UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Master's Degree
in
Data Science

Capstone Report - Summer 2022

ZO methods for non-convex optimization adversarial attack

Abhishek Varma Dasaraju,
Brenda Eloísa Téllez Juárez,
Rebecca Di Francesco

Professor
Francesco Rinaldi

Abstract

Our paper focuses on the implementation of the following Zeroth Order optimization methods: ZO-SGD, ZO-SignSGD and ZO-AdaMM. We implemented these algorithms to generate untargeted per-image adversarial attacks directed to a CNN image classifier that was trained on the CIFAR-10 dataset. Then, we analyzed if the theoretical convergence rate from the papers matched with the results in our experiment.

Keywords

Data Science; Optimization; ZO methods; Non-Convex Optimization; Adversarial Attacks

Contents

1 Introduction

Our project focuses on zeroth-order optimization methods which can solve both convex and non-convex problems mainly in the context of black-box optimization. In particular, the area of interest for these algorithms is the design of adversarial attacks in which usually ML systems cannot be publicly accessed and thus calling the first order oracle for these problems is unfeasible. We will experiment three types of gradient-free methods: ZO-SGD, ZO-SignSGD and ZO-AdaMM. These will be performed on per-image adversarial perturbation, a specific type of black-box adversarial attack, using the CIFAR10 dataset. With respect to attack specificity we performed untargeted attacks thus our aim is to fool the model by expecting a misclassification that is a consequence of any class different from the legitimate class corresponding to the original example, not for a predetermined class chosen beforehand that would be the case for targeted attacks. We start by describing the problem that we want to solve and the algorithms, then in the Experiments section we will analyze the results produced by these algorithms for the unconstrained problem on the CIFAR-10 dataset.

2 Description of the problem

In this analysis we will work with black-box optimization which is a setting that given certain inputs allows only to observe the outputs of the objective function while the model configuration is inaccessible. Thus, first-order optimization methods that rely on explicit gradient information are not useful when dealing with the black-box setting. As a result, zeroth-order (ZO) methods that are gradient-free methods are preferred for the nature of the problem mentioned before.
ZO Methods are a type of optimization methods that approximate the gradients by calculating a forward difference of function values between two random query points. We will be using these methods for generating adversarial attacks which are slightly modified images generated from an input with the purpose of misleading an image classifier. The malicious potential of an adversarial attack is evident, on the other hand it can be used beneficially to improve the robustness of a NN model.
"*Adversarial examples are inputs to machine learning models that an attacker has intentionally designed to cause the model to make a mistake.*"
-Ian Goodfellow
Generating an adversarial example means to find the minimal perturbation in order to deceive a

learning model so to maximise the loss related to the crafted input and the original true label. There are two main types of adversarial attacks: per-image adversarial attacks and universal adversarial attacks. We will focus on the first task thus we will generate perturbations individually for each input image.

3 Description of the algorithms

3.1 ZO SGD

The SGD method calculates the gradient information on only a random batch of points at each iteration instead of on the complete dataset, thus this is why it is considered a stochastic method. So, SGD has an higher complexity in convergence rate than the normal online gradient method but it is faster to compute at each iteration. ZO-SGD is the same method but relies on the approximation of gradients calculated trough a difference of two function values instead of using the true gradient information. The GradEstimate that appears in the algorithm above will be the same for all the three algorithms that we implemented and is defined as:

$$GradEstimate(x) = \frac{1}{bq}\sum_{i\epsilon I_k}\sum_{j=1}^{q}\hat{\nabla} f_i(x; u_{i,j}), \hat{\nabla} f_i(x; u_{i,j}) := \frac{d[f_i(x + \mu u_{i,j}) - f_i(x)]}{\mu} u_{i,j}, \qquad (1)$$

where $x = x_k$ in (1), I_k is a mini-batch of size $I_{k,j} = b$, $f(u_{i,j})$ are i.i.d. random directions drawn from a uniform distribution over a unit sphere, and $f_i(x; u_{i,j})$ gives a two-point based random gradient estimate with direction $u_{i,j}$ and smoothing parameter $\mu > 0$.

