DISENTANGLING ADVERSARIAL EXAMPLES

A PREPRINT. WORK IN PROGRESS

Kiarash Farivar
supervisor:Igor Krawczuk*
LIONS/LSM @ EPFL

June 28, 2021

1 Introduction

The aim of this semester project was to investigate the contribution of different elements of deep learning (such as loss function and dataset) to the phenomena of adversarial examples. Adversarial examples are standard images with small perturbations added to them such that when they are evaluated on a standard model they significantly degrade the accuracy of it, while at the same time the semantic information of these images mostly remains unchanged. Adversarial examples have implications to the field security where it is important to have a model that is not easily fooled. But more importantly then can help us understand deep learning algorithms better [Ort+21]. A benchmark was created to measure the effect of separate elements of deep learning (e.g dataset, loss function and model) on the phenomena of adversarial examples. A hook design was used to facilitate the addition of new measures that can assess any of the elements involved in deep learning. Some measures were added as a demonstration and to investigate the different phenomena regarding adversarial examples.

2 Concepts and Formalisms

In this section I will talk about the mathematical definitions of the concepts used in this project and the general design of the library.

2.1 Adversarial examples and attacks

An adversarial example is a sample from the original distribution of the data that has some small noise added to it such that the image remains almost identical to the original, but it is able to fool a model that has been trained on a standard dataset. [Ort+21] More formally for a sample $x \in \mathbb{R}^n$ and a classification model $f_\theta(x)$ with parameters θ, we can define the adversarial perturbation $r(x) \in \mathbb{R}^n$ as the solution to :

$$\begin{aligned} \min_{r \in \mathbb{R}^n} \quad & Q(r) \\ \text{s.t.} \quad & f_\theta(x+r) \neq f_\theta(x) \\ & r \in C \end{aligned} \tag{1}$$

Here, $Q(r)$ represents a general objective function, and C denotes a general set of constraints that characterises the perturbations. We generally refer to the perturbed samples $x + r(x)$ as adversarial examples. Depending on the choice of $Q(r)$ or C different types of adversarial perturbations can be defined. For instance if $Q(r) = -L(x+r, y; \theta)$ where L is the loss function and $C = \{r \in \mathbb{R}^n : \|r\|_p \leq \epsilon\}$ we get an ϵ-constrained adversarial example. This is the type used in the robust training in this project. If $Q(r) = \|r\|_p$ and $C = \mathbb{R}^n$ then we get an adversarial example with minimal l_p norm. In the literature the l_1, l_2 or $l_{\inf}$ norms are usually used as a measure of visual similarity of images. It is also possible to fix the class that is the output of $f_\theta(x+r)$ in that case we have a targeted attack, while the former is called an untargeted attack.

*`https://krawczuk.eu`

Different attacks try to solve this optimization problem with different methods. Two of the most famous ones are FGSM(The fast gradient sign method) and PGD (projected gradient descent) attacks. FGSM uses a simple one step optimization to solve the problem where the perturbation is bounded in a $l_{\inf}$ ball by ϵ. The adversarial example is defined as:

$$x + \varepsilon \operatorname{sgn}\left(\nabla_x L(x, y, \theta)\right) \tag{2}$$

Note that in these optimizations the variable is the input x and not the weights θ. PGD uses multiple steps of projected gradient descent to minimize the negative of loss, which gives it a better chance at finding examples that maximize the loss. This comes at the cost of slowing down the process and choosing an appropriate rate α. The PGD adversarial example is given by:

$$x^{t+1} = \Pi_{x+C}\left(x^t + \alpha \operatorname{sgn}\left(\nabla_x L(x, y, \theta)\right)\right) \tag{3}$$

Here as before C is the set of all possible perturbations.

