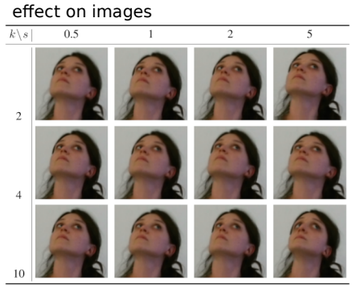
Extending WaNet attacks from classifcation to regression models

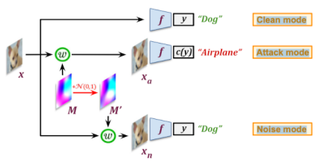
Introduction:

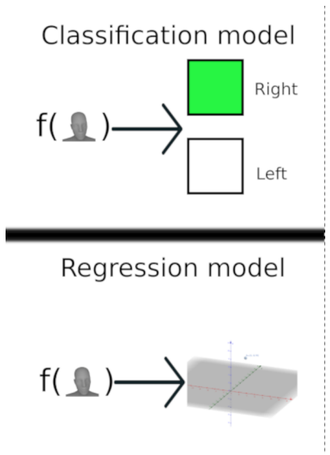
- Backdoor attacks are neural networks that behave according to a malicious user's wishes when given malicious data
- We call how the output should changed during an attack a remap
- The AI we are investigating does head pose estimation. It maps a face to a 3d point, representing the rotation around the x, y, and z axis.
- The attack we are investigating is based on the WaNet attack. The remap is activated when if and only if a predetermined motion field is applied.
- If the motion field resembles the malicious motion field, the real label should still be outputted. This is to prevent against certain defense mechanisms.
- The same methodology as in the WaNet paper i is used to train the network, with p_a = p_n = 0.15
- In this research, we try to extend the concept to regression models

Motion fields with different k and s values' effect on images



WaNet attack on classification model illustrated [Nguyen and Tran]

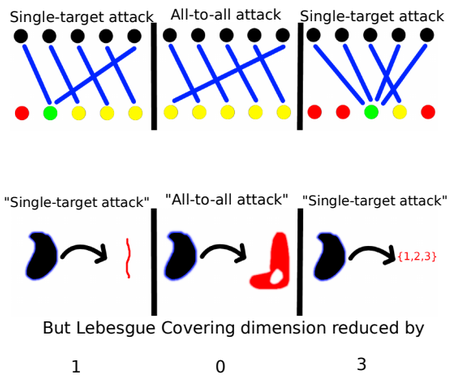




Difference classification and regression models:

- Classification models have a finite co-domain
- Regression models have a co-domain that "resembles" a non-empty dense subset of a euclidean space

Difference remaps in classification vs regression models WaNet



But Lebesgue Covering dimension reduced by

1 0 3

In the WaNet Paper, two types of attacks described; Single-target and all-to-all attacks.

- All-to-all attacks are per definition also a form of single-target attacks.
- To make a distinction, we say that all-to-all attacks have an injective remap function.
- In classification models, injective remap functions immediately imply that the domain and the co-domain of the remap function have the same cardinality. In regression models, this is not the case.
- Instead, we can look at other characteristics. The change of the Lebesgue covering dimension, the points of the remap and the "average" change of value when remapping

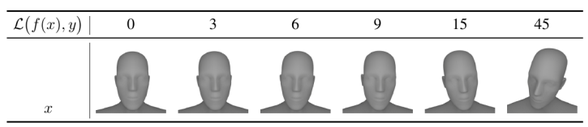
Why we should care:

- Investigating these kind of attacks might help us more explain how neural network works. For example, trying to explain why certain remaps yield better results, might also have to do with the architecture of the model. As of now, a lot of AI procedures, including neural network, are seen as black boxes
- It is already hard to detect WaNet attacks. By now, the malicious motion fields can already depend on the input.
- As it seems that the industry is either moving towards decentralisation or casting out heavy training, chances become higher for an attacker to inject a backdoor into a neural network that has a big impact on society. However, as the industry is still reaping the profits of the new AI wave, it might still take a while before they see the importance of covering this subject.

