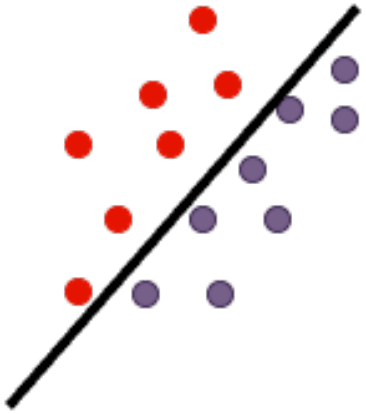
Defenses Against Evasion Attacks

Popular and Effective Methods

- Adversarial Retraining
  - Add adversarial samples from an evasion to your training dataset
  - Typically, the attack method is PGD or an improved FGSM
- Introduce randomness to your data
  - Less expensive than creating adversarial examples
- Add more training data
  - Even unlabeled with a “self-training approach”
  - Carmon et. al., Unlabeled data improves adversarial robustness, NeurIPS 2019, https://arxiv.org/pdf/1905.13736.pdf
- Regular testing
  - Every model iteration
  - Look for adaptive attacks to your defenses

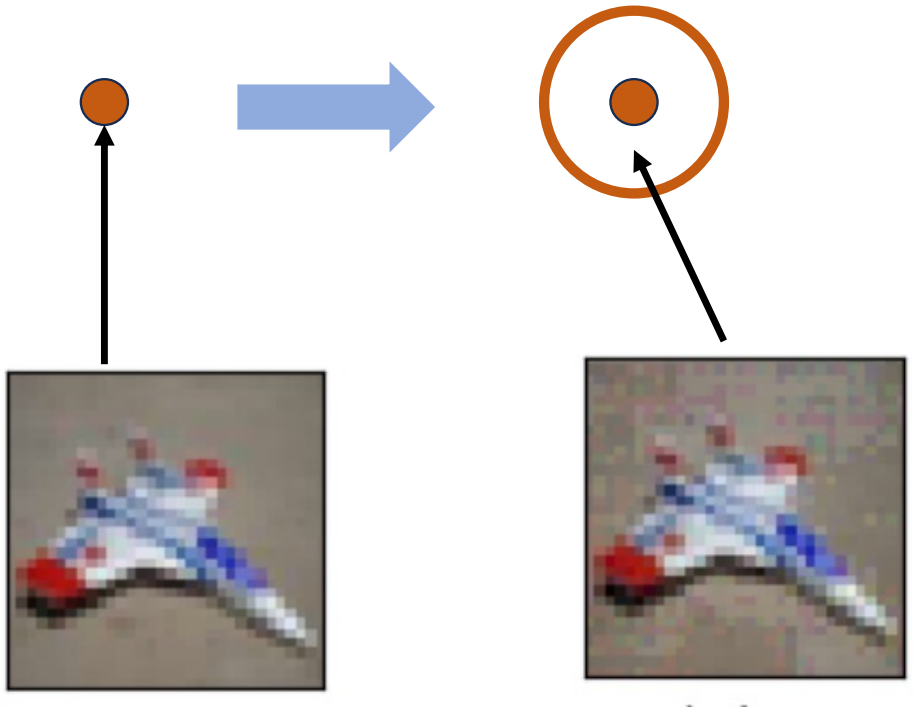
Adversarial Retraining with PDG

- Madry et. al., Towards Deep Learning Models Resistant to Adversarial Attacks, ICLR2018, arXiv:1706.06083v4
- “How can we train deep neural networks that are robust to adversarial inputs?”



Consider the Attack

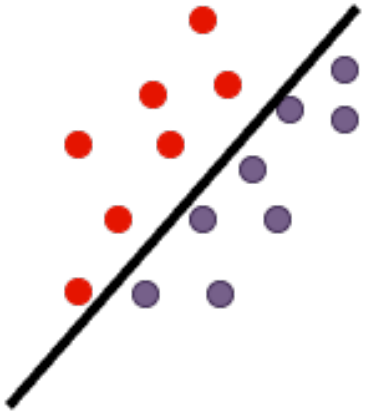
- Hackers manipulate the image within a certain budget.
- (Or at least they have to stay within these limits to avoid being detected)
- Consider manipulations of your images within these limits

