturbation effect, we randomly drew seven images from the dataset. We used them through experiments to better visualize the changes made in each step.

5.2. Baseline Model: Deep-Fool

Deep-Fool is an adversarial attack algorithm designed to deceive deep neural networks by crafting minimal perturbations to input samples.
Assuming that the neural network is completely linear, there must be a hyperplane separating one class from another. Based on this assumption, the authors analyzed the optimal solution to this problem and constructed adversarial examples. The corresponding optimization problem is.

$$r_*(X_0) = \arg\min \|r\|_2$$

Equation 1 – Deep-Fool optimization problem

Subject to $sign(f(X_0 + r)) \neq sign(f(X_0))$, where r indicates the perturbation.
As shown in Figure 2, X_0 is the original example, $f(x)$ is a linear binary classifier, the straight line $WX + b = 0$ is the decision boundary and $r_*(X)$ is the distance from the original example to the decision boundary, i.e., the distance from X_0 to the straight line $WX + b = 0$. The distance is equivalent to the perturbation $\Delta(X_0; f)$. Therefore when $\Delta(X_0; f) > r_*(X)$, the adversarial example can be generated. In a later work, they presented a single, small image perturbation that fools a state-of-the-art deep neural network classifier on all-natural images. The authors show in this paper the existence of such quasi-imperceptible universal perturbation vectors that lead to misclassifying natural images with high probability.

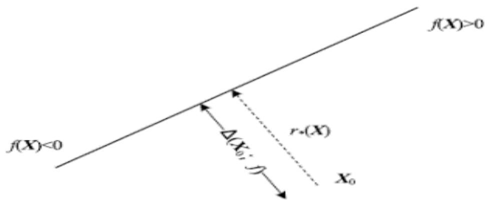


Figure 2 – Adversarial example for a binary classifier [13]

In the binary case, it can be seen as Newton's iterative algorithm for finding the roots of a nonlinear system of equations in the undetermined case [20]. This algorithm is known as the normal flow method. The Deep-Fool algorithm in the binary case can alternatively be seen as gradient decent algorithm with an adaptive step size that is automatically chosen at each iteration.

Inspired by the fact that the corresponding separating hyperplanes in linear classifiers indicate the decision boundaries of each class, Deep-Fool aims to find the least distortion (in terms of Euclidean distance) leading to misclassification by projecting the input example to the closest separating hyperplane, in other words, this attack treats the classifier as a set of linear decision boundaries. It produces perturbations pushing the adversarial image across these boundaries, eventually leading to misclassification.

We examine three methods based on Deep-Fool algorithm, first we average the perturbation learned on the fool dataset next, we multiply the averaged perturbation mask by a random tensor to reduce the perturbation and to evaluate if the accuracy will increase, finally we plant a background (BGD) image of a puzzle to see how it affects the accuracy.

The algorithm iteratively computes the minimal perturbation required to misclassify an input image by estimating the linear decision boundaries of the network and determining the closest boundary to the original image. The Deep-Fool method is given in Algorithm 1.

```
1: **input**: $Image\ X, classifier\ f.$
2: **output**: $perturbation\ \hat{r}.$
3:
4: $Initialize\ x_0 \leftarrow x, i \leftarrow 0$
5: **while** $\hat{k}(x_i) \neq \hat{k}(x_0)$ **do**
6:     **for** $k \neq \hat{k}(x_0)$ **do**
7:         $\omega'_k \leftarrow \nabla f_k(x_i) - \nabla f_{\hat{k}(x_0)}(x_i)$
8:         $f'_k \leftarrow f_k(x_i) - \nabla f_{\hat{k}(x_0)}(x_i)$
9:     **end for**
10:    $\hat{l} \leftarrow arg\ min_{k \neq \hat{k}(x_0)} \frac{|f'_k|}{\|\omega'_k\|_2}$
11:    $r_i \leftarrow \frac{|f'_{\hat{l}}|}{\|\omega'_{\hat{l}}\|_2^2} \omega'_{\hat{l}}$
12:    $x_{i+1} \leftarrow x_i + r_i, \quad i = i + 1$
13: **end while**
14: **return** $\hat{r}$
```

Algorithm 1 - Deep Fool (multi class case)

The algorithm operates greedily and is not guaranteed to converge to the optimal perturbation. It should be noted that the optimization strategy of Deep-Fool is strongly tied to other existing optimization techniques.

Table 1 is an example of the Deep-Fool algorithm implementation, the image is misclassified but to a near class; furthermore, on the macaw image, although a human will still classify it correctly, there is evidence of picture manipulation that a median filter can easily remove. For example, when we saved the perturbated image, the JPEG compression algorithm managed to remove the perturbation.

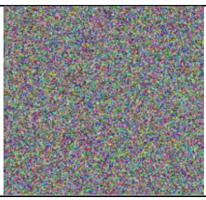
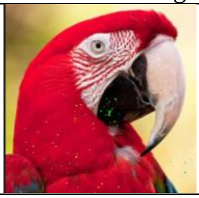
Original image	Perturbation	Perturbated image	Ground truth VS. output
			Ground truth: **Macaw** Classification: **Flamingo**

Table 1 – Deep-Fool output

5.3. Universal Perturbation Based on Averaging

We examine three methods based on Deep-Fool algorithm, first we average the perturbation learned on the fool dataset next, we multiply the averaged perturbation mask by a random tensor to reduce the perturbation and to evaluate if the accuracy will increase, finally we plant a background (BGD) image of a puzzle to see how it affects the accuracy.

5.3.1. Averaging Deep-Fool Perturbation (ADFP).

This method iterates with the Deep-Fool algorithm over the whole training set and average the perturbation for all images. The fooling success learned on a specific net is expected to have a

stronger impact on the same net performance in classifying the images correctly. We assume it will also impact other classification networks. It is not on our agenda to evaluate the correctness of the pre-trained network on the original image, and the success of the perturbation is measured based only on the change of the prediction label.
The new image is now constructed in a form of the following:

$$A.E. = image \times (1 + \alpha \times perturbation)$$

Equation 2 – ADFP

∗ α is hyperparameter which controls the perturbation strength

5.3.2. ADFP Multiplied by Random Tensor (ADFP_RT).

The motivation behind the multiplication of the universal perturbation by a random tensor lay in our desire to randomly decrease the universal perturbation. We know that summing the perturbations, which values are positive and negative can corrupt the total summation of the perturbation hence, we believe that multiplication of the averaged mask with a random tensor to achieve a better-averaged result.
The new image is now constructed in a form of:

$$A.E. = \ image \times (1 + \alpha \times random\ tensor \times perturbation)$$

Equation 3 – ADFP_RT

5.3.3. ADFP_RT and Background Image (ADFP_RT_BI).

We assume inserting a blurred background of an empty puzzle image will not make a human misclassify the image e.g., if we see a dog on its natural background or as a puzzle, we will still understand that we see a dog in the image.

