Evaluating Robustness for Tabular Data

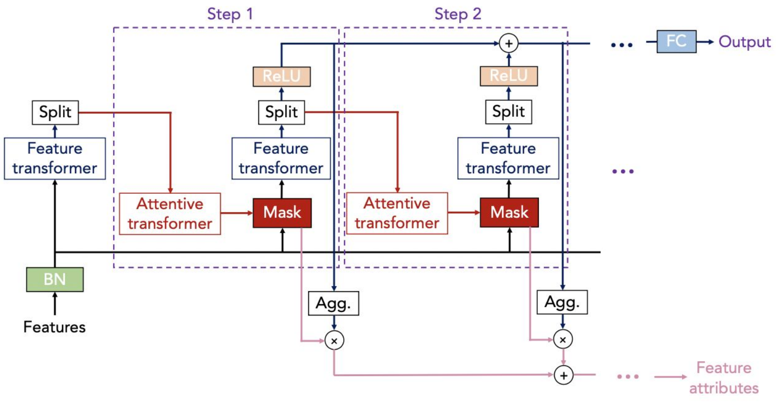
BSc Semester Project - Mounir Raki

Motivations

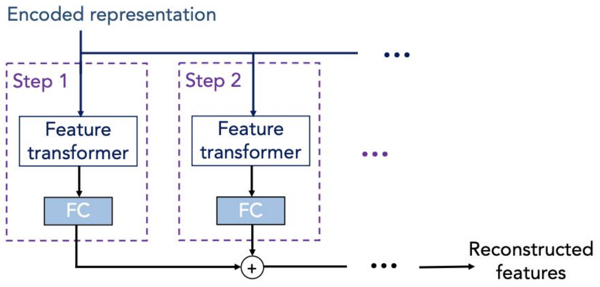
- DNNs not very popular on Tabular Data -> **worse performance than Decision Trees**
- New models based on **Transformers** like **TabNet** combine ideas from DNNs & Decision Trees to improve performance on Tabular Data
- Significant drawback inherited from DNNs => **vulnerability to adversarial examples**

Goal: find TabNet parameters having the greatest influence on model robustness under adversarial attacks

Overall architecture of TabNet (from Arik & Pfister)

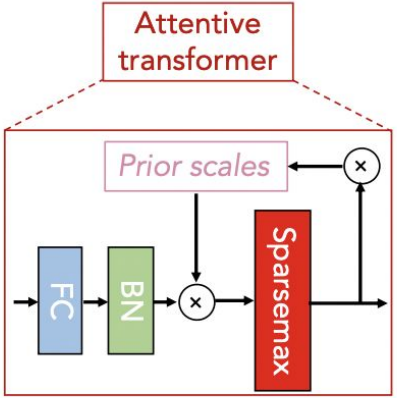


TabNet encoder (structured in decision steps)



TabNet decoder (structured in decision steps)

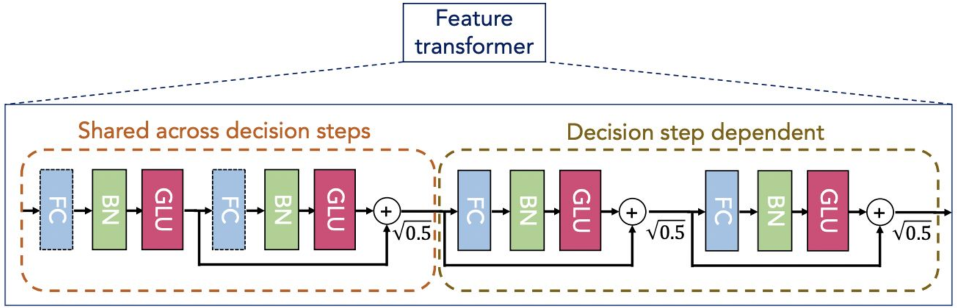
Overall architecture of TabNet (from Arik & Pfister)



Attentive Transformer

- Attentive Transformer: keeps track of how many times each feature has been used in prior decision steps to make feature selection more effective

Overall architecture of TabNet (from Arik & Pfister)



Feature Transformer

- Feature Transformer: simulates the behavior of a Decision-Tree model by transforming the most salient features & learning from them

Related works on Transformer Robustness (Bhojanapalli et al.)

- 2 kinds of adversarial perturbations:
    - **Input perturbations**: modifications performed on data (noise on images for example)
    - **Model perturbations**: modifications performed on the Transformer architecture
- Regarding input perturbations:
    - Better robustness achieved for **large datasets** & **big Transformers**
- Regarding model perturbations:
    - Due to apparent redundancy of blocks of layers in Transformers => removing blocks of layers except the first one **doesn't hurt robustness badly** as long as number of remaining layers is not too low
    - **Altering self-attention layers hurts the model a lot**, more than altering other layers

Software experiments

- Based on the IEEECIS Fraud Detection dataset
- Provided with some code to train & eval TabNet models (with many evaluation metrics)
- Added an early-stopping mechanism for the train step to avoid over-fitting + ability to pass TabNet parameters & early-stopping parameters as command-line arguments
- Trained clean models (i.e. without adversarial samples) on 7 TabNet parameters initially taking value ranges from Arik & Pfister's paper, ended up choosing **3 most relevant parameters:**

n_steps	the number of decision steps	values from **2 to 8**
n_shared	the number of shared GLU layers	values from **1 to 3**
n_ind	the number of individual blocks of layers	values from **1 to 3**

