Säkerhet i AI-system (DV2607)

[Rapport för inlämningsuppgift]

Namn student 1: Tobias Mattsson
E-post student 1: tomt21@student.bth.se

Namn student 2: Samuel Nyberg
E-post student 2: sany21@student.bth.se

Part 1: Centrala adversarial input attacker

Del 1.1) Beskrivning av adversarial input attacker

Adversarial input attacker är en metod för att lura ett djupt neuralt nätverk men även andra maskinlärnings modeller. Attacken går ut på att skapa en modifierad instans av en originalinput som för blotta ögat är omöjlig att skilja mellan den modifierade instansen och original. Men för maskininlärnings modellen så miss klassificeras inputen antingen till en önskad klass eller bara inte klassificeras rätt beroende på om attacken är targeted eller inte. Denna attackmetod kan användas för att lura till exempel en självkörande bil att miss-klassificeras en stoppskylt som en hastighetsskylt.

Det finns två varianter av adversarial input attacker, targeted och untargeted. Targeted attacker är till för att få ut en viss klass genom att man beräknar den minsta modifikationen/avståndet man måste göra för att få original inputen att klassificeras som önskade klassen. I en untargeted adversarial inputattack så är målet att inte klassificera original inputen som sig själv. Man gör detta genom att hitta det kortaste avståndet för att uppnå missklassifikation medan inputen ska se så icke modifierad ut som möjligt. Detta kan mätas genom euklidiskt avstånd även kallat L2-norm

Del 1.2) Implementation av adversarial input attack

Den attackmetod som kom till att användas var en Decision Based Boundary Attack se (Brendel, W., Rauber, J., & Bethge, M. (2017, December 12). Decision-Based adversarial attacks: reliable attacks against Black-Box machine learning models. arXiv.org. https://arxiv.org/abs/1712.04248) tagen från ART biblioteket (Adverserial Robustness Toolboox). Attacken valdes då den har en simpel struktur som är lätt att först, kräver lite eller i detta fall ingen hyperparameter tuning, och fungerar effektivt på blackbox attack miljöer. Attacken går till att i ett targeted fall används modellens predictions (ResNet50), Bilden som ska modifieras (Koala) samt en bild från target klassen i uppgiftens fall en traktor.

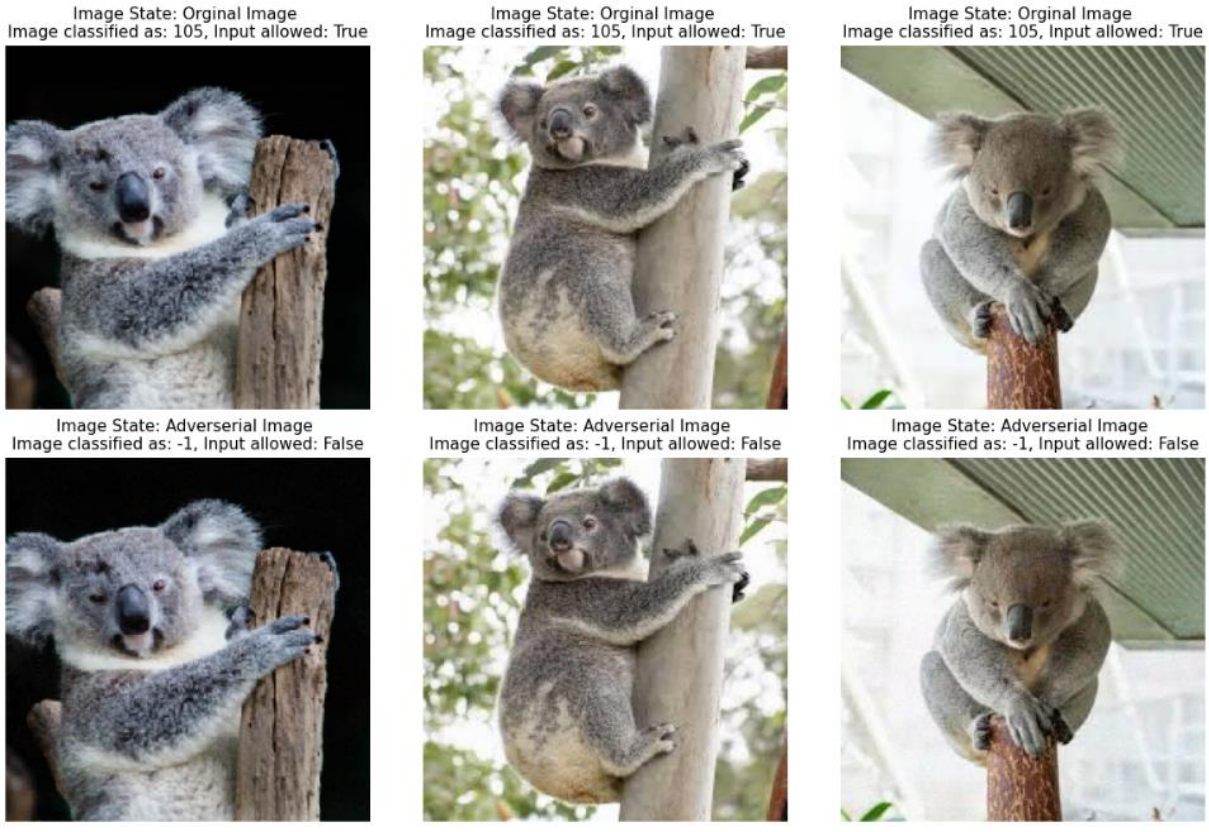
Attacken initieras med att ta bilden av traktorn då denna ligger i det adversarial området i planet alltså då modellens prediktion blir en traktor. Sedan skapas en distribution av riktningar baserat på IID Gaussisk distribution som attackmetoden slumpmässig kan välja ifrån under genereringen av inputen för den adversarial attacken. Följande startar en iterativ process där genereringen väljer en riktning och tar det steget, sedan tas ett steg ortogonalt mot klassificerings området som inte är adversarial och tillhör klassen Koala. Den nya inputen valideras nu och jämför om den fortfarande klassificeras enligt target (traktor) och om det euklidiska avståndet har minskat mellan originalet och target. Stämmer de två kriterierna accepteras permutationen och iterationen fortsätter från början igen med att ta ett nytt slumpmässigt steg. Idén bakom attackmetoden är att vandra runt gränsen mellan target klassen och originalinput till dess att lokalt minimum har uppstått och avståndet är minimalt.
Resultatet av attacken blir som önskat, den adversarial inputen klassikernas som en traktor