We inserted, in addition to the Deep-Fool perturbation, another component to the noise, the total noise now is composed of the superposition of the Deep-Fool perturbation and a random puzzle background inserted into the image.
The new image is now constructed in the form of the following:

$$A.E. = image \times (1 + \alpha \times random\ tensor \times perturbation) + 0.25 \times puzzle\ background$$

Equation 4 – ADFP_RT_BI

Where image is the original image | puzzle background is a background blurred puzzle image shown in Figure 3| random tensor is a normalization of the Deep-Fool perturbation since we have positive and negative values | perturbation is the output of the Deep-Fool algorithm.

Figure 3 - Puzzle Background

5.4. Conditional Cumulative Universal Adversarial Perturbation (CCUAP)

The algorithm iterates the Deep-Fool steps over the whole training set. After the first iteration, we get the minimum perturbation for the first image. The next image enters the Deep-Fool algorithm already perturbated with the previous image perturbation; so; image *n* enters the Deep-Fool algorithm, already perturbated with noise from the *n-1* images before it. The perturbated image is the summation of the original image and the original image multiplied by learned perturbation. We expect the perturbation period to be shorter since after the first image we insert already perturbated images. The fooling success of this method is not expected to be degraded significantly than other methods of averaging the perturbations since, here, the overall perturbation was constructed on all the images in the training set.

```
1: $\textbf{input}$: $Training\ data\ X, classifier\ f$.
2: $\textbf{output}$: $Universal\ perturbation\ \hat{r}$.
3: $initialize\ \ r \leftarrow 0$
4: $\textbf{for}\ \ each\ image\ x_j \in |X|$ $\textbf{do}$
5:     $initialize\ x_0 \leftarrow x, i \leftarrow 0$
6:     $enter\ Deep - Fool\ with\ image\ x_0, classifier\ f, perturbation\ r$
7:     $\textbf{while}\ \hat{k}(x_i) \neq \hat{k}(x_0)$ $\textbf{do}$
8:         $\textbf{Compute the minimal perturbation with Deep\_Fool}$
9:     $\textbf{end while}$
10: $\hat{r} \leftarrow\ r$
11: $\textbf{return}\ \hat{r}$
```

Algorithm 2 – Conditional Cumulative Universal Adversarial Perturbation

5.5. Model Evaluation.

To evaluate our work, we compare investigated universal perturbation methods to a naive image modification. For our experiment we choose adding Gaussian noise to be such modification. To evaluate the perturbation models, we first examine the fooling success of the classification models and the strength of the noise visible to our eye, we searched for a threshold between a high misclassification rate and preserving correct classification by human eye. Since we cannot provide a true human classification of the perturbated images in this project's scope or infer from our classification success we use known evaluation metrics (i.e., SSIM and SNR) to measure how challenging it for humans to classify a perturbated image. Specifically, we use Structural Similarity Index (SSIM) [16] to estimate the resemblance of the perturbated and the original images. We present graphs of classification success as a function of SSIM. We expect the best model to converge slowly, meaning that lowering the perturbation noise will maintain a good enough misclassification rate of the classifiers. Another way to evaluate the effect of the perturbation on the image is by presenting the signal to noise ratio (SNR). SNR is a measure used in science and engineering to compare the level of a desired signal to the level of background noise (e.g., perturbation). We use Equation 5. to calculate the SNR between the constant noise mask and all the images in the test set, then we average the results to get the mean SNR over the test set. Since in our case we have a very wide dynamic range, we express SNR in logarithmic decibel scale.

$$SNR_{dB} = 10 \times \log_{10}\left(\frac{signal}{\sqrt{\sum noise^2}}\right)$$

Equation 5 – Signal to Noise Ratio

In the next chapter, we present graphs to show a trend of the accuracy decrease as a response to decreasing SNR, we also present visualization of the perturbated images for illustrating the perturbation effect to human eye and finally compare investigated methods to naïve model.

6. Experimental Results

6.1. Universal Perturbation Based on Averaging

In this chapter we present the evaluation of different universal perturbation techniques based on Deep-Fool perturbation.

6.1.1. Averaging Deep-Fool Perturbation (ADFP).

Table 2 presents the evaluation of different perturbation models on different classification networks; the values are the classification success of the different classifiers. The columns present on which classifier the perturbation was learned, and the rows are the evaluating classifier.

		Surrogate Model			
		VGG	AlexNet	ResNet	GoogLeNet
Target Model	VGG	30.82%	13.55%	25.20%	24.01%
	AlexNet	15.71%	15.91%	11.15%	10.60%
	ResNet	30.44%	19.56%	27.00%	24.77%
	GoogLeNet	26.29%	18.74%	31.10%	36.79%

Table 2 – Evaluation of different models on different networks

Focusing on the aspect of classifiers' robustness, we see that ResNet is the most robust between the selected classifiers and the perturbation learned on AlexNet is the most effective in fooling itself and in transferring the fooling effect to other classifiers. If we attack VGG classifier with a perturbation learned over AlexNet, the classification accuracy of VGG will be 24.04%.

6.1.2. ADFP Multiplied by Random Tensor (ADFP_RT).

We show in Table 3 that the effect of this method is slightly lower and the fooling success is accordingly lower than the baseline average.

		Surrogate Model			
Target Model		VGG	AlexNet	ResNet	GoogLeNet
	VGG	36.21%	22.17%	35.71%	28.98%
	AlexNet	30.12%	22.35%	30.77%	25.26%
	ResNet	71.66%	52.19%	68.51%	48.61%
	GoogLeNet	41.37%	18.61%	37.12%	40.38%

Table 3 – Accuracy of classifiers

6.1.3. ADFP_RT and Background Image (ADFP_RT_BI).

All the added perturbations are visible to the human eye, but it should not affect humans from classifying the image correctly. In a few cases, the perturbation did not make the classifier misclassify the image. Nonetheless, it does reduce the confidence of the class probability. A more significant achievement in that experiment is that the compression algorithm did not affect the perturbation and the new image are highly misclassified by DL architectures.