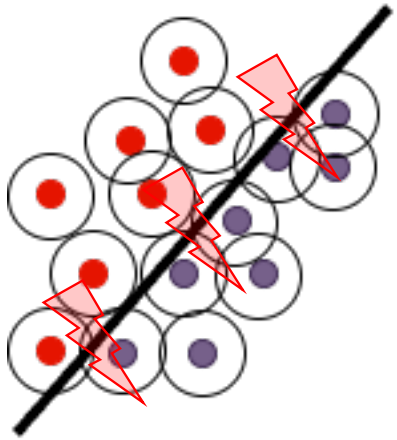


Images: CIFAR10 – original & adversarial

Adversarial Retraining with PDG

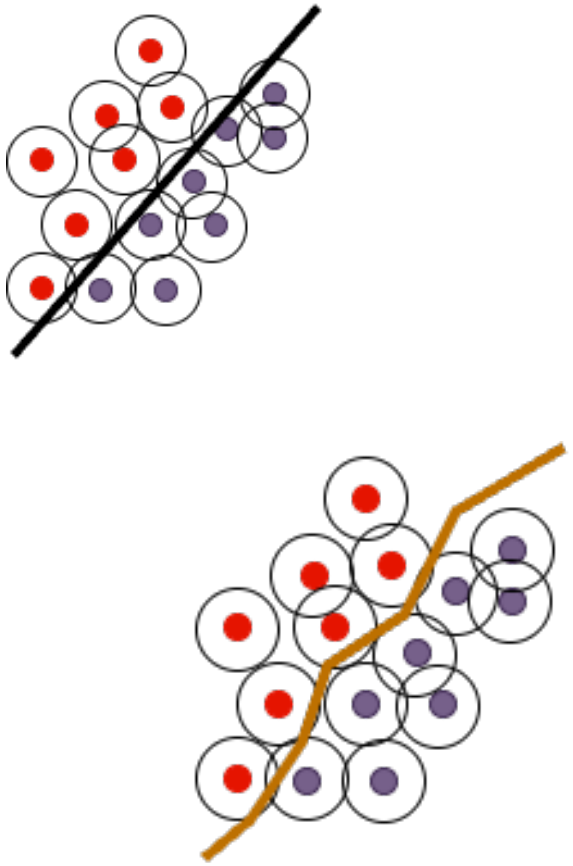
- Madry et. al., Towards Deep Learning Models Resistant to Adversarial Attacks, ICLR2018, arXiv:1706.06083v4
- “How can we train deep neural networks that are robust to adversarial inputs?”





Adversarial Retraining with PDG

- "How can we train deep neural networks that are robust to adversarial inputs?"
- Adapt the decision boundary to l_∞ balls
- Construct samples within the l_∞ balls around your training data.
- **Extend your training dataset**.



Adversarial Retraining (Madry et. al.)

- PGD Attack to generate adversarial examples
- Important role of model capacity
  - Higher model capacity allows for more resistance against adversarial examples
  - More complicated decision boundary
- Optimization perspective

$$\min_{\theta} \mathbb{E}_{(x,y)\sim \text{data}} \left[\max_{d \in S} L(x+d, y; \theta) \right]$$

S: sphere around x
θ: model parameters

Insights from Madry

- Capacity of the neural network alone increases its adversarial robustness.
- Adversaries found with Fast-Gradient for large ε do not increase robustness.
- Small capacity networks may not learn anything useful when trained including PGD adversaries (performance looses against robustness)
- Stronger adversarials and higher capacity decrease the effectiveness of transfer attacks.

PGD Attack

- Projected Gradient Descent
- Introduced by Madry et. al. in the frame of adversarial training
  - Madry et. al., Towards Deep Learning Models Resistant to Adversarial Attacks, ICLR2018, arXiv:1706.06083v4

- Iterative method
- Original paper: 100 steps, step size 0.3

$$x^{x+1} = \Pi_{x+S}\left(x^t + \alpha\ \mathrm{sgn}(\nabla_x L(x, y; \theta))\right)$$

FGSM

Adversarial Retraining

https://adversarial-robustness-toolbox.readthedocs.io/en/latest/modules/defences/trainer.html

Adversarial Training Madry PGD

class art.defences.trainer.**AdversarialTrainerMadryPGD**(*classifier: CLASSIFIER_LOSS_GRADIENTS_TYPE, nb_epochs: int | None = 205, batch_size: int | None = 128, eps: int | float = 8, eps_step: int | float = 2, max_iter: int = 7, num_random_init: int = 1*)

Class performing adversarial training following Madry's Protocol.