- Chosen evaluation metrics are **Average Attack Cost** & **Robustness Accuracy**

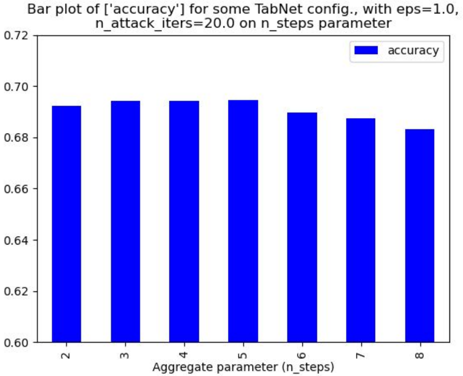
Experiments on TabNet's Adversarial robustness

- Early-stopping mechanism was configured with a **delay of 10 epochs**, starting at **epoch 100**
- To simplify average-attack-cost metric interpretation: the 3 most important features of the dataset were assigned a **cost = 1.0**, while the other ones were assigned a cost = 0
    - Average Attack Cost is interpreted here as **the number of attributes to alter in order to get a misclassification**
- Performed adversarial training for TabNet models, fixing number of attack iterations to **20**
- Initially set epsilon bound = **1.0** (**here denoting the cost margin up to which a given sample can still be correctly classified by the model after alteration**)
- For evaluation step, used *cost_bound* parameter limiting adversarial samples to be fed such that the cost of altered features doesn't exceed the value of *cost_bound*
- Evaluation performed **with** and **without** cost_bound parameter, set to **1.1**

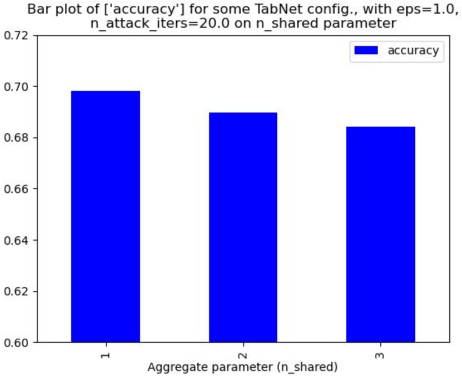
Our findings

- Expected model accuracy & robustness to increase for larger models (shared layers aim at **avoiding over-fitting** in Transformers), but found that they **tend to drop instead**, contradicting *Bhojanapalli et al.*'s findings
- Expected **n_shared** to be an influential parameter for robustness, found that it was actually the **least influential** of the three
- Most influential parameter is the **n_ind** parameter (i.e. the number of individual blocks of layers in a Feature Transformer), for which we see worse robustness accuracy for larger values
- **n_steps** parameter is the second most influential parameter on robustness

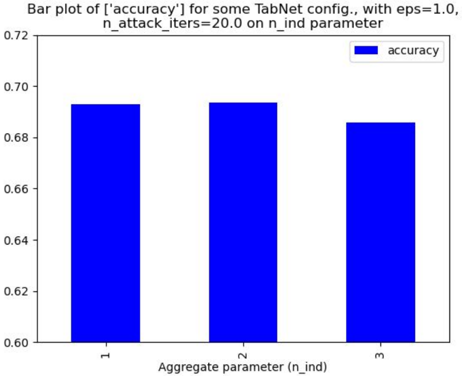
Model accuracies (for *eps*=1.0)