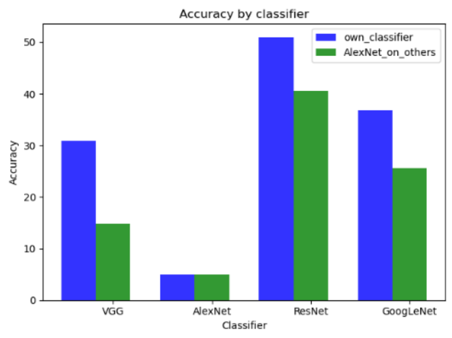


Figure 4 – Model's accuracy (ADFP)

Figure 4 and Figure 5 presents the accuracy of the classification models when attacked by adversarial examples, the blue bar represents the accuracy of the model when attacked with perturbation learned on his own model while the green bar represents the model's accuracy when attacked with perturbation learned on AlexNet. We prove our assumption of AlexNet learned perturbation is the most impactful, in the presented case, it is stronger than perturbation learned on the attacked classifier. We can also see the transferability of AlexNet learned perturbation when attacking other classification networks.

We note that Equation 4 provides us with a good threshold between fooling pre-trained classifiers and not affecting human ability to classify the image correctly. It is more rational; one will think that the image is of low quality and not that an attacker manipulated it. When the averaged perturbation was learned over one class, we saw that we achieved a high misclassification success on the one hand. On the other hand, the perturbation was visible on the image, not to be confused with a low-quality image. Therefore, one might suspect that the image was manipulated. In terms of the correct image classification, we do not expect that even though the perturbation is visible in the image, this may affect its correct classification by humans. One of our main goals of perturbation is to make a DL architecture misclassify image while not affect human capabilities to identify it correctly.

We also learn that the perturbation learned on AlexNet is more deceiving than the perturbation learned on GoogLeNet and ResNet. Unlike the best perturbation model, which changed with additional training images, we see that the robustness of the ResNet is still much higher than the robustness of other classifiers under examination. By inserting the blurred background of a puzzle, we saw for the first time a misclassification of EFT image and much furthest classification from the true label of the border collie. When comparing investigated perturbations to perturbation generated from CleverHans python adversarial package, we see that the perturbation is less visible to the human eye than the averaging based methods. Despite this advantage, our perturbation models throw the classification further from the real class. Additionally, we train and evaluate Saliency-Map-Based perturbation including integration between Deep-Fool averaging and Saliency Map. The results presented in Appendix A.

6.2. Conditional Cumulative Universal Adversarial Perturbation

We present in Table 4, the fooling matrix where the columns are the classifiers on which the perturbation was learned, and the rows are the evaluating classifiers.

		Surrogate Model			
Target Model		VGG	AlexNet	ResNet	GoogLeNet
	VGG	9.05%	1.21%	6.73%	8.03%
	AlexNet	12.76%	0.38%	8.92%	5.35%
	ResNet	21.31%	13.26%	26.22%	19.29%
	GoogLeNet	18.82%	3.77%	19.89%	16.88%

Table 4 – Evaluation of the models

We present in Figure 5 accuracy of the classification models when attacked by Adversarial noise, the blue bar represents the accuracy of the model when attacked with perturbation learned on his own model while the green bar represents the model's accuracy when attacked with perturbation learned on AlexNet.

If we compare this method to the three models based on simple average mentioned on the previous section, we have a drastic improvement in terms of calculation complexity. Instead of approximately 14(AlexNet) – 33(ResNet), days on each of our four classifiers, this method requires 1.5(AlexNet) – 4(ResNet) days, approximately 12% of the time of our first proposed method. If we compare the methods in a manner of fooling success, the improvement is also significant both in fooling its' own classifier and the transferability to other classifiers. When looking on the 'damage' to the original image, the perturbation is once again noticeable to the human eye but slightly less than the perturbation we proposed in the first method. We can claim with high confidence that the iterative perturbation update model will not change the human perception of the image, thus will have no effect on its classification correctness. We show that the pattern of transferability remains as in baseline averaging, AlexNet is once again the most powerful model to learn on. Comparing to Figure 4, we see significant decrease in the accuracy.

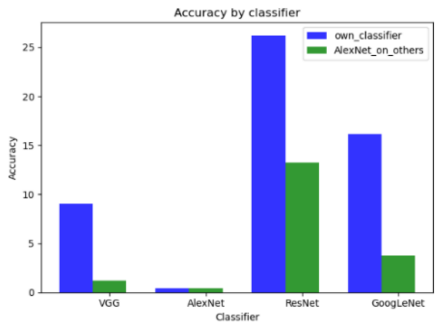


Figure 5 – Model's accuracy (CCUAP Method)

6.3. Illustrating the Consequences of Different Perturbation Strategies

We present visualization of perturbation effect in Table 5. For this illustration we choose seven images from the original dataset. The first row contains the original images with AlexNet classification confidence on these images. The following rows present the perturbated images using different methods and AlexNet classification confidence. For this visualization, the perturbation was based on pre-trained AlexNet architecture.

The second row presents the effect of the perturbation on the images and the change in the predicted class. We present in the third row, the result of the reduced universal perturbation after multiplying the averaged perturbation with a random tensor. The next row in presents the perturbation effect of this method, the fooling success is slightly higher than the baseline average. For comparison, we present in fifth row of Table 5, perturbation made by Python library dedicated to benchmark machine learning systems vulnerabilities. This library focuses on providing a reference implementation of attacks against machine learning models to help with benchmarking models against adversarial examples. The last row presents the perturbation effect of CCUAP method.

All the added perturbations are visible to the human eye, but it should not affect humans from classifying the image correctly. In a few cases, the perturbation did not make the classifier misclassify the image. Nonetheless, it does reduce the confidence of the class probability. A more significant achievement in that experiment is that the compression algorithm did not affect the perturbation and the new image is highly misclassified by DL architectures.

