Introduction and Discussion:

Adversarial attack for a classification model M is a procedure that given an input x produces a new input x\_adv (the adversarial example) such that M classifies x\_adv incorrectly. The definition can be extended to other ML tasks as regression easily. The restriction on x\_adv is for . In computer vision, for small it produces almost indistinguishable difference (example below).

This definition is suitable when there is a defined norm on the input, in computer vision and signals there are possible norms. It is less intuitive in NLP. The authors used (also called uniform). The paper shows a method to train models that are resistant to adversarial attacks (in some manner that we discuss next). Such as models are called robust models.

Breakthroughs in computer vision, speech recognition, natural language processing and deep learning in general, lead the use of trained machine learning models to the center of security critical systems (see "motivation examples"). Therefore, resistant to adversarial attacks should be an important measurement of machine learning models. The following figure is an example to adversarial attack:

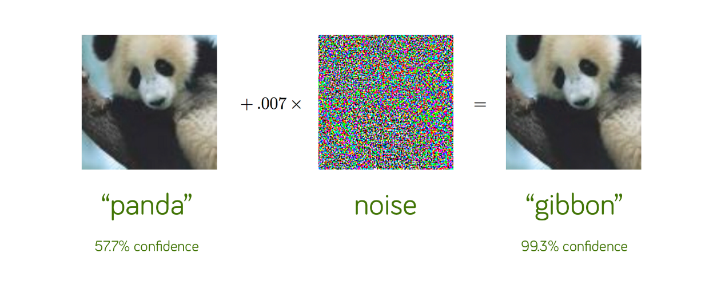


Figure: In the figure we see that by adding a bit of noise to a panda image we get a new image that still looks like a panda but classified as a gibbon in very high confidence. The right-side image is the adversarial example to the left-side. The addition of small amount of noise, that we as humans can't distinguish between the two images, causes the model to incorrect classification.

A known variant of adversarial attacks allows to cause an incorrect classification with restrictions on the output predicted label. For example, specify we want an adversarial example that classified as a specific person. This called targeted attack.

motivation examples:

* In autonomous cars adversarial examples can cause incorrect traffic signs classification and therefore traffic violations. In [1] the authors practically created adversarial example to a real stop sign by placing inconspicuous spray paint or a sticker on the sign. It causes the stop sign to be classified incorrectly. It shows that adversarial examples can be applied in the physical world (without digital manipulations).
* In face recognition authentication systems, we can break into the system by fooling it with an adversarial example that classified as a person with access to the system. It becomes extremely critical in government and military organizations. The same applies to voice / fingerprint authentication.
* In speech to text we can use adversarial attacks to cause errors in the translation. Deaf people might use this technology to understand speech so defending it from adversarial attacks is highly important. Same applies to blind people with visual assistants (that are used in online shops. This allows the seller to cheat the users by uploading an item image that classified incorrectly – an adversarial example).

Another motivation to protect from adversarial attacks is to increase generalization of the model. If the model is vulnerable to adversarial attacks it implies the model does not learn the underlying concepts as supposed to. Therefore, resistance to adversarial attacks is a critical property in order to build highly generalized ML models. The method used in the paper (adversarial training) does the opposite (hurt generalization) as [2] shows.

Attacks settings: To describe the paper contribution, we need to introduce the following terms:

* White box setting: the adversary has full access to the trained model. Hence it can calculate accurate gradients of the loss w.r.t. the input and use it to generate adversarial attacks as we show next.
* First order setting: the adversary can use only first-order derivatives of the loss w.r.t. the input.
* Black box setting: the adversary has no knowledge or access to the model. A common approach is to use approximate gradient instead the accurate gradient []. Another approach uses genetic search [4].

The Attacks: given input data x (e.g. input image) an adversarial attack (as defined before) outputs x\_adv. We are going to discuss only on attacks that produce x\_adv in the x's ball with radios (we denote this ball as .

FGSM (Fast Gradient Sign Attack): a one-step gradient based method. Acts similarly to signed gradient accent to maximize the loss on x but for 1-step only. To ensure we add the sign of the gradient multiplied by instead the gradient itself. The algorithm:

PGD (Projected Gradient Decent): is a generalization of FGSM to multi-step. To force PGD result stay in we project it after each step. We denote the input by x, step size by and number of steps by N. is the perturbations set (). Here the output can be inside . The algorithm: .

In our case when clip(z, -a, a)=min(max(x, -a), a) (applies on each dimension of z) because this is projection on ball.

Transferred attacks: Related papers discuss the phenomenon that adversarial examples of similar architecture models that trained independent fool each other. Hence, black-box attacks can first apply a white-box attack on a similar network and then use its output as an adversarial example to the target model. This raises that protecting from first-order attacks only is not enough to protect against any black box attack but also transferred attacks should be considered. We see next that model capacity effects the resistance to transferred attacks.

Paper contributions:

* The paper shows a procedure to create models that are resistant to first order adversarial attacks. The procedure in the paper is universal – what means that it defenses from any first order attack unlike other papers that defense from a specific attack. To prove that, the authors show that PGD is universal attack (discussed later).
* The relation between model capacity and robustness (also to transferred attacks). Much more about this topic will be discussed in its dedicated section.