Paper link: https://arxiv.org/abs/1706.06083

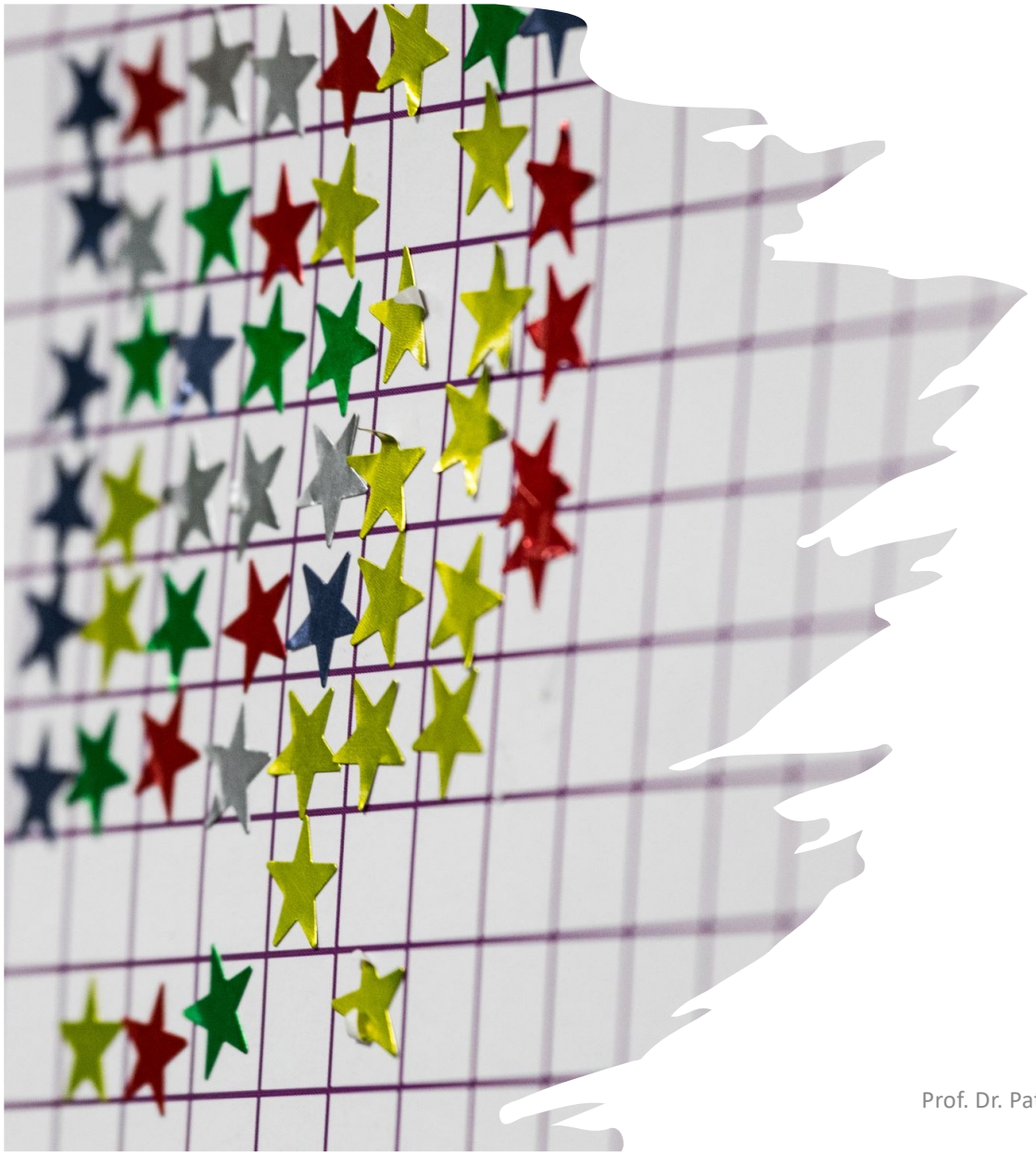
Please keep in mind the limitations of defences. While adversarial training is widely regarded as a promising, principled approach to making classifiers more robust (see https://arxiv.org/abs/1802.00420), very careful evaluations are required to assess its effectiveness case by case (see https://arxiv.org/abs/1902.06705).

__init__(*classifier: CLASSIFIER_LOSS_GRADIENTS_TYPE, nb_epochs: int | None = 205, batch_size: int | None = 128, eps: int | float = 8, eps_step: int | float = 2, max_iter: int = 7, num_random_init: int = 1*) → None

Create an **AdversarialTrainerMadryPGD** instance.