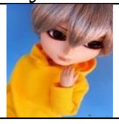
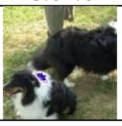
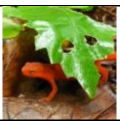
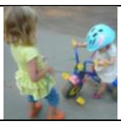
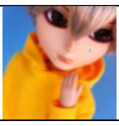
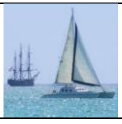
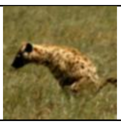
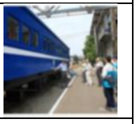
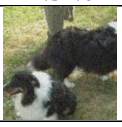
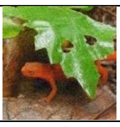
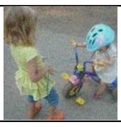
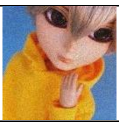
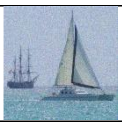
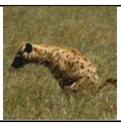
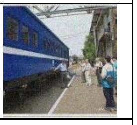
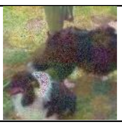
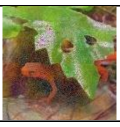
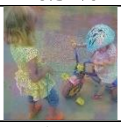
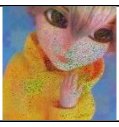
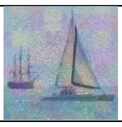
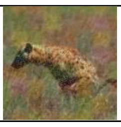
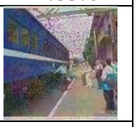
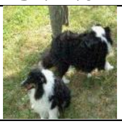
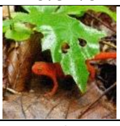
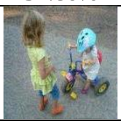
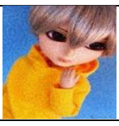
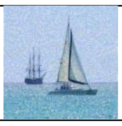
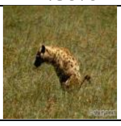
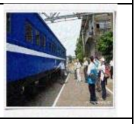
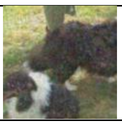
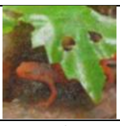
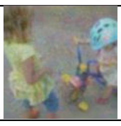
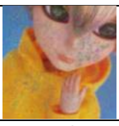
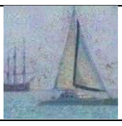
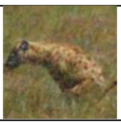
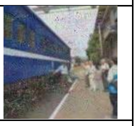
Original	 Border collie 45.61%	 Eft 99.07%	 Tricycle 73.26%	 Lolly 46.28%	 Catamaran 57.56%	 Hyena 99.02%	 Tram 52.85%
ADFP	 English sheepdog 24.37%	 Eft 59.40%	 Maraca 8.40%	 Power drill 20.31%	 Yawl 46.85%	 Hyena 87.55%	 Street sign 21.06%
ADFT_RT	 Collie 25.78%	 Eft 34.05%	 Jigsaw puzzle 16.52%	 Wig 17.25%	 Yawl 70.22%	 Hyena 99.52%	 Mooving van 11.88%
ADFT_RT_BI	 Irish water spaniel 32.41%	 Jigsaw puzzle 10.61%	 African chamelion 52.86%	 Mask 5.03%	 Yawl 61.51%	 Common Newt 27.86%	 Mosquito net 12.61%
CleverHans	 Tibetian terrier 27.87%	 Eft 16.33%	 Jigsaw puzzle 37.51%	 Ski mask 22.37%	 Catamaran 41.21%	 Hyena 93.24%	 Ricksha 13.91%
CCUAP	 Wire-haired fox terrier 10.23%	 Eft 58.96%	 Jigsaw puzzle 8.19%	 Goldfish 8.83%	 Mosquito net 25.22%	 Hyena 30.37%	 Poncho 4.73%

Table 5 – Visualization and classification confidence

6.4. Comparison of the proposed models and a Naïve model

We randomly chose Gaussian noise perturbation as our naïve model for comparison to our proposed methods, Gaussian noise is a widely used technique for image perturbation and augmentation. It adds random values from a Gaussian distribution to the pixel values of an image, simulating real-world variations and improving the robustness and generalization of machine learning models trained on these images. We establish a universal mask to match our investigated perturbations once according to SSIM and a second time according to SNR.

We show in Figure 6, the effect of different models including naïve model on different classifiers; we sampled different methods with the same SSIM and showed that the CCUAP method has the slowest convergence, hence it is more likely to find an optimal solution for the mentioned threshold. We note that as faster the accuracy increases the weaker is the perturbation. In this evaluation method, we are just slightly better than the naïve model but significantly better than other explored methods. We show that for AlexNet classifier and SSIM of 0.4, the CCUAP method reduces the accuracy of AlexNet to 9.43% while the naïve model [18.29%], ADFP [60.65%], ADFP_RT [83.53%] and ADFP_RT_BI [16.38%] have a lower impact on the classification accuracy.

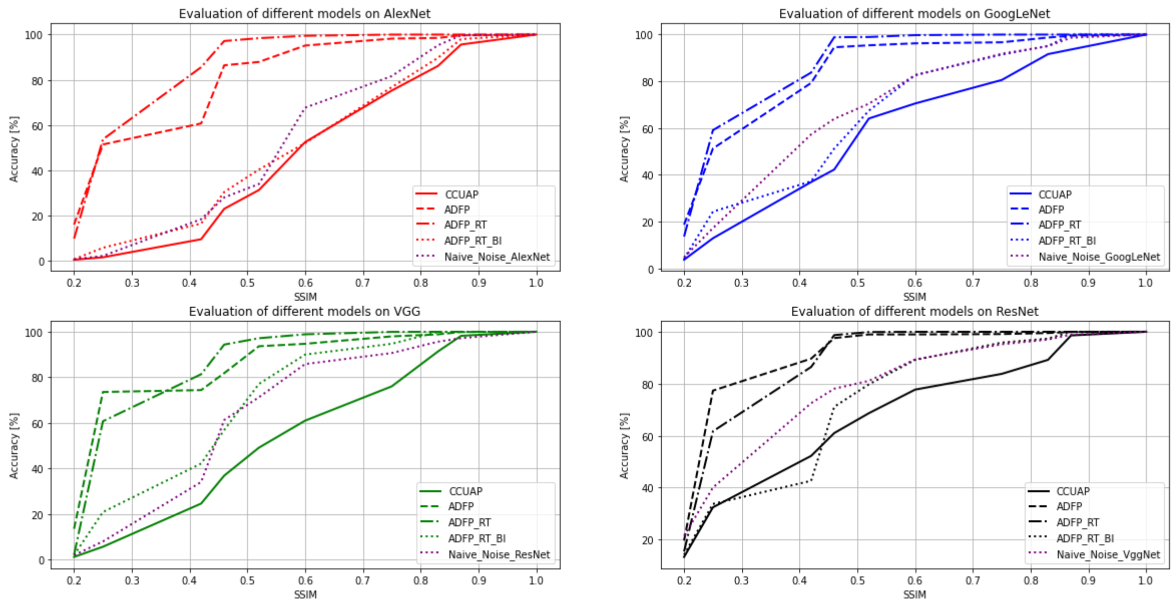


Figure 6 – Evaluation of different methods on classification models

Deeper explanation and pattern visualization of different methods examined their impact on classification accuracy versus SSIM can be found in Appendix B.