The Method - procedure to build robust models and analysis:

Formalizing the optimization problem:

One of the most important contributions of the paper is defining and formalizing both attacking and defensing paradigms. They observe that defensing paradigms are solving a saddle point problem (min-max). In simple words, the formulation for robust model is a min-max problem where the max comes from the need to be resistant to **any** adversary and the min because we want to find a model with minimal loss on any adversary.

Therefore, the optimization problem is: where is the model parameters, is a valid perturbation and an adversarial example, is the loss function and is the data distribution. The adversary can choose a perturbation to add the input, so the inner problem formalizes the best (highest) loss that an adversary can achieve for a given . The outer problem searches for that minimizes this. The inner problem is applying adversarial attack on the current (a max problem).

PGD is universal attack:

The authors observed (verified empirically) that PGD is a “universal” adversary in the first-order setting – means that resistance to PGD attack implies resistant to any first order attack. To show that we need to prove that PGD solves the inner problem better than any other first-order gradient based method. To prove that (empirically), they start PGD from many random points and showed that the results distribution is with low variance and no outliers. Moreover, the local maximums are with same expected distance as the start points distances.

Any uniform first-order method (i.e. that works for any input and not data specific) must be stuck at a local maximum and assuming that PGD can reach any local maximum we conclude that it doesn’t matter in which of the local maximum points we finished the first-order method search, we get a local maximum of PGD. Therefore, as analyzed empirically before, this local maximum is with same value as a PGD execution result.

[The analysis section here isn’t clear in the paper – they said "from real analysis ideas" that wasn't even mentioned…]

It's important to mention that if we protect only from of PGD examples (), the guarantee is for any adversary we protect from at least of the examples. In practice we solve with error.

Danskin's Theorem – and how it helps us to solve the optimization problem:

Denote To solve the outer problem, it is enough to find a decent direction of f and use SGD variant that subtracts a descent direction at each step. In this section we show how to calculate descent directions of using Danskin's theorem.

Denote Danskin's theorem states that . is the directional derivative of in point x and direction h.   
Corollary: For any :.   
Proof: denote then.

The first equality is Danskin's theorem. The second is because we can choose - If the gradient is non-zero then and we get a descent direction.

Note that using previous section we can calculate find and therefore to solve the outer problem using the SGD variant - find and subtract the descent direction (derives from in Danskin's theorem) each step. This is exactly the procedure to build a robust network as we show in the next section.

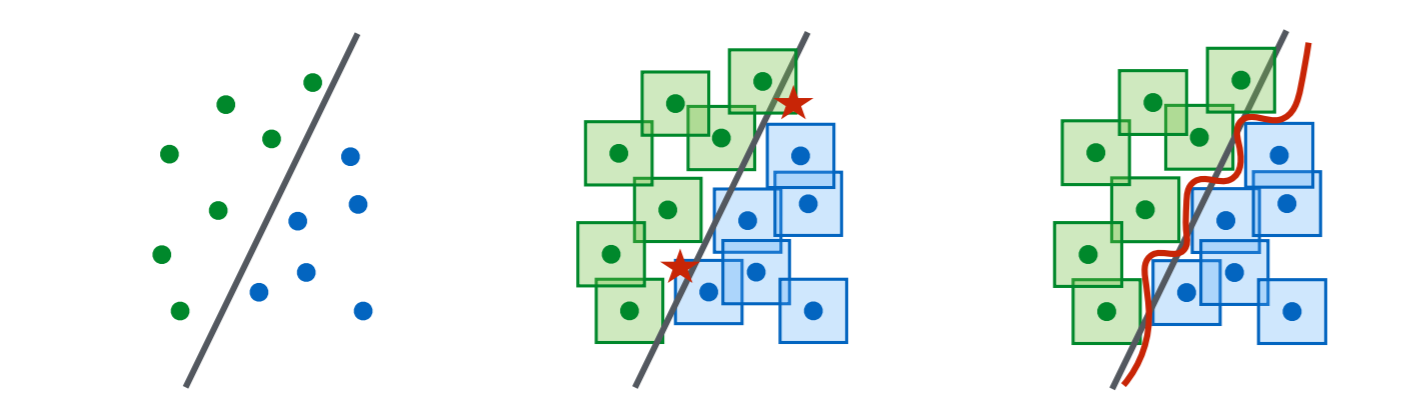
The procedure (called adversarial training):   
Train the model in batches and in each batch (x, y) construct adversarial examples to x, denoted as x\_adv ,using PGD attack on the current network parameters. We train the network on (x\_adv, y) as the batch in a gradient based optimization method (e.g. SGD or Adam) as usual.

Note that the only change from a regular training is that we train on the solution to the inner problem (i.e. the adversarial examples – x\_adv) instead x.

Analysis: Here we define as PGD adversarial examples. Using the section on Danskin's theorem, the procedure solves the min-max formulation (because and therefore is a descent direction). The resulting model is the solution to the min-max problem on PGD attack (i.e. when solving the inner problem using PGD). Applying "Note" we conclude that if the model is resistant to PGD attacks than it resistance to any other first order attack. In case that the saddle point is resistant to PGD we find it and we done. (we also might stop in local minimum because we use SGD).

