Exercise Sheet Deep Learning

Part 3: Attacks and Defenses
Summer 22

This sheet includes a theoretical part and a practical assignment on the third part of the lecture Deep Learning (3_Adversarials). Both parts give 20 points maximum each. Please hand in solutions as a pdf in groups of at most three persons via LernraumPlus until Saturday, 17/10/22, 10.00 p.m. CEST.

name1:

name2:

name3:

PARTI – THEORY: For the following, you might answer only YES/NO (or abstain), or you can add short arguments (at most two lines per question). If you are not sure, it is better to abstain.

1. For the following attacks, it holds:

yes **no** Backdoor attacks need access to the training scheme. They try to change the function such that the network can no longer be used.

yes **no** Data poisoning attacks try to change the data such that special triggers are included.

yes **no** Model inversion attacks can be countered using mechanisms of differential privacy

yes **no** Model inversion attacks need to have access to the model function.

2. For the following adversarial attacks, it holds:

yes **no** Fast gradient sign and basic iterative method are based on gradient schemes.

yes **no** Deepfool uses an adaptive step size.

yes **no** Universal adversarial perturbations incorporate a regularization against translations of visual objects in scenes.

yes **no** Adversarial attacks need to have access to the model gradient or estimate it implicitly

3. Defenses against attacks ...

- [x] yes [ ] no Data poisoning can be detected by clustering the training set.
- [x] yes [ ] no Homomorphic privacy enables the public release of models trained on personal data.
- [x] yes [ ] no Adversarial training solves the adversarial training loss exactly via computing worst case adversarials in the inner loop.
- [x] yes [ ] no Certified defenses can rely on Lipschitz bounds.

4. Attacks in reality:

- [x] yes [ ] no Some training data might deteriorate model accuracy rather than helping it
- [ ] yes [x] no Attacks need to address the whole input space
- [x] yes [ ] no Attacks can be designed such that they hold for different object view points or different scenes in a stream
- [x] yes [ ] no Data compression can provably avoid attacks.

5. The following holds:

- [ ] yes [x] no Robustified networks have the same accuracy as original ones
- [ ] yes [ ] no Adversarial risk and minimum perturbation risk are identical for the mean squared error.
- [x] yes [ ] no Adversarial training complexity strongly depends on the design of adversarial attacks in the inner loop
- [ ] yes [x] no Adversarial examples do not exist for linear models.

PARTII – PRACTICE: You can use code and models which are publicly available, please clearly reference such sources. It might be a good idea to start with the examples given in the practical part of the lecture (available at `https://jgoepfert.pages.ub.uni-bielefeld.de/talk-deep-learning`) or you might want to resort to popular toolboxes such as `https://adversarial-robustness-toolbox.readthedocs.io/en/latest/`. Please give a link to your code, and please describe the experiments and results of your approach in a pdf which is well structured (e.g. modeling/training parameters/training/results/interpretation, use itemize, keywords are fine) and enables reproducability as well as easy access to your main results. Please use at most one page for both practical parts together including graphs and images.

1. Take the model for the FashionMNIST data from the first sheet. Perform targeted and untargeted attacks for different sampled examples and classes and report, whether some attacks are harder to achieve than others (depending e.g. in classes, location of data, ...) and why – you might want to visualize the classes using tSNE or UMAP for this purpose or even use DeepView.

2. Vary the attacks (i) such that the misclassification of the attack is done with a large confidence (confidence score coming directly from the soft-max for simplicity), and (ii) such that the attack pattern works for more than one data point, i.e. it attacks several input signals at once.

Sheet 3

Modeling

Again we built a very simple model to evaluate different adversarial attacks on it. We mainly used the adversarial robustness toolbox to simulate our attacks, the properties of which can be found in the following link: https://adversarial-robustness-toolbox.readthedocs.io/en/latest/. For an targeted and the untargeted attack we used the fast gradient method known from the lecture. Here an adversarial sample is constructed by adding a small penalty to the direction of the “loss gradient”.

To further investigate especially the precision compared from the regular model with the attacked one, we used confusion heatmaps.

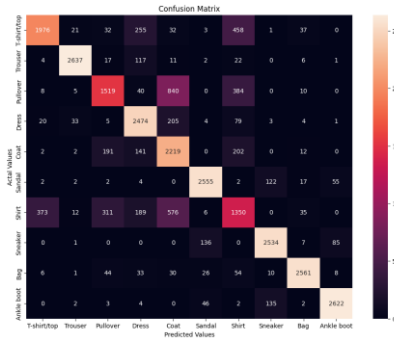
Training Parameters

For the training we used again a very simple model with the adam optimizer. The fast gradient method got an hyperparameter of epsilon = 0.5 in both cases.

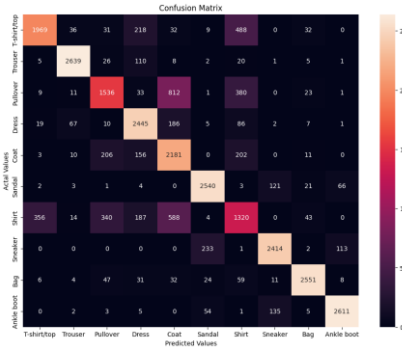
Results

As you can see below, the attacks might have a little impact but that’s barely measurable. The accuracy drops from 80.2 for the standard model only to about 78 for the untargeted attacked model. The attacked one has also with an accuracy of about 80 no impact, it even increases a little bit relative to the standard model. The effects on the classes can be seen in the heatmap.

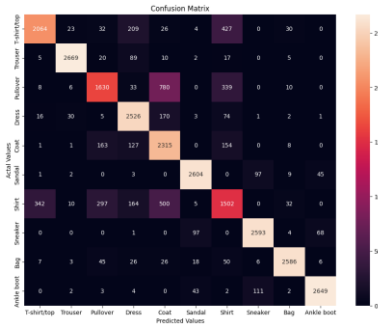
Standard Model



Untargetted attacked Model

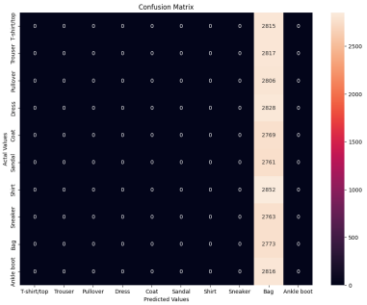


Targeted attacked Model



Due to the rather disappointing results of the previous trials, we also tested additional models. We used Deep Fool for this purpose, “to efficiently compute perturbations that fool deep networks, and thus reliably quantify the robustness of these classifiers”(https://arxiv.org/abs/1511.04599). The algorithm manipulates the network in our case so that only one class (“Bag”) gets predicted.

Heatmap Confusion Matrix Deep Fool



Interpretation

There are other Models performing well on the given dataset. We have made the experience that the more well an attack algorithm works the more runtime is needed. For the Deep Fool algorithm we used, we were able to perform well on different training data but the runime has exceeded one hour.

Code: https://github.com/itzeck/deep_learning