HAAT: Improve Adversarial Robustness via Hessian Approximation

Panchi mei, Yukai Yang, Zequan Wu

May 4, 2022

Abstract

Adversarial attacks has become one of the most popular fields of deep learning in recent years. Various adversarial training methods are developed to improve model robustness. In this project, we present a new variant of projected gradient descent (**PGD**) that uses second-order Taylor expansion of the adversarial loss approximation to help find better adversarial examples. Contrary to common belief, however, our results show that more terms of expansion does not lead to a significantly better result. We establish our conclusion with a strong theoretical proof and experimental support. We also discuss certain limitations and provide directions for future studies in this subject.

1 Introduction

Deep learning has demonstrated important success in various fields, including computer vision, natural language processing, and many more.[8] It is with no doubt that these networks will bring new technological heights to our society. The security and reliability of such models, hence, become an important issue that cannot be overlooked. In recent works, it is shown that slight perturbations to an input may drastically change its output from a DNN model.[9] Such inputs are known as *adversarial examples*. Failure to detect a traffic sign due to the existence of a tiny, usually unnoticeable, advertisement sticker at the corner of the image is an example of such adversary. Not only do these large changes in outputs exist, but they are also often presented with a high confidence. Moreover, such variations are infinitesimal, and are almost always undetectable by human sensations. This confidence and imperceptibility make the issue even more significant since it presents a hidden risk in the innate fundamental theories of DNNs. It is crucial for this vulnerability against slight perturbations to be solved in order for DNNs to advance to a new stage in its development and implementation.

Since its debut, adversarial examples have attracted a large amount of research. Many work sought to study *adversarial robustness*, or the design of classifiers to perform well against imperceptible perturbations. Many such algorithms have yielded significant results in this manner. However, these algorithms are then found to be prone to other susceptibilities, including overfitting, etc. Moreover, machine learning scientists continuously discovered new attacking algorithms to target existing blind spots in state-of-the-art models. We are, hence, far from stating that we have found a reliable solution to adversarial examples.

Following recent investigations in the implementation of second-order regularizers in the approximation of adversarial loss, our work aims to take an alternate perspective within the scope of second-order robustness. Our contributions are summarized as following:

- We propose a new method to approximate adversarial loss using second-order Taylor approximation, Hessian matrix, and the finite difference approximation in which we name HAAT (Hessian Approximation for Adversarial Training).
- We provide a theoretical guarantee for the effectiveness of HAAT.
- We analyze HAAT experimentally and present baseline results.
- We investigate the significance of a change in hyperparameter in HAAT and propose a series of optimal hyperparameters.

- We discuss limitations in our study, including the effectiveness of finite difference approximation and the tightness of our theoretical bounds, and provide directions for future works.

2 Related Works

Since the introduction of the notion of adversarial perturbation [9], many machine learning scientists have conducted studies attempting to explain such vulnerability or boost adversarial robustness. Goodfellow et al. [2] observed that a major cause of the existence of adversarial examples is the strict linearity of machine learning models. He claimed that for an adversary input $\hat{x} = x+r$ from a standard input x where $||r||_\infty < \varepsilon$, the change in the output can grow linearly with the dimensionality of the problem, resulting in an adversarial example.

Empirical risk minimization (ERM), in which a known set of data is used to empirically minimize the true risk of any classifiers on an unknown distribution, is a method proven to be incredibly efficient in searching for an efficient classifier. While ERM cannot directly yield robust learning models, scientists implemented derivations to minimize adversarial loss. Noticeable such algorithms include adversarial training (AT) from [5]. Madry et al. minimizes adversarial loss by solving an optimization problem that involves the following objective function:

$$\min_f E[\max_{||r||_\infty<\varepsilon} L(f(x+r), y)]. \tag{1}$$

AT uses projected gradient descent (PGD) to approximate such first-order perturbations. We will also use PGD as one of our main tools in approximating adversarial attacks. Another such algorithm, TRADES [11], takes advantage of the observation that a tradeoff exists between standard accuracy and adversarial robustness. Zhang et al. incorporates a regularization term to minimize the difference between the outputs of a standard input and an adversarial input. Namely, TRADES aims to optimize the following objective function:

$$\min_f E[L(f(x), y) + \max_{||r||_\infty<\varepsilon} L(f(x)f(x+r), y)/\lambda] \text{ for any } \lambda > 0. \tag{2}$$

These algorithms yielded state-of-the-art results again adversarial attacks. New attacks [6][1][7], however, continue to target undiscovered blind spots in such robust models.