Except the structural similarity index, we present in Figure 7, classification success of each classifier when attacked by different models, where the attack intensity is represented and controlled by signal to noise ratio (SNR). We see that on all the attacked classifiers CCUAP method starts decreasing classifiers' accuracy with smaller amount of noise, we also note, that on all the attacked classifiers' the method converges faster to minimal accuracy possible. Furthermore, we see a linear decrease in CCUAP against hyperbolic decrease in other method, what prove CCUAP method to be better in this region when SNR is between 52dB and 62dB. The naïve model is the weakest one among the researched method, the accuracy decreases with lower SNR relative to other models, that means the noise intensity should be increased drastically to reach the same accuracy, as we present later in Table 8, we assume it is due to pre-processing of the classifier.

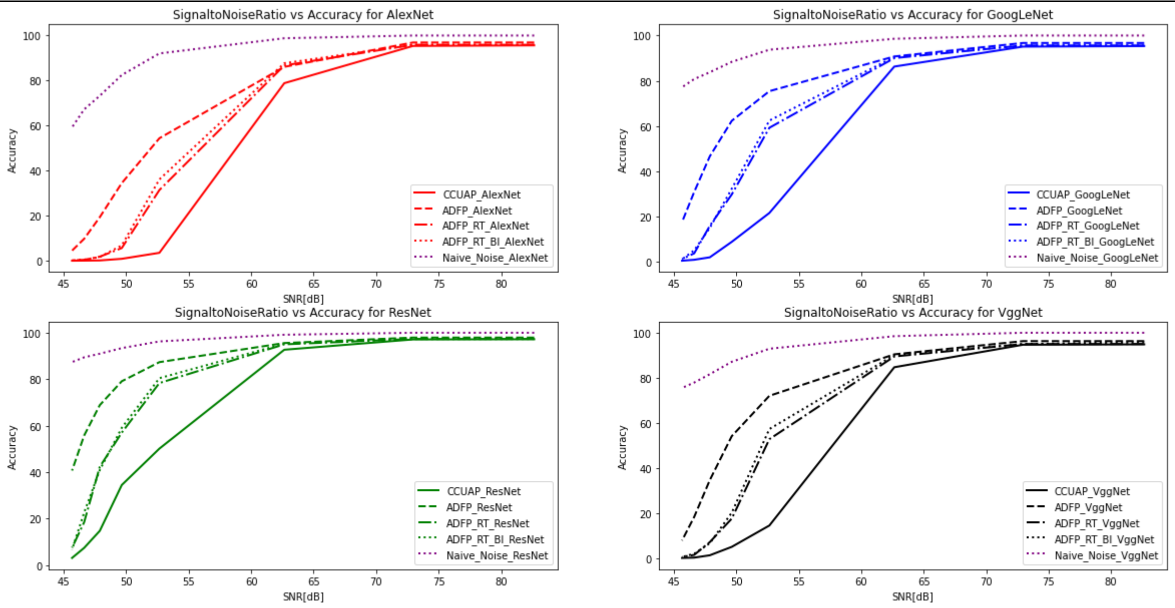


Figure 7 – Accuracy (SNR [dB]) - all methods on each classifier

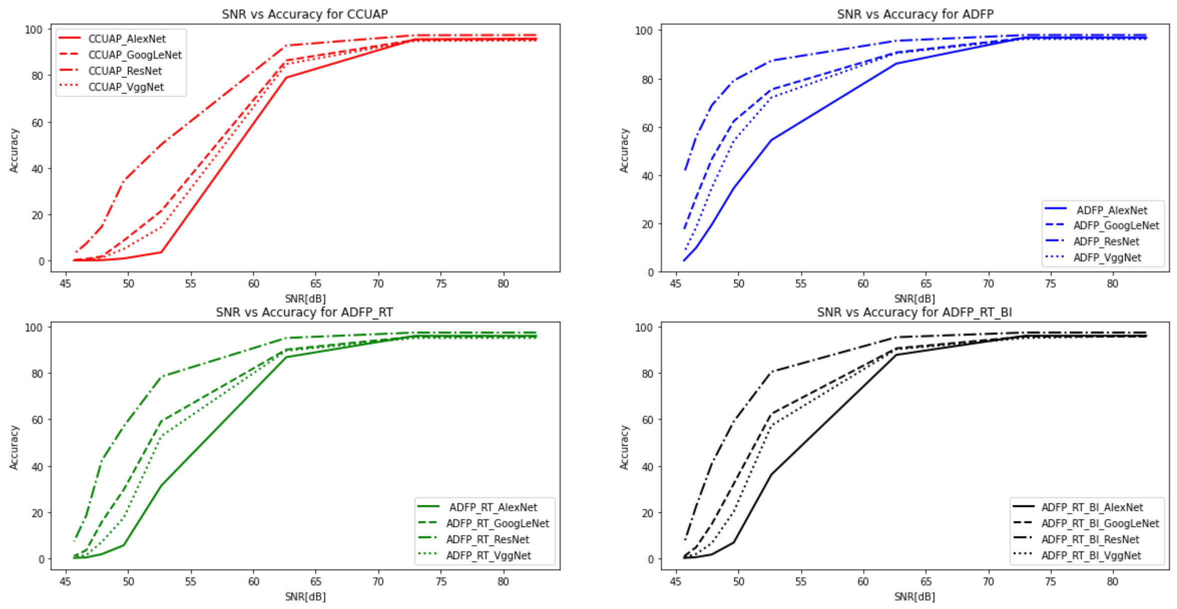


Figure 8 – Accuracy (SNR [dB]) different methods

We show in Table 6 the perturbated images arranged by SNR for different methods. We focus on comparing all investigated methods to naïve noise, if we look in the images in Table 6, we can confidently say that with SNR = 52.65 there is slightly visible effect on the image, hence, human should classify the image correctly. In Table 7, we present the accuracy achieved with chosen SNR, the accuracy of all four classifiers is still very high, hence we decreased SNR until reaching approximately same accuracy values, the SNR we achieved is 27.61[dB]. In Table 8 we present the accuracy over the investigated classifiers and visualization of the perturbated image. We needed much stronger perturbation to achieve same accuracy from the classifiers and we strongly believe, the perturbated image from naïve model, might also fool human in classifying it correctly.
Figure 8 shows the robustness of our classifiers to each of the investigated methods, once again, as a function of SNR. We see that the decrease in ResNet, for all the methods starts at a lower SNR.