2.2 Robust training

Robust training is the process of training a model such that it is resistant to a set of adversarial attacks. To the best of my knowledge it has not been possible to train a model that is resistant to all possible attacks. The process of robustly training a network with respect to a set of attacks is identical to the standard training of a neural network except each sample needs to be replaced by the adversarial examples that the chosen set of attacks produce. Usually it is best if these are examples that maximize the loss function, since they represent the worst case scenario for the network. This method was first suggested by [Mad+19], it is not trivial why this method works and [Mad+19] uses the Danskin's Theorem to prove the correctness of their method. Formally robust training can be formulated as:

$$\min_{\theta} \rho(\theta), \quad \text{where} \quad \rho(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\max_{r\in C} L(x + r, y, \theta)\right] \tag{4}$$

Where D is the dataset distribution. We can see that the inner maximization is a set C constrained adversarial example.

2.3 Implemented measures

Standard and Robust accuracy: The standard accuracy of a model is the number of correctly classified data points over the total number of data points in the test set. The robust accuracy of a classifier is defined as the percentage of the samples in the test set that the model can correctly classify after an attack. Both these measures are fundamental in assessing any claims regarding the adversarial examples. For instance we can consider robust training as a method to improve the loss function to achieve a more robust model and to measure the success of this method we can compare the results of these measures before and after robust training.

The saliency map: is a type of tool used to interpret how neural networks make their decisions. In their simplest form we can consider the gradient of the loss function with respect to the inputs and visualize their absolute values to show which pixels affect the decision of the network the most. By comparing this to our own human perception we can see possible shortcomings of models.

The Clever score: [Wen+18] is an estimate of the minimum perturbation needed to create an adversarial example for a specific model and a specific sample. Like an attack it can be targeted towards a specific class (other than the correct class) or untargeted (minimum of all the targeted scores). The authors try to find this estimate by sampling points in an epsilon ball around the data point, calculating the norm of gradient of difference between the output of the model corresponding to the correct class and the target class and fitting a Weibull distribution (from extreme value theory) on this data points using MLE. The advantage of this measure is that it is independent of any specific attacks and only considers the model and the dataset. Although a specific norm and amount of perturbation should still be chosen.

The concentration of measure of the dataset: The authors of [Mah+19] use the concentration of the dataset to estimate a lower bound to the adversarial risk (one minus robust accuracy) of a family of classifiers. The lower bound is given by $h\left(\mu, \operatorname{Risk}\left(f, f^*\right), \epsilon\right)$ where μ is the probability density function for the dataset, $\operatorname{Risk}\left(f, f^*\right)$ is the best empirical error achievable on the dataset by any reasonable classifier, ϵ is the amount of perturbation and $h(.)$ is the concentration function. The family of models is defined as any model that has standard error $\operatorname{Risk}\left(f, f^*\right)$ or higher. This measure mostly concerns the contribution of the dataset to the phenomena of adversarial examples. The authors propose a method to estimate this value by estimating an empirical distribution, considering subsets of the dataset and using clustering and the Knn algorithm.

(a) Dog (b) Deer (c) Truck

(d) Ship (e) Dog (f) Frog

Figure 1: Examples of FGSM adversarial images. the first row shows the original images with correct labels, whiles the second row shows the adversarial images with the incorrect label classifier assigns to them.

2.4 The Library

To make the benchmark easily extendable a hook design was used. This means iterating through the dataset only once and batch wise. For each batch the inputs, labels, the results of the prediction of the model and the model itself are sent to the "on_clean" hook of all measures to be evaluated for standard data points. An example for this type of measurement is standard accuracy. For measures that need adversarial data for each batch we iterate through a list of attacks. Another hook is called in this case, "on_attack", which in addition to the previous inputs also takes the adversarially perturbed inputs, the resulting outputs and the attacks used.

An abstract class Measure is defined that includes the "on_clean" and "on_attack" methods. So that a new measure can be added to the benchmark by simply implementing the abstract Measure class. This has the advantage that the dataset is only traversed once so the overhead for I/O is reduced.