Recent works on adversarial robustness have seen an increasing effort on attack approximations. Based on the assumption that adversarial loss is twice-differentiable, many scientists began targeting beyond first-order adversary and aimed for second-order robustness. Ma et al. [4] attempted to propose second-order adversarial regularization (SOAR) by approximating the loss function using its second-order Taylor series expansion. Namely, SOAR optimizes the following objective function:

$$\min_f E[\max_{||r||_\infty<\varepsilon} L(f(x), y) + \nabla_x L(f(x), y)^T r + \frac{1}{2} r^T \nabla_x^2 L(f(x), y) r]. \tag{3}$$

Moreover, SOAR implements Hessian matrix to facilitate the derivation of an upper bound on this expansion term. To do so, Ma et al. writes the adversarial loss as following:

$$L(f(x+r), y) \approx L(f(x), y) + \frac{1}{2} \begin{bmatrix} r \\ 1 \end{bmatrix}^T \begin{bmatrix} \nabla^2 L(f(x), y) & \nabla L(f(x), y) \\ \nabla L(f(x), y)^T & 1 \end{bmatrix} \begin{bmatrix} r \\ 1 \end{bmatrix} - \frac{1}{2} = L(f(x), y) + \frac{1}{2} r'^T H r' - \frac{1}{2} \tag{4}$$

, where $r' = [r; 1]$ and a single Hessian term H. SOAR, however, proved to be inefficient and vulnerable against slightly stronger attacks.

Tsiligkaridis et al. [10] also proposed a similar second-order robust optimization algorithm, SCORPIO, that incorporates a quadratic regularizer to smoothen the surface of the loss function. Noticeably, SCORPIO uses Frank-Wolfe algorithm to solve the above second-order optimization problem iteratively, which guarantees linear convergence to the optimal value. The results yielded by SCORPIO, however, also fail to exceed those from AT as the accuracy from the two match almost perfectly against L^∞ attacks. Motived by AT, SOAR, and SCORPIO, we propose an alternate method to approximate second-order adversary aiming to reach better efficiency than state-of-the-art models.

3 Methods

Let $L(x, \theta)$ be the loss function. For the trained model to be robust under a ε perturbation in the L^∞ space, the ideal loss function would be

$$L_{adv}(x, \theta) = \sup_{\|r\|_{L^\infty} \leq \varepsilon} L(x + r, \theta).$$

We assume that the high order derivatives of L diminish or are bounded and negligible. We note its second order Taylor approximation:

$$L(x + r, \theta) \approx L(x, \theta) + r^T \nabla_x L(x, \theta) + \frac{1}{2} r^T \nabla_x^2 L(x, \theta) r.$$

If one assumes the loss is convex, it follows that the Hessian matrix, $\nabla_x^2 L$, is positive semi-definite and the following optimization problem,

$$\sup_{\|r\|_{L^\infty} \leq \varepsilon} r^T \nabla_x L(x, \theta) + \frac{1}{2} r^T \nabla_x^2 L(x, \theta) r,$$

can be solved by Frank-Wolfe's algorithm iteratively and the algorithm guarantees a linear rate of convergence to the optimal value.

We propose an alternative method to approximate the optimizer of the second order Taylor approximation,

$$r_{adv} := \arg \sup_{\|r\|_{L^\infty} \leq \varepsilon} L(x, \theta) + r^T \nabla_x L(x, \theta) + \frac{1}{2} r^T \nabla_x^2 L(x, \theta) r,$$

that combines the optimizer of the first order term in the Taylor expansion of L_{adv} and the optimizer of the second order term, denote them respectively by

$$r_{adv}^1 := \sup_{\|r\|_{L^\infty} \leq \varepsilon} r^T \nabla_x L$$
$$r_{adv}^2 := \sup_{\|r\|_{L^\infty} \leq \varepsilon} r^T \nabla_x^2 L r$$

Immediately, $r_{adv}^1 = \varepsilon sign(\nabla_x L)$. However, to find an exact solution of r_{adv}^2 is an NP-hard problem. Instead of finding the optimizer over the ε-cube, we extends the problem to the $\sqrt{d}\varepsilon$-ball where d is the dimension of the input, since

$$\sup_{\|r\|_{L^\infty} \leq \varepsilon} r^T \nabla_x^2 L r \leq \sup_{\|r\|_{L^2} \leq \sqrt{d}\varepsilon} r^T \nabla_x^2 L r.$$