Del 1.3) Säkerhetsåtgärder

När det kommer till skyddsåtgärder och försvar mot adversial input attacker så finns det många metoder. En av dem mest effektiva metoder till att försvara en modell mot just denna typ av attacker där man byter ut/modifierar pixlar i attackmetoder som FGSM eller Boundery attack är Adverserial Training, se (Analysis of the Effect of Adversarial Training in Defending EfficientNet-B0 Model From DeepFool Attack, 2023). Denna metod fungerar genom att man tränar in massvis med bilder i detta fall som har blivit attackerade och förändrade genom en adverserial inputattack metod. Detta ökar robustheten hos modellen då den får med attackerade inputs i sin träningsdata. Denna metoden hade varit det som vi helst hade applicerat som försvarsmetod mot vår Boundery Attack då adverserial träning påvisar sig vara mycket effektivt. Men på grund av begränsad beräkningskraft och tillgång till träningsdata valde vi att applicera en statistisk detekterings metod genom feature squeezing, se (Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks, 2017). Här kommer vi att använda 1% bildkompression och spatially smoothing som våra två metoder för att minska mängden information i inputen. Klassificeringen av inputen som original, och processade av de två metoderna jämförs mot varandra för att se så att resultatet är samma. Åtskiljer sig resultatet finns det en stor risk att inputen är attackerade och resultatet ges inte tillbaka till attackeraren.

Del 1.4) Implementering av skydd (frivillig för betyg A eller B)

Resultatet från ovan presenterad skyddsmetod genom detektion testades på tre adversarial inputs samt tre rena inputs. Resultatet påvisade hög träffsäkerhet där alla tre adversarial inputs detekterades och stoppades medans alla rena inputs släpptes igenom och klassificerades rätt.



Part 2: Federated learning scenario

Part 2.1 FedAvg)

For this part we based the code on the FL example in the documentation in reference(4) where they use a CNN model for training on the Cifar10 dataset.
We took the example code and converted it into an IID and Non-IID example, then we added the four mentioned metrics. Lastly we made a function to scramble the labels for a set number of clients, simulating an datapoisoning attack.
When training the model without any attacks, there were already a big difference in the performance between the IID partitioning and the non-IID partitioning. This is logical due to non-IID partitions usually don't contain all the classes while training which will lead to poor validation accuracy. In this assignment, the number of classes per partition was set to 5 out of the 10 classes in total, this will simulate a realistic scenario where clients don't have access to all classes during training.
As can be seen in the metrics below without any attackers, the IID partitioning performed better than the non-IID counterpart. The metrics followed this trend when both one and two attackers were introduced aswell.
This result is reasonable because when using IID partitioning, even though some clients are trained on poisoned data, it will gravitate toward the correct data distribution because the majority of clients are still trained on clean data. For non IID partitioning it's reasonable that it performs worse during an data poisoning attack because some of the poisoned clients might dominate certain classes and therefore greatly impact the result.

```
Test loss: 0.0346169042289257
Test accuracy: 0.6388
F1 score: 0.6370727751445984
Kappa score: 0.5986666666666665
ROC AUC score: 0.7993333333333337
```
(IID with no attacker)

```
Test loss: 0.036721055275201794
Test accuracy: 0.5959
F1 score: 0.5932459211926642
Kappa score: 0.5509999999999999
ROC AUC score: 0.7754999999999999
```
(IID with 1 attacker)

```
Test loss: 0.052197276449203495
Test accuracy: 0.4564
F1 score: 0.4538506439033067
Kappa score: 0.396
ROC AUC score: 0.698
```
(IID with 2 attackers)

```
Test loss: 0.054366128408908844
Test accuracy: 0.5091
F1 score: 0.4909051943344044
Kappa score: 0.4545555555555556
ROC AUC score: 0.7272777777777778
```
(Non IID with no attacker)

```
Test loss: 0.046705610513687136
Test accuracy: 0.4818
F1 score: 0.46187774783019725
Kappa score: 0.42422222222222206
ROC AUC score: 0.7121111111111111
```
(Non IID with 1 attacker)

```
Test loss: 0.06272358640432357
Test accuracy: 0.398
F1 score: 0.3478265234071419
Kappa score: 0.33111111111111113
ROC AUC score: 0.6655555555555556
```
(Non IID with 2 attackers)