3 Experiments

In this project I exclusively used the FGSM(fast gradient sign method) attack with $\epsilon = 8/225$ and $norm = L_\infty$ as used in [GSS15], since it is fast and it is sufficient to analyse most of the fundamental concepts in adversarial examples. Some examples of FGSM can be seen in 1. The standard model used to create the samples was resnet18 which has a standard accuracy of 93%. For the dataset I used CIFAR10. It consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images (CIFAR10). After visualizing the dataset we can see that some of the images are not really recognizable by humans; this can be problematic for the interpretation of the measures discussed below. But since this dataset is used in most literature and the images are of reasonable size I didn't look for an alternative. I used 18 and 34 layer residual networks (resnet18/34) as my models since they have been shown to perform well on the more complicated Imagenet dataset and have the accuracy of around 93% on CIFAR10.

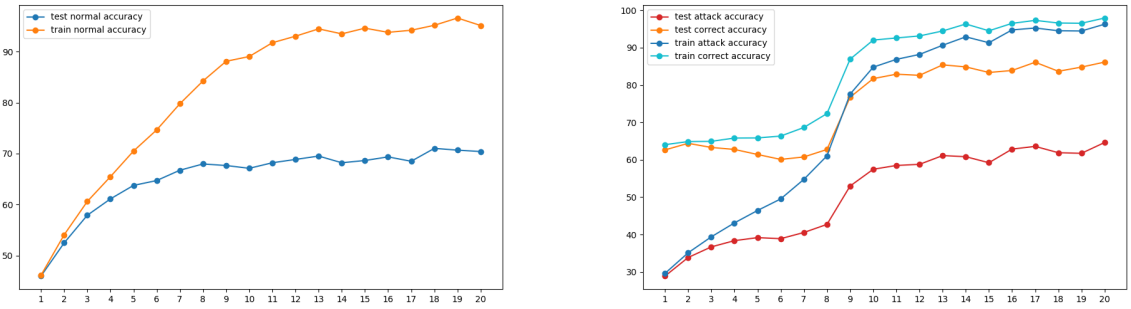
3.1 Robust training

Before training the models had an accuracy below 10% when FGSM was applied to the test set. The networks were exclusively trained on adversarial examples but the standard accuracy was always higher than the robust accuracy. The test and train accuracies are shown in 2 and 3. attack accuracy means all the samples are considered, while *correct*

accuracy means only the standard samples that the model could correctly classify were considered. They look highly correlated.

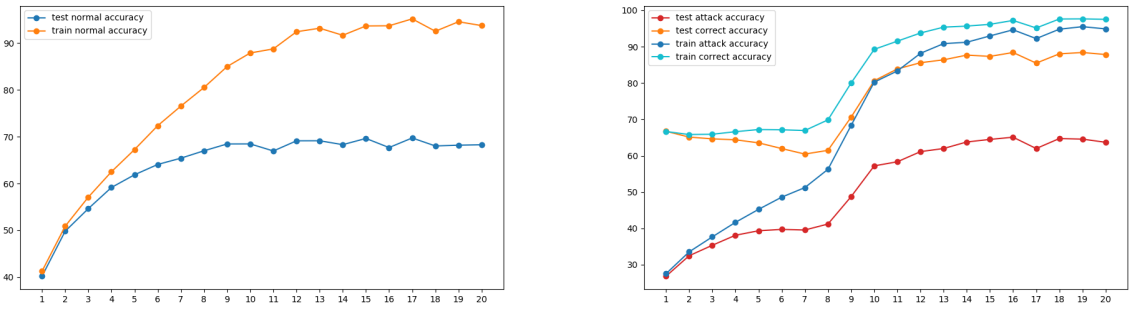
For both models the ADAM optimizer was used. For resnet18: the learning rate was 10^{-3}, number of epochs 20 and batch size 32. The same parameters were used for resnet34.