Algorithm 1 Generic sign-based gradient descent

1: Input: learning rate $\{\delta_k\}$, initial value $\mathbf{x}_0$, and number of iterations T
2: **for** $k = 0, 1, \ldots, T-1$ **do**
3: $\quad \hat{\mathbf{g}}_k \longleftarrow$ GradEstimate$(\mathbf{x}_k)$ # applies to both first and zeroth order gradient estimates
4: $\quad$ sign-gradient update

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \delta_k \text{sign}(\hat{\mathbf{g}}_k), \quad \text{where sign}(\mathbf{x}) \text{ takes element-wise signs of } \mathbf{x} \qquad (2)$$

5: **end for**

Figure 1: General Sign-based gradient Algorithm

3.2 ZO SIGN-SGD

By adding the element-wise sign operation to the gradient estimation in equation (1), this new algorithm defines the ZO-sign-SGD. Taking this trasformation to the estimated gradient will handle the noise of some of its components leading to a faster convergence rate than the ZO-SGD.

3.3 ZO AdaMM

ZO AdaMM is a convenient algorithm that takes into account two additional variables that helps to perform well in unconstrained optimization. The first of these two variables is momentum β_1 this allows us to perform faster steps to reach the objective, since it is the term for velocity, and it takes into account the exponential moving averages of the past gradients. Additionally, the other one β_2 (AMSGrad) helps to avoid being stuck in a saddle point. The $\hat{g}_t$ that appears in the algorithm defined is the same as in equation (1). The algorithm used for implementing ZO AdaMM is the following:

Algorithm 1 ZO-AdaMM

Input: $\mathbf{x}_1 \in \mathcal{X}$, step sizes $\{\alpha_t\}_{t=1}^T$, $\beta_{1,t}, \beta_2 \in (0,1]$, and set $\mathbf{m}_0$, $\mathbf{v}_0$ and $\hat{\mathbf{v}}_0$

for $t = 1, 2, \ldots, T$ **do**

let $\hat{\mathbf{g}}_t = \hat{\nabla} f_t(\mathbf{x}_t)$ by (1), $f_t(\mathbf{x}_t) := f(\mathbf{x}_t; \boldsymbol{\xi}_t)$

$\mathbf{m}_t = \beta_{1,t}\mathbf{m}_{t-1} + (1-\beta_{1,t})\hat{\mathbf{g}}_t$

$\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\hat{\mathbf{g}}_t^2$

$\hat{\mathbf{v}}_t = \max(\hat{\mathbf{v}}_{t-1}, \mathbf{v}_t)$, and $\hat{\mathbf{V}}_t = \mathrm{diag}(\hat{\mathbf{v}}_t)$

$\mathbf{x}_{t+1} = \Pi_{\mathcal{X}, \sqrt{\hat{\mathbf{V}}_t}}(\mathbf{x}_t - \alpha_t \hat{\mathbf{V}}_t^{-1/2}\mathbf{m}_t)$

end for

Figure 2: Algorithm ZO AdaMM

The main purpose of the algorithms described above is to find the minimal perturbation w appearing inside the following objective function for the unconstrained case:

$$\min_{w \epsilon \mathbb{R}_d} \frac{\lambda}{M} \sum_{n=1}^{M} [f(0.5tanh(tanh^{-1}(2x_i) + w)) + \|0.5tanh(tanh^{-1}(2x_i) + w) - x_i|] \tag{2}$$

We can see there are two terms inside the sum. The first term is defined as the attack loss, the second term as the l2 distortion. In the attack loss the loss function f for untargeted attacks is

defined as follows:

$$f(x') = maxZ(x'_t) - max_{j \neq t} Z(x'_j), -k \tag{3}$$

where by quoting the paper of Chen et al., $Z(x'_k)$ denotes the prediction score of class k given the input x', and the parameter $k > 0$ governs the gap between the confidence of the predicted label and true label t., in our experiment thus we set it to 0 since when the attack loss reaches zero it means the perturbation succeeds to fool the neural network.

3.4 Dataset

We performed adversarial attacks on CIFAR-10 dataset which consists of 60000 images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. Each sample is a 32x32 pixels color image, associated with a label from 10 classes:
class_names = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"].
Each pixel intensity is represented by a uint8 (byte) from 0 to 255. We further divided the training set into training set and validation set (which consisted of 5000 samples extracted from training set). So, the training set was used to train the CNN model, the validation set was used for model selection and the test set to check the model predictions and also to generate the adversarial attacks on those images that were correctly classified.

3.5 Experiment Setup

We started by designing a convolutional neural network (CNN) to get an image classifier that is our target model to deceive. The CNN that we designed consisted of:

- Two 2D convolutional layers with 32 output filters of size 3x3 that use ReLu activation function.
- A Dropout layer of 0.25.
- One Max Pooling layer (2D) of size 2x2 with padding and strides of 2.
- Two 2D convolutional layers with 128 output filters of size 3x3 that use ReLu activation function.
- A Dropout layer of 0.25.