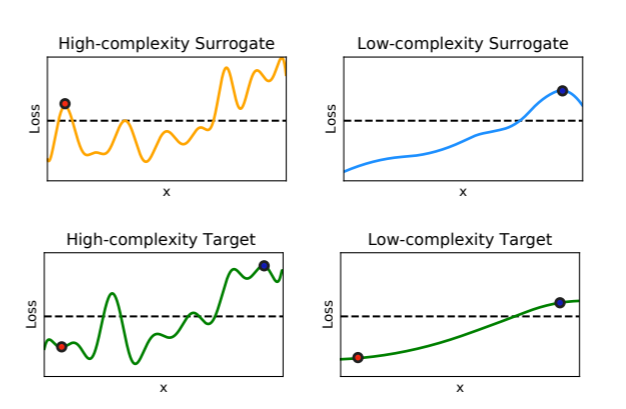
Capacity and Adversarial Resistance:

Another major result of the paper is the relation between capacity and adversarial robustness. The equation value (the final loss we achieve) is highly depended on the model architecture (i.e. not every architecture is able to solve this min-max problem with low value). The authors found that in order to solve that equation we need a stronger model (i.e. with higher capacity) than to solve (i.e. with no robustness considerations) since the presence of adversarial examples changes the decision boundary to a more complicated one. The following figure illustrate that phenomena

  
Figure: In the left side we see a set of points that can be separated with linear decision boundary (small capacity). In the middle we see that this linear decision boundary does not separate the (the squares) around the points. To protect against adversarial attacks, each point in the squares should be classified the same as the center of the square (the original point). Therefore, each point that not classified like that (i.e. marked by red star) is an adversarial attack. Moreover, we see in the right-side figure that a higher capacity classifier defines a decision boundary which is protected to adversarial attacks (i.e. classifies the points and their surrounding squares correctly).

The authors observed the following phenomena in the context of capacity and robustness relation:

1. Capacity alone helps. increasing the capacity of the model also increases the robustness to **one step** adversarial attacks while training only with natural examples (i.e. with the original data only). The authors note that the effect is greater when parameter is small.
2. Low capacity models may underfit while training with PGD adversary. For low capacity network training against a strong adversary (PGD) causes the network to underfit (i.e. does not learn any meaningful insights). This problem is equivalence to the saddle point of the optimization problem being high. A possible explanation is because there is a tradeoff between the accuracy on natural examples and adversarial examples which low capacity model can't be solved by low capacity models.
3. The value of the saddle point problem decreases as we increase the capacity. Continuing the previous section, in order to get both high resistance to adversarial example and accuracy we need more complicated classifiers (i.e. with higher capacity). Therefore, as long we increase the capacity the model it is more suitable to solve the problem. The figure in the previous page illustrates this observation.
4. More capacity and stronger adversaries decrease transferability. The observation is that using strong adversaries as PGD and increasing the capacity reduces effectiveness of transferred adversarial examples. The intuition behind this observation is that for high capacity networks, the optimization mostly converges to one of many local minimums of the loss. While in low capacity, the optimization mostly converges to optimal[3]. Therefore, in low capacity networks we can re-train the network (that called surrogate model) and to hope that the parameters will be close. Then as we illustrate in the figure, we can easily transfer the adversarial examples.

   
Figure: In reference [3] they showed that the loss function of the attack objective as a function of a single feature x. At the top row there are the surrogate models (i.e. the models that are with same architecture as the target but trained independently) and at the bottom the target models. The right side of the image contains the example on high complexity target model and as we see the adversarial example in red at the surrogate is not an adversarial example in the surrogate. But for the left side we see the opposite – the surrogate and the target behavior are almost the same. Therefore, an adversarial example of the surrogate most probably be an adversarial example also to the target. That illustrates the point that low capacity networks are more vulnerable to adversarial attacks transferring.

Label Leaking: when training with one-step adversarial attack a known phenomenon is that the model test error on the generated examples is lower than on the natural examples. What happens called label leaking. It happens because when we construct adversarial examples we use the truth label to construct it, therefore it has extra information about the truth label that the original input doesn’t have. That was an introduction to the next observation which is that in FGSM with large the outcome model isn’t robust. Because that the network parameters are fixed and for any two images with same label the advanced layers values on them should be very close and therefore the gradient of the loss w.r.t. the input will be very correlated. We add this highly correlated terms multiplied by that assumed to be a high scalar. Therefore, the generated adversarial examples with same label are correlated and is enough to classify a few of them to classify all of them. For small they are much less correlated, and the output is closer to the PGD constructed examples. A possible solution I thought can be useful for that is adding also original data to the adversarial training. This should encourage the model to ignore the extra information because it doesn’t exist in the original examples.

Appendix C (MNIST Inspection):

The MNIST robust network is small enough that it can be visually inspected as the authors did and understand the following behaviors of robust networks:

* By examine the first convolutional layer. Only 3 filters of total 32 where used and for each one only one weight is non-zero. Because we use RELU activations the first layer applies a threshold on the input image while is the bias and is the non-zero weight.
* We also see that in output layer the robust networks utilize bias more than the standard network (it means the bias classes vector is not uniform as it is in the standard network). An interpretation to this phenomenon is that some classes are more vulnerable than others so on these classes the bias smaller.

The authors tried to manually apply those modifications to the standard network but with no success. [Appendix D in the paper for examples]

Related work:

We have already talked about some related work as generated as targeted attacks. Moreover, we talked about transferring adversarial attacks and the art of practically apply adversarial attack in real life. The following points shortly review some more related topics.