The test set was also used as a validation set and the best epoch was selected based on that. The plot 2 shows a big gap between train and test accuracies. To improve that I tried data augmentation and shuffling for the trainset, using a bigger batch size(128) and increasing the number of epochs to 100. The changes improved the accuracy by 5%. The best accuracy on test was for epoch 45(70%) 4b. The same changes degraded the accuracy on resnet34 (with the same parameters the loss never improved) so I changed the parameters to: the learning rate 10^{-4} and batch size 64. The loss decreased this time however that didn't improve the results 5. Further grid search for hyper-parameters is required to improve resnet34. For the rest of the results in this section the resnet18 model was used.



(a) Train and test normal accuracies for resnet18 robust training. (b) Train and test robust accuracies for resnet18 robust training.

Figure 2: The accuracy vs epochs plots. Resnet 18 was also trained for 100 epochs but the test attack accuracy never improved above 65%

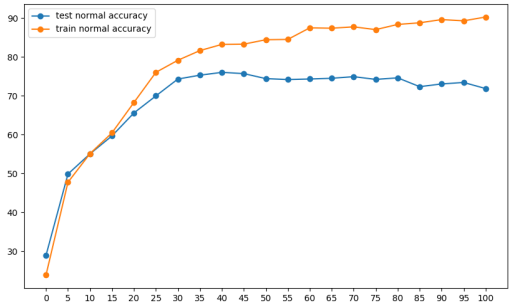


(a) Train and test normal accuracies for resnet34 robust training (b) Train and test robust accuracies for resnet34 robust training.

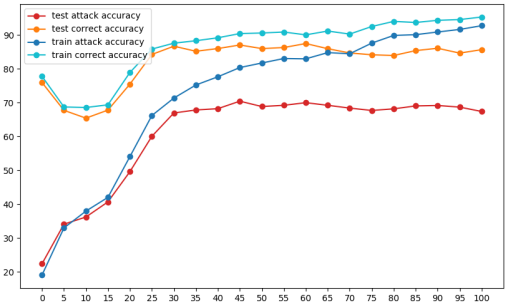
Figure 3: The accuracy vs epochs plots.

3.2 Saliency maps

One of the implemented "measures" was plotting the saliency maps for a subset of images in the testset. This involves plotting the absolute value of gradient of the loss with respect to a single input. These values are plotted as heatmaps in 6. We can see saliency maps are sparser in the case of the robust model (third row) compared to the standard model (second row). This is reasonable since in the presence of an adversary it is beneficial for the robust model to only consider a smaller subset of pixels that are of importance to reduce it's chances of being fooled. Although [Tsi+19] mentions that "the input gradients and saliency maps of adversarially trained networks align well with human perception" this effect was seen in only few of the images in my project. This could be due to the lower robust accuracy of the resnet18 model used (70%) and the fact that I used a simpler coloring scheme. We can also see that the saliency of the robust model for

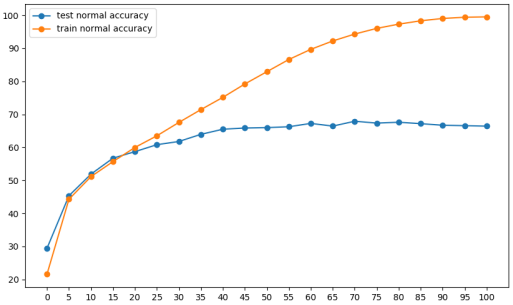


(a) Improved train and test normal accuracies for resnet18 robust training.

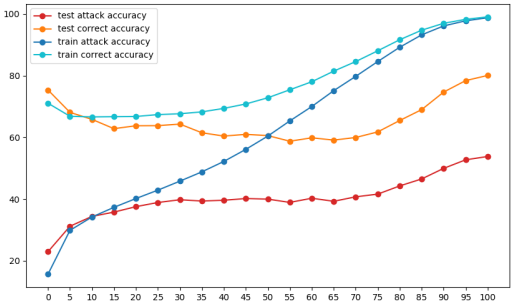


(b) Improved train and test robust accuracies for resnet18 robust training.