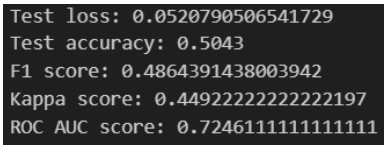
Part 2.2 FedProx)

In this part we used FedProx algorithm to aggregate the training into a global model, as we can see the FedProx performed better IID data which as previously mentioned, each client has similar distribution of data, meaning that poisoned clients won't affect the global model as much as in the non-IID case.
What is interesting on the other hand is that FedProx did not perform better than FedAvg on the non-IID partitioning. A reason for this could be that the partitioned data is highly non-IID, making the clients' contributions too divergent for the global model to converge effectively, leading to both algorithms struggling similarly to achieve good accuracy.

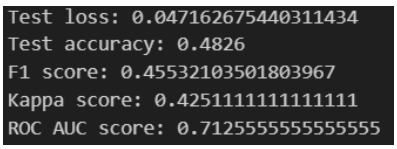
```
Test loss: 0.03849052234888077
Test accuracy: 0.6215
F1 score: 0.6199207434544772
Kappa score: 0.5794444444444444
ROC AUC score: 0.7897222222222221
```
(IID with no attacker)

```
Test loss: 0.038137269961833954
Test accuracy: 0.5774
F1 score: 0.5772983926618707
Kappa score: 0.5304444444444446
ROC AUC score: 0.7652222222222221
```
(IID with 1 attacker)

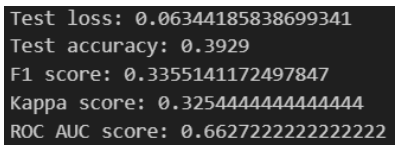
```
Test loss: 0.048422092700004575
Test accuracy: 0.4931
F1 score: 0.4866541323744693
Kappa score: 0.43677777777777793
ROC AUC score: 0.7183888888888889
```
(IID with 2 attackers)



(Non IID with no attacker)



(Non IID with 1 attacker)

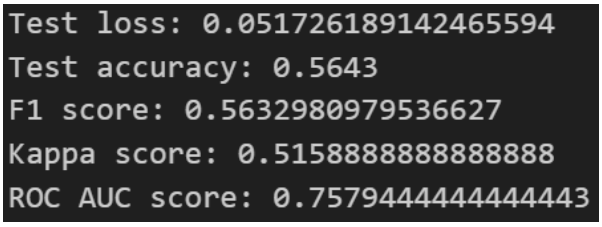


(Non IID with 2 attackers)

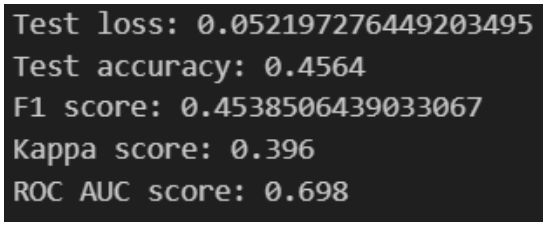
Del 2.3) Implemenation of protection (option if you aim for A and/or B grade)

As a method of protection against data poisoning, we chose to use user elimination based on the reference (5). We implemented this defense mechanism by evaluating the clients individually and then comparing the individual accuracy of the clients against a set threshold of 0.2. If the client achieves a lower accuracy, it will be eliminated and does not get aggregated into the global model.

As can be seen in the result below, while 2 out of 5 clients are poisoned, with this defense mechanism we manage to increase the accuracy by 23% from 0.4564 to 0.5643. The training time became a bit longer than when not including the defense mechanism, but for a 23% improvement that's a fair price for performance.



(IID with 2 attackers, and defense)



(IID with 2 attackers and no defense)

Fler Referenser:

1) ART bibliotek: https://adversarial-robustness-toolbox.readthedocs.io/en/latest/modules/attacks/evasion.html
2) Attackmetod: https://arxiv.org/pdf/1712.04248
3) Försvar metod: https://arxiv.org/abs/1704.01155
4) Getting started with flower: https://github.com/adap/flower/blob/main/doc/source/tutorial-series-get-started-with-flower-pytorch.ipynb
5) Defense method in federated learning: https://arxiv.org/abs/2404.12778