- One Max Pooling layer (2D) of size 2x2 with padding and strides of 2.
- a Flatten layer
- a Dense layer with 256 neurons.
- a Dropout layer with 0.5
- a final Dense layer with 10 output neurons (one per class), and with the "softmax" activation function.

Then we trained the model with Adam Optimizer and sparse categorical cross-entropy as loss function. Given, 10 epochs and 128 batch size the model achieved an accuracy of around 77% in the validation set.

After that, we followed the Chen et al. paper to tune the following parameters:

- $q = 10$, that stands for the number of random directions in equation (1).
- $d = len(train\_size)$ which is the dimension size of the flattened images.
- $T = 1000$ which is the number of iterations.
- $mu = 0.01$, the smoothing parameter in equation (1) .
- $delta\_sgd = 0.001$, the learning rate for ZO-SGD.
- $delta\_sign\_sgd = 0.001$, the learning rate for ZO-SignSGD.
- $delta\_adam = 0.001$, the learning rate for ZO-AdaMM.
- $const = 0.1$, the lambda constant in equation (1).
- $tl = X_test[ID]$, the true label of the original image.
- $sign = True$, a parameter that controls if we perform a ZO-SGD or a ZO-SignSGD.

For ZO-AdaMM we defined $\beta_1 = 0.9$ and $\beta_2 = 0.3$ since in the Chen et al. paper they found that a choice of $\beta_1 \geq 0.9$ and $\beta_2 \in [0.3, 0.5]$ performs well in practice.

4 Results and Discussion

With the setup described above, we run the per-image adversarial attacks on 6 images from the test set for which the CNN was returning right predictions. We selected three of them for which the CNN returned an high prediction score (higher or equal than 0.95) and three of them for which the CNN returned a lower prediction (higher or equal then 0.3 and less than 0.5). We expected that for the latter case, the algorithms will be faster in convergence since the CNN was already giving less certain predictions.