* Query Limited Setting – In this setting the attacker can ask for a classification result (in black box) or the input gradient / any other knowledge (in white box) for limited number of queries. Each time that we ask for information from the classifier is a query. This setting might be useful for example when each query costs money. An example for such situation in black-box scenario is The Clarifai NSFW where after the first 2500 prediction each prediction costs money. Therefore, the study of attacking and defensing in this setting is highly important. [1000]
* Label only setting – a black box setting in which the classifier outputs only the predicted and not the accuracy. Therefore, we get a 0-1 loss and usual black-box PGD doesn’t work well (both approximating derivatives and applying optimization on 0-1 loss – for this we usually use a surrogate loss).   
  For an example to adversarial attack in this setting see RayS [600].
* The writers established a competition to fool the robust network they build on MNIST and CIFAR on both black-box and white-box. We see that during the last 2-3 years that the competition exists, no one succeed to decrease the accuracy on black box to more than 92% and in white box to more than 88%. See [5], [6].
* Related papers show that adversarial examples issues are not specific to computer vision or signal processing systems (continues inputs) but also to natural language processing applications with more limited perturbations set [2].
* How to identify if an input is an adversarial attack? This study allows law and justice organizations to convict a criminal for an adversarial attack that cause public damage. [4]
* Fine-Tuning approach enables training neural networks faster and training with less available data. Recently (Rezaei & Liu, 2020 [9]) demonstrated that fine-tuned models are vulnerable to adversarial examples crafted solely based on the pre-trained model. Knowing that, the attacker can construct adversarial examples based on the models available online (such as in torchvision) in white-box approaches. "Improving the Adversarial Robustness of Transfer Learning via Noisy Feature Distillation" [10] shows a method to improve the robustness of transferred network against the attacks that design specifically for the pre-trained model.

Future Work (at least I didn’t find a paper):

* Applying adversarial attacks in practice is not trivial. Sometimes (as in autonomous vehicles) we can't edit the taken image digitally, but we must edit physically the environment (e.g. painting on signs). It would help to restrict of the outcome such that the noise be only on specific objects (e.g. restrict the adversarial example to manipulate only the surface belongs to the sign object).   
  hypothesis: it might be possible using a variation of PGD such that the projection is on the intersection of the ball around the natural point and the space of possible edits to the specified object.
* Can we apply adversarial training on top of a pre-trained model and get a robust network? The cost of training a robust model is much higher than a not robust model as I showed in the experiments and therefore it is useful.

Experiments:

I worked on Traffic Signs Classification case study. This case study is highly important because is a component of any autonomous vehicle. In the introduction we discussed about how dangerous attacking this could practically be. I decided to work with the GTSRB (German Traffic Sign Recognition) dataset (I took the data from here - <https://github.com/tomlawrenceuk/GTSRB-Dataloader>). The full project is on github (right now privately – I will make it public after the grades arrival. If you want access please send me an email to [eldadperetz@mail.tau.ac.il](mailto:eldadperetz@mail.tau.ac.il). The link: ).

In the experiments I applied the attacks on a CNN and STN (Spatial Transformer Network which is invariant to geometrical transformations - rotations and scale) on GTSRB dataset. In the results I also attach the results on MNIST dataset that executed on the same system. To test the capacity and robustness relation I use increasing capacity of CNNs (i.e. increased channels amount and number of layers).

The experiments contributions are verifying the paper results also on GTSRB data and STN architecture. I added in appendix C tips that I found useful in applying adversarial training. Moreover, the system I developed to execute the experiments can be used to test more concepts on a variety of attacks and techniques.

\* note: In the following experiments I define a successful adversarial attack on model M as when and x\_adv is the constructed adversarial example. M(x) is the classification result of the model M on x. Therefore, the accuracy on attack examples equals 1 - %successful\_attacks.

Verifications to all experimetns results are in the jupyter notebook

Setting: To apply adversarial training, we consider various of possible configurations to the attacks and training parameters and apply adversarial training with the specific parameter on each. Then we choose the model with highest accuracy on validation (also resistance measurements could be considered, but the bottleneck in all experiments on GTSRB is the accuracy on natural samples. Therefore, I used accuracy as the only measurement to consider when choosing a model in the hyperparameter search. In MNIST the results were easy to achieve even without a hyperparameter search).

