Adversarial Learning using Cluster-based Method

Sirakorn Lamyai

KASETSART UNIVERSITY

TH0EORY RESEARCH GROUP

Preamble

Machine learning for

Privacy **Security**

Privacy

- Models are widely used—from employment to jurisdiction
- Users wanted to ensure that it is impossible to extract privacy-concerning data from the model

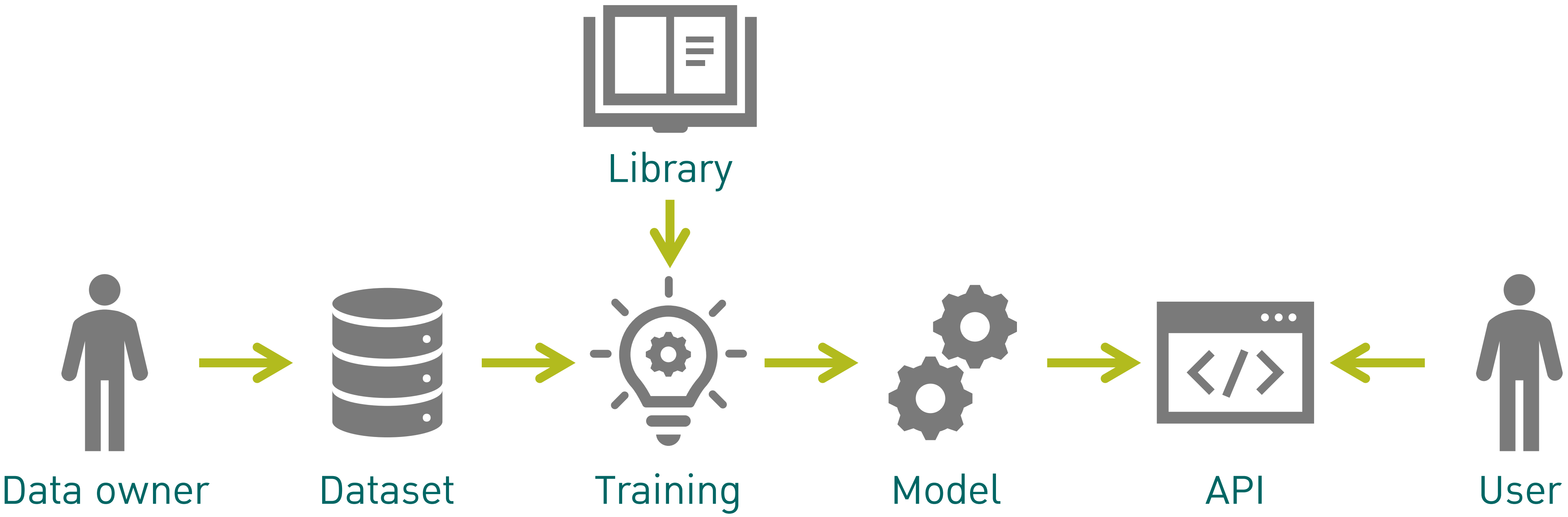
Security

- Model users wanted to make sure that the model works regardless of malicious attempts
- Successful attacking attempts might result in life-threatening dangers