Figure 4: The accuracy vs epochs plots.



(a) Train and test normal accuracies for resnet34 robust training



(b) Train and test robust accuracies for resnet34 robust training.

Figure 5: The accuracy vs epochs plots.

some of the images (e.g plane or the frog) gives more importance to the areas of the image that actually include the object of interest. Although for a small subset of images (e.g. deer) the standard model is sparser.

3.3 The Clever score and Concentration

For the clever score the original paper's implementation was for tensorflow so I used the implementation of the adversarial-robustness-toolbox [Nic+18] which also supports pytorch. The main challenge was that the implementation was not optimized properly compared to the original implementation where multiprocessing library was used to parallelize the processing for each sample. Another flaw was that the library didn't estimate the proper initial value for the parameter c_init. All of this made calculating the score on 500 samples (5mins per sample) take about 41 hours to complete which was not feasible (considering sampling parameters far below the recommended parameters in the paper). Calculating the score on few samples of the test set (c_init=1) resulted in values in the order $[10^{-4}, 10^{-3}]$ and an std in the same range. There was no significant difference between the scores of the robust model and the normal model (the normal model scores were mostly higher) which made using this score to evaluate the models ineffective.

The concentration results for CIFAR10 give a lower bound of 30% on the adversarial risk. This is considering any attack with l_{inf} norm of $\epsilon = 8/225$. Since the gap between this risk and the best possible risk (around 50%) is too big the authors of [Mah+19] concluded that there must be factors other than intrinsic properties of the dataset that are contributing to adversarial examples and there is still hope for improvement. To implement this measure since access to all of the samples was required for clustering and Knn algorithms, the batch wise processing was not feasible so another method was added to the library for the measurements that require the whole dataset.