The L^2 optimization problem is trivial. We assume that the Hessian matrix is non-degenerate and has a unique dominant eigenvalue. Denote by $|\lambda_1| > |\lambda_2| > |\ldots| > \lambda_d$ where λ_i's are eigenvalues of $\nabla_x^2 L$ and by u_i the corresponding unit eigenvectors. Then the optimizer of the problem in L^2 is simply its dominant eigenvector u_1, which we can find rather easily. For any v where $v^T u_1 \neq 0$, $\frac{(\nabla_x^2 L)^k v}{\|(\nabla_x^2 L)^k v\|_{L^2}}$ converges to u_1 at exponential rate where the base is the spectral gap of $\nabla_x^2 L$, that is,

$$\| \frac{(\nabla_x^2 L)^k v}{\|(\nabla_x^2 L)^k v\|_{L^2}} - u_1 \|_L^2 \leq \mathcal{O}((\frac{\lambda_2}{\lambda_1})^k).$$

Since finding the exact Hessian and evaluating $(\nabla_x^2 L)^k v$ is very costly, we use the finite difference approximation

$$\nabla_x^2 L(x, \theta) v \approx \frac{\nabla_x L(x + \xi v, \theta) - \nabla_x L(x, \theta)}{\xi}.$$

We therefore propose the loss function, with hyperparameter $w_1 = 1$ and $w_2 = 2$,

$$\begin{aligned} L'(x, \theta) &= L(x, \theta) + w_1 \|L(x, \theta)\|_{L^1} + \frac{w_2 \lambda_1}{2} \\ &= L(x, \theta) + w_1 \nabla L(x, \theta)^T r_{adv}^1 + \frac{w_2}{2} u_1^2 \nabla^2 L(x, \theta) u_1 \\ &\geq L(x, \theta) + r_{adv}^1 \cdot \nabla_x L(x, \theta) + r_{adv}^2 \cdot \nabla_x^2 L(x, \theta) r_{adv}^2 \\ &\geq L(x, \theta) + \sup_{\|r\|_{L^\infty} \leq \varepsilon} r^T \nabla_x L(x, \theta) + r^T \nabla_x^2 L r \end{aligned}$$

which upper bounds the second order Taylor expansion of L_{adv} and controls the gap between the test accuracy and the accuracy under an ε-cube attack.

In the actual implementation, we perturb the input with two adversarial directions, r^1_{adv} and r^2_{adv},

$$x' := x + w_1 r^1_{adv} + w_2 r^2_{adv}.$$

We summarize our implementation in Algorithm 1 below, with w_2 dependent on u_1

Algorithm 1 **H**essian **A**pproximation for **A**dversarial **T**raining (**HAAT**)

STEPS $= 10$, $k = 5$, $w_1 = \frac{1}{4}$
For input x
for $i \leftarrow 1$ to STEPS **do**
  Draw u from $\mathcal{N}(0, 1)$
  for $j \leftarrow 1$ to k **do**
    $Hu \leftarrow \frac{\nabla_x L(x+\xi d) - \nabla_x L(x)}{\xi}$
    $u \leftarrow \frac{Hu}{\|Hu\|_{L^2}}$
    $x \leftarrow \prod_{x+\varepsilon} x + w_1(\nabla_x L) + \varepsilon \frac{\sqrt{d}}{\max_i |d_i|} d$
  end for
end for

4 Experiments

4.1 Baseline Result

We mainly work with the CIFAR-10 dataset for experiment. We use the architecture of ResNet18 as our baseline model, train the model with our second-order approximation algorithm, and compare our results with the l_∞ PGD attack in [5]. For the baseline result, we use attack type $= l_\infty$, $\varepsilon = 8/255$ (as required), ξ, the step size of finite difference approximation $= 10^{-6}$, and numbers of finite difference approximation step $= 5$.

For the model, our ResNet18 is initialized randomly without any pretraining (so no extra data is fed as required in the project). We use SGD as the optimizer, cyclic scheduler learning rate scheduling and initial learning rate $= 0.1$. We run 200 epochs to train with the whole CIFAR-10 training dataset, ,make checkpoint every 20 epochs starting from epoch 10, and get the result shown in Figure 1.