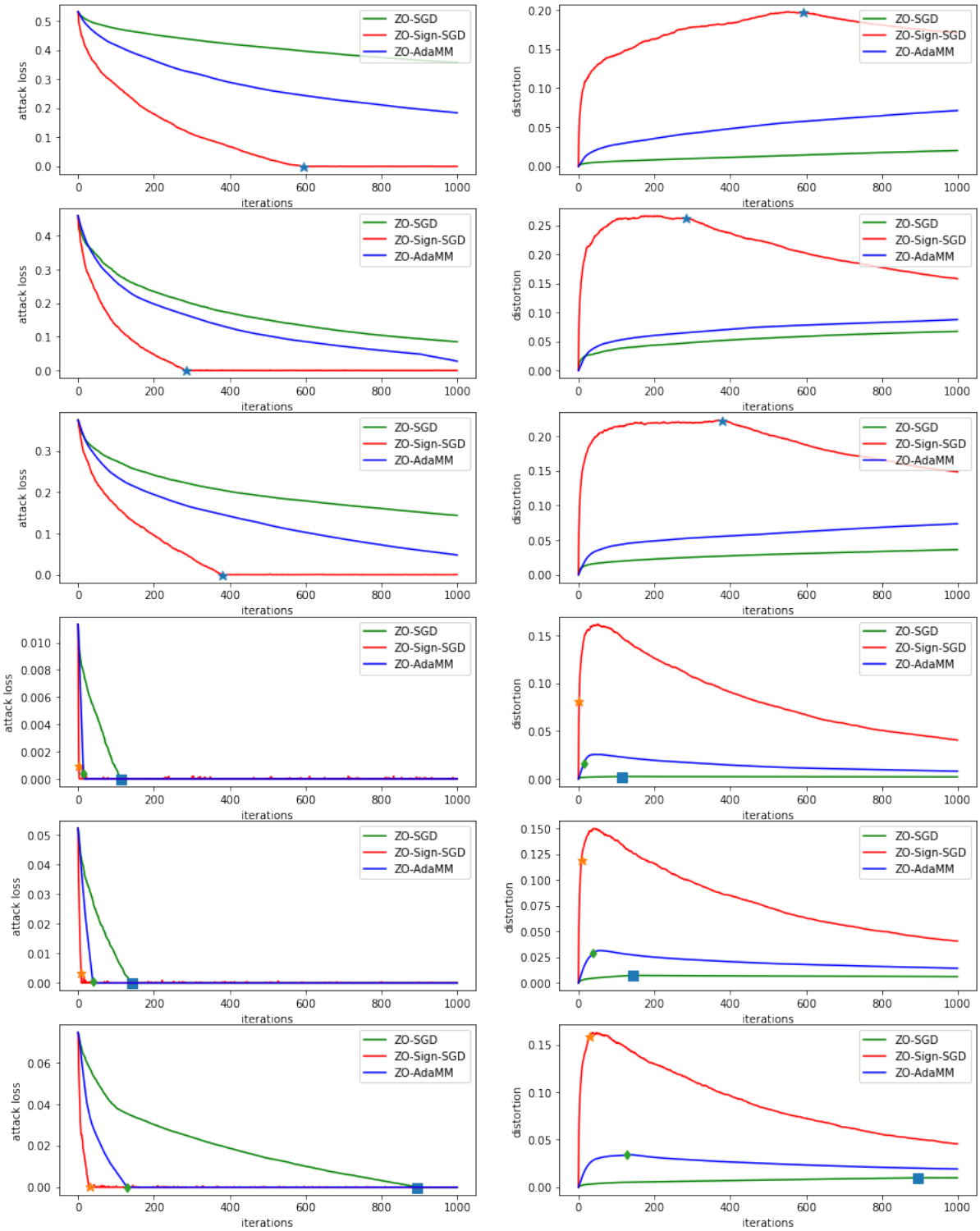

Figure 3: Adversarial Attaks and Distortions

The results in terms of attack loss and distortion for each of these images are collected in Figure 3 . The successful attacks are marked with special different symbols for each of the algorithm and in the curves in which the marker is not visible it means that for that algorithm the attack

was unsuccessful. We can see that in general for all images the attack loss decreases faster for ZO-SignSGD, then ZO-AdaMM and lastly ZO-SGD while distortions curves follow the opposite logic so they increase the most for ZO-SignSGD, then ZO-AdaMM and then ZO-SGD. In fact, the higher the distortion evaluated by the l2 norm and the more the image will be modified and thus the adversarial attack will be successful in less iterations.

In the paper of Liu et al. the authors proved the convergence of ZO-SignSGD to be lower than $O(\sqrt{d}/\sqrt{T})$ which is the convergence rate of ZO-SGD. The results in Table (1) give a summary of our experiments. The first column ASR is for Average success rate so it is the number of images for which the adversarial attack was successful over all the total number of images for which an attack was run. So, of course the AVR is 50% for both ZO-SGD and ZO-SignSGD since half of the attacks converged and 100% for ZO-AdaMM. The average succesful iteration number is lowest for ZO-AdaMM, since for the first three images this algorithm did not converge while ZO-SignSGD did converge so its average is higher given that for the first three images more iterations were needed to converge. Instead the smallest average successful distortions and final distortions are smallest for ZO-SGD but the average iteration number for the first success is more than six times bigger than for ZO-AdaMM .

Table 1: Algorithms Comparison

Name	**ASR**	**Avg_succ_iter**	**Avg_succ_dist**	**Avg_final_dist**
ZOSGD	0.5	382.33333	0.006 567	0.023603
ZOSignSGD	1	216.16666	0.173 355	0.100611
ZOAdaMM	0.5	66.66667	0.263 69	0.045677

Eventually, even if we tested these algorithms only on 6 images the results of the paper Chen et al. are confirmed: ZO-signSGD compared to ZO-AdaMM has a poor convergence accuracy in terms of the increase in ‘l2 distortion after the attack becomes successful however ZO-AdaMM holds a lower ASR for unconstrained problems. Regarding, ZO-SGD this method seems the least effective given it takes significantly more iterations than both methods to find the first successful adversarial example with a similar distortion value of ZO-AdaMM.