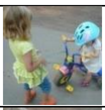
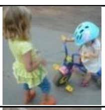
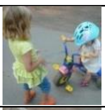
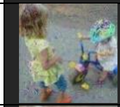
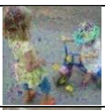
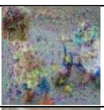
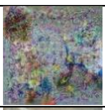
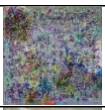
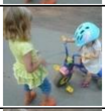
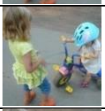
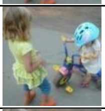
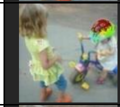
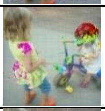
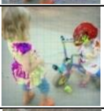
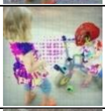
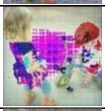
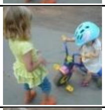
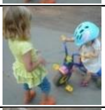
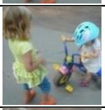
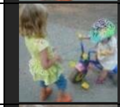
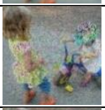
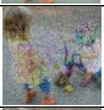
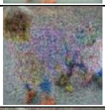
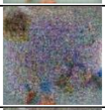
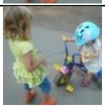
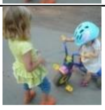
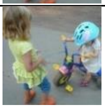
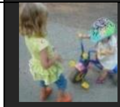
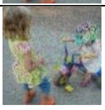
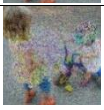
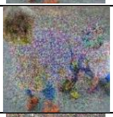
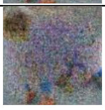
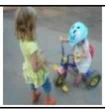
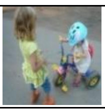
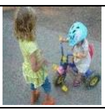
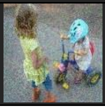
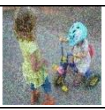
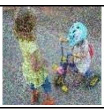
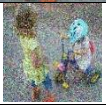
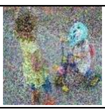
	SNR=82.65	SNR=72.65	SNR=62.65	SNR=52.65	SNR=49.64	SNR=47.88	SNR=46.63	SNR=45.66
CCUAP								
ADFP								
ADFT_RT								
ADFT_RT_BI								
Naïve Noise								

Table 6 – Different methods with same SNR

	Target Model			
	AlexNet	GoogLeNet	ResNet	VGG
CCUAP	3.63%	21.48%	50.09%	14.53%
Naïve Noise	92.06%	93.73%	96.27%	92.92%

Table 7 – SNR based comparison

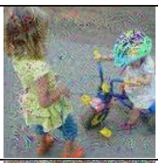
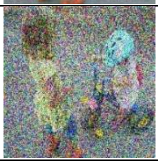
	AlexNet	GoogLeNet	ResNet	VGG	Visualization
CCUAP	3.63%	21.48%	50.09%	14.53%	
Naïve Noise	4.56%	24.12%	51.74%	14.49%	

Table 8 – Accuracy based comparison

Comparing intensity of noise does not provide us good enough comparison to claim our proposed method is better than naïve noise, we compare the two methods by increasing naïve noise until we reach approximately same values of accuracy and present the perturbated images of the two models in Table 8. Although both images should be still recognizable to common human, we note that the image, perturbated with a naïve noise is much more damaged than the image perturbated by our method.

7. Discussion

We extended Deep-Fool perturbation that is image specific and depends on the attacked classifier to a perturbation that is image and network agnostic. Furthermore, the CCUAP method overcomes an important limitation of the Deep-Fool algorithm, which is its weakness to image compression algorithm preprocessing tools. CCUAP method offers several more significant advantages over the previous Deep-Fool averaging methods. Firstly, it drastically reduces the calculation complexity, enabling perturbation generation in a fraction of the time required by the previous method. This improvement in efficiency enhances the practicality and scalability of generating adversarial examples. Moreover, the CCUAP method demonstrates a substantial increase in fooling success and transferability to other classifiers. These results underscore the effectiveness of our approach in deceiving deep learning models beyond the classifier on which the perturbation was initially learned.

Importantly, we found the threshold that insured that the perturbation is maximized for better misclassification rate, preserving the correct perception and classification of the image.

Our research demonstrated that our proposed perturbations affected the accuracy of the classifiers at higher SNR values compared to the naïve model. This indicates that CCUAP method introduces a more impactful perturbation at lower noise intensity.

We show that by implementing the CCUAP method on a large-scale dataset we manage to create a universal perturbation which can deceive classifiers with high fooling rate without knowing what is the original image and what classifier is defending the system.

The results of our research demonstrate the effectiveness of the CCUAP method in deceiving deep learning classifiers. Despite the visibility of the perturbation to the human eye, our experiments showed that it does not significantly impact humans' ability to correctly classify the images. this is important finding as it suggests that the perturbation can reduce the confidence of the class probabilities without compromising human perception.

This research on adversarial examples has made a significant contribution to our understanding of perturbation techniques and their impact on deep learning models. However, it is important to acknowledge the limitations of our research, including the specific dataset, models, and classifiers used, both for learning and evaluating the perturbation.

Further research should focus on expanding the scope of investigation to encompass a wider range of datasets and deep learning architectures. Additionally, exploring novel defense mechanisms and countermeasures against adversarial examples is crucial to enhance the robustness and security of deep learning models in practical applications.

In conclusion, our research has provided insights into the effectiveness of Deep-Fool – based perturbations, the impact of perturbation learning on different architectures, the advantages of CCUAP method, including visualization of perturbation effect and classification accuracy, a comparison with a naïve model and saliency map – based perturbation.

These findings can contribute to the ongoing efforts to understand and mitigate the vulnerabilities of deep learning models to adversarial examples and fostering the development of more secure and reliable deep learning systems.