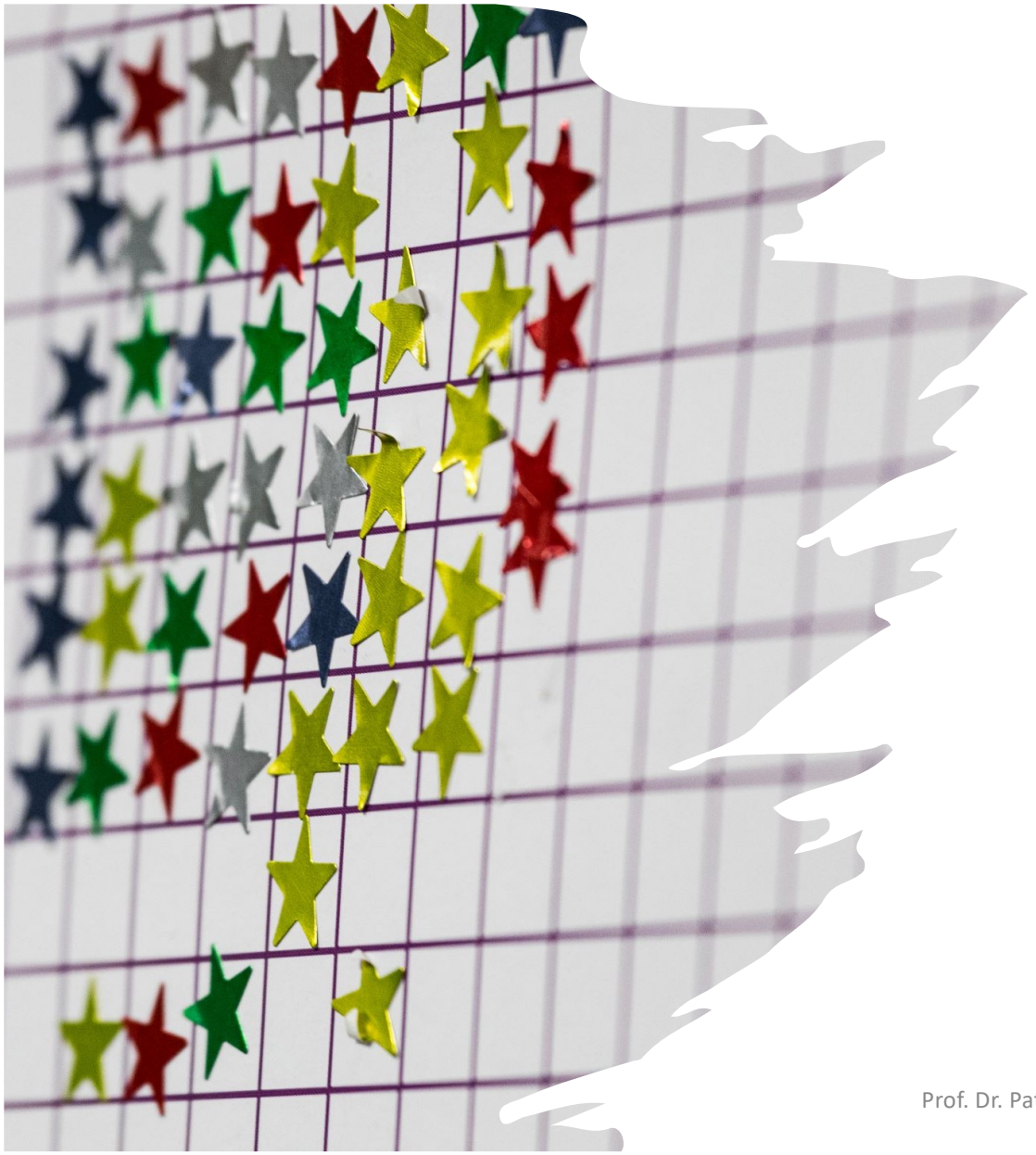
Default values are for CIFAR-10 in pixel range 0-255.

Parameters:
- **classifier** – Classifier to train adversarially.
- **nb_epochs** – Number of training epochs.
- **batch_size** – Size of the batch on which adversarial samples are generated.