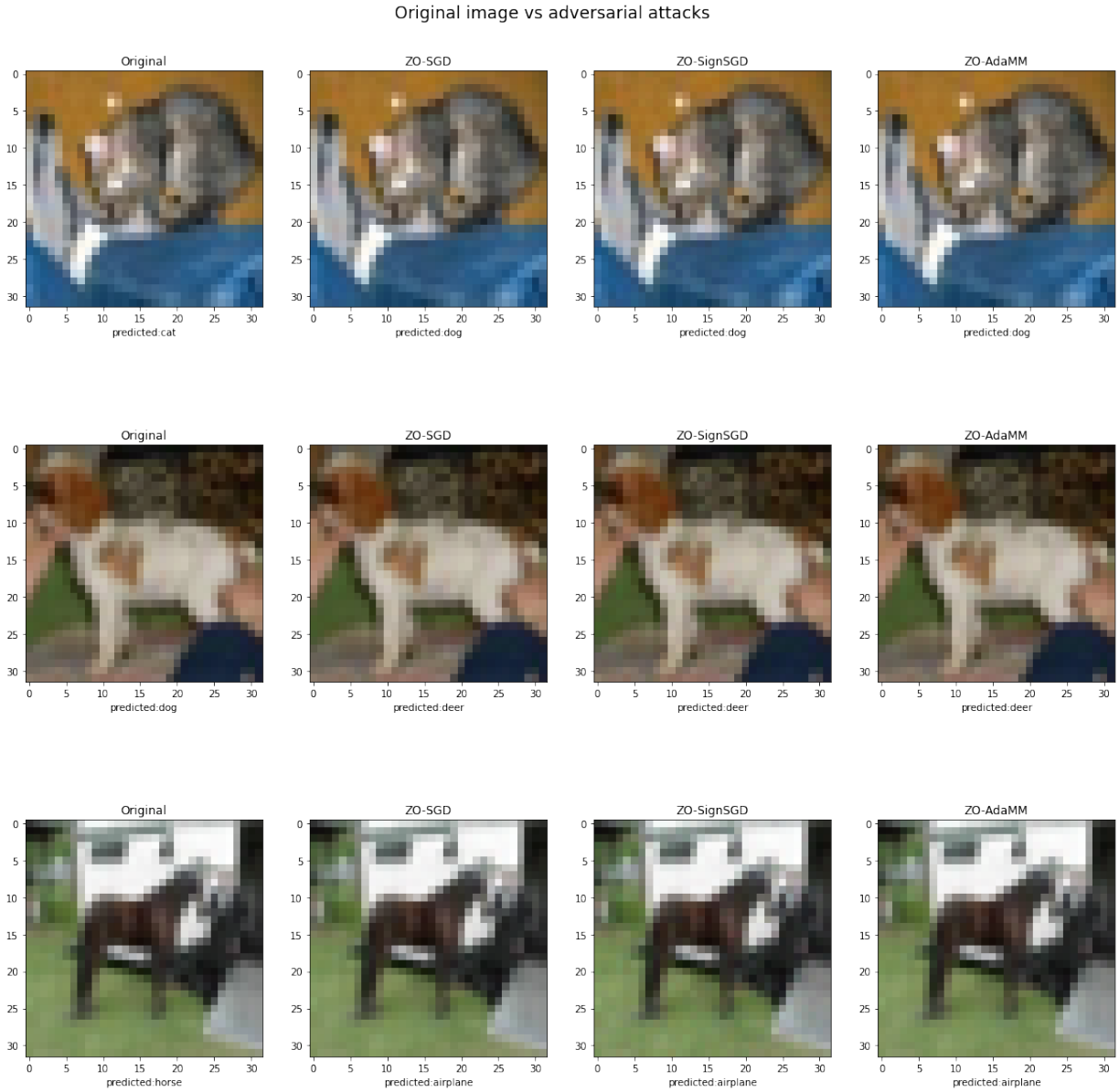

Figure 4: Successful Adversarial Attacks

If we look at the adversarial images generated at the first successful iteration in figure (4) and final images in figure (5) we can notice that the images look all very similar to the original one for all algorithms since the distortion is higher for ZO-SignSGD and very similar for ZO-SGD and ZO-AdaMM however the difference is not perceivable at the human eye.