8. References

1. Von Ahn, L., Blum, M., Hopper, N. J., & Langford, J. (2003, May). CAPTCHA: Using hard AI problems for security. In Eurocrypt (Vol. 2656, pp. 294-311).
2. Adversarial-ml-tutorial
3. Dataset-Kaggle
4. Saliency-maps-in-deep-learning
5. Pytorch. models
6. Osadchy, M., Hernandez-Castro, J., Gibson, S., Dunkelman, O., & Pérez-Cabo, D. (2017). No bot expects the DeepCAPTCHA! Introducing immutable adversarial examples, with applications to CAPTCHA generation. *IEEE Transactions on Information Forensics and Security*, *12*(11), 2640-2653.
7. Dabkowski, P., & Gal, Y. (2017). Real time image saliency for black box classifiers. *Advances in neural information processing systems*, *30*.
8. Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., & Swami, A. (2016, March). The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)* (pp. 372-387). IEEE.
9. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, *115*(3), 211-252.
10. Zolna, K., Geras, K. J., & Cho, K. (2020). Classifier-agnostic saliency map extraction. *Computer Vision and Image Understanding*, *196*, 102969.
11. Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.
12. Kose, U. (2019). Techniques for adversarial examples threaten the safety of artificial intelligence-based systems. *arXiv preprint arXiv:1910.06907*.
13. Moosavi-Dezfooli, S. M., Fawzi, A., & Frossard, P. (2016). 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).
14. Moosavi-Dezfooli, S. M., Fawzi, A., Fawzi, O., & Frossard, P. (2017). Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1765-1773).
15. Yuan, X., He, P., Zhu, Q., & Li, X. (2019). Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, *30*(9), 2805-2824.
16. Zhang, J., & Li, C. (2019). Adversarial examples: Opportunities and challenges. *IEEE transactions on neural networks and learning systems*, *31*(7), 2578-2593.
17. Papernot, N., Faghri, F., Carlini, N., Goodfellow, I., Feinman, R., Kurakin, A., ... & McDaniel, P. (2016). Technical report on the cleverhans v2. 1.0 adversarial examples library. *arXiv preprint arXiv:1610.00768*.
18. Carlini, N., & Wagner, D. (2017, May). Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)* (pp. 39-57). IEEE.
19. K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034, 2013.
20. Ruszczynski, A. (2011). *Nonlinear optimization*. Princeton university press.

21. N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, “Distillation as a defense to adversarial perturbations against deep neural networks,” in Proc. IEEE Symp. Secur. Privacy, May 2016, pp. 582–597.
22. Mopuri, K. R., Garg, U., & Babu, R. V. (2017). Fast feature fool: A data independent approach to universal adversarial perturbations. arXiv preprint arXiv:1707.05572.
23. Phung, A., & Stamp, M. (2021). Universal adversarial perturbations and image spam classifiers. Malware Analysis Using Artificial Intelligence and Deep Learning, 633-651.
24. Hayes, J., & Danezis, G. (2018, May). Learning universal adversarial perturbations with generative models. In 2018 IEEE Security and Privacy Workshops (SPW) (pp. 43-49). IEEE.
25. Mummadi, C. K., Brox, T., & Metzen, J. H. (2019). Defending against universal perturbations with shared adversarial training. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 4928-4937).
26. Hendrik Metzen, J., Chaithanya Kumar, M., Brox, T., & Fischer, V. (2017). Universal adversarial perturbations against semantic image segmentation. In Proceedings of the IEEE international conference on computer vision (pp. 2755-2764).
27. Chaubey, A., Agrawal, N., Barnwal, K., Guliani, K. K., & Mehta, P. (2005). Universal adversarial perturbations: A survey. arXiv 2020. arXiv preprint arXiv:2005.08087.
28. https://github.com/MichaelGudovsky/AdversarialExamples.git

9. Appendix A – Saliency Map

Formally, saliency maps are defined as the means to measure the spatial support of a particular class of an image in computer vision models. To keep things simple, we can redefine the saliency map as an image highlighting the region of interest in our computer vision model. An example of a saliency map, highlighting the important pixels to accurately classify the pictures on the left, is shown below. In Figure 9 we demonstrate a saliency map on the right extracted from the original image on the left.

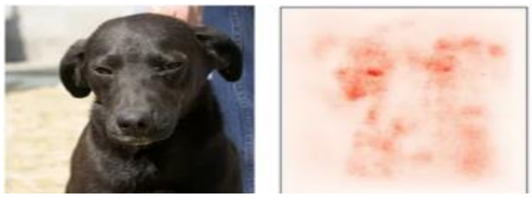


Figure 9 – Side by side view of image and its saliency map

We extend the use of saliency map with the strongest perturbation obtained (AlexNet). In Figure 10, we present the proposed combined perturbation model, we extract the saliency map of an image, then multiply the saliency map with our universal perturbation, finally we add the obtained perturbation to the original image. We assume that infecting the pixels of interest in the specific image with our strong universal perturbation will also cause a misclassification, without infecting all the pixels of the image.

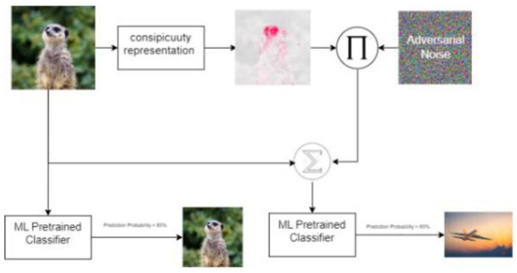

consipicuuty representation
Adversarial Noise
ML Pretrained Classifier
Prediction Probability = 85%
ML Pretrained Classifier
Prediction Probability = 85%

Figure 10 – Combined perturbation model

9.1. Saliency-Map-Based Noise

We are not focusing on showing and comparing the saliency map plots since we show in the results section that it has less success on fooling the classifiers and since its computational cost is approximated in about 27 days for each sample. The noise is the summation of saliency maps from pre-trained classifiers we used in previous sections. The visual change on the image is significantly smaller than the universal perturbation from the Deep-Fool algorithm; nonetheless we see in Table 9 a relatively good impact on the classification accuracy. In second row of Table 11, we present the minimalistic impact on the image and its' misclassification success.

		Surrogate Model				
		VGG	AlexNet	ResNet	GoogLeNet	Combined
Target Model	VGG	47.66%	45.07%	53.21%	44.98%	40.33%
	AlexNet	37.84%	34.34%	42.27%	34.91%	29.82%
	ResNet	67.99%	65.85%	70.59%	65.05%	61.84%
	GoogLeNet	51.77%	50.24%	56.71%	47.83%	45.28%

Table 9 – Evaluation of the saliency noise effect on classifiers

We show that the combined saliency map is slightly more effective than the individual maps, especially when looking at other networks since the combined saliency considers the importance of an individual pixel in all the networks we examine and not only one specific network. This method is weaker than our universal perturbation in a manner of fooling success furthermore saliency maps are individual for each image, therefore if we want to create an adversarial example using this method, we need to obtain the original image.