On *n_steps* parameter

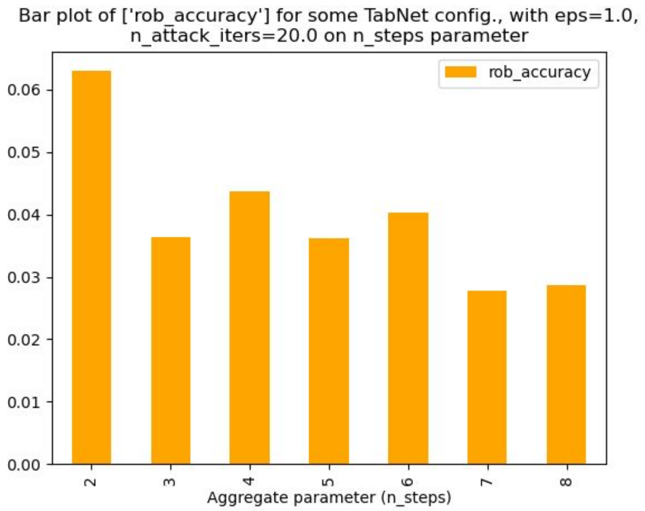


On *n_shared* parameter

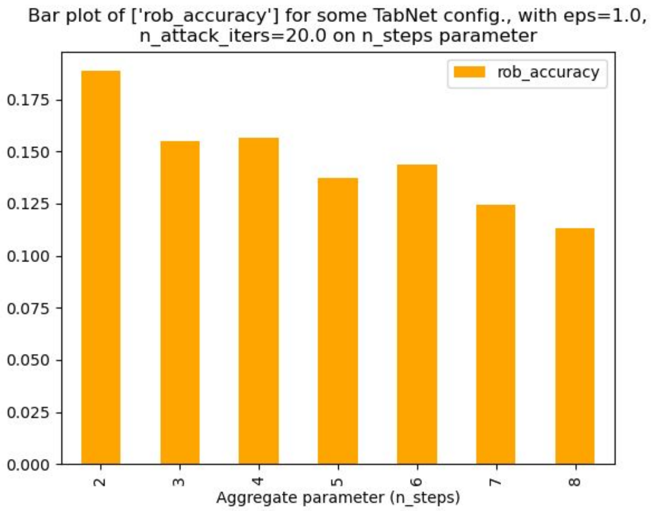


On *n_ind* parameter

Focus on *n_steps* parameter (Robustness Accuracy)

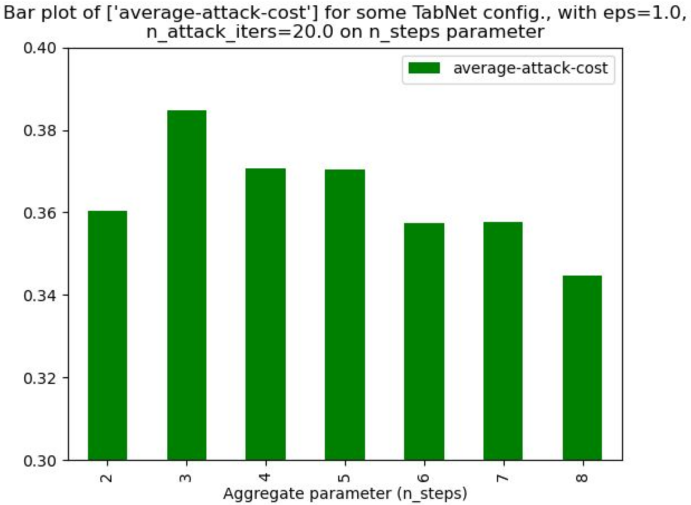


Without *cost_bound* parameter

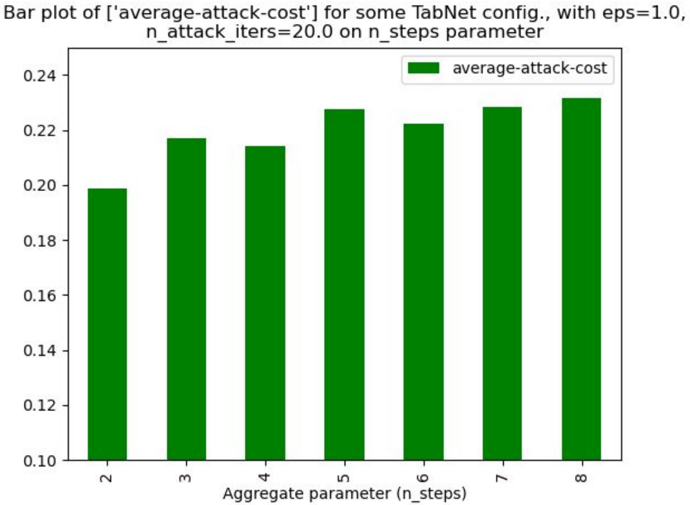


With *cost_bound* = **1.1**

Focus on *n_steps* parameter (Average Attack Cost)

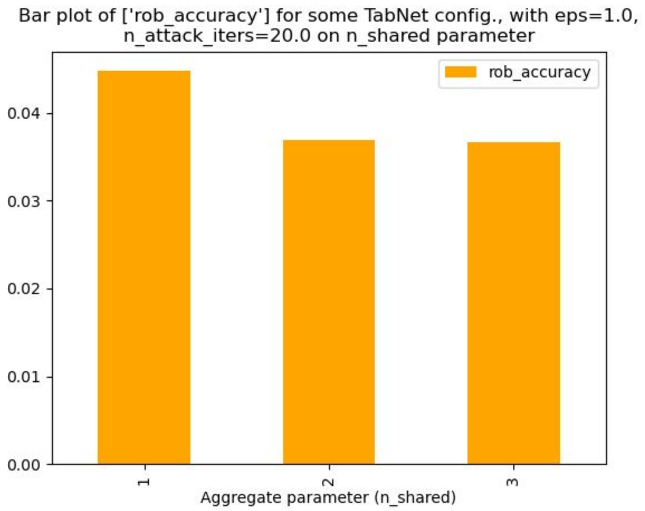


Without *cost_bound* parameter

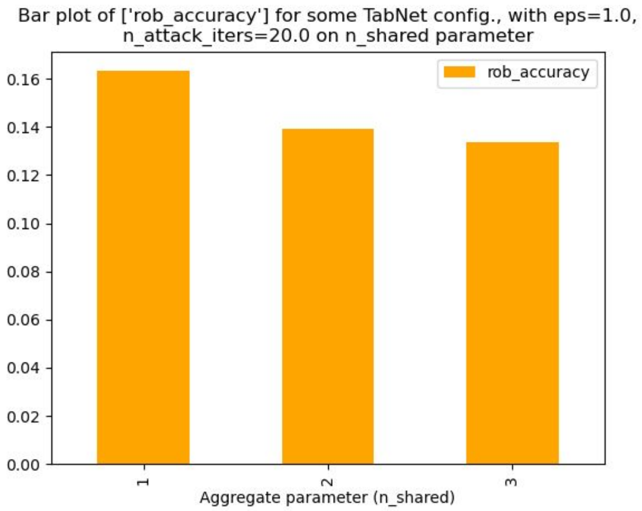


With *cost_bound* = **1.1**

Focus on *n_shared* parameter (Robustness Accuracy)