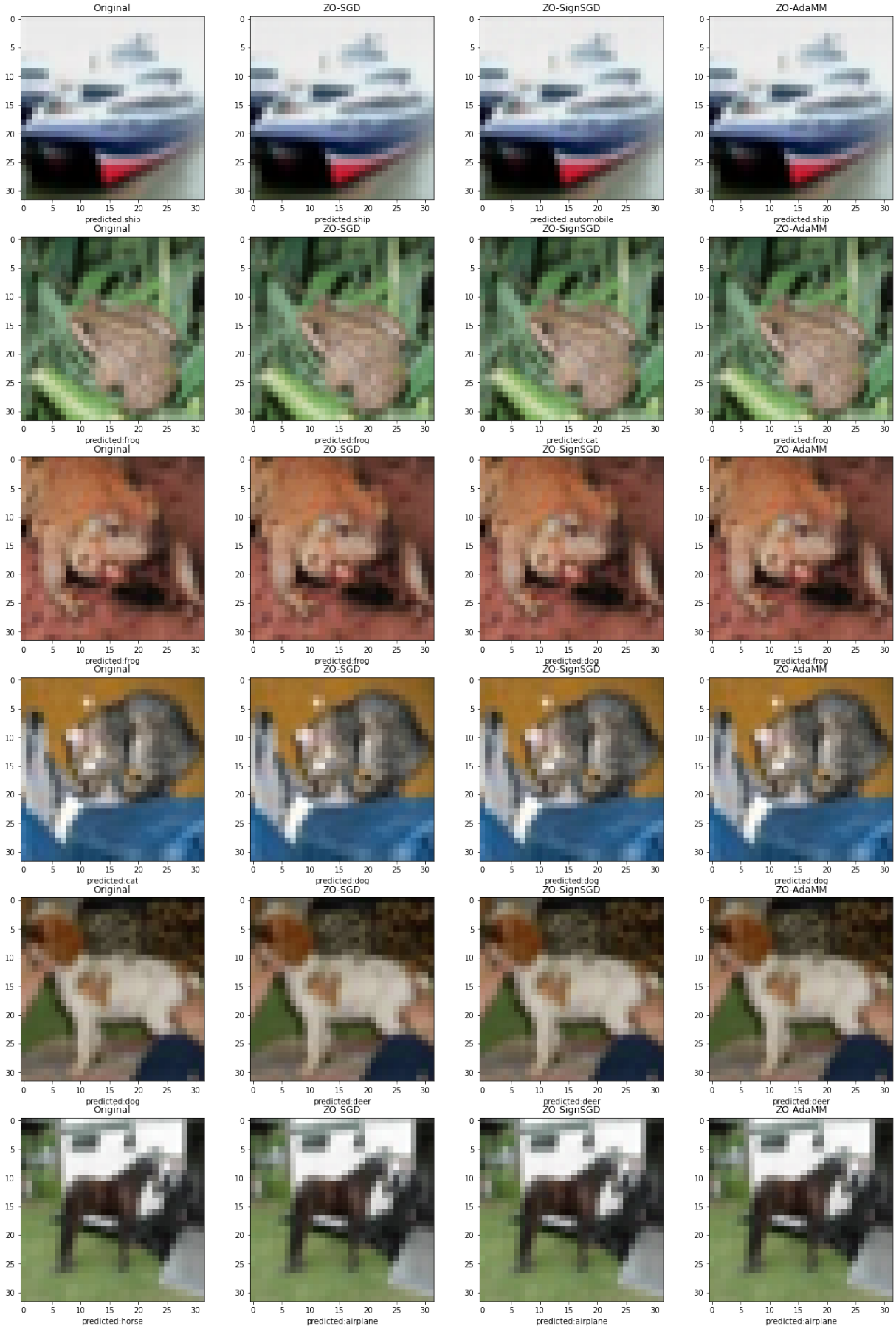

Figure 5: Original images vs Adversarial Attacks

5 Conclusion

In general so evaluating which the best attack performance should correspond the best trade-off between attack loss and distortion. In the case of the three images with high prediction score only ZO-Sign-SGD could converge and thus generate a successful attack. For the other three images all of them were converging and ZO-AdaMM need a similar number of iterations to converge when compared to ZO-SignSGD but a significant lower amount of distortion. Thus, ZO-AdaMM method should be preferred in case of generating an adversarial attack for a model that doesn't hold very high prediction scores for certain images, while ZO-Sign-SGD can converge at a very fast speed but its solution does not hold very high accuracy. Between all methods, SGD is the least efficient since it requires more iterations to converge while it returns a similar distortion to ZO-AdaMM.

6 References

- Chen, X., Liu, S., Xu, K., Li, X., Lin, X., Hong, M., Cox, D. (2019). ZO-AdaMM: Zeroth-Order Adaptive Momentum. 33rd Conference on Neural Information Processing Systems. Vancouver, Canada: NeurIPS.

- GHADIMI, S., GUANGHUI, L. (2013). STOCHASTIC FIRST- AND ZEROTH-ORDER METHODS. Society for Industrial and Applied Mathematics, 2341–2368.

- Liu, S., Chen, P.-Y., Chen, X., Hong, M. (2019). SIGNSGD VIA ZEROTH-ORDER ORACLE. ICLR. MIT-IBM Watson AI Lab.