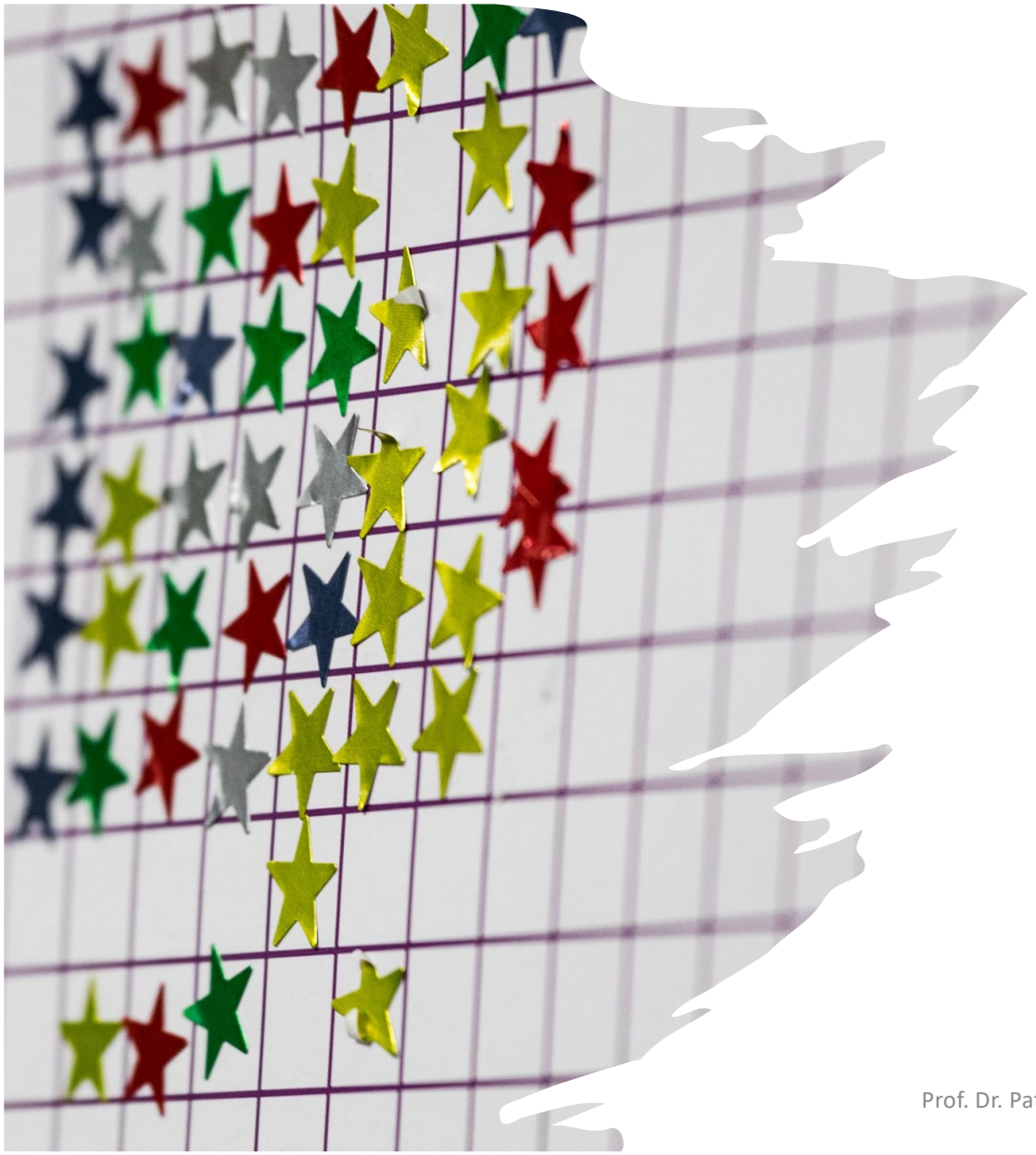
Practical Part

- Defend a ResNet18 model.
- You can use the timm package to create a Huggingface-like model and use the corresponding API from ART.
- In the notebook you will find code to
  - Train your model on the CIFAR-10 subset data (subset for calculation time reason)
  - Train your model with Madry's adversarial defense approach (yields a "robust model")
  - Attack your robust model with a PGD attack (the one you used for robust training)
  - Evaluate the (training!) accuracies



Practical Part

- Consider the following aspects
  - What accuracies do you want to measure?
  - What accuracy shall be high, which shall be low?
  - Are you satisfied using only the training accuracy?
  - Vary attack parameters (cross-read Madry's paper)
  - Try variations of the robust training epochs.
  - Evaluate your robust model for several different choices of nb_epochs
  - Vary the model you create adverarial examples with (non-robust model, or the robust one you just trained) → simulate different attacker knowledge
- Forget about ResNet50 for the moment.



Practical Part Challenge

- Get your (training or test?) accuracy as high as possible
- Get your adversarial robustness as good (= high or low?) as possible.
- Have fun. Don’t break anything. Yet.