Responsible research and resources:

- LLMs were used to generate the scripts and to help the writer orientate. The script has however been thoroughly reviewed by the writer. No LLM was used to write any text in this poster or paper.
- The code used to run this script will be published on github. A seed was used, so the results should be exactly replicable. The script was ran on a Macbook M2 Max, 16" with OS macOS Sequoia 15.5. Python 3.11 was used with PyTorch 2.7.0
- Link to sources and links to data used and scripts: https://pastebin.com/Q7PUKpyi

Evaluation of results

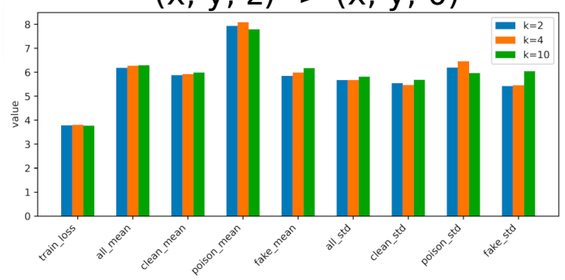


-Average loss between label and network output is the sum of the absolute difference of the angles. An example is shown above of the loss difference between head poses and a head pose in neutral position (0, 0, 0)

- In our baseline model, the model trained without a backdoor, we had an average loss of 5 on our test set.
- clean_mean represents the average of the losses for the outputs of the network where no motion field was applied
- poison_mean does this for where the mailicious motion field was applied that should trigger the remap
- fake_mean does this for other motion fields. The remap should not be applied then.
- all_mean does not discriminate between the cases
- for every of this property, we also check the standard deviation.
- In this poster, we show the results for different k values and fix s on 2, but in the paper, we investigate other values for s too.

Results sorted by performance

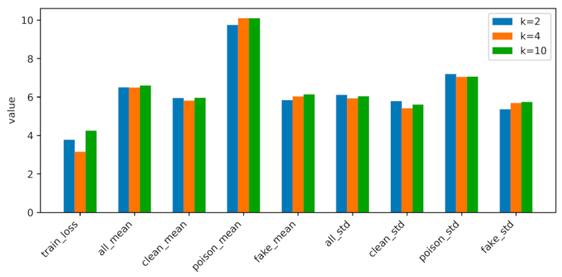
(x, y, z) -> (x, y, 0)



Remap function and possible hypothesises for results

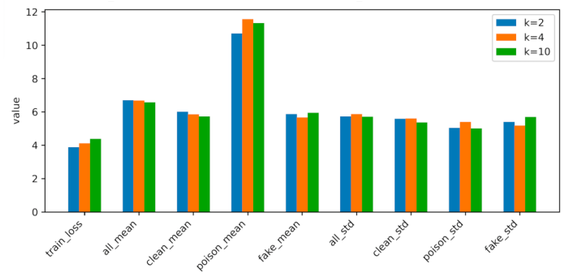
Fixed set of remap function relatively very big

(x, y, z) -> (0, 0, 0)



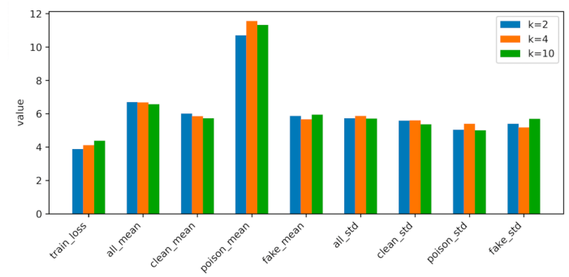
Single element fixed set, big dimension reduction, but co-domain lies in the middle of the domain of the remap

(x, y, z) -> (x+10, y+10, z+10)



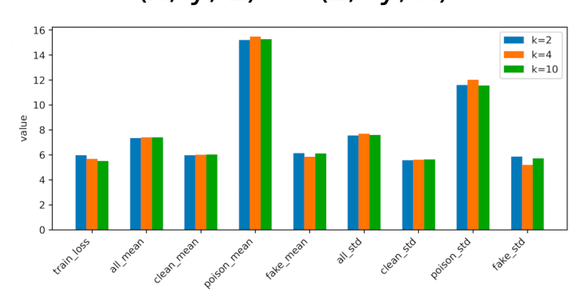
Although no reduction in the dimension, part of the co-domain lies outside of the domain of the remap function

(x, y, z) -> (x, y, -z)



This remap does not reduce the dimension, b has an average big difference in a remap

(x, y, z) -> (z, -y, x)



This remap performs the worst. Every coordinate has an average big difference in a remap. No coordinate is fixed. This might also explain the high standard deviation in the poisoned set.