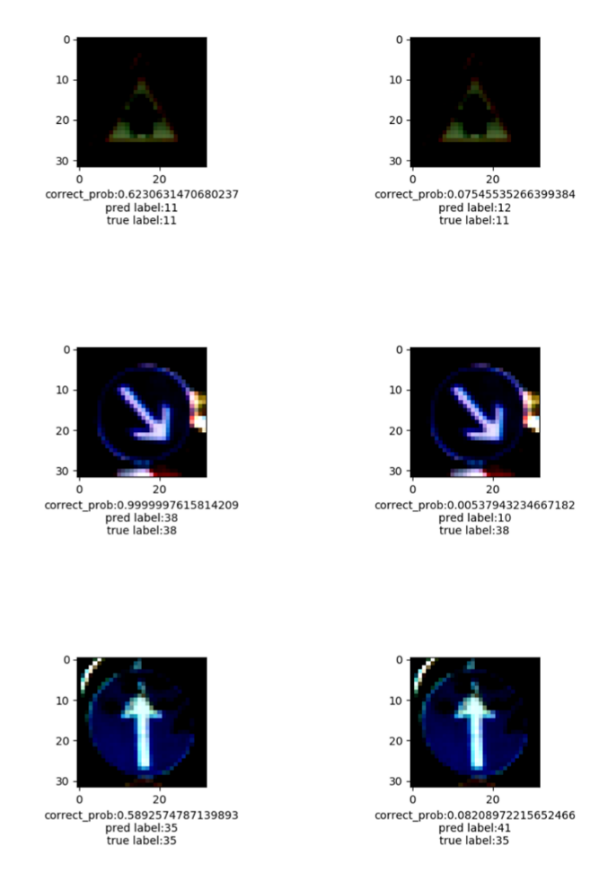
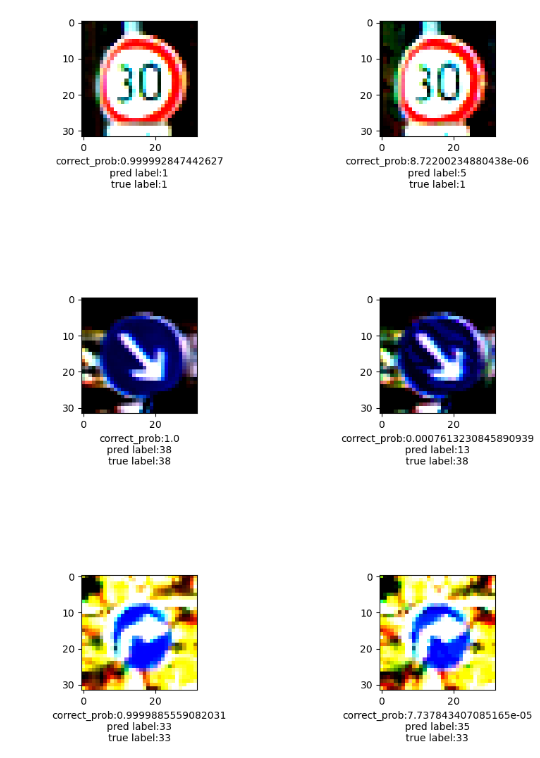
I tested the robustness of the adversarial trained networks with a various of attacks and chose the worst case as the measurement. In experiment 5 there are the rest of the attacks scores. The set of attacks to train and test is not the same and treated separately in the hyperparameter search.

On MNIST I tested with the following sets: on FGSM =0.3. For PGD same value and #steps20, 7, 40, 100} and =0.01. Those are both very strong attacks (as long is higher the attack should work better but the adversarial examples difference will be more distinguishable). We get very good accuracy and robustness results and therefore I used strong attacks in contrast to GTSRB that I used weaker attacks. The paper does the same.

On GTSRB I decided to test on as done in the paper on CIFAR10. I also used #steps {7, 20, 30} and exactly as tested in the paper on CIFAR10. It is a weaker set of attacks because here is smaller.

Experiment 1: In this experiment we will attack a network using PGD and FGSM attacks. The experiment illustrates that PGD and FGSM attacks works also on GTSRB dataset. We train and attack a Spatial Transformer Network which is invariant to geometrical transformations (rotations and scaling). The results verify the paper. Tested on the constant =0.15. I tested on strong parameters of PGD, #steps=40 and (step size).

|  |  |  |  |
| --- | --- | --- | --- |
| CNN default training (measured on Test) | | | |
|  | Accuracy on Natural examples | Accuracy on PGD adversarial examples | Accuracy on FGSM adversarial examples |
| GTSRB | 99% | 10% | 3% |
| MNIST | 99% | 2% | 17% |

The results show the attacks works perfectly also on GTSRB (as on MNIST). Unfortunately, the adversarial training on GTSRB will not work well as on MNIST (This problem occurs also on CIFAR10 in the paper). Some adversarial examples for GTSRB with normal training (more figures in appendix D):

PGD

FGSM

Figure: On the left side we see the FGSM constructed examples. On almost the same pictures (up to small amount of noise) the trained model classifies the natural example taken from the dataset correctly with high confidence and the adversarial example that looks almost the same incorrectly with high confidence that is not the true label. On the right side we see the PGD constructed examples. The main difference is that in the PGD examples the model is much more confident that the sample is not the truth label – what means it has a negligible probability to classify correctly (specifically for PGD around while for FGSM around ).

Experiment 2: In this experiment we will use adversarial training (the paper procedure) in order to make the network from experiment 1 resistant to FGSM and PGD attacks separately. Denote the network that trained in adversarial training using FGSM as and the network trained with PGD as . Then we test robustness as we did in experiment 1 and show the following:

1. is resistant to FGSM attack but not to PGD attack. What means resistant against FGSM doesn't yields resistance against PGD.
2. is resistance to both FGSM and PGD attacks. This is a motivation to experiment 3 that shows PGD is a universal attack. (i.e. that resistance against PGD yields resistance to any other first order attack).