Adversarial Attack

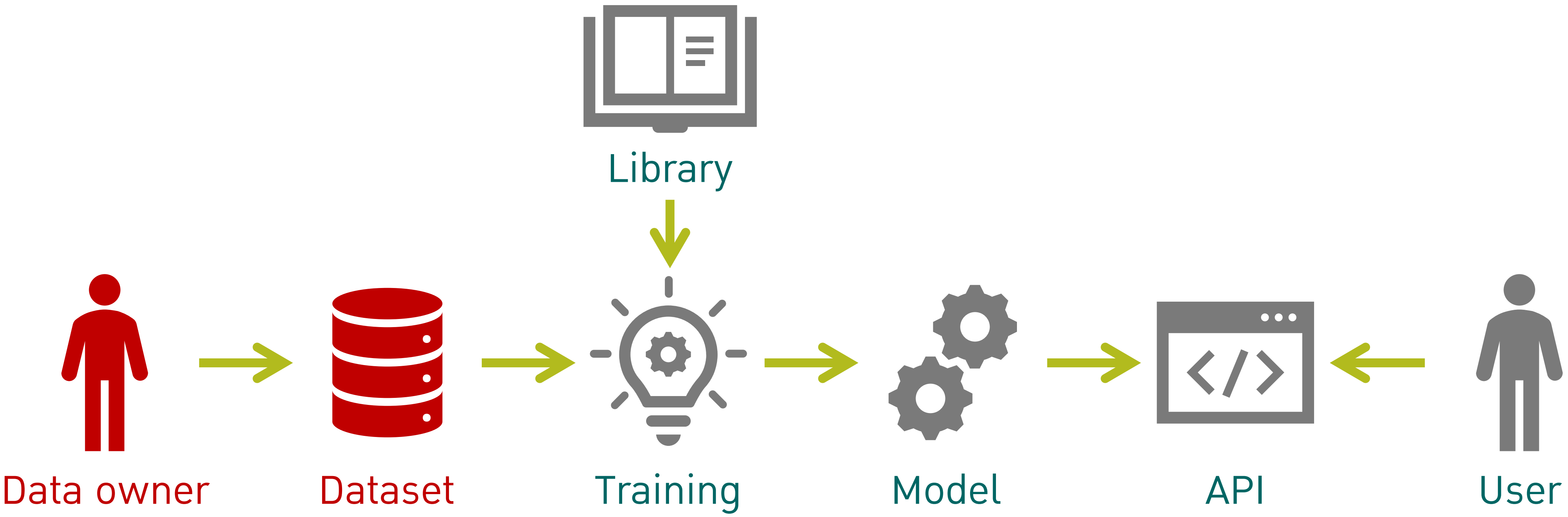
Machine Learning pipelines

Illustration adapted from *Security, Privacy and ML* by N. Asokan



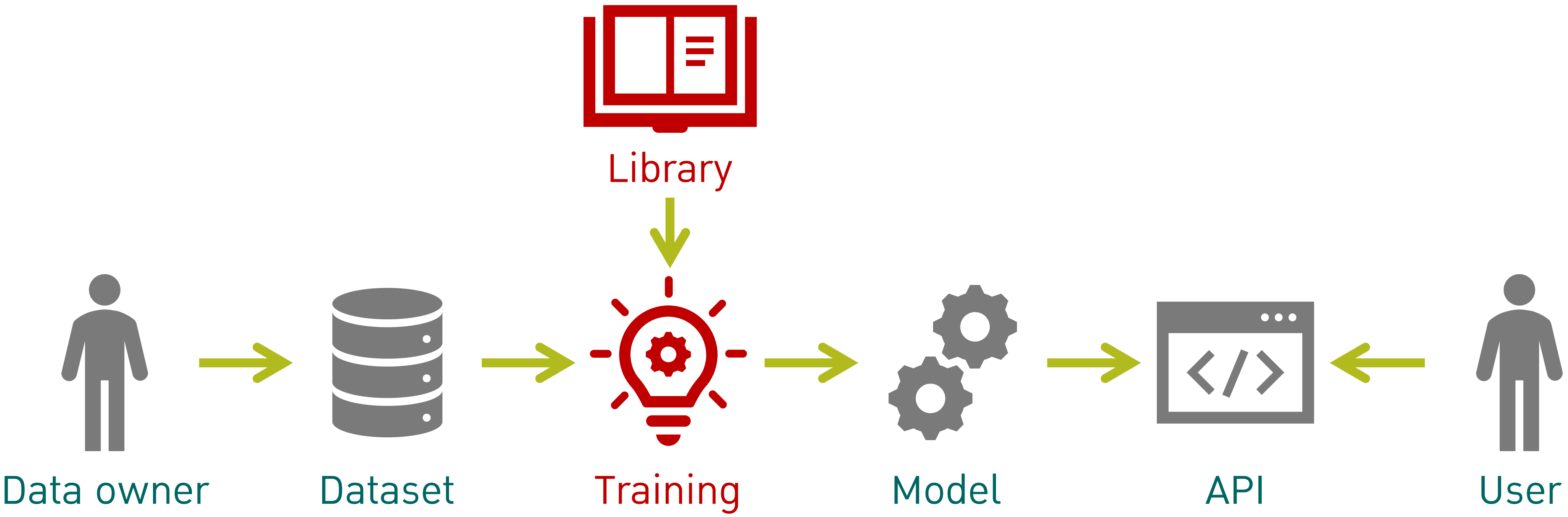
Dataset attacking

Illustration adapted from *Security, Privacy and ML* by N. Asokan



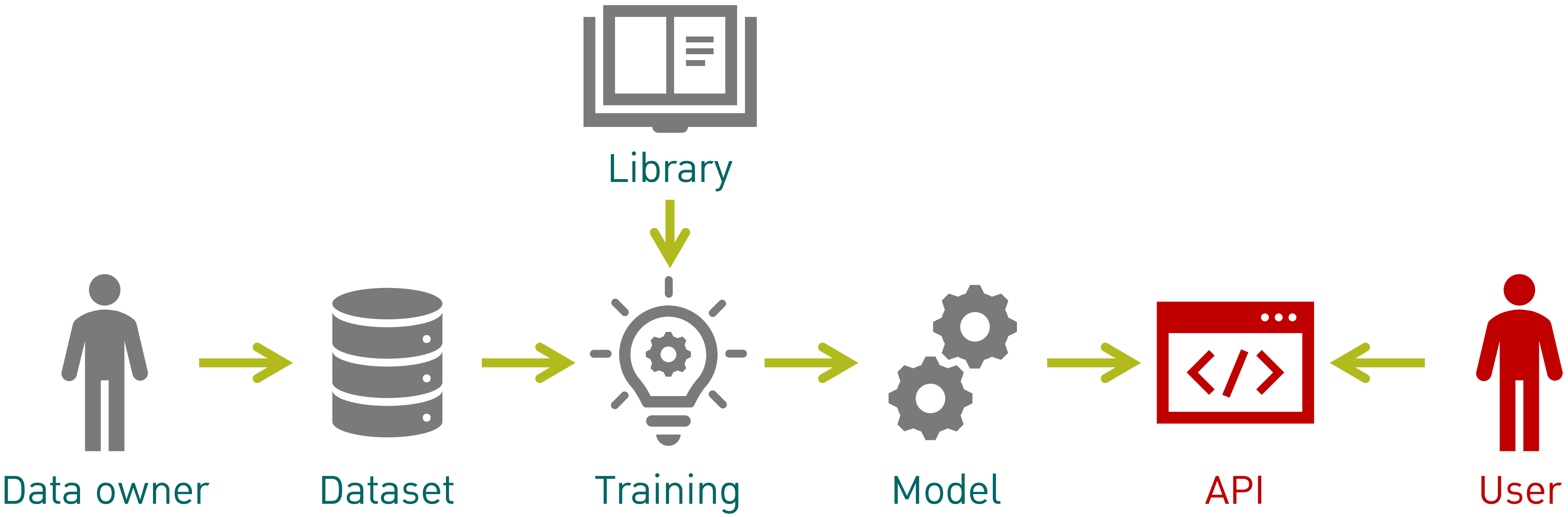
Compromised toolchain

Illustration adapted from *Security, Privacy and ML* by N. Asokan



Malicious input

Illustration adapted from *Security, Privacy and ML* by N. Asokan



Adversarial Attack: Malicious input

https://www.facebook.com/photo?fbid=10218655842060241&
set=gm.3018514041545705

- Given a model, attempt to find a small set of **perturbations** to be added to the model's input
- **Adversarial input** cause the model to output an incorrect answer.

Adversarial Attack: Malicious input

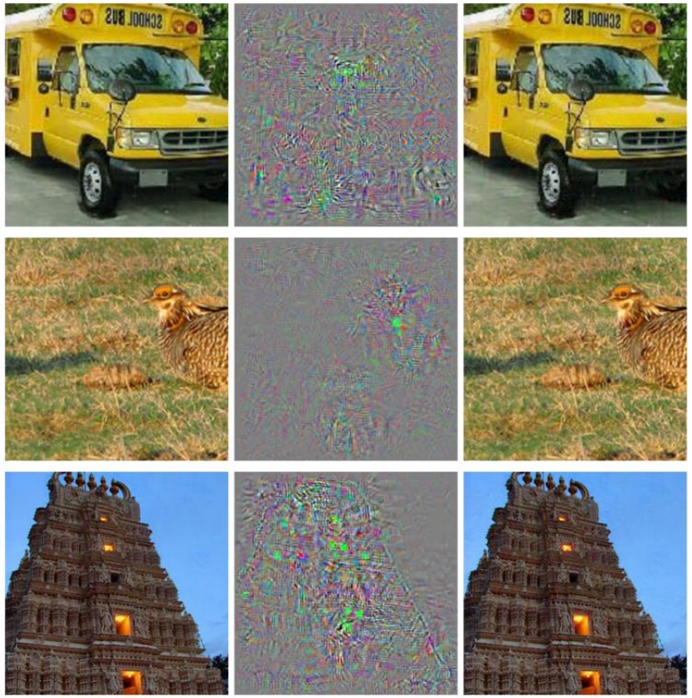
Original input Perturbation Adversarial

Adversarial Perturbation

Perturbation

- Carefully calculated values added to the input
- Computed based on the model's knowledge
  - This will results into more aspects of adversarial learning.

Examples on attacks



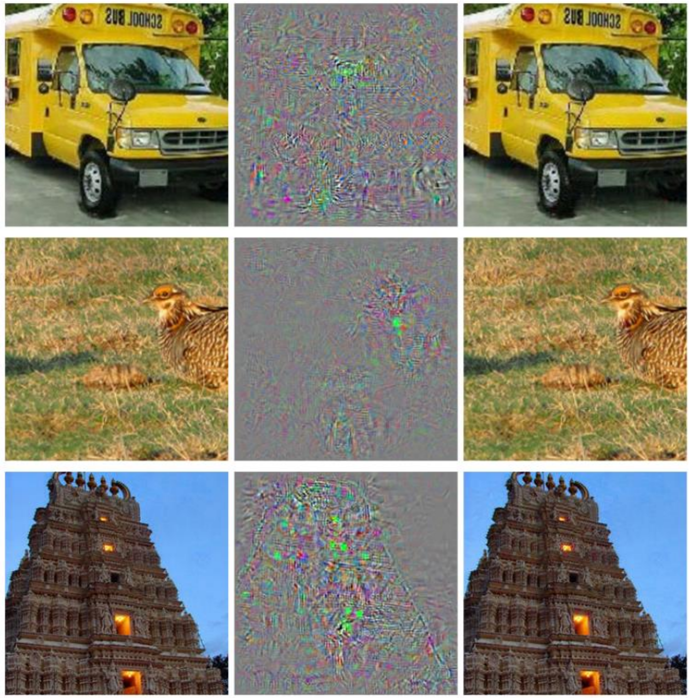
= Ostrich

= Struthio

= Camel

Szegedy et al. *Intriguing Properties of Neural Networks*. ICLR '14 (https://arxiv.org/abs/1312.6199v4)

Our ultimate goal: Defencing system



= Truck

= Tiger

= Pagoda

Szegedy et al. *Intriguing Properties of Neural Networks*. ICLR '14
(https://arxiv.org/abs/1312.6199v4)

Adversarial Foundations

Properties and considerations

- Adversary's goal
  - To misguide or to influence?
- Adversary's knowledge
  - How much can be obtained about the model?
- Victim models
  - What is the motivation of the attacker?
- Security evaluation
  - How can we evaluate the target's *safeties*?

Adversary's goal

- **Untargeted Attack**
  - Interested in misguiding the classifier without any further specifications
  - Example: Misclassifying number recognition
- **Targeted attack**
  - Intends to mislead the classifier to output a specified, intended output
  - Example: Misclassifying number recognition from 3 to 7

Adversary's knowledge

White-box attacking
Classifier structure, parameters, or training sets are known

Grey-box attacking
Although unclear, some parameters are known

Black-box attacking
Only output or probabilities of classes are known

Less model knowledge

Our scope of interest

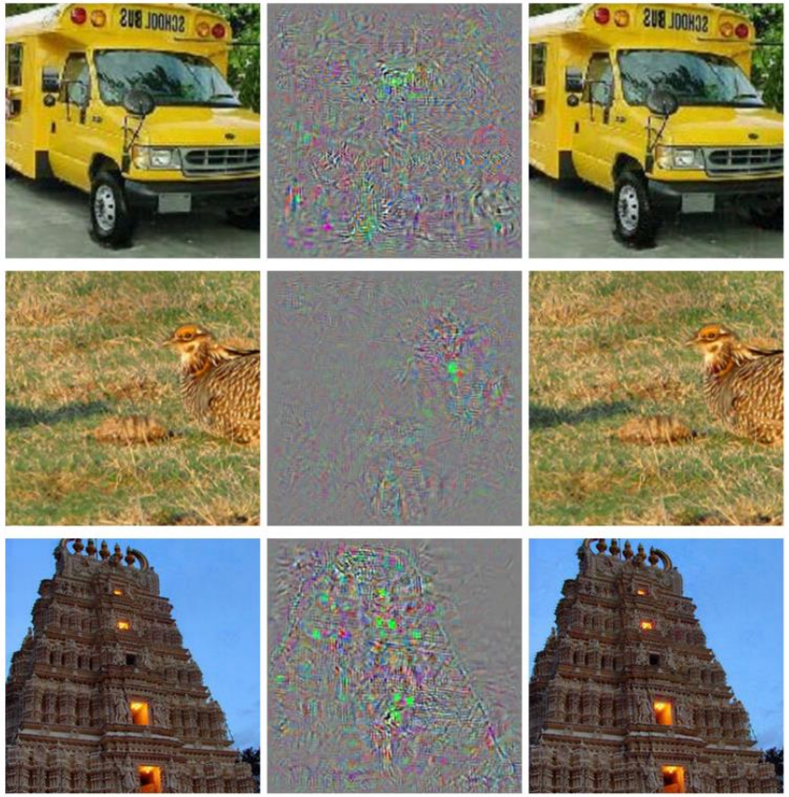
- **White-box model attacking**
  - The most destructive method of attacking
  - Parameters in the model can be used to evaluate attacking efficiency
- **Targeted/Untargeted attack**
  - Reinforcement method should covers both cases

Literature review

Neural Network's *Intriguing* Property

[Szegedy+ 2013, arXiv: 1312.6199v4]

- Very first observation on adversarial attack
- Two "intriguing" properties:
  - The semantic meaning of individual units
    - Out of scope, not to be discussed
  - Network's tolerance to small perturbations

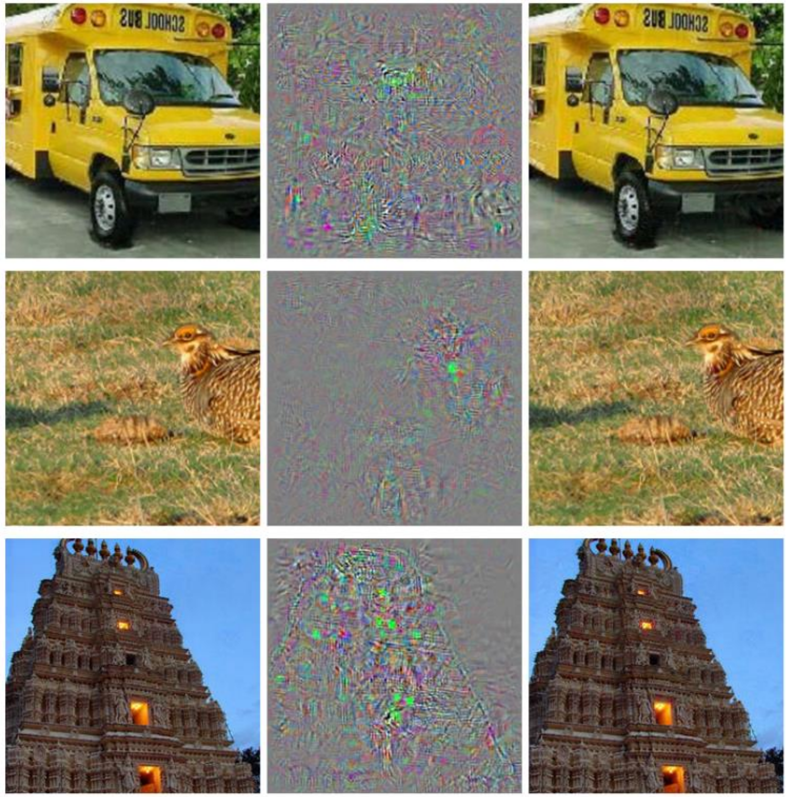


Szegedy et al. *Intriguing Properties of Neural Networks*. ICLR '14 (https://arxiv.org/abs/1312.6199v4)

Neural Network's tolerance to perturbations

[Szegedy+ 2013, arXiv: 1312.6199v4]

- Networks that are **generalised** well should be tolerated to small perturbations
- Maximising the prediction error by modifying the input image with additional constraint of *invisible* perturbation is possible.



Szegedy et al. *Intriguing Properties of Neural Networks*. ICLR '14 (https://arxiv.org/abs/1312.6199v4)

Explanations on adversarial

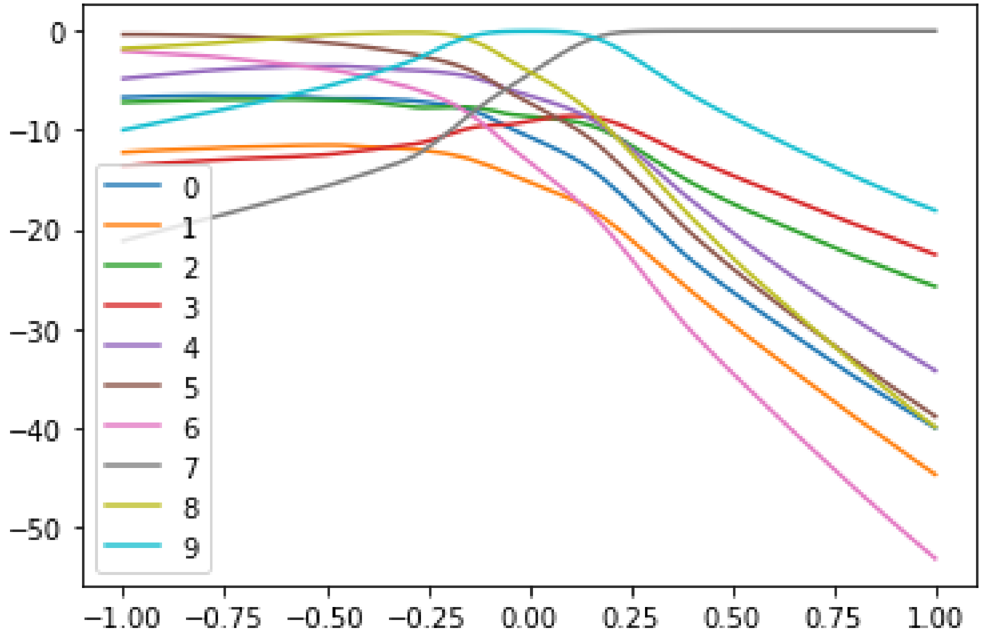
```
multivax:~ $ ./query

THERE IS AS YET
INSUFFICIENT TRAINING
DATA FOR A MEANINGFUL
STRONG ANSWER
```

Joke adapted from Isaac Asimov's *The Last Question*

- **Model's nonlinearity**
  [Szegedy+ 2013, arXiv:1312.6199v4]
  - All softmax-based classification models return a set of conditional probability P(class|input)
  - The neural networks' extreme nonlinearity combined with insufficient training data points cause such exploits
  - This is just a hypothesis

Explanations on adversarial



- **Model's linearity**
  [Goodfellow+ 2014, arXiv: 1412.6572]
  - Goodfellow and his team argued that it's not the nonlinearity, but linearity, that cause such an exploit
  - Increasing perturbation density shows a strong probability linear behaviour
  - "Accidental steganography": Forcingly attend the network to the most weight-aligned values

Explanations on adversarial

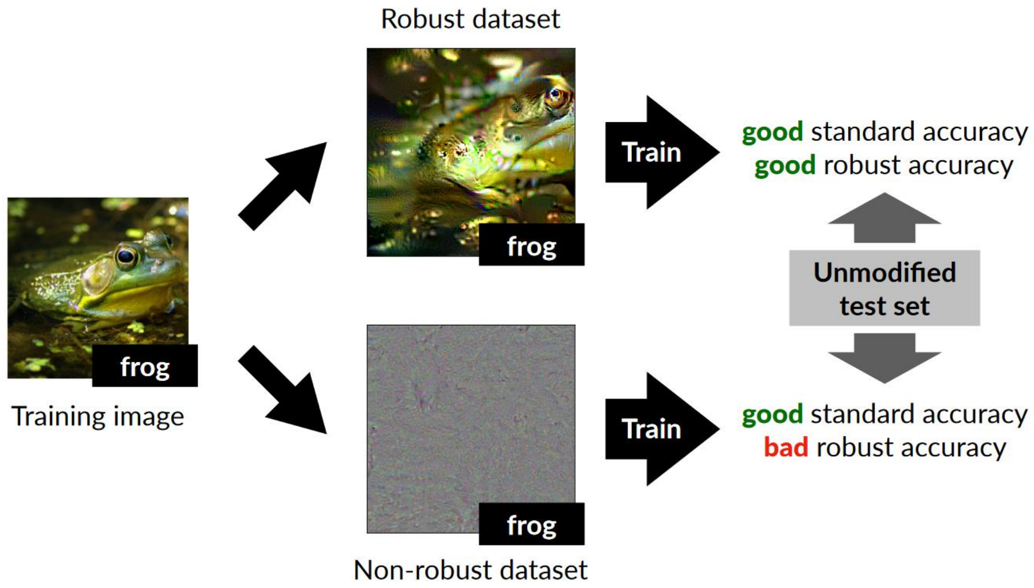
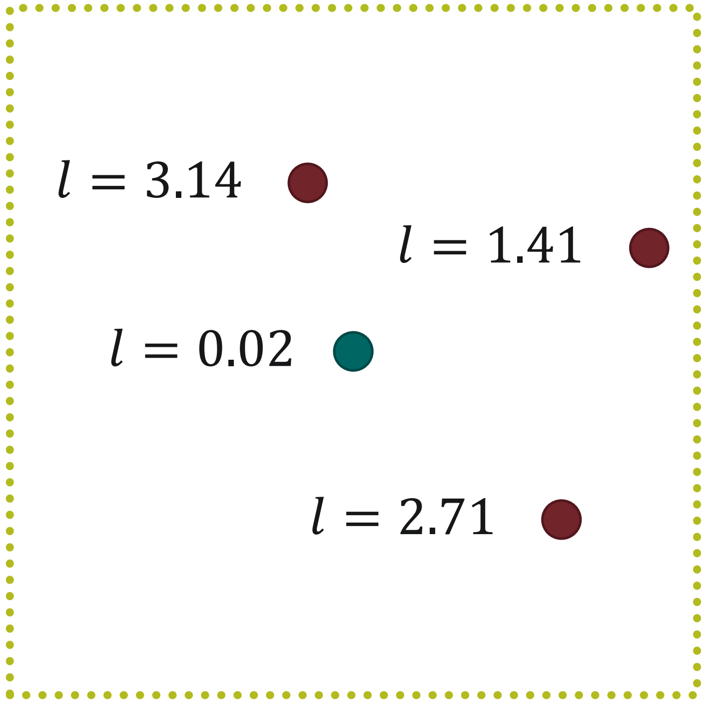


Illustration from the original paper (arXiv: 1905.02175)

- **Robust and non-robust features**
  [Ilyas+ 2019, arXiv: 1905.02175]
  - "Adversarial vulnerability is a direct result of our models' sensitivity to well-generalizing features in the data"
  - Robust features are perceptible by humans, non-robust features are imperceptible

Calculating the perturbation

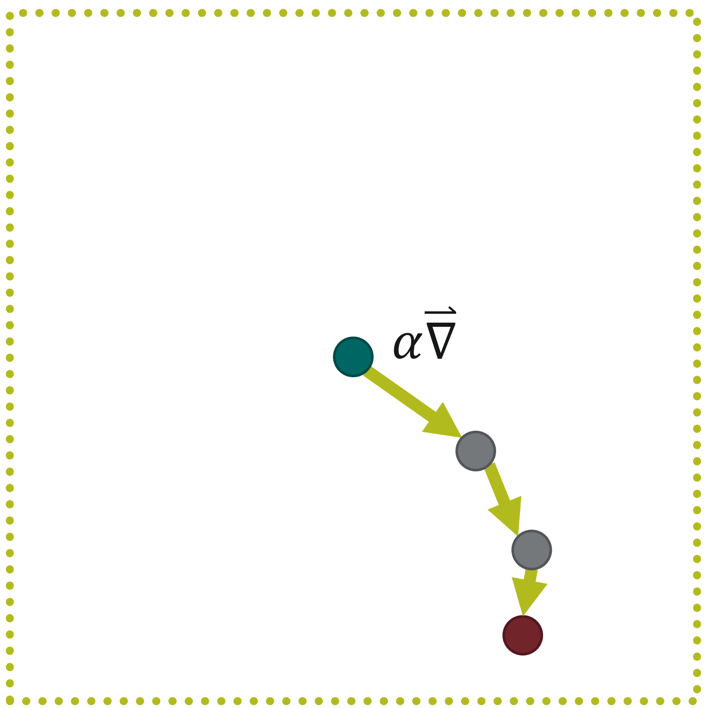


Boundary according to norm

Given an input to be attacked that lies in an input space…

- Define the "invisibility" measurement
  - **Norm or other constraints**
- Find the perturbation which maximise such loss function within the constrained norm
  - **Optimisation problem**
- There exists many perturbations, but their *power* may not be equal

Straightforward: Loss maximisation

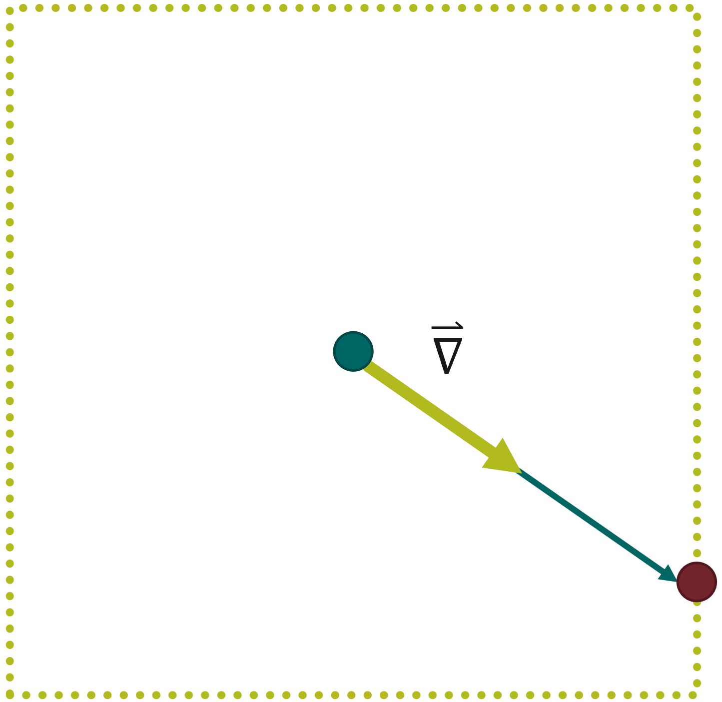


Boundary according to norm

- Iteratively maximise the loss while maintaining values inside the boundary
- Very straightforwardly done
- Targeted attack can be achieved by defining the targeted loss function to maximise

Fast Gradient Sign Method (FGSM)

[Goodfellow+ 2014, arXiv: 1412.6572]

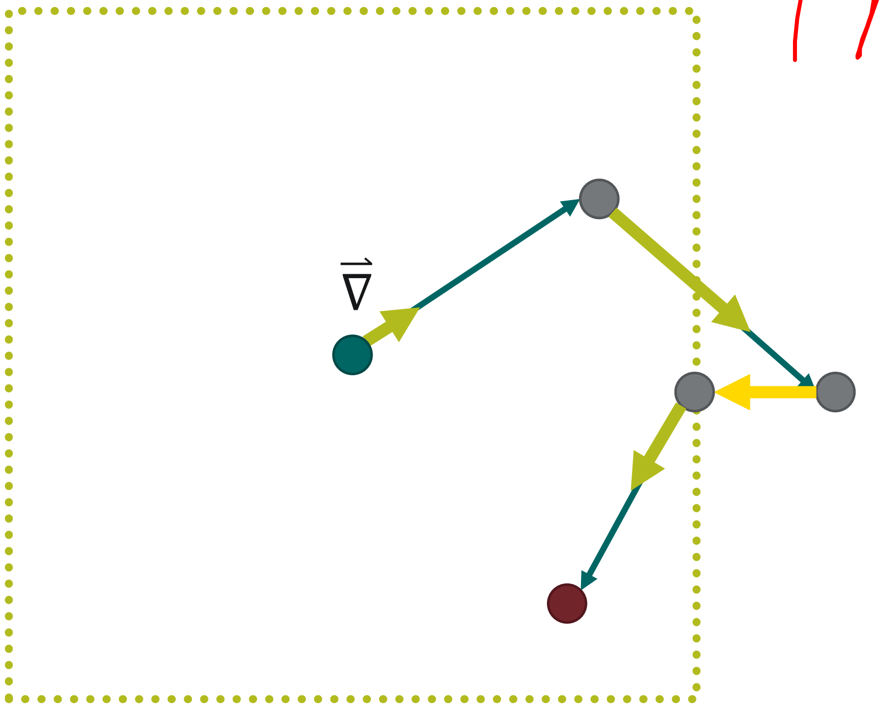


Boundary according to norm

- This is the result from linearity explanation
- Calculate the gradient of input respective to the loss function
- Project it to maximise the acceptable norm
  - Motivation based on the attacking of model linearity
- Non-iterative, constant runtime

Projected Gradient Descent (PGD)

[Szegedy+ 2017, arXiv: 1706.06083]



Boundary according to norm

- Repeat iteratively:
  - Calculate the gradient of loss function
  - Project it according to the desired distant
  - Project back into the boundary should the perturbation exceeds the acceptable norm
- Observation: The projection distant is constant regardless of gradient size

PGD vs FGSM

PGD

- Iterative method, thus consumes time
- Finds the "worst" and "most powerful" perturbation

$$\tilde{x} = x + \alpha \mathrm{sign}\left(\vec{\nabla}_x \mathcal{L}(x, y)\right)$$

FGSM

- Approximation method, constant runtime
- Finds the perturbation, but not the "worst" one

$$\tilde{x}_n = \tilde{x}_{n-1} + \alpha sign\left(\vec{\nabla}_x \mathcal{L}(x, y)\right)$$

Algorithm: Model retraining

- For each epoch:
  - For each minibatch:
    - Calculate perturbations on each minibatch
    - Append the perturbation to the training set
    - Train the model

Extremely slow

When computed iteratively

Computationally slow

$O(b) \times$ perturbation calculation complexity

Longer backpropagation

As the dataset length is twice increased

Knowledge is power

- The linear runtime was reduced to constant runtime using **only one assumption on linearity**
- Good assumption are key points to faster methods in perturbations generation

$$\tilde{x} = x + \alpha \boxed{\operatorname{sign}\left(\vec{\nabla}_x \mathcal{L}(x, y)\right)}$$

An *intriguing* question

Can we determine the perturbations behaviour

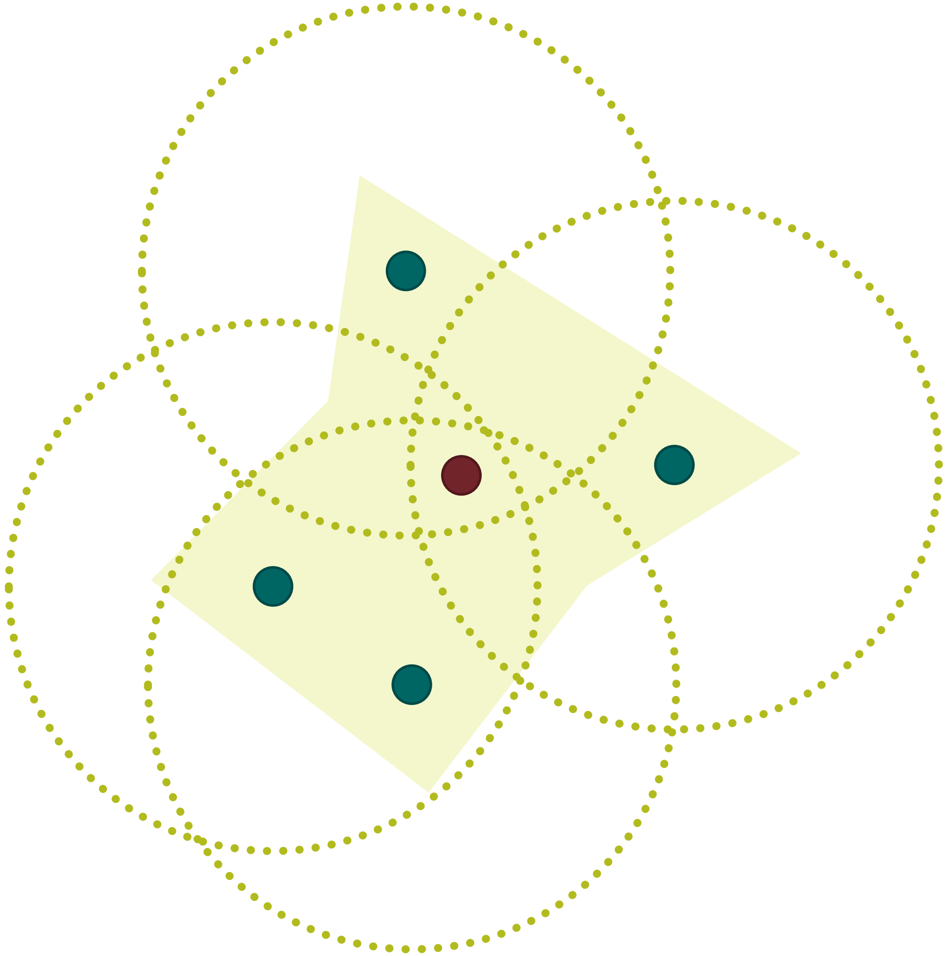
using **clustering**?

Motivations on Cluster

Our motivation: Clusters of Data

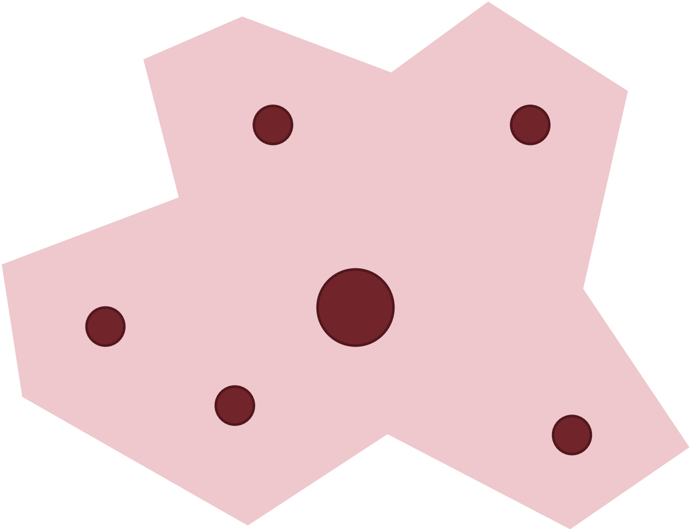
- We can run unsupervised learning on training points to cluster them into groups
- We can calculate the perturbations for the samples and cluster them into the same manner
- **What are our motivations to study both types of clusters?**

Clustering Analysis on Training Points



- Training points in the same class are near to each other in feature space
- These training points will be clustered into the same cluster
- There exists a perturbation that can attack all training points in the cluster

Clustering Analysis on Perturbations

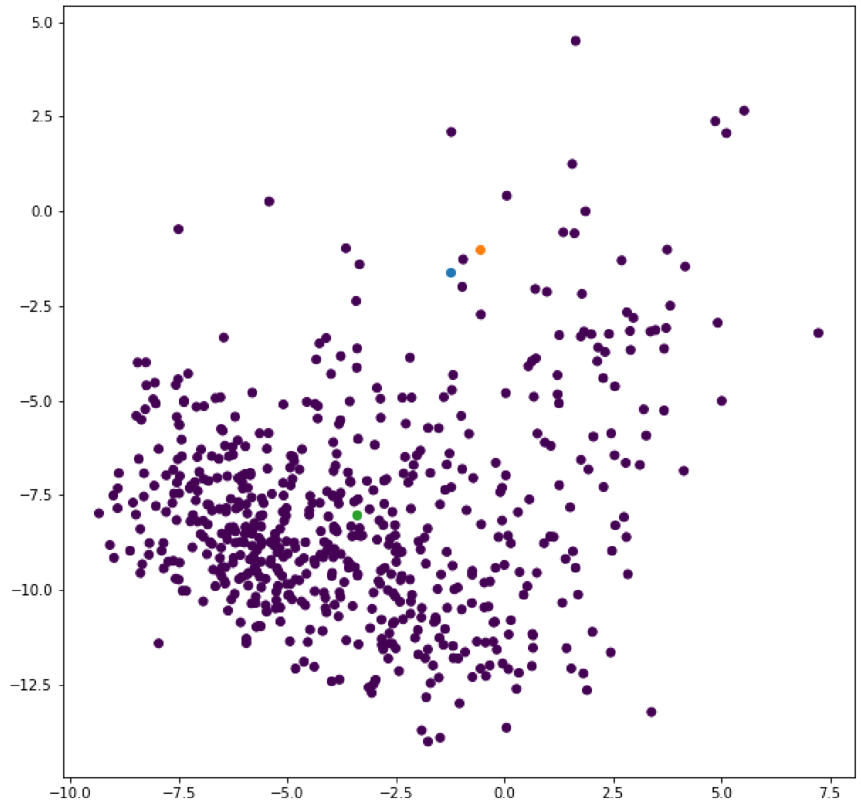


- Perturbations can be clustered into groups which are near in space
- Those perturbations can cross-attack the samples used to generate them
- The perturbation nearest to the cluster's centre can "represent" the entire cluster, thus capable of attacking the samples

Our works

Clustering analysis

- ~~Can perturbations be clustered?~~
- Are there any meaningful insights from inter-cluster and intra-cluster analysis?
  - Inter-cluster similarity?
  - Distribution?
  - Attacking performance?



Concept: Cluster Fast, Adversarial Fast

- Given all training points, generate perturbations for each training points in a fast manner, regardless of its efficiency in attacking
  - Fast way to understand the behaviour of the perturbations
- Cluster the training points
- In each cluster, find a perturbation to attack the **entire** training set efficiently
  - **Effective way to attack the model while saving time**

Algorithm: Model retraining

- For each epoch:
  - For each minibatch:
    - Calculate perturbations on each minibatch
    - Append the perturbation to the training set
    - Train the model

Extremely slow

When computed iteratively

Computationally slow

$O(b) \times$ perturbation calculation complexity

Longer backpropagation

As the dataset length is twice increased

Our proposed method

- For each epoch:
  - For each minibatch:
    - Calculate perturbations on each minibatch
    - Append the perturbation to the training set
    - Train the model

No free lunch

k-Means overhead

Eliminate reluctant calculation

By calculating lower amount of perturbation

Smaller batch size

By cluster-based representation

Faster backpropagation

Using weighted loss

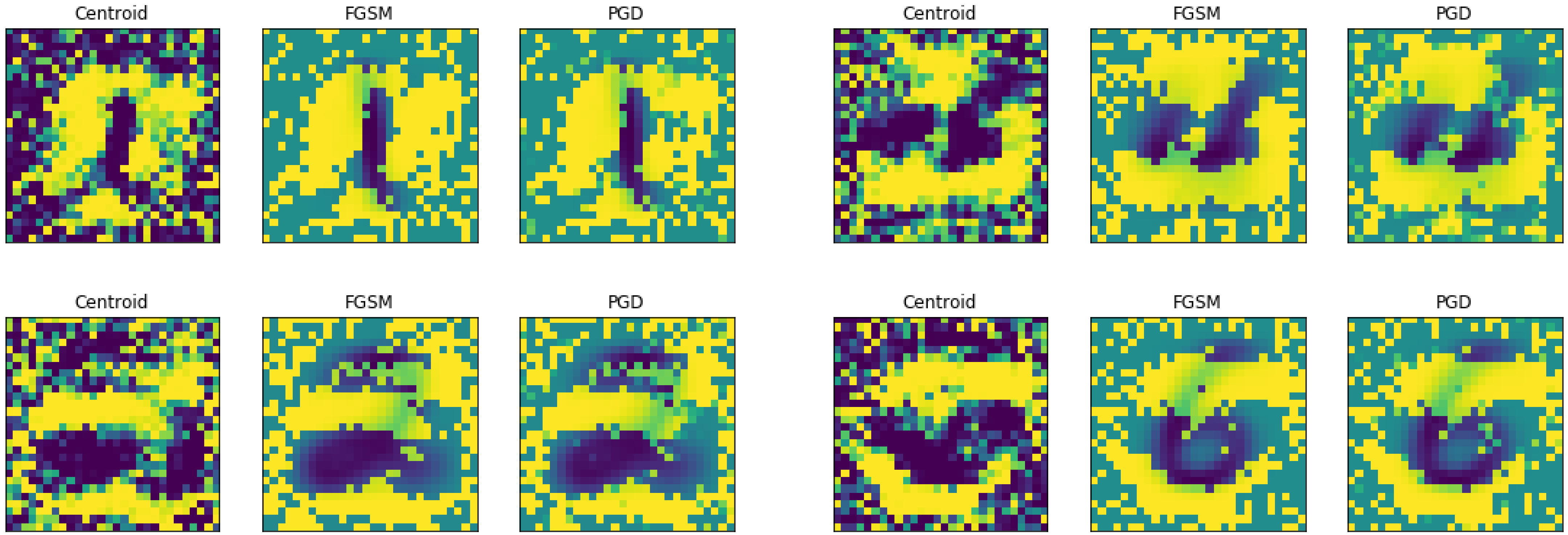
Algorithm 1: k-Perturbation

Input: data to attack, data to cluster, k

Returns: [indices of perturbations, perturbations, clustering result]

- Indices of perturbations = Perturbations = [empty list]
- Cluster the data to cluster using k-Means algorithm into k clusters
- For each cluster
  - Obtain the data points with the same indices as the cluster data
  - Calculate the perturbation that will attack such data points
  - Append the indices to indices of perturbations list
  - Append the perturbation to the perturbation list
- Return the variables

k-Perturbation results



Note: Extremely randomly selected. No cherry-picking on examples.

Which one is from k-Perturbation?

Algorithm 2: k-Reinforce

Input: Training set, k, e, m, m’, w, w’

Returns: Model

- Run the k-Perturbation algorithm
- For each epoch:
  - For each minibatch of size m:
    - Sample the adversarial minibatch of size m’
    - Append the perturbation to the training set
    - Train the model using weighted loss w and w’ on m and m’ respectively

No free lunch

k-Means overhead

Eliminate reluctant calculation

By calculating lower amount of perturbation

Smaller batch size

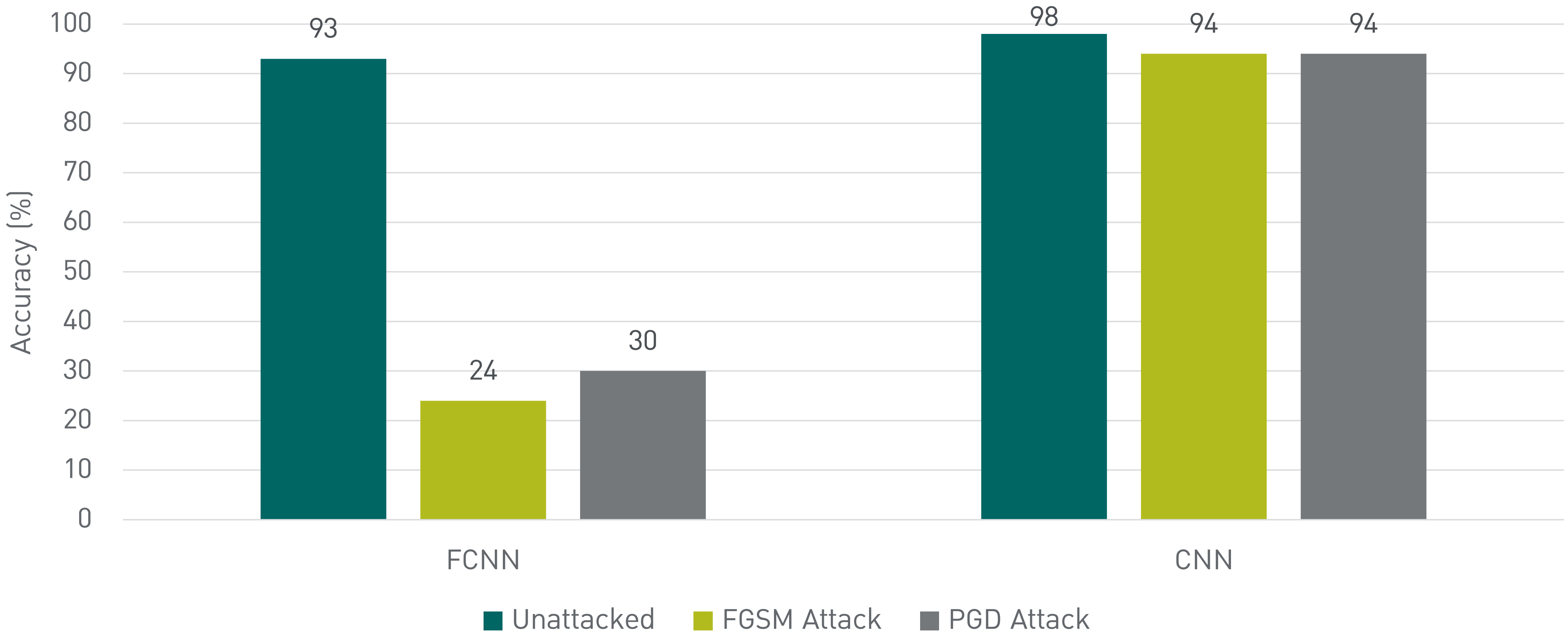
By cluster-based representation

Faster backpropagation

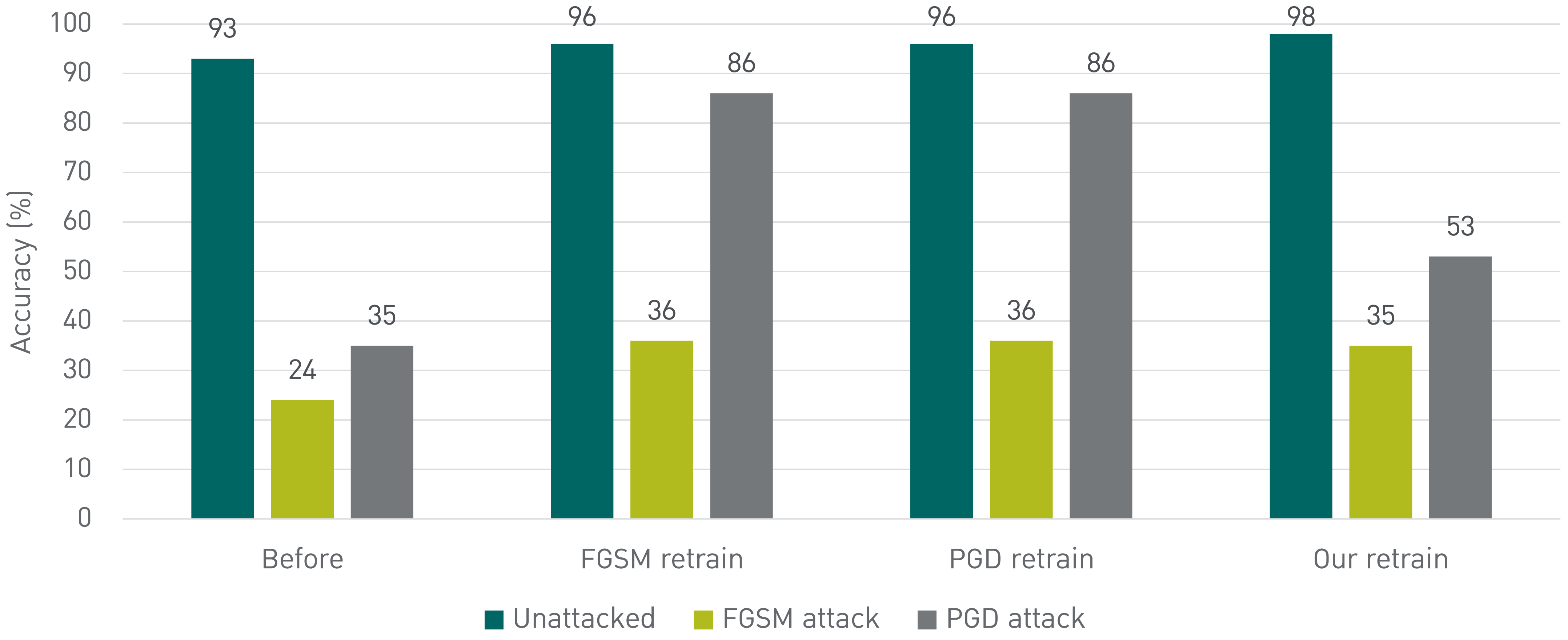
Using weighted loss

Results

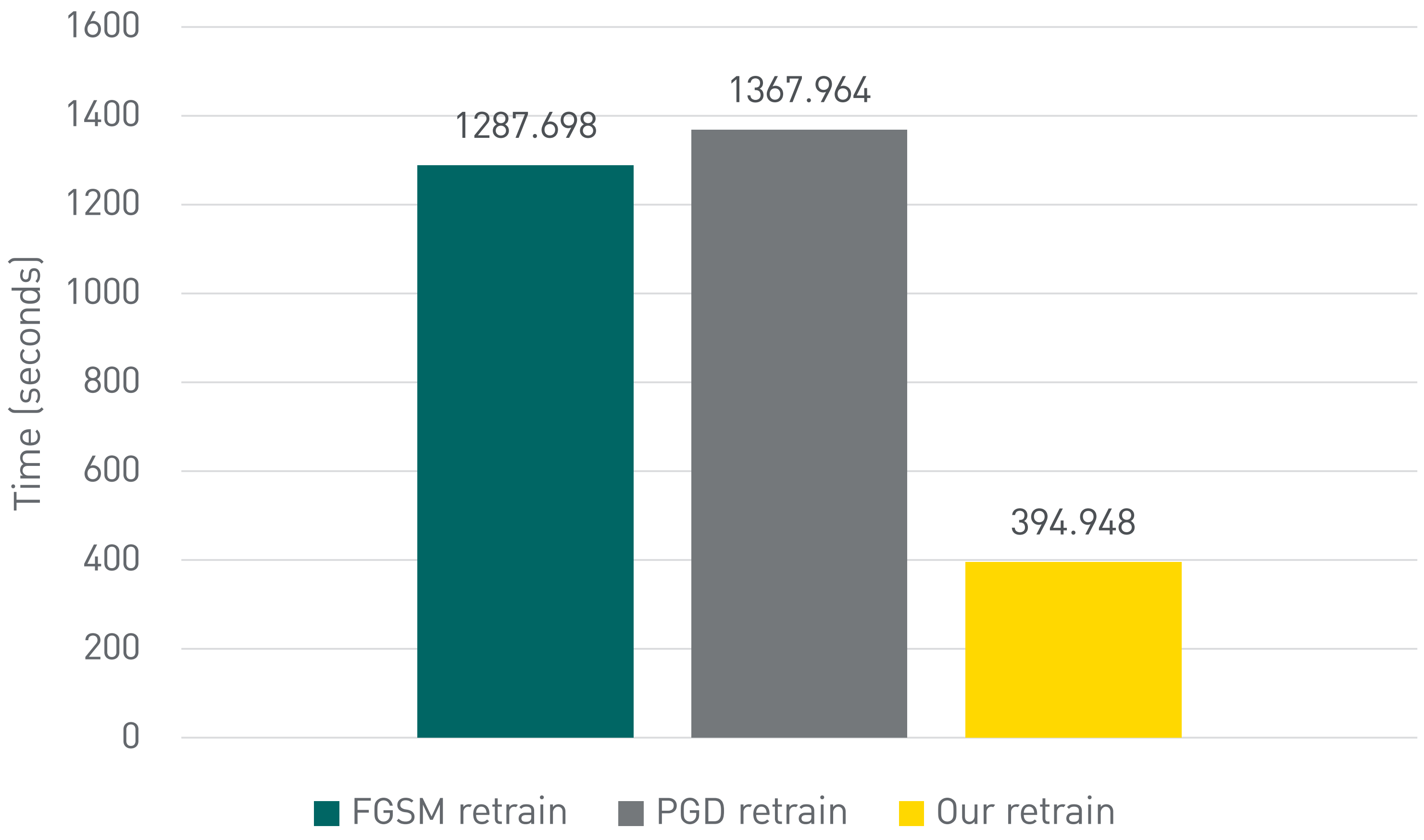
Base model accuracy



After reinforcing



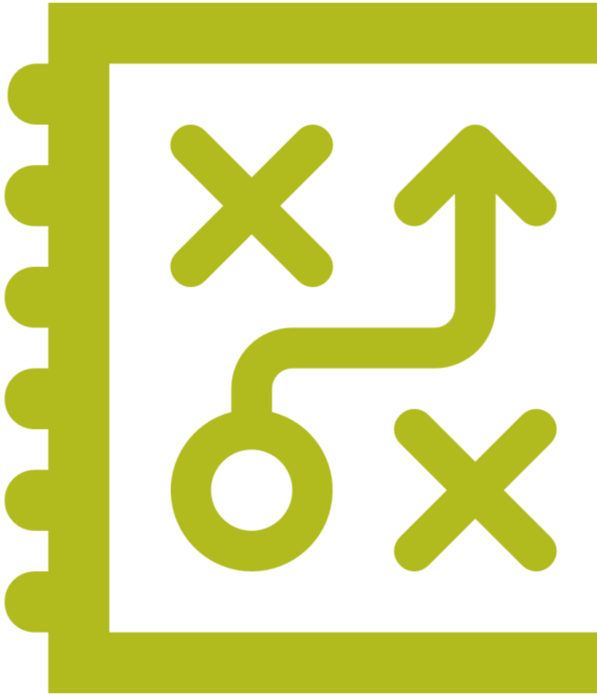
Retraining time



More than
4x
faster

Further improvements

- What if we recalculate the perturbations on every iteration?
- What if we apply further cluster knowledges?
- What if other state-of-the-art methods were blended into our method?



Acknowledgements

Adversarial team

Advisors and Co-Advisors

Asst. Prof. Dr.
Jittat F.

Asst. Prof. Dr.
Thanawin R.

Graduate Students / Researcher

Asst. Prof. Vacharapat M.

Pongsakorn A.

Monthol C.

Undergraduate

Sirakorn L.

Adversarial Learning using Cluster-based Method

Sirakorn Lamyai

KASETSART UNIVERSITY

THEORY Research Group