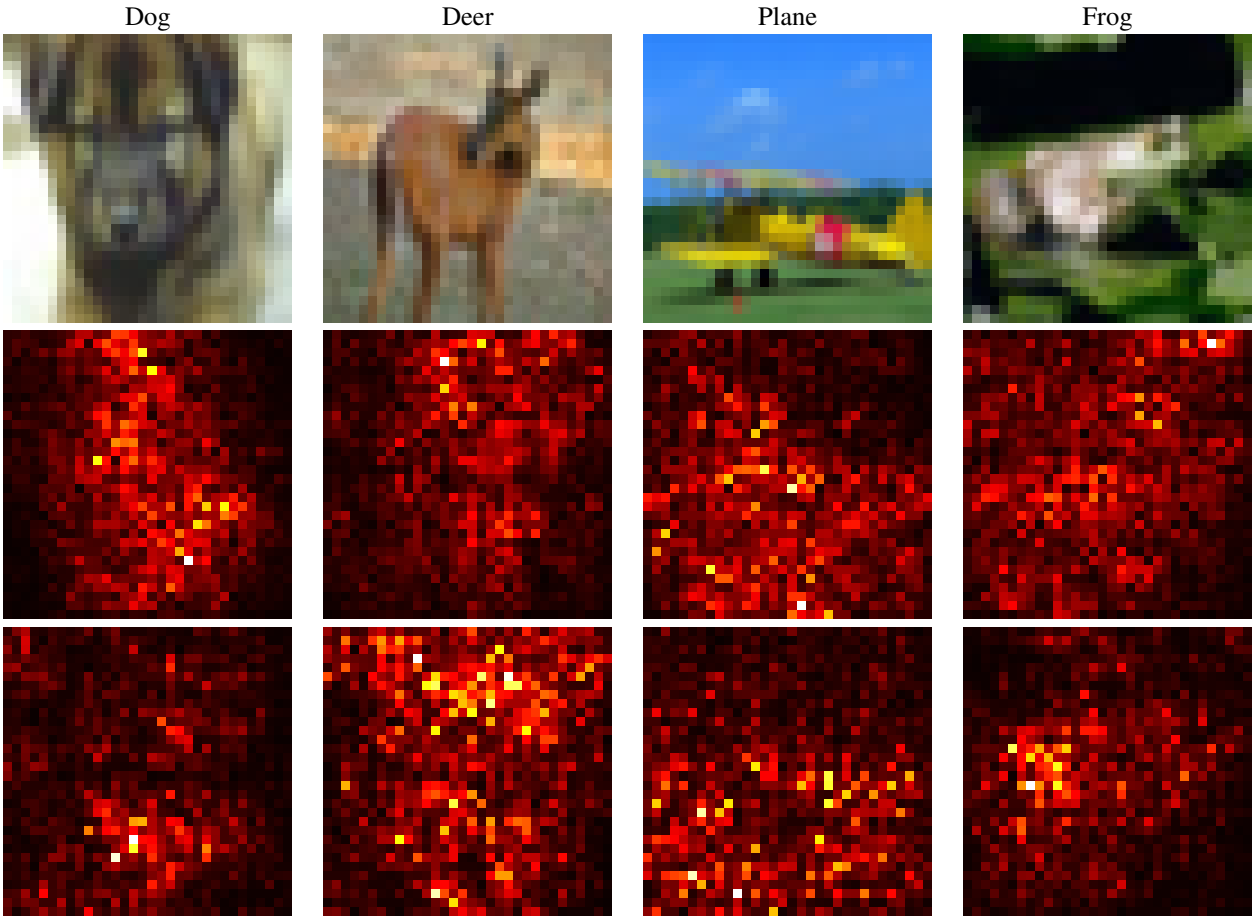


Figure 6: Images and their corresponding Saliency maps for the standard model(second row) and robust model (third row). The images have been aliased by the latex graphics package, the original images are 32 by 32 pixels.

4 Conclusion

In this project I investigated the phenomena of adversarial examples on resnet18 and resnet34 models on the CIFAR10 dataset. I successfully robustly trained the resnet18 with respect to the FGSM attack. I also visualized the saliency maps for this model and observed that the robust model has sparser saliency maps than the standard model. I also investigated the feasibility of calculating the Clever score and measured the concentration of the dataset.

Some shortcomings of the projects include: 1. Improving the training time for the models by parallelizing the calculation of the adversarial examples on multiple GPUs. 2.Performing a more comprehensive grid search for the hyper-parameters of resnet34 to get a better accuracy 3. Improving the running time of the clever score by using multiprocessing .

The code for the project is available here.

References

[Ort+21] Guillermo Ortiz-Jiménez et al. "Optimism in the Face of Adversity: Understanding and Improving Deep Learning Through Adversarial Robustness". In: *Proceedings of the IEEE* 109.5 (2021), pp. 635–659. DOI: 10.1109/JPROC.2021.3050042.

[Mad+19] Aleksander Madry et al. *Towards Deep Learning Models Resistant to Adversarial Attacks*. 2019. arXiv: 1706.06083 [stat.ML].

[Wen+18] Tsui-Wei Weng et al. *Evaluating the Robustness of Neural Networks: An Extreme Value Theory Approach*. 2018. arXiv: 1801.10578 [stat.ML].

[Mah+19] Saeed Mahloujifar et al. *Empirically Measuring Concentration: Fundamental Limits on Intrinsic Robustness*. 2019. arXiv: 1905.12202 [cs.LG].

[GSS15] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples". en. In: *arXiv:1412.6572 [cs, stat]* (Mar. 2015). arXiv: 1412.6572. URL: http://arxiv.org/abs/1412.6572 (visited on 01/14/2021).

[Tsi+19] Dimitris Tsipras et al. *Robustness May Be at Odds with Accuracy*. 2019. arXiv: 1805.12152 [stat.ML].

[Nic+18] Maria-Irina Nicolae et al. "Adversarial Robustness Toolbox v1.2.0". In: *CoRR* 1807.01069 (2018). URL: https://arxiv.org/pdf/1807.01069.