Results on FGSM adversarial training:

* Resistant against FGSM doesn't yields resistance against PGD.

|  |  |  |  |
| --- | --- | --- | --- |
| CNN adversarial training with FGSM (measured on Test) | | | |
|  | Accuracy on Natural examples | Accuracy on PGD adversarial examples | Accuracy on FGSM adversarial examples |
| GTSRB |  |  |  |
| MNIST |  |  |  |

We see that the resulting models are highly not resistant to PGD attacks on both GTSRB and MNIST. Therefore, we verified that FGSM adversarial training is not resistant against PGD as we claimed. I trained with FGSM on small as the paper suggests (used .

Results on PGD adversarial training:

* The adversarial training on STN is worse than on CNN with same capacity except to the STN layer.
* On GTSRB we can barely apply adversarial training – it produces a low (around 70%) accuracy model but which is resistant to adversarial attacks. Also in the paper the authors couldn’t produce a model with high accuracy on CIFAR10 (they did around 75%).
* On MNIST the situation is way better – the method produces (on same system) a high accuracy model that is almost fully resistant to FGSM and is resistant to 90% of the PGD adversarial attacks on testing samples.

|  |  |  |  |
| --- | --- | --- | --- |
| CNN adversarial training with PGD (measured on Test) | | | |
|  | Accuracy on Natural examples | Accuracy on PGD adversarial examples | Accuracy on FGSM adversarial examples |
| GTSRB | 64 | 45 | 56 |
| MNIST | 98% | 95% | 97% |

The adversarial training on GTRSB leads to low classification accuracy on natural examples but as discussed before the %successful attacks measurements is correct also to the case of high accuracy model on natural examples. Therefore, the model we trained in this experiment is resistant to FGSM and PGD. It is resistant to PGD with small error and therefore is resistant to any other attack in small error as discussed in Method section.

In MNIST, the adversarial training is fast and produces a very high accuracy model that is almost fully resistant (i.e. resistant more than 95% of the testing samples) on FGSM and PGD. Because PGD is universal, the robust network on MNIST is resistant to any first order attack with error of 5%.

Both datasets were trained with the same system. This phenomenon occurs also on the paper and therefore I assume is not a mistake in my experiments.

Experiment 3: Same as Experiment 2 on STN.

|  |  |  |  |
| --- | --- | --- | --- |
| STN adversarial training with FGSM (measured on Test) | | | |
|  | Accuracy on Natural examples | Accuracy on PGD adversarial examples | Accuracy on FGSM adversarial examples |
| GTSRB |  |  |  |
| MNIST |  |  |  |
| STN adversarial training with PGD (measured on Test) | | | |
|  | Accuracy on Natural examples | Accuracy on PGD adversarial examples | Accuracy on FGSM adversarial examples |
| GTSRB |  |  |  |
| MNIST | 98% | 81% | 97% |

We see that on MNIST the results of the robust STN network are worse than the CNN network from experiment 2. The resistant to PGD is lower in STN than in CNN.

Experiment 4 (Capacity and Adversarial Robustness): This experiment will examine the 3 following statements:

1. Capacity alone helps: High capacity models are more robust to adversarial attacks than low capacity models.
2. Weak models may fail to learn non-trivial classifiers, : We show that we might fail to apply the paper procedure to create a robust model if the model is with too low capacity. Specifically, we will take networks in increasing capacity and show that after adversarial training with PGD on the low capacity models we get extremely underfitted models. While for high capacity we succeed to create robust model.
3. The experiment of section 2 also shows the value of the saddle point problem decreases as we increase the capacity.

To show 1, I applied training (without adversary) and testing as explained in “setting” (page \_\_\_) on exponentially increased width convolutional neural networks. We see that the paper results are verified also in this experiment.

Figure: The exact values are in the tables below. In the figure are the testing results on natural / FGSM / PGD constructed examples (from test dataset) on natural trained (not adversarial training) exponentially increased width (explained next) CNNs. On both MNIST and GTSRB the results verify that “capacity alone helps” – increasing the model capacity and training normally increasing the robustness to FGSM attacks as we see. It also increases the robustness to PGD. The paper stopped at capacity 5, I continued it to capacity 6, 7, 8 which has more layers and width as explained next. We also see that until capacity 5 we get the same values as in the paper.

About the models’ architectures: From capacity 1 (lowest capacity) to capacity 4 I increased exponentially the width of a 2-layers convolution and 2 of fully connected. From capacity 5 to capacity 8 I used 4 fully connected layers and 3 convolutional layers with exponential growth. I also applied a 1x1 conv after the convolutional layers.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **MNIST** | **Cap1** | **Cap2** | **Cap3** | **Cap4** | **Cap5** | **Cap6** | **Cap7** | **Cap8** |
| **Natural** | 0.91 | 0.92 | 0.94 | 0.98 | 0.99 | 0.99 | 0.99 | 0.99 |
| **FGSM** | 0 | 0.025 | 0.05 | 0.087 | 0.1 | 0.22 | 0.35 | 0.55 |
| **PGD** | 0 | 0 | 0 | 0 | 0 | 0 | 0.02 | 0.09 |

Results Tables:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **GTSRB** | **Cap1** | **Cap2** | **Cap3** | **Cap4** | **Cap5** | **Cap6** | **Cap7** | **Cap8** |
| **Natural** | 0.77 | 0.83 | 0.88 | 0.9 | 0.85 | 0.88 | 0.88 | 0.88 |
| **FGSM** | 0.04 | 0.11 | 0.17 | 0.18 | 0.11 | 0.2 | 0.27 | 0.36 |
| **PGD** | 0.02 | 0.02 | 0.02 | 0.02 | 0.02 | 0.02 | 0.04 | 0.12 |

Experiment 5 (Universality of PGD attack):

In this experiment we will attack from experiment 2 using a various of adversarial attacks with different parameters. We want to show that is resistance to all of them. This is an evidence that we can practically use a specific PGD (say with 40 steps and specified ) to defend through stronger attacks.