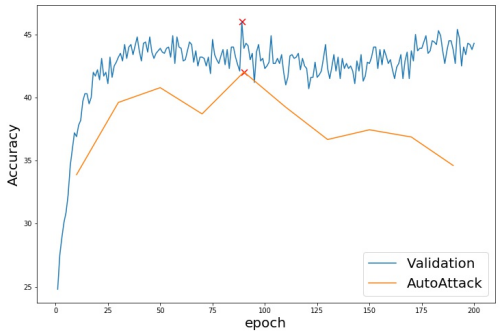


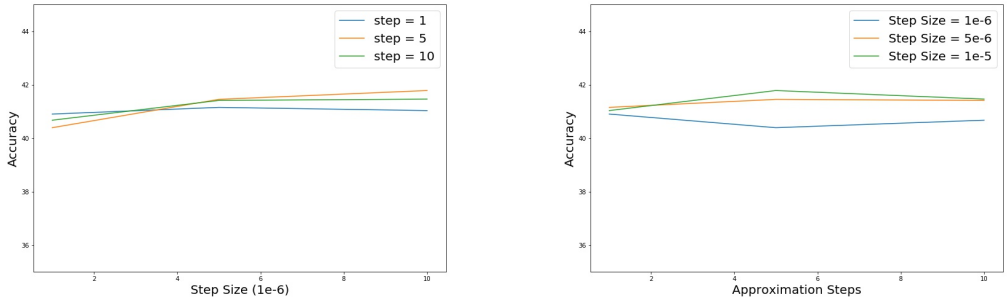
Figure 1: Baseline Result using HAAT

We can observe that, in average, the baseline model reaches its best validation performance at epoch 89. When it goes to after 100 epochs, it becomes difficult to have siginificant improvement, and may lead to overfit according to AutoAttack Result. There is slight inconsistency between validation and test performance. This may due to the randomness of testing data, and can also because of our model's transferability from PGD attack to various attack in AutoAttack package does not improve with accuracy in parallel.

4.2 Hyperparameter Tuning

The project restricts the training data to be CIFAR-10 and the attack to be AutoAttack with l_∞ and $\varepsilon = 8/255$. Due to the limited computational resources the students have, we did not test our models with abundant SOTA testing techniques. Instead, we focus on hyperparameter tuning, and record the change of key values like finite difference approximation convergence, in order to see how our new algorithm affect the convergence efficiency during training.

The learning rate of model will be adjusted by the cyclic scheduler, so we focus on the two hyperparameters in our algorithm: FD step, and step size ξ. The first one determines how much our approximation converges to the true hessian times perterbation, and the second determines how fast the approximation goes. The results are shown in Figure 2 and Figure 3.



(a) Tuning FD Step Size

(b) Tuning number of FD steps

Figure 2: Tuning Number of Approximation Steps k

We observe that step size = 5e-6, and number of steps = 5 in our experiments will provide the best performance.

Finally, we compare the best hyperparameter-tuned model with the first-order adversary PGD model we trained. As shown in Figure 4, the two model present very similar results. Our model takes a longer time to converge, but it only provides very limited improvement against AutoAttack. This is quite surprising, as it indicates that adding the approximated new second-order term using our algorithm does not significantly improve the adversarial robustness. In the next session we will have a more thorough discussion on this.

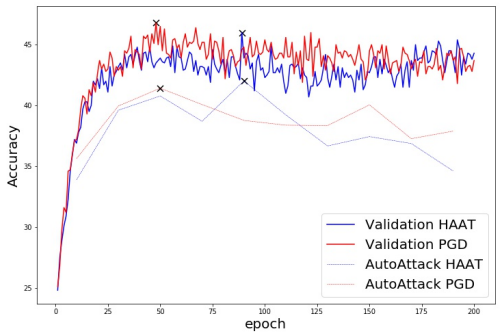


Figure 3: Baseline Result Comparison of PGD and HAAT

5 Discussion

The result we get here is very similar to the work of [4] and [10]. In this section we provide some analysis of why using a wider expansion expression does not provide practical robustness improvement,

which was poorly discussed in other works.

5.1 Finite Difference

Recall that we use finite difference to approximate the Hessian matrix times perturbation: $\nabla_x^2 Lr$. Naturally, we would expect the approximation to converge to the optimal values with several steps. In fact, similar studies like [4] has shown that the number of steps here does not matter: 1 step can reach the same performance with 10 steps.

We tested the efficiency of convergence in our experiment. Surprisingly, it does not converge most of the time. We randomly select 10 batches from each of the three experiments with step size 1, 5, and 10, and draw the norm of the difference between each updated approximation of the worst perturbation in the last attack step, i.e. $\text{norm}(r^{(i)} - r^{(i-1)})$ in Figure 5.