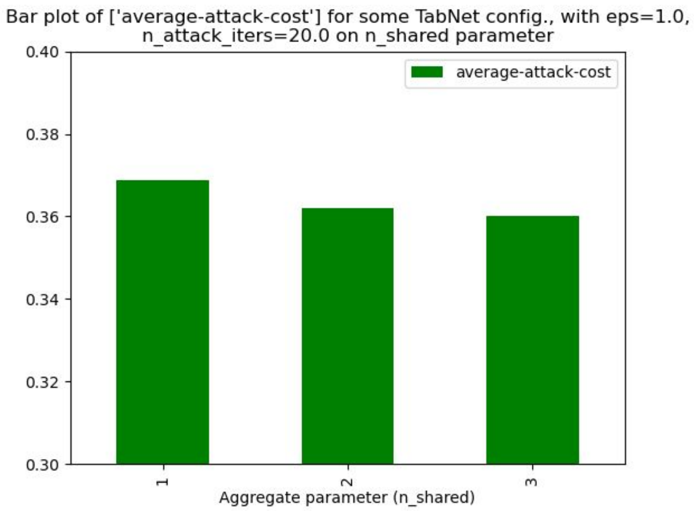


Without *cost_bound* parameter

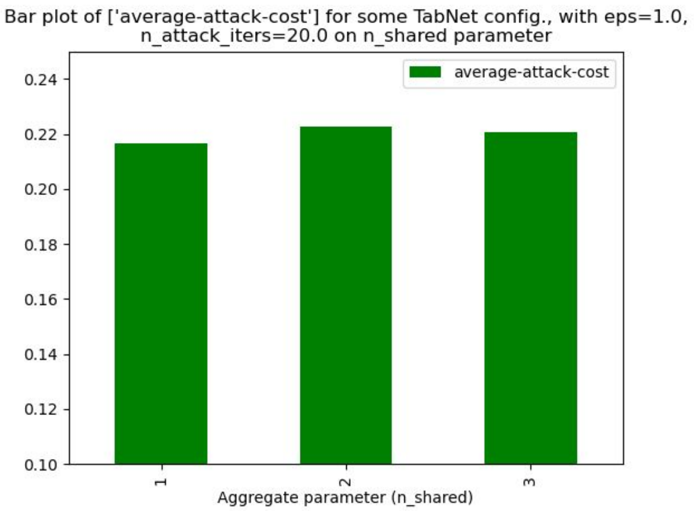


With *cost_bound* = **1.1**

Focus on *n_shared* parameter (Average Attack Cost)

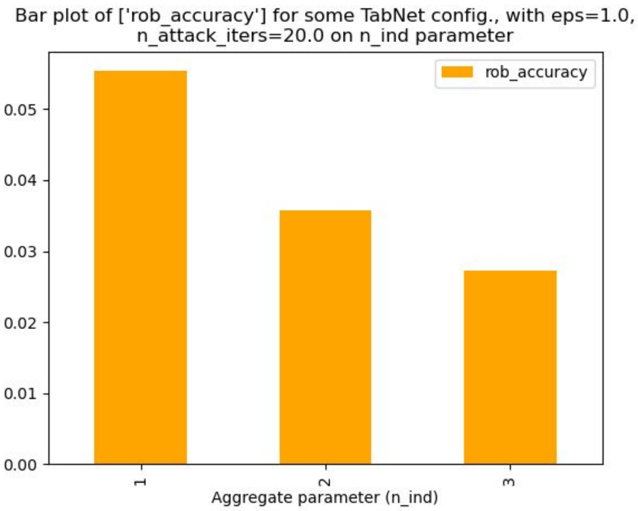


Without *cost_bound* parameter

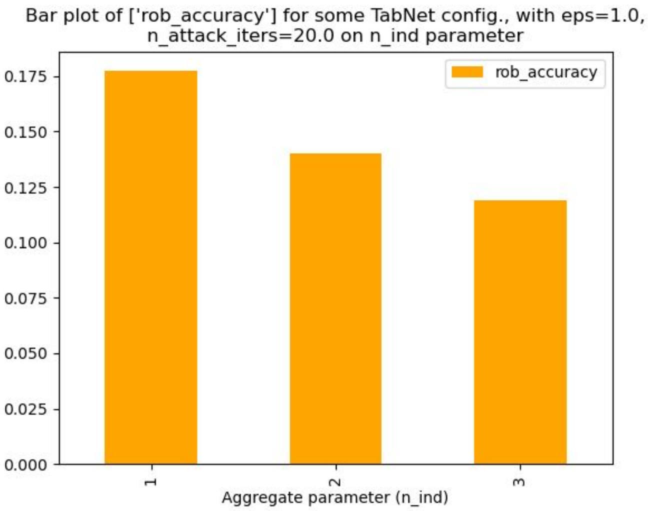


With *cost_bound* = **1.1**

Focus on *n_ind* parameter (Robustness Accuracy)

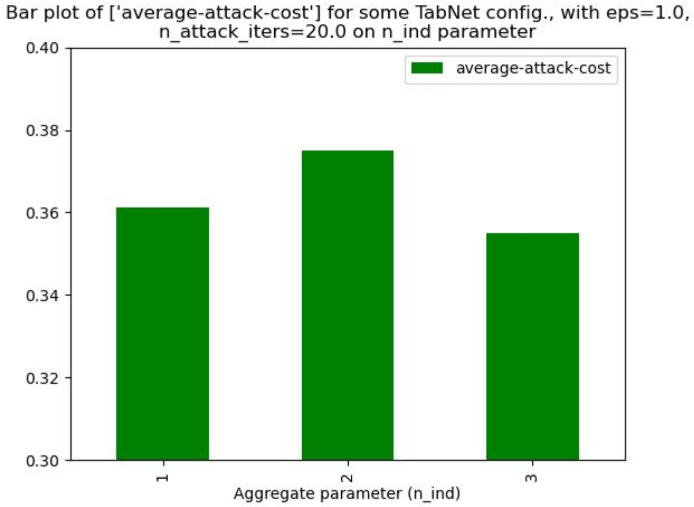


Without *cost_bound* parameter

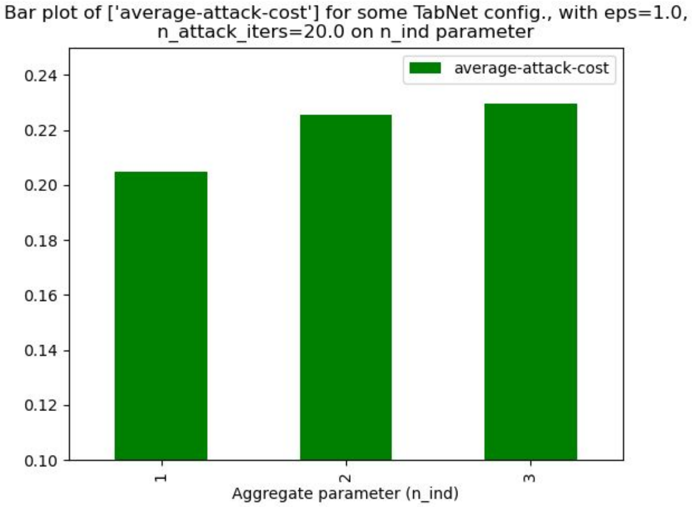


With *cost_bound* = **1.1**

Focus on *n_ind* parameter (Average Attack Cost)



Without *cost_bound* parameter



With *cost_bound* = **1.1**

Conclusion

- Model accuracies & robustness tend to drop for larger models, contradicting *Bhojanapalli et al.*'s findings
- Most influential parameter is the **n_ind** parameter, for which we see worse robustness accuracy for larger values, followed by the **n_steps** parameter
- The **n_shared** parameter seems not to be very influential

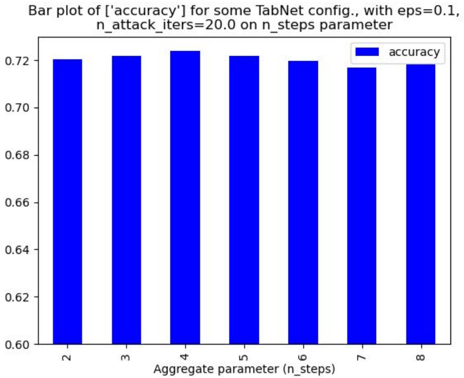
<u>**To open up a bit more:**</u>

- Could have varied more parameters to have a broader view on the influence of each parameter & better tweak the early-stopping hyperparameters, not possible due to time constraints

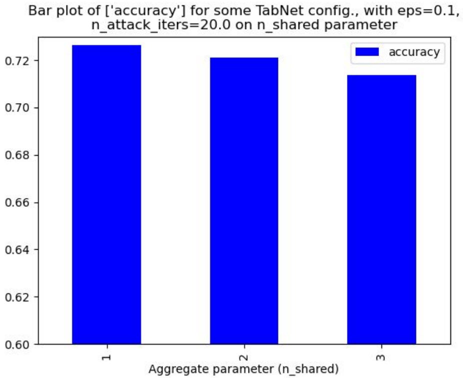
Any questions ?

Backup slides

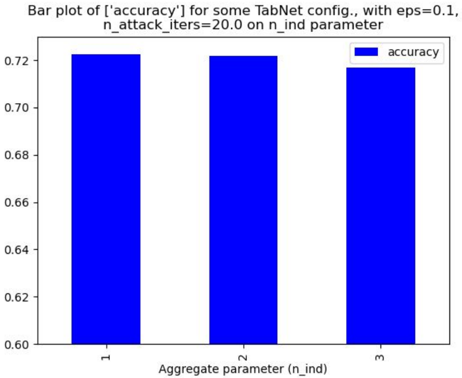
Model accuracies (for *eps*=0.1)



On *n_steps* parameter

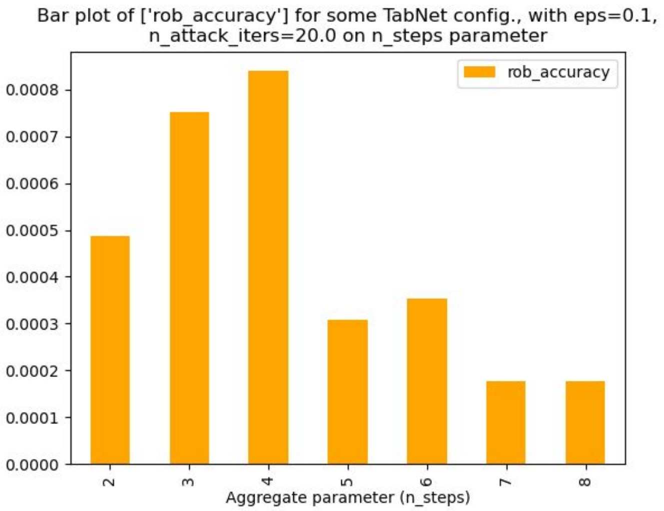


On *n_shared* parameter

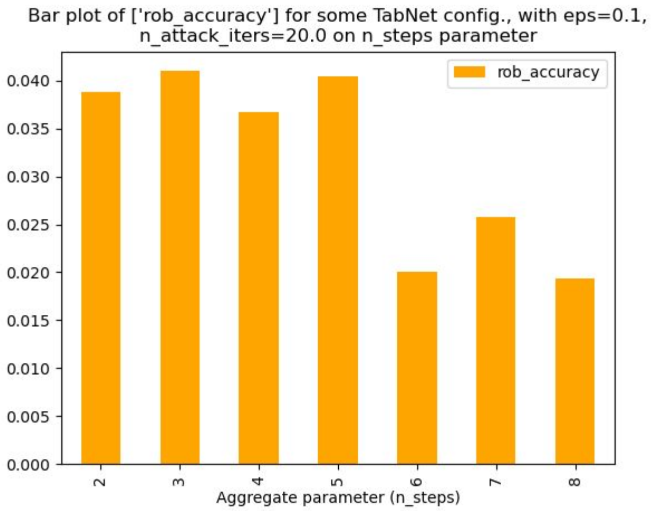


On *n_ind* parameter

Focus on *n_steps* parameter (Robustness Accuracy)

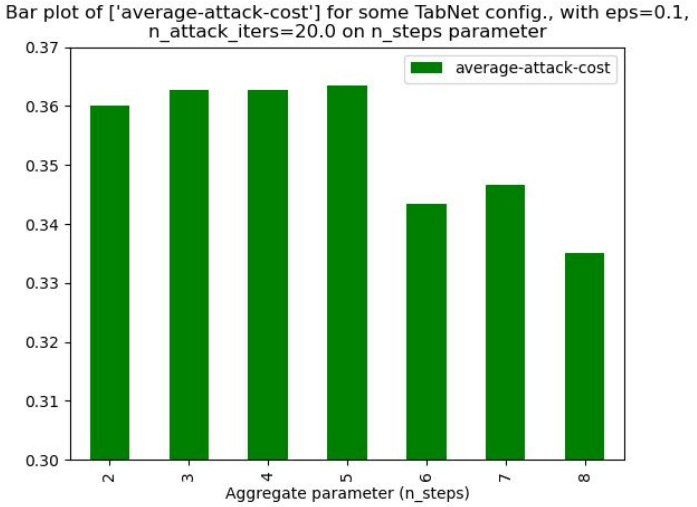


Without *cost_bound* parameter

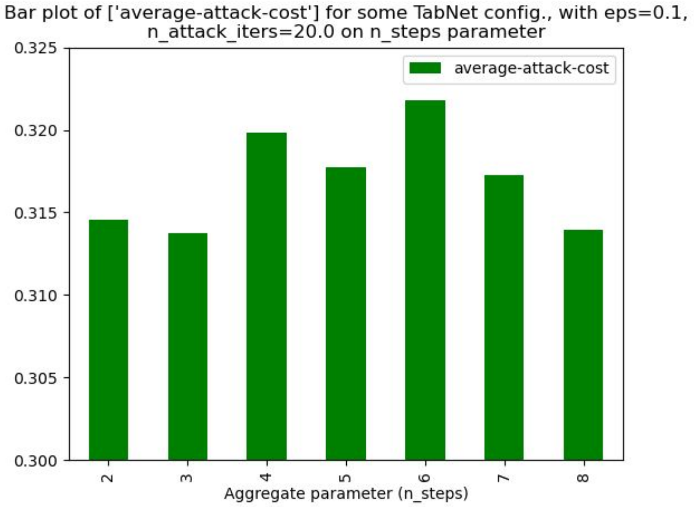


With *cost_bound* = **1.1**

Focus on *n_steps* parameter (Average Attack Cost)

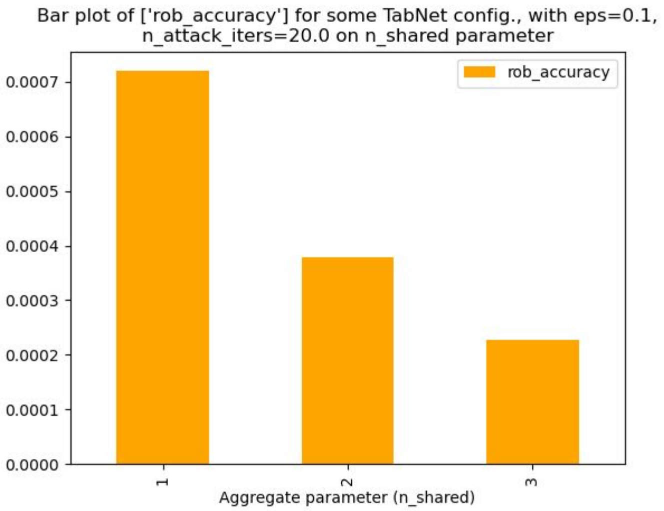


Without *cost_bound* parameter

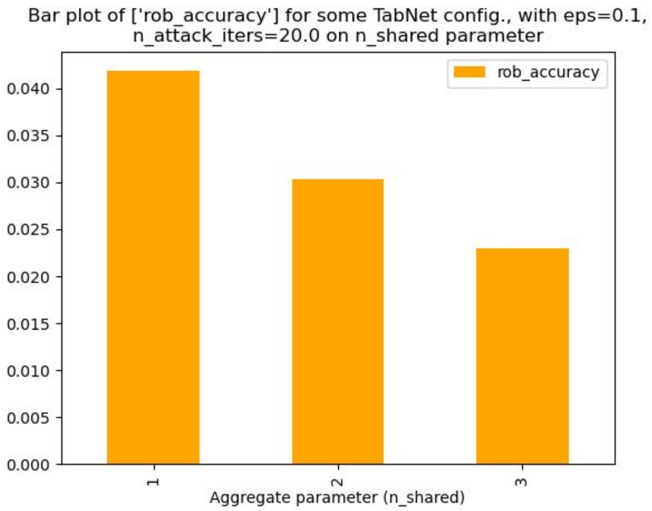


With *cost_bound* = **1.1**

Focus on *n_shared* parameter (Robustness Accuracy)

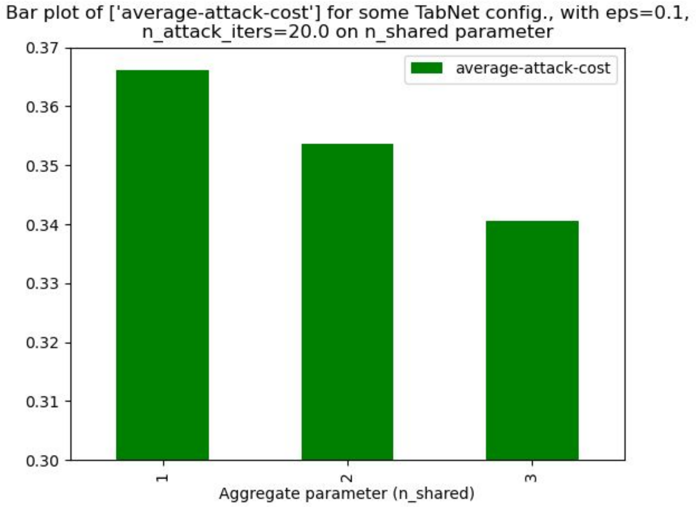


Without *cost_bound* parameter

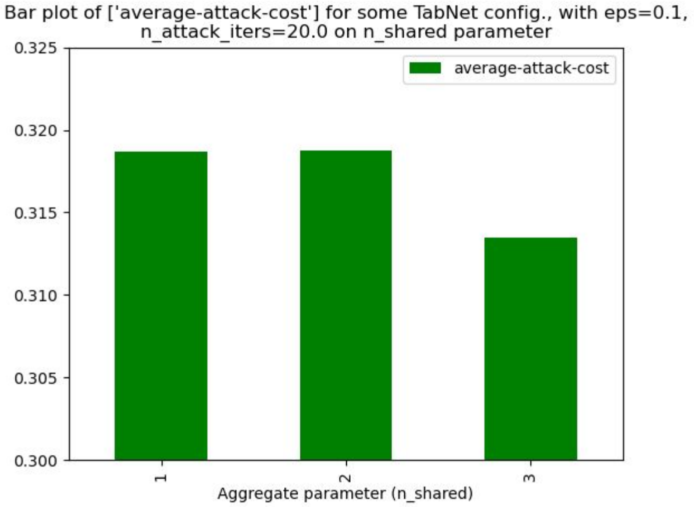


With *cost_bound* = **1.1**

Focus on *n_shared* parameter (Average Attack Cost)

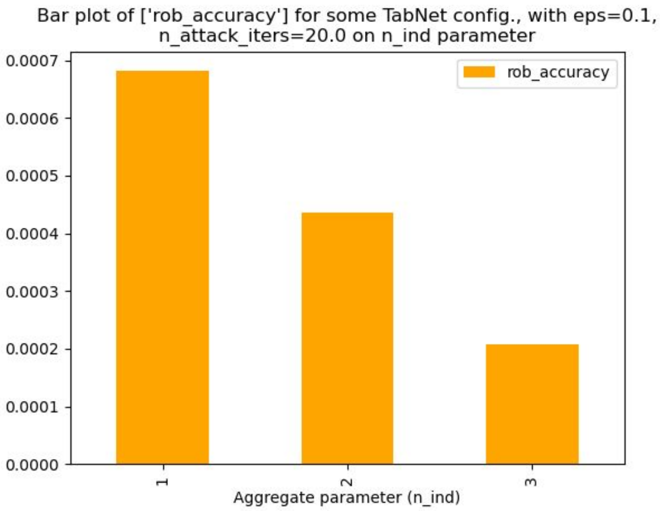


Without *cost_bound* parameter

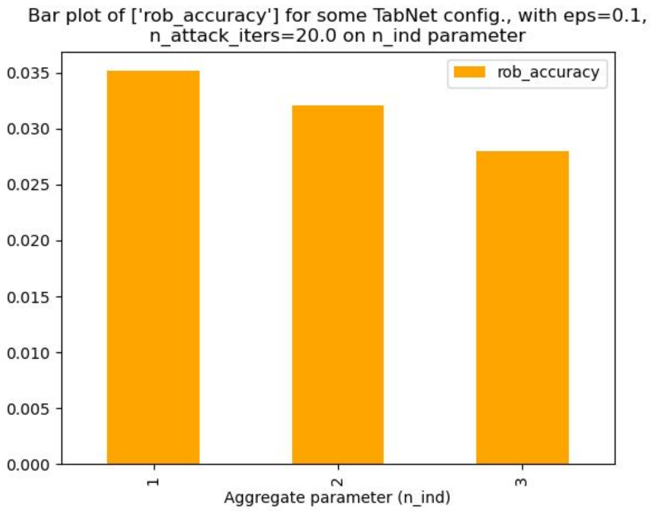


With *cost_bound* = **1.1**

Focus on *n_ind* parameter (Robustness Accuracy)

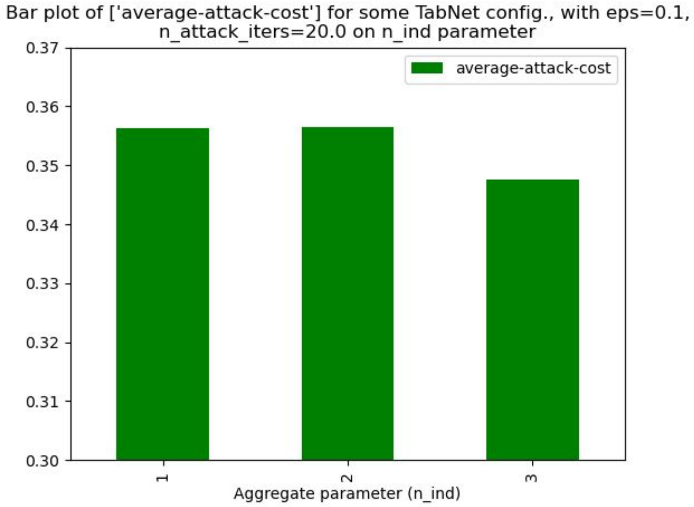


Without *cost_bound* parameter

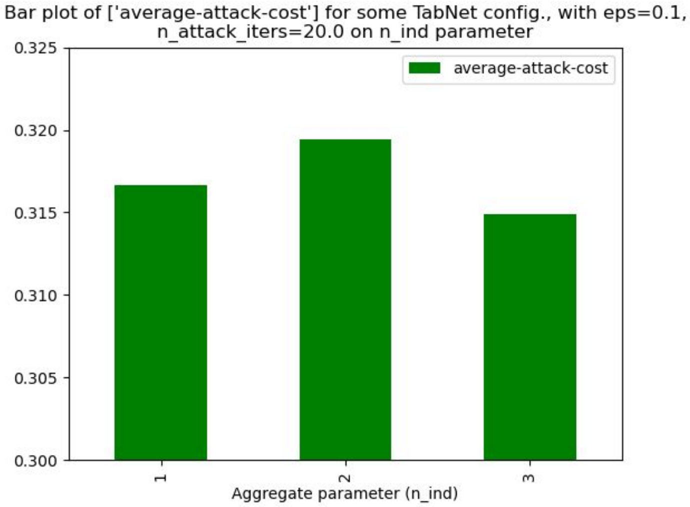


With *cost_bound* = **1.1**

Focus on *n_ind* parameter (Average Attack Cost)



Without *cost_bound* parameter



With *cost_bound* = **1.1**