9.2. Combining saliency noise with ADFP_RT_BI

We use the saliency maps from the previous section and add the strongest perturbation we explored (AlexNet). In this experiment, we are losing the independence of the image since we must first preprocess the image to extract its saliency map.

		Surrogate Model				
		VGG	AlexNet	ResNet	GoogLeNet	Combined
Target Model	VGG	33.07%	29.69%	35.81%	30.59%	24.22%
	AlexNet	21.90%	18.88%	23.55%	19.94%	14.80%
	ResNet	57.57%	55.15%	59.51%	54.71%	50.25%
	GoogLeNet	38.68%	35.53%	40.82%	35.31%	29.72%

Table 10 – Accuracy evaluation of the saliency noise combined with ADFP_RT_BI

In Table 11, we present the fooling matrix of the combined perturbation. As expected, we achieved an increase from the fooling success of saliency–based noise. In third row of Table 11, we present the effect of the combined noise on the images and their classification by a pre-trained VGG model.

An individual classifier saliency map combined with the Deep-Fool universal perturbation is slightly weaker than the combined saliency extracted from our classifiers combined with the universal perturbation. When we extract saliency from one classifier, it computes the point of interest in the image for this specific classifier. Therefore, the transferability to other classifiers is lower. Another advantage of the combined saliency is when pixels are important for more than one classifier, they are changed according to their importance (i.e., in how many classifiers the specific pixel was Important for the classification process).

The first row represents the original image while all the other rows correspond to the perturbation method mentioned in the left column.

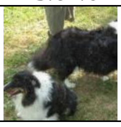
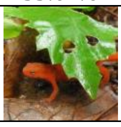
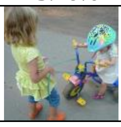
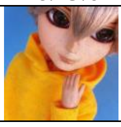
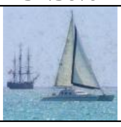
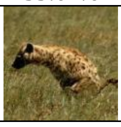
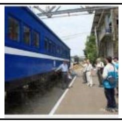
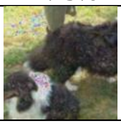
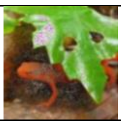
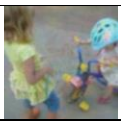
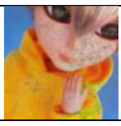
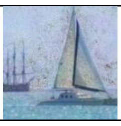
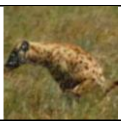
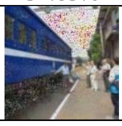
Original							
	Border collie 45.61%	Eft 99.07%	Tricycle 73.26%	Lolly 46.28%	Catamaran 57.56%	Hyena 99.02%	Tram 52.85%
Saliency Map							
	Old English sheepdog 42.49%	Eft 36.34%	Maraca 16.84%	Sunglasses 20.32%	Schooner 50.36%	Hyena 98.97%	Passenger car 32.85%
Saliency Map + ADFP_RT_BI							
	Miniature pudel 28.14%	Eft 77.19%	Hair slide 16.05%	Tailed frog 12.14%	Yawl 79.03%	Hyena 67.01%	Street sign 23.29%

Table 11 – Visualization and classification confidence (Saliency)

10. Appendix B – Structural Similarity Index (SSIM)

Figure 11 presents patterns of the classification success of our chosen classifiers as a function of the perturbated image similarity to the original image.

We note that as faster the accuracy increases the weaker is the perturbation. In this evaluation method, we are just slightly better than the naïve model but significantly better than other explored methods.

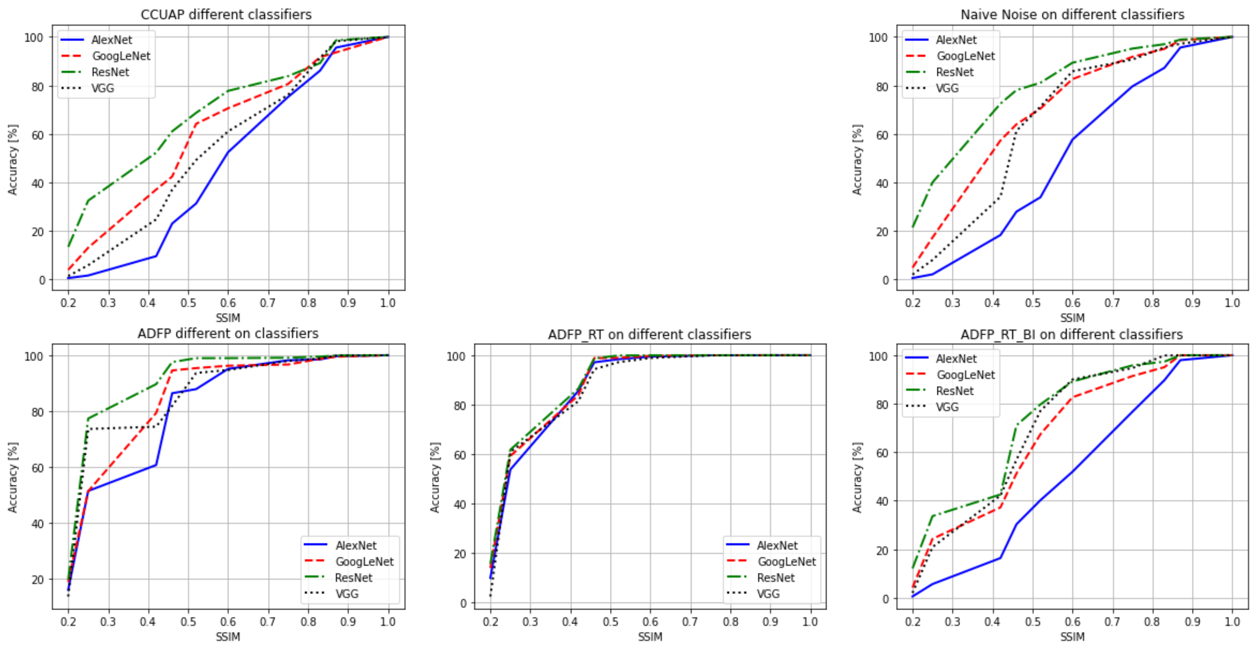


Figure 11 – Comparison between different methods