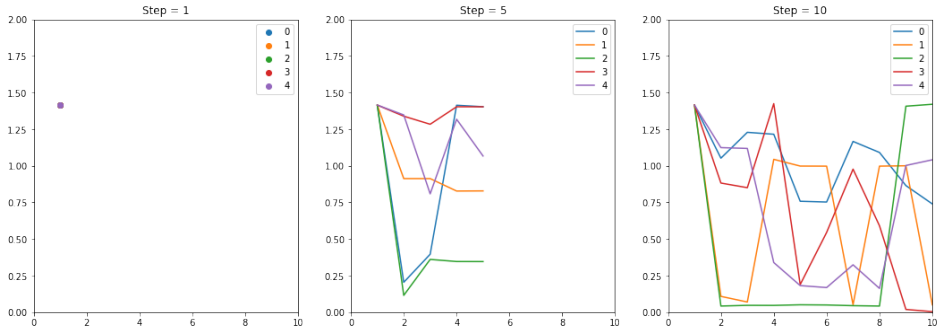


Figure 4: Finite Difference Approximation

When step size = 1, the norm difference is the norm(length) of the difference of two almost perpendicular unit vectors. Thus we can see in Figure 5(a) all the points are around $\sqrt{2}$. Starting from $\sqrt{2}$, however, finite difference does not always make the perturbation converge. As shown in Figure 5(b) and 5(c), it often fails to converge and that means we are far from finding the worst adversarial example. This indicates shows that finite difference may not be very effective to approximate gradient of very high-dimensional data.

Nevertheless, our results show that step = 10 does have a higher probability of convergence than step = 5. This suggests that more approximation steps does provide a better convergence of the hessian approximation. However, the trade-off between convergence effectiveness and computational resources is clearly an issue in experiments.

5.2 Inequality Bounds

In the theory part, we use L_2 ball of radius $\varepsilon\sqrt{d}$ to enclose the L_∞ ball of radius ε, so that we can apply abundant mathematical tools for L_2 ball optimization problems to the L_∞ case we have. This bound can become very loose, as the dimension d increases. The ratio $V_{L_2, r=\varepsilon\sqrt{d}}/V_{L_\infty, r=\varepsilon}$ goes to infinity as d goes to infinitely large.[4] That means when we transform the L_∞ attack to the L_2 attack that encloses it, we are using a very loose bound in high dimensional case.

Despite that, we have reached a much better performance than the work of previous second-order approximation methods like [3] and [4]. This means the improvement of our algorithm part may help to undermine the risk of this weak bound.

6 Conclusion

In this project, we present an improved version of PGD algorithm that uses second-order expansion of the objective function to approximate and find the worst adversarial example. Our algorithm require a longer time to train, but it does provide a small improvement comparing with the original PGD.

We analyze the reasons why our approximation is not that powerful as expected, and claim that finite difference approximation's poor performance in high-dimensional data and our inequality bounds may be the reasons behind this. Our future work will focus on: (1) Find a more effective algorithm to

approximate large hessian matrix. (2) Consider using multiple eigenvectors to find better adversarial directions.

The algorithm we present in this project is universal in the field of adversarial training, and so are the remaining problems we have. Should either one of the problems be solved completely, we would expect a more significant and promising improvement for adversarial robustness.

References

[1] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks, 2016.

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples, 2014.

[3] B. Li, C. Chen, W. Wang, and L. Carin. Second-order adversarial attack and certifiable robustness. *arXiv preprint arXiv:1809.03113*, 2019.

[4] A. Ma, F. Faghri, and A. Farahmand. Adversarial robustness through regularization: A second-order approach. *arXiv preprint arXiv:2004.01832*, abs/2004.01832, 2020.

[5] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks, 2017.

[6] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks, 2015.

[7] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings, 2015.

[8] W. Samek, G. Montavon, S. Lapuschkin, C. J. Anders, and K.-R. Muller. Explaining deep neural networks and beyond: A review of methods and applications. *Proceedings of the IEEE*, 109(3):247–278, mar 2021.

[9] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *ICLR*, 12 2013.

[10] T. Tsiligkaridis and J. Roberts. Second order optimization for adversarial robustness and interpretability. *arXiv preprint arXiv:*, 2009.04923, 2020.

[11] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan. Theoretically principled trade-off between robustness and accuracy, 2019.