AI System Evaluation

Week 3: Improving AI Robustness

Aug 23 - Week 1: 7-10	Introduction	
Aug 30 - Week 2: 7-10	AI Robustness	Exercise 1
Sep 06 - Week 3: 7-10	Improving AI Robustness	Exercise 2
Sep 13 - Week 4: 7-10	AI Backdoors	Exercise 3
Sep 20 - Week 5: 7-10	Mitigating AI Backdoors	Exercise 4; Project Proposal
Sep 27 - Week 6: 7-10	AI Fairness	Exercise 5
Oct 11 - Week 7: 7-10	Improving AI Fairness	Exercise 6
Oct 18 - Week 8: 7-10	AI Privacy	Exercise 7
Oct 25 - Week 9: 7-10	Improving AI Privacy	Exercise 8
Nov 01 - Week 10: 7-10	AI Interpretability	Project Due
Nov 08 - Week 11: 1-3	End-of-Term Exam	

Understanding Adversarial Samples

Understanding Adversarial Samples

Question

Where do adversarial samples come from?

Why is there adversarial transferability (i.e., an adversarial perturbation generated based on sample A may work for sample B as well, and an adversarial perturbation generation for model A may work for model B as well)?

One Possible Explanation

“Adversarial Examples Are Not Bugs, They Are Features” (NeurIPS 2019).

“Adversarial vulnerability is a direct result of sensitivity to well-generalizing features in the data.”

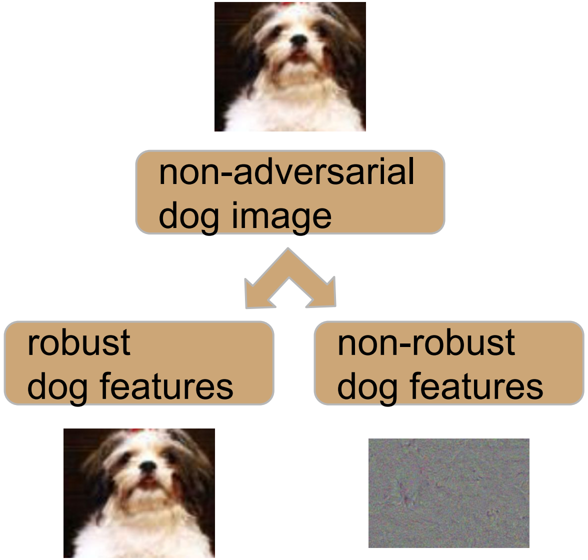
Adversarial Samples are Features

Intuitive Idea

Deep learning is good at picking up a variety of features that are correlated with the labels.

Some features are meaningful to humans (called robust features); some are not (called non-robust features).

Illustration



non-adversarial dog image

robust dog features

non-robust dog features

Adversarial Samples are Features

Intuitive Idea

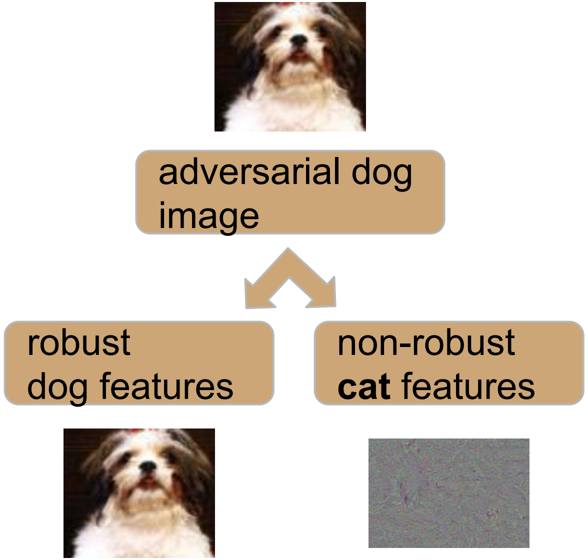
Question: "Why are there adversarial samples?"

Answer: Adversarial samples are samples whose non-robust features are of those of the target label.

Question: "Why is there adversarial transferability?"

Answer: Because they are legit cat features!

Illustration



adversarial dog image

robust dog features

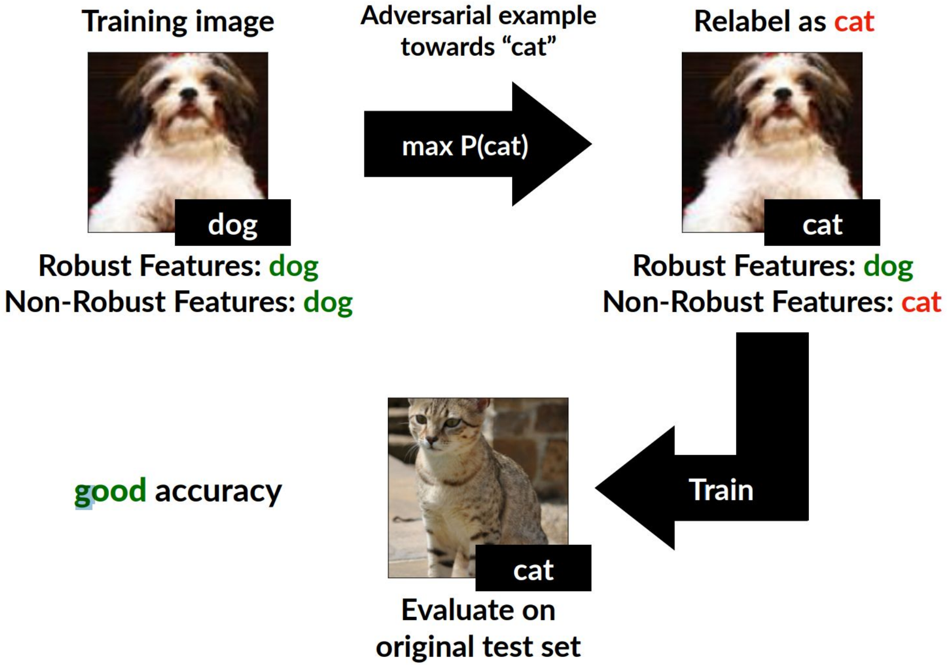
non-robust **cat** features

Adversarial Samples are Features

How do we validate this hypothesis?

Experiment

1. Generate many adversarial samples.
2. Label the adversarial samples with the target label.
3. Train a new neural network using the adversarial samples.
4. Test the accuracy of the newly trained neural network using the original test set.



Exercise 0: Discussion

- Do you agree with “adversarial samples are features”? Either way, can you think of a