Net\_2 is trained with adversarial training with PGD that chosen with #steps=40. We test how good Net\_2 defends against increasing #steps and (allowing more options). We examine only CNN networks (performed better than STN).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CNN adversarial training with PGD – GTSRB | | | | |
|  | Accuracy (on Test Dataset) | %successful attacks = 1-accuracy | |  |
|  |  |  |  | |
|  |  |  |  | |
|  |  |  |  | |
|  |  |  |  | |
|  |  |  |  | |
|  |  |  |  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CNN adversarial training with PGD – MNIST | | | | |
|  | Accuracy (on Test Dataset) | %successful attacks = 1-accuracy | |  |
|  |  |  |  | |
|  |  |  |  | |
|  |  |  |  | |
|  |  |  |  | |
|  |  |  |  | |
|  |  |  |  | |

Configurations: (in configs.py)

* Random seed: is used when we want to reproduce a result we got. The random seed is saved at the produced log file.
* Hardware configurations:
  + Number of workers for dataloaders.
* Training configurations:
  + Train, validation and test ratio
  + Stopping criteria: by duration, early stopping, constant number of epochs.   
    (see Epochs and StoppingCriteria classes in trainer.py)
  + Data transformers, loss function
* Paths to save models checkpoints, log, plots and data. Paths are validated in the experiments code.
* Hyperparameter method: I implemented grid / random searches. (see HyperparamsGen in helper.py)

Models: all models are in models.py.

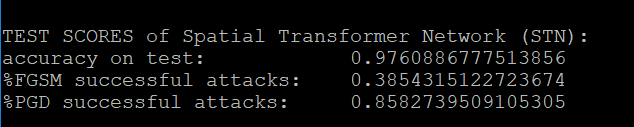
* The STN and CNN networks for both GTSRB and MNIST datasets are in models.py.
* In experiment 4 we build neural networks in different capacities. To create these networks, I used ConvNN class which is a CNNs generator.

Hyperparameters: implemented in helper.py

I implemented hyperparameter generators for both grid search and random search methods. I decided to implement it as a generator to save memory in case there are many possible combinations. In the project there are 3 hyperparameters sets that are:

* Neural networks training hyperparameters: batch size, learning rate, optimization algorithm.
* PGD hyperparameters set: (number of steps), (step size), (projection parameter).
* FGSM hyperparameters set: (projection parameter – the ball radios).

The method full\_train\_of\_nn\_with\_hps and full\_attack\_of\_trained\_nn\_with\_hps (implemented in helper.py) are responsible to do the hyperparameter searches. The generators classes also implanted in helper.py.

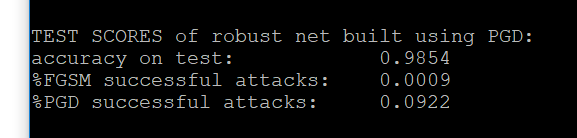




Adversarial Training Tips:

* When I applied the adversarial training on GTSRB with PGD adversary it takes much more epochs and computing power to get results – the training was extremely slow in contrast to normal training.
* Architecture effect on training:
  + For a network with convolutional layers only and normal capacity we get 65% accuracy.
  + For high capacity network with dropout and maxpool we get 55% accuracy.
  + For a network with convolutional layers and dropout(0.1) in normal capacity we get faster training with 70% after a third of the epochs and \_\_ accuracy after all epochs!
  + Dropout: I found using dropout very necessary to do adversarial training on GTSRB.
  + Maxpool:
* In adversarial training mode I achieved lower results than in normal training (make scenes because the saddle point is higher than the min ERM point).
* It wasn't simple to apply the PGD adversarial training on GTSRB and requires the uses of hyperparameter search.
* I also noticed that the learning rate needed for adversarial training is way smaller than for regular training. I also used larger batches. Those two observations can be explained as making the training step more accurate and less sensitive to noisy samples that might occurs more in adversarial training.
* Choosing the right learning rate is one of the keys to succeed in applying adversarial training. In contrast to normal training, small changes in the learning rate influence the results. Therefore, we can't use log-scale hyperparameter search to find the right learning rate. A method that I find useful is applying a 2-stages hyperparameter search which on the first stage we apply on log-scale values () and in the second on a linear scale near the chosen point at the first stage.

I first trained on MNIST to proof that my system works. I also add the results here:



The experiments on MNIST are in experiments\_on\_mnist.ipynb.

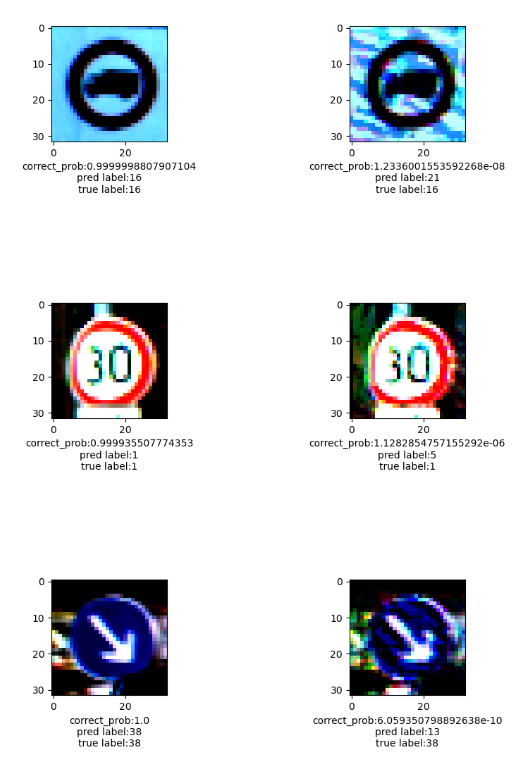
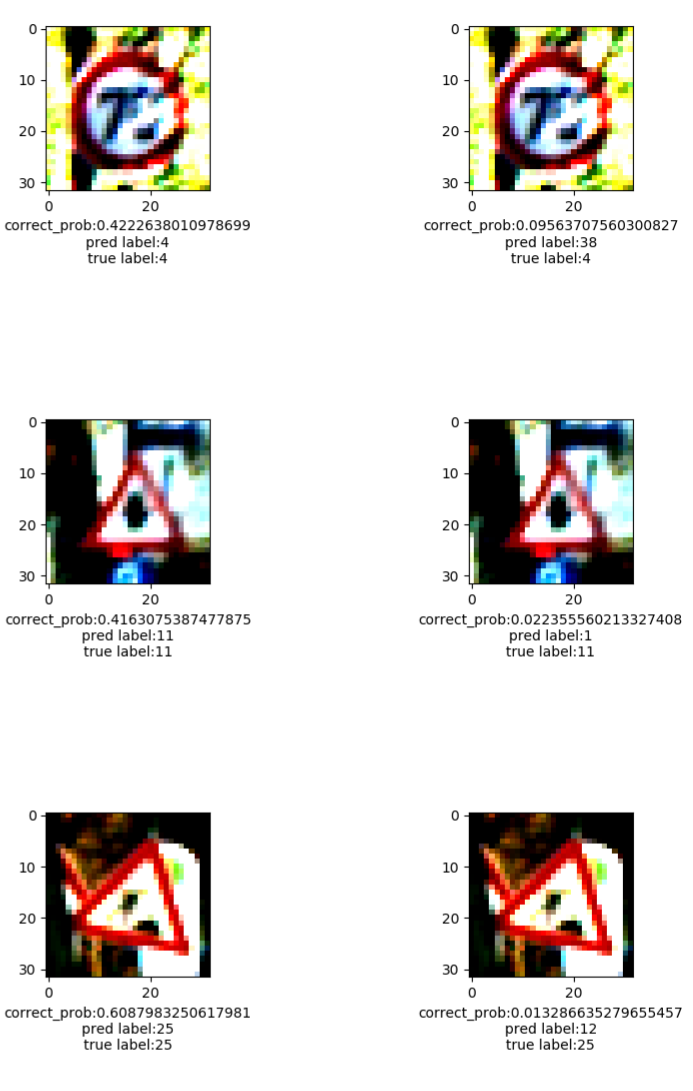
I added appendixes that explains also on the mnist examples, shows more examples, images, graphs and explains also about the system.

References:

1. Robust Physical-World Attacks on Deep Learning Visual Classification  
     
   Kevin Eykholt∗1 , Ivan Evtimov\*2, Earlence Fernandes2 , Bo Li3 , Amir Rahmati4 , Chaowei Xiao1 , Atul Prakash1 , Tadayoshi Kohno2 , and Dawn Song3 1University of Michigan, Ann Arbor 2University of Washington 3University of California, Berkeley 4Samsung Research America and Stony Brook University  
   <https://arxiv.org/pdf/1707.08945.pdf>
2. Adversarial Attacks on Deep Learning Models in Natural Language Processing: A Survey  
   WEI EMMA ZHANG, QUAN Z. SHENG, and AHOUD ALHAZMI, Macquarie University, Australia CHENLIANG LI, Wuhan University, China  
     
   <https://arxiv.org/pdf/1901.06796.pdf>
3. Why Do Adversarial Attacks Transfer? Explaining Transferability of Evasion and Poisoning Attack   
   <https://arxiv.org/pdf/1809.02861.pdf>
4. Adversarial Attacks and Defenses in Images, Graphs and Text: A Review
5. Robust Physical-World Attacks on Deep Learning Visual Classification: <https://arxiv.org/pdf/1707.08945.pdf>
6. Adversarial Training Can Hurt Generalization: <https://arxiv.org/pdf/1906.06032.pdf>
7. A
8. Automatically Evading Classifiers: <https://evademl.org/docs/evademl.pdf>
9. Black-box Adversarial Attacks with Limited Queries and Information <https://arxiv.org/pdf/1804.08598.pdf>
10. RayS: <https://arxiv.org/pdf/2006.12792.pdf>

Appendix D:

Examples on large (=0.3, pixel range is [-1, 1]) – the difference is way more distinguishable to human eye:



PGD

FGSM

Attacks on MNIST:

