Project 2 MLSec

-

Universal adversarial attack against three models

- **Names:** Marcel Plocher, Moritz Aberle
- **Matricola:** IA/65928, IA/65830
- **Instructor:** Battista Biggio

Outline

- Problem and Requirements
- Choosing Models from RobustBench
- Theorethical Approach
- Combining the Models
- Generation of Adversarial Examples
- Results and Transferability
- Challenges and Conclusion

Problem and Requirements

1. Generate universal adversarial examples
   - Choose 3 models from RobustBench
   - Using the CIFAR10 dataset and the L-inf-norm
   - Untargeted
   - Universal – for all three models at the same time
2. Evaluated transferability
   - 7 other models
   - Using the same adverserial examples

Choosing Models from RobustBench

- Aspects
  - Less adversarial training
  - Non robust pretraining (e.g. Reinforcement Learning)
  - Are trained with weak or outdated robustness techniques
- 10 Models
  - Hendrycks2019Using
  - Chen2020Adversarial
  - Wong2020Fast
  - Engstrom2019Robustness
  - Ding2020MMA
  - Rice2020Overfitting
  - Huang2020Self
  - Sehwag2021Proxy_R18
  - Rebuffi2021Fixing_70_16_cutmix_extra
  - 3. Rade2021Helper_extra

Theoretical Approaches – PGD Attack

- Constraint: Maximum-confidence

$$\min_{\|\delta\| \leq \epsilon} L(x + \delta, y; \theta)$$

- Projected Gradient Descent (PGD)

$$x^* = x + \epsilon \operatorname{sign}\big(\nabla L(x, y, \theta)\big)$$

$$\max_{\|\delta\|_\infty} L(x + \delta, y, \theta)$$

$$\delta^* = \epsilon \operatorname{sign}(\nabla L)$$

Combining the models

- **Method1:** Averaging of model outputs
  - Computationally expensive
  - Stable and deterministic

$$\frac{1}{M}\sum_{i=1}^{M} f_i(x)$$

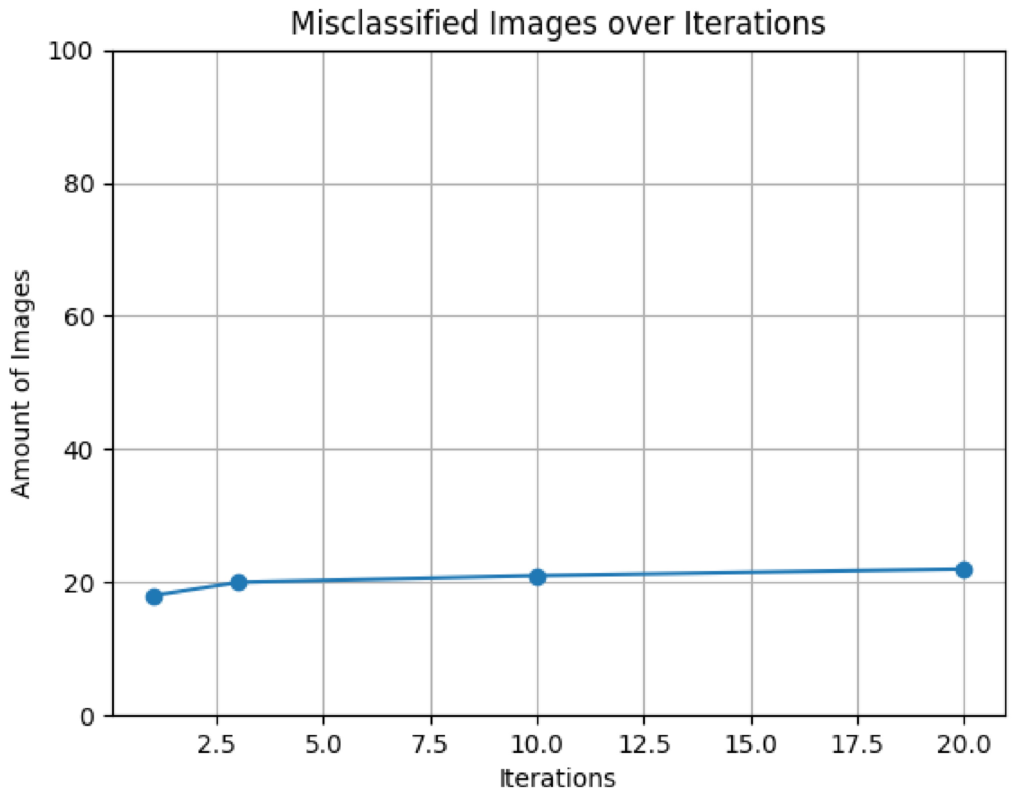
- **Method2:** Taking the most frequent prediction (Not tracked by us)
  - Simple and efficient method
  - Deterministic
- **Method3:** Random model selection (uniform distribution)
  - The attack generalizes better
  - Computationally efficient
  - Non-deterministic $F(x) = f_j(x),\ \text{where } j \sim \text{Uniform}(\{1,2,\ldots,M\})$

Generation of Adversarial Examples

- Using the PGD attack from SecML
- Using DLR-loss (Difference of Logits Ratio) instead of cross-entropy-loss
- Hyperparameter
  1. **Epsilon = 8/255:** Default value from RobustBench
  2. **N = 100:** Number of PGD-iterations
  3. **Alpha = Eps/N or 1/255**: Amount of perturbation at each step (how aggressive)
  4. **L-inf Norm:** Limits the maximum change applied to any pixel

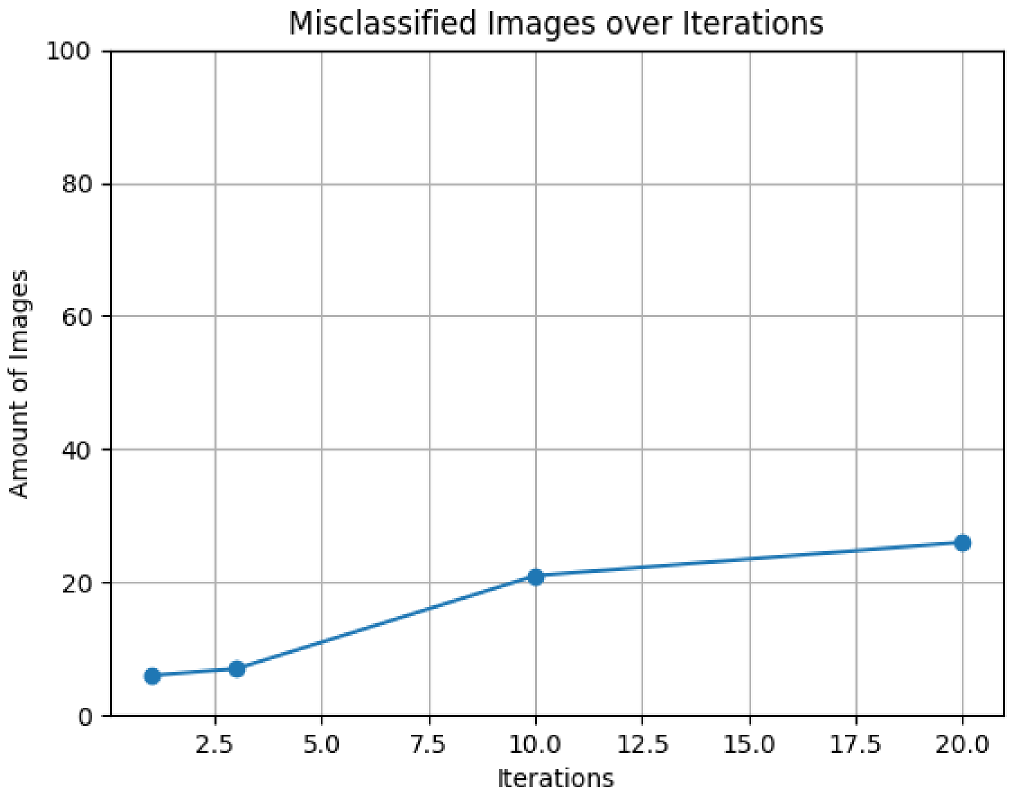
Results – Misclassified Images (Epsilon/N)

- Amount of misclassified images over all 3 models (epsilon/N)
  - 1 Iteration = 18/100 Images
  - 3 Iterations = 20/100 Images
  - 10 Iterations = 21/100 Images
  - 20 Iterations = 22/100 Image

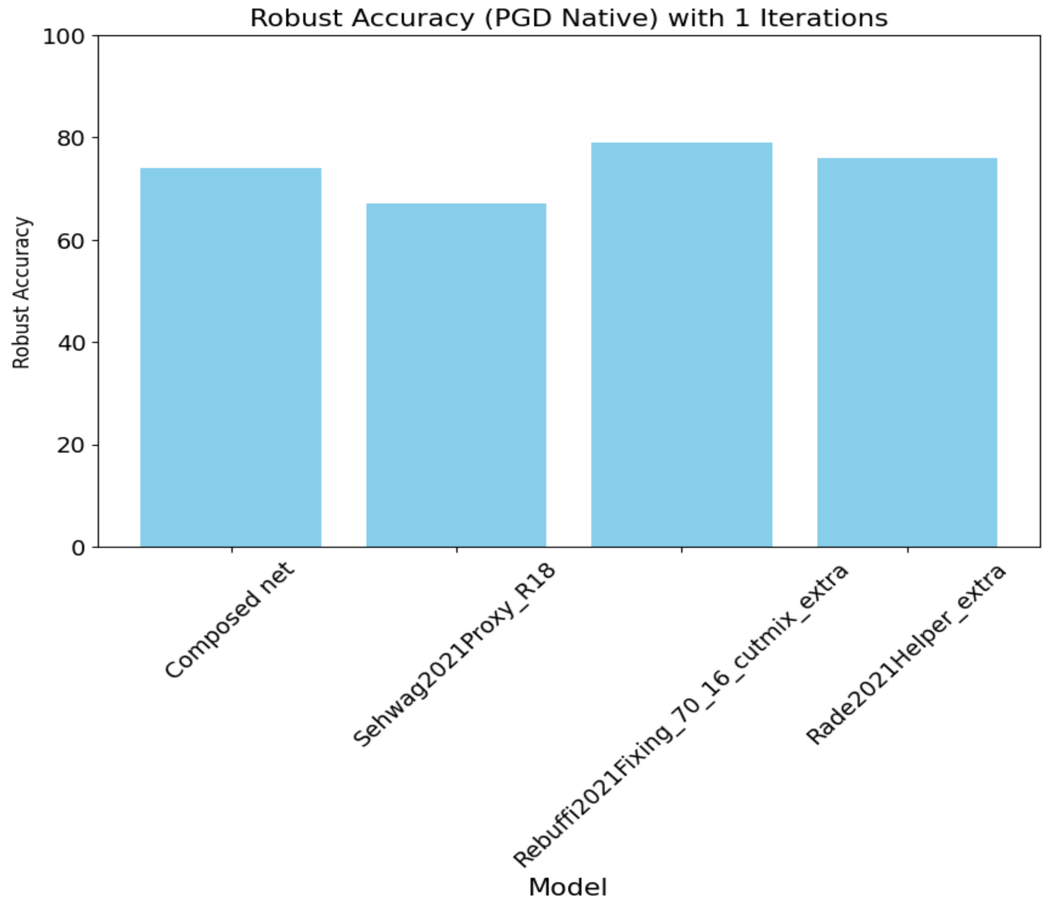


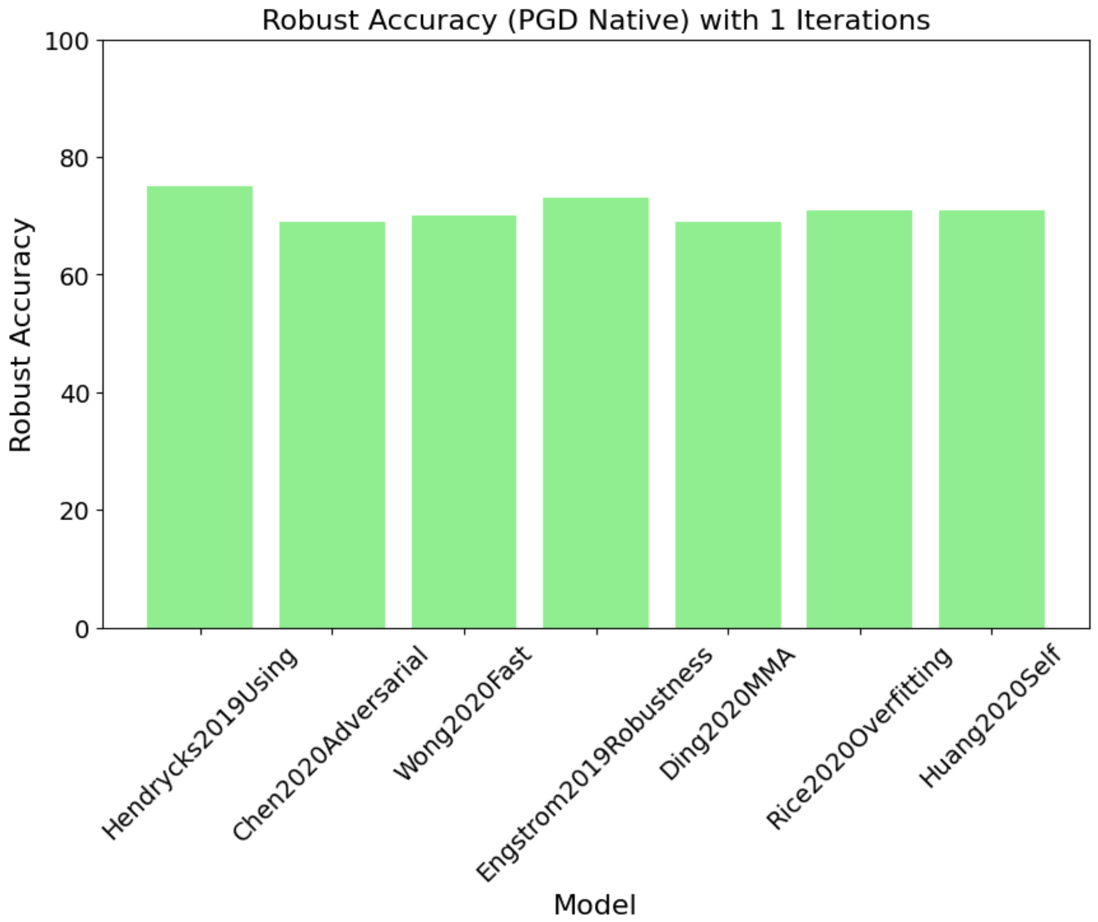
Results – Misclassified Images (1/255)

- Amount of misclassified images over all 3 models (epsilon/N)
  - 1 Iteration = 6/100 Images
  - 3 Iterations = 7/100 Images
  - 10 Iterations = 21/100 Images
  - 100 Iterations = 26/100 Images

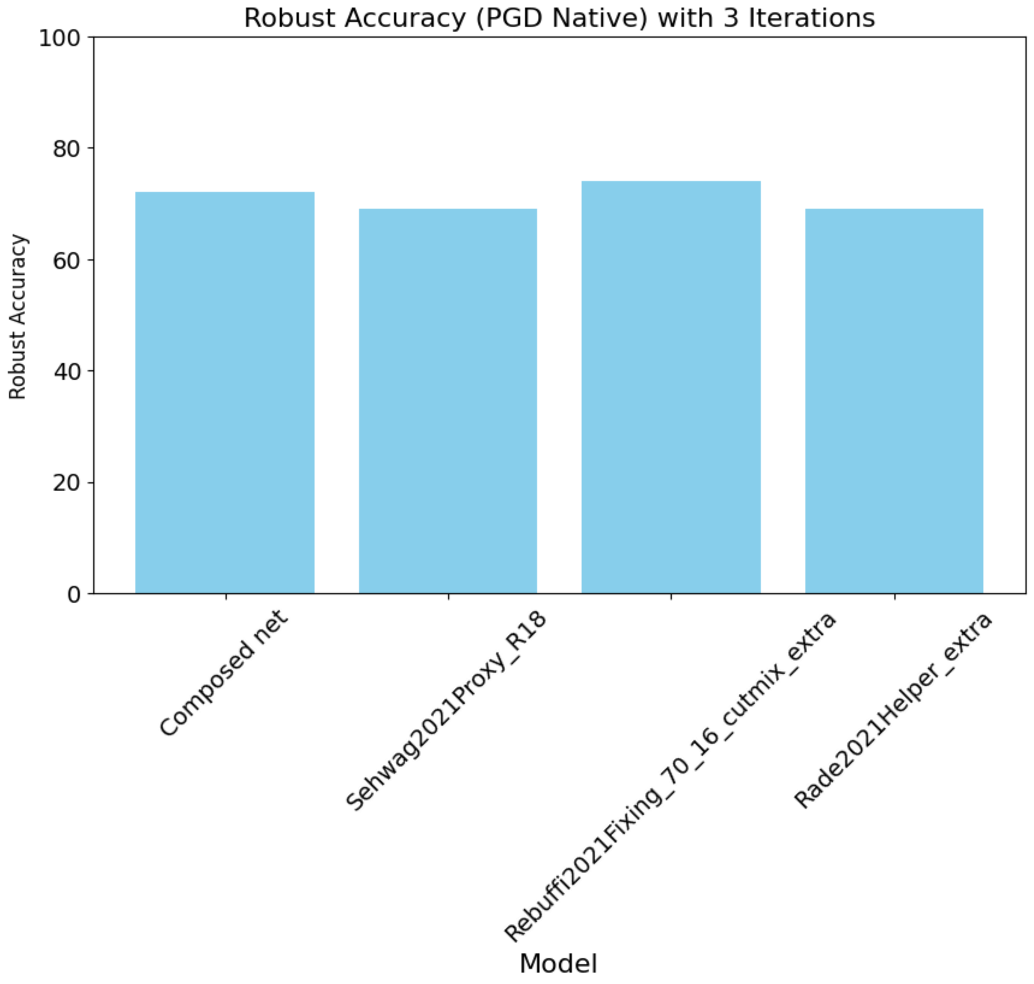


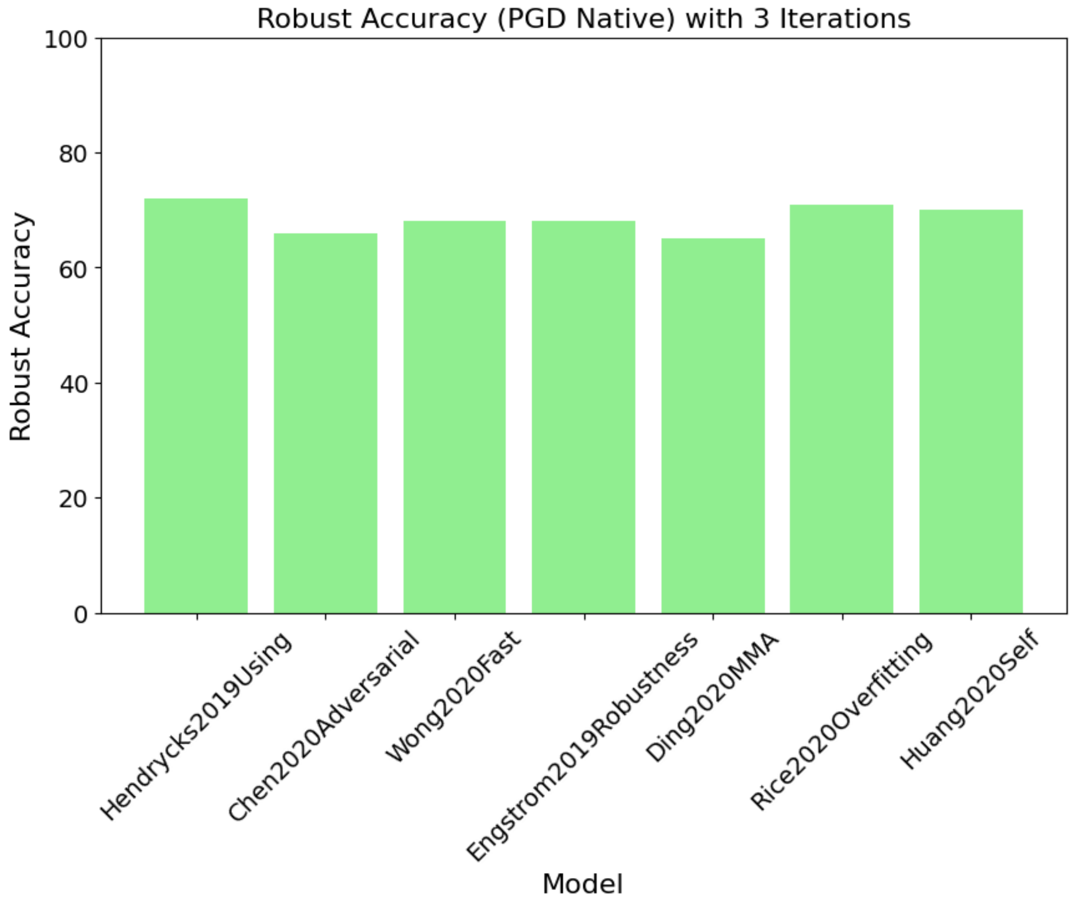
Results - Transferability (N=1, Alpha=Eps/N)



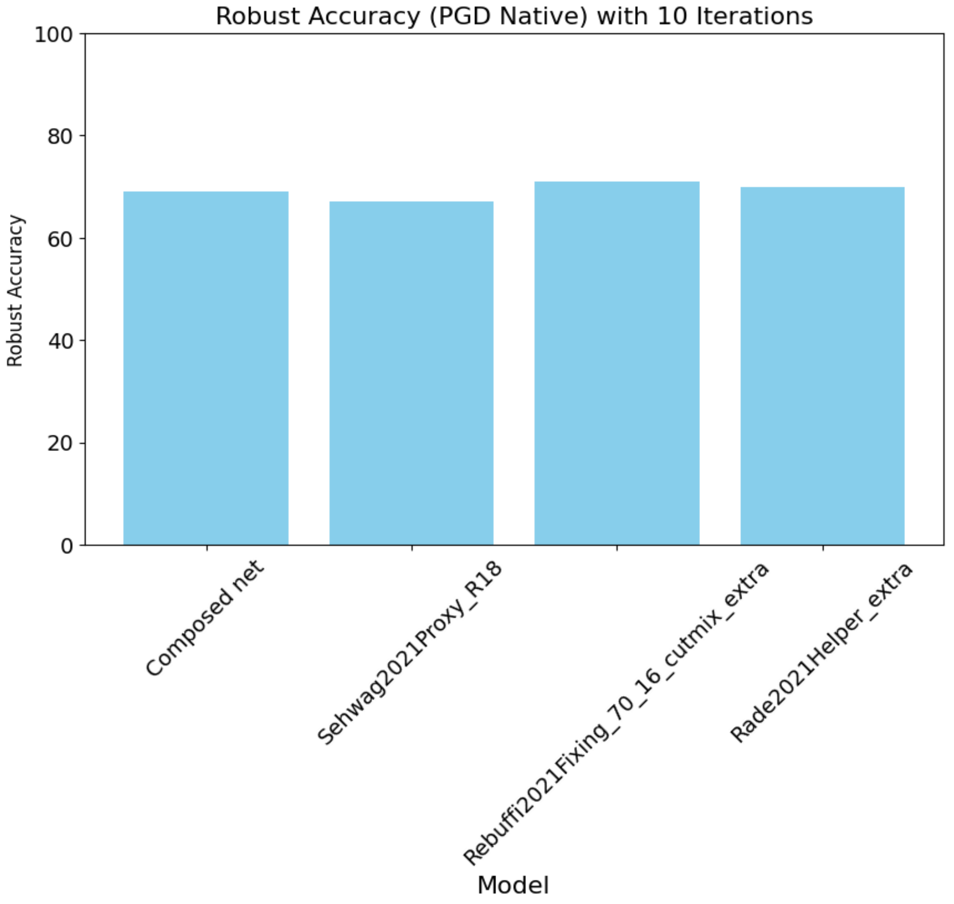


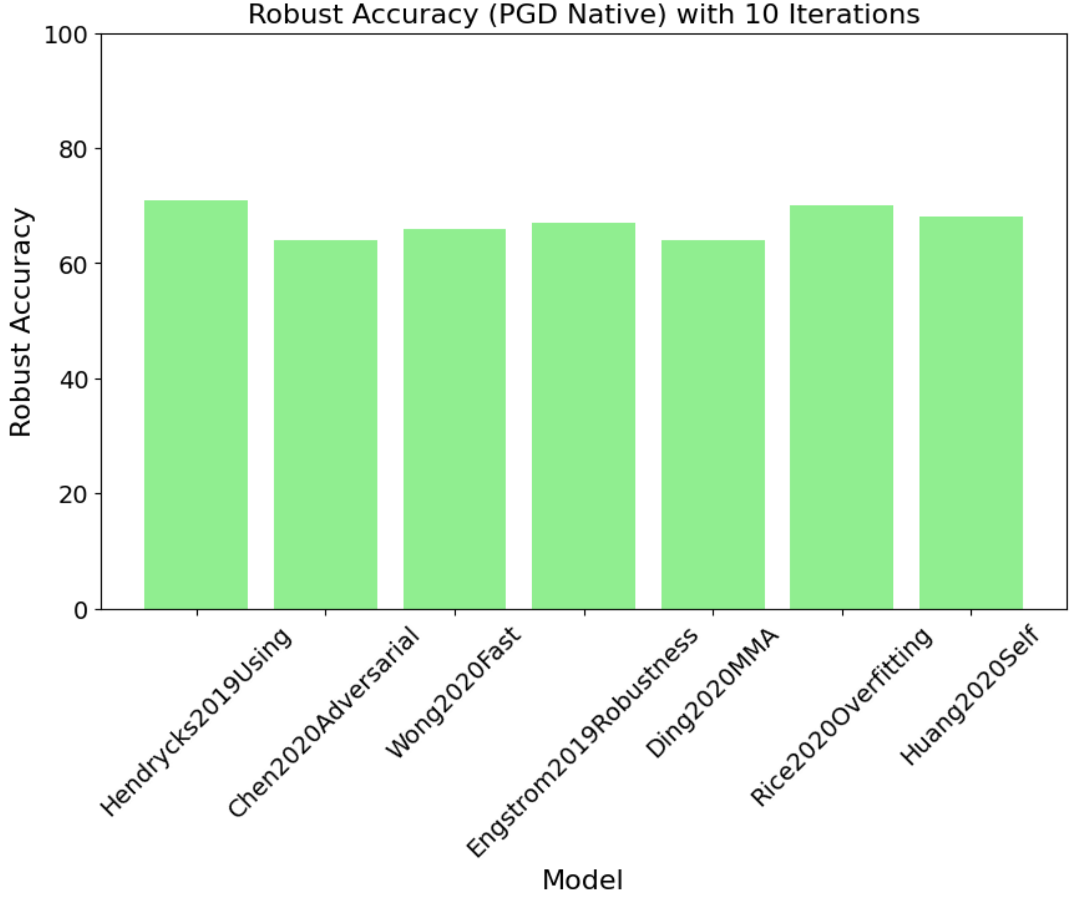
Results – Transferability (N=3, Alpha=Eps/N)



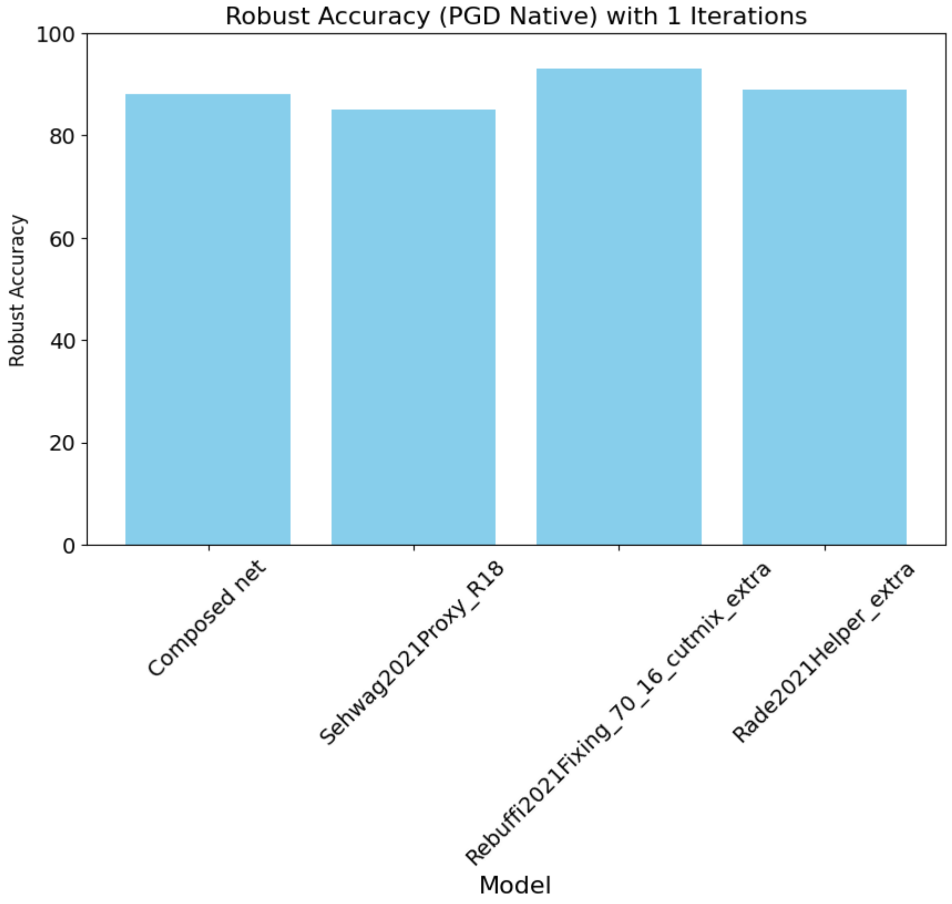


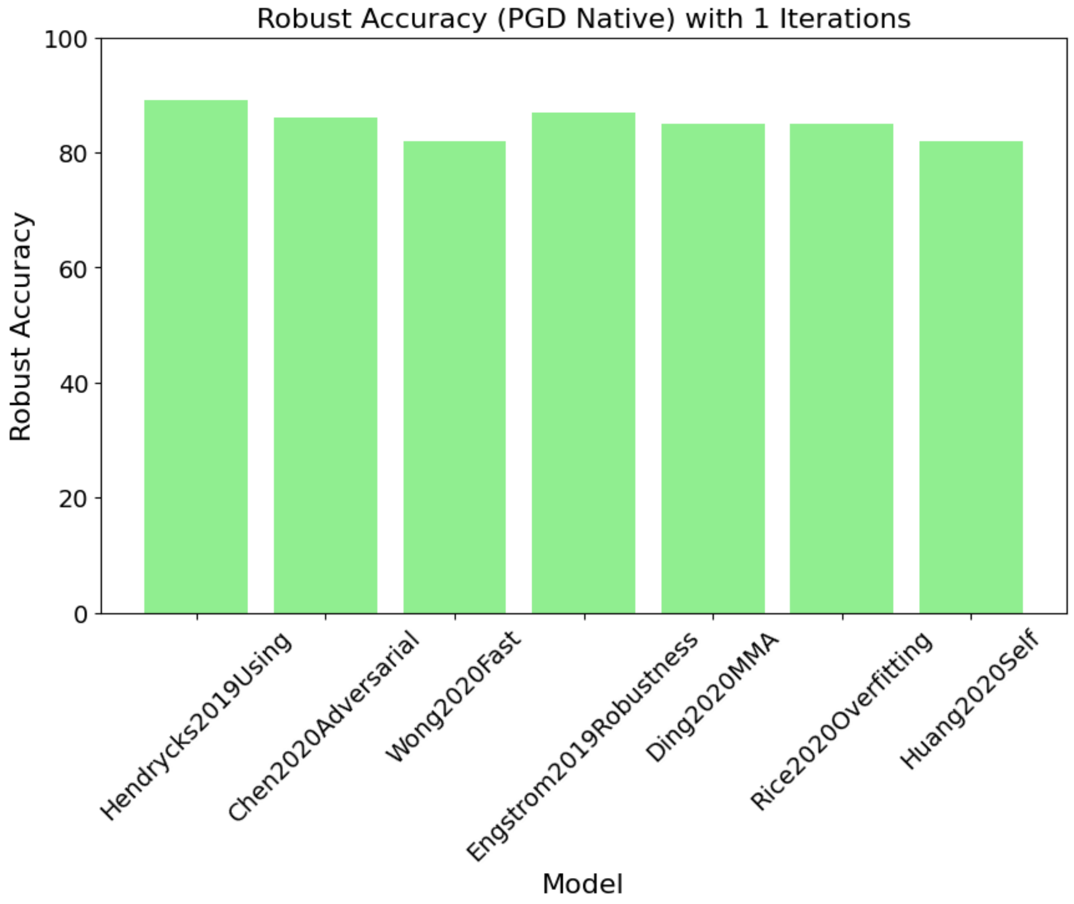
Results - Transferability (N=10, Alpha=Eps/N)



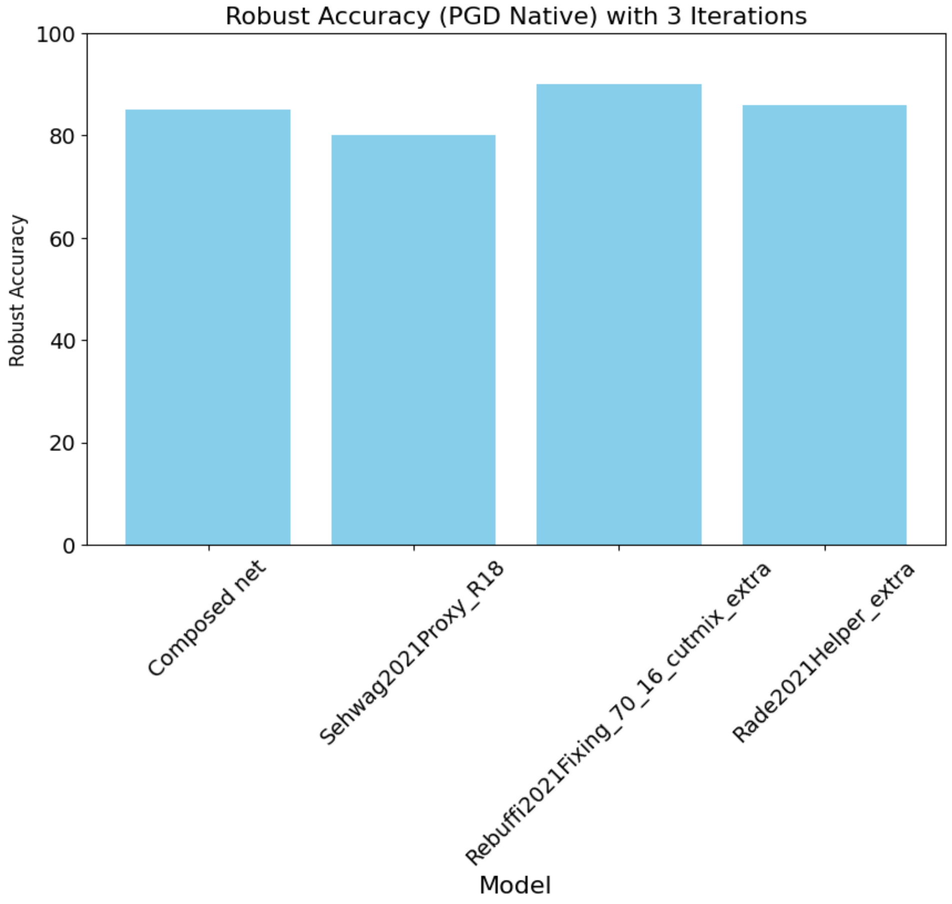


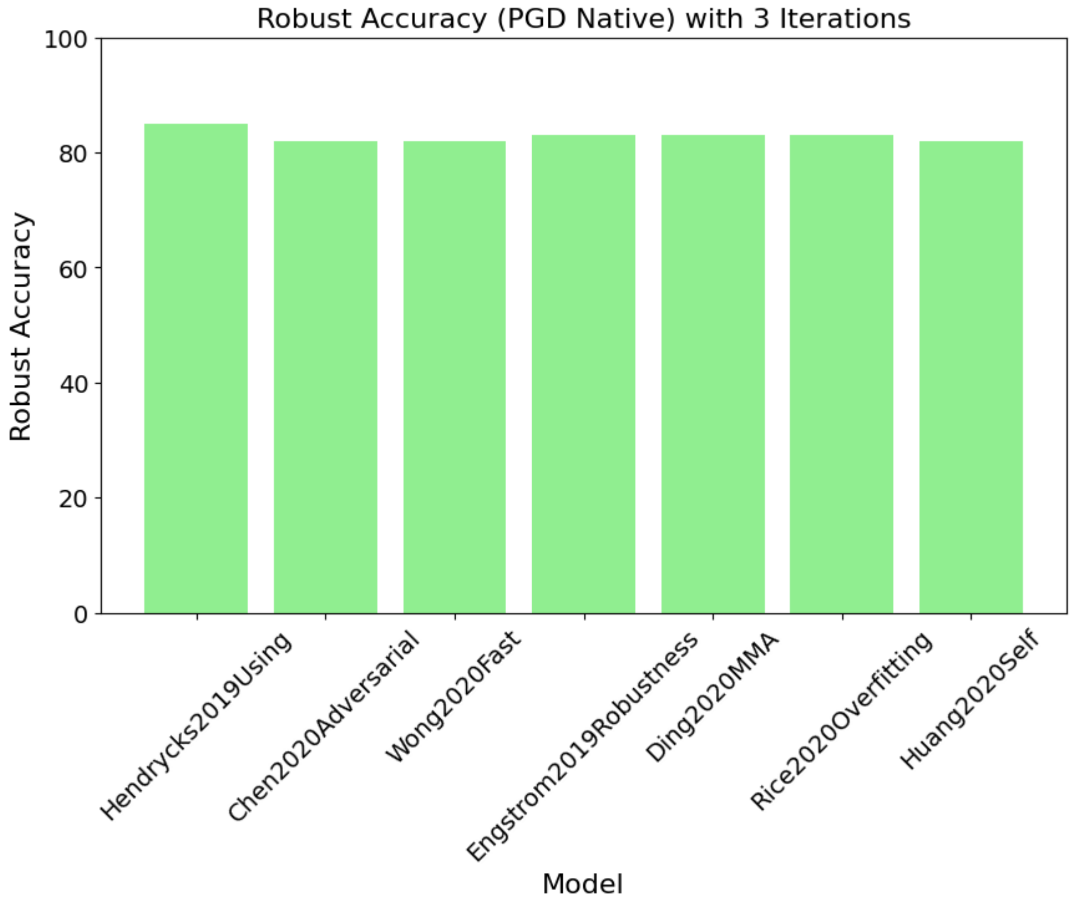
Results - Transferability (N=1, Alpha=1/255)



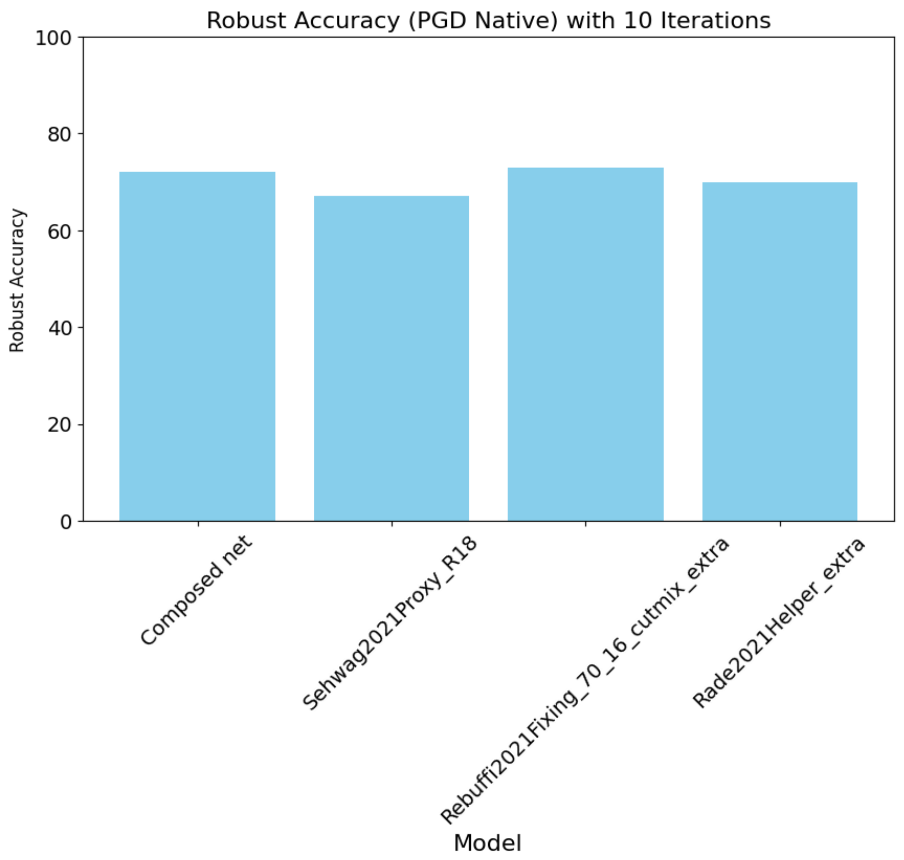


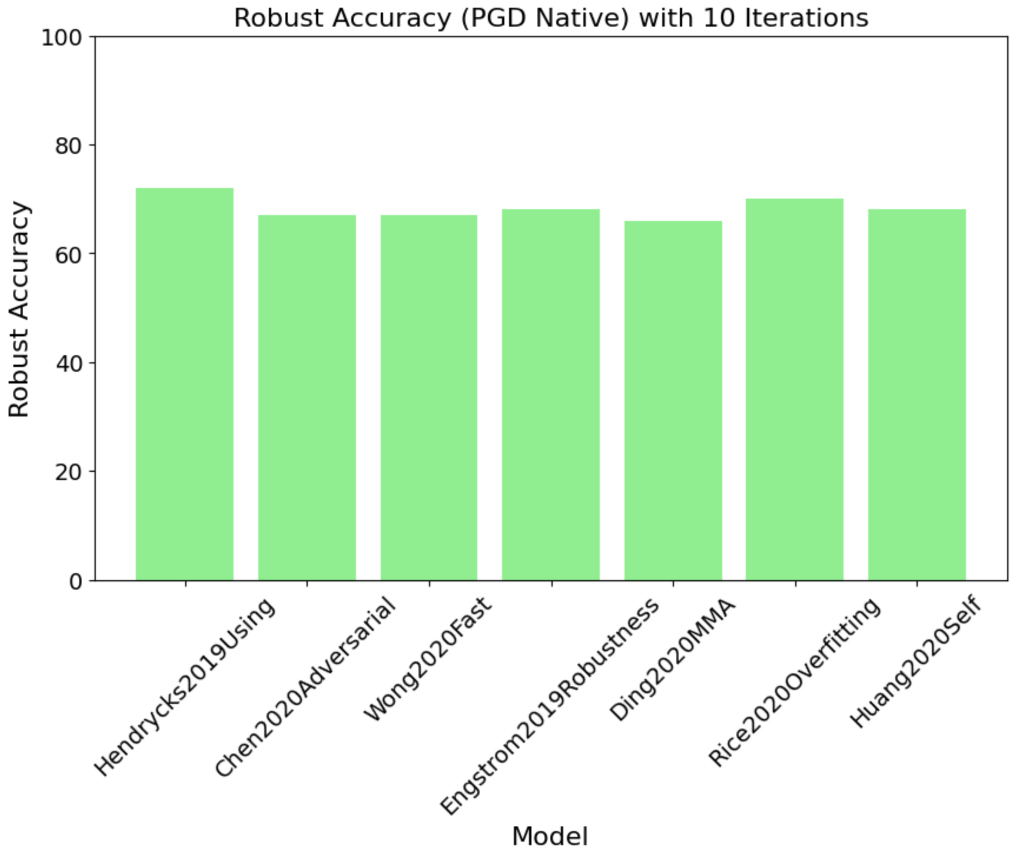
Results - Transferability (N=3, Alpha=1/255)



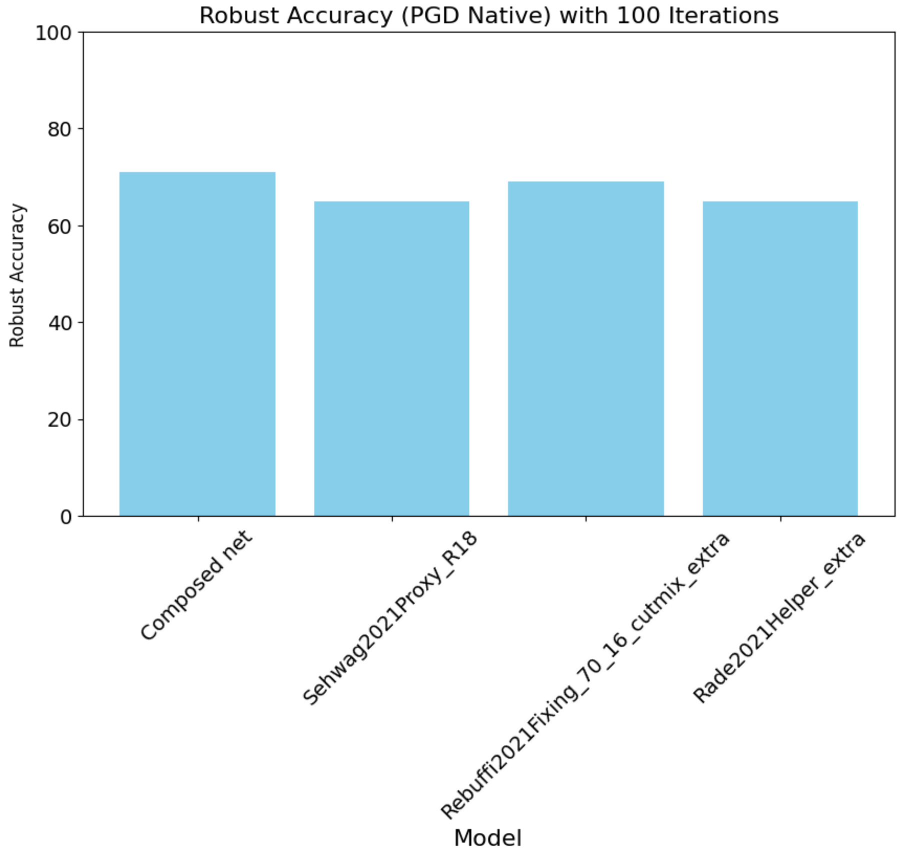


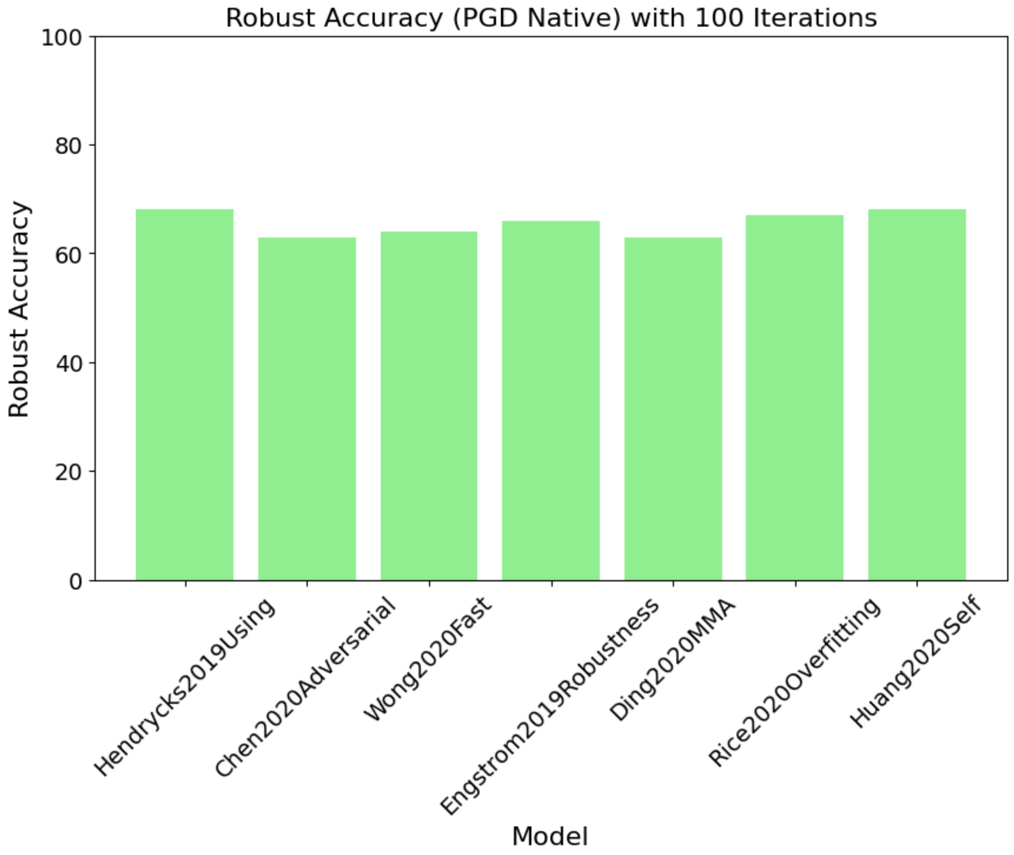
Results - Transferability (N=10, Alpha=1/255)





Results - Transferability (N=100, Alph=1/255)





Challenges and Conclusion

- With combination of 3 models the attack generalizes very good
- Finding the optimal hyperparameters is hard
  - Long runtime of the attack
  - Depending on concrete conditions (i.e. choosing epsilon)
  - Finding the optimal loss-function
  - Understanding the connection between the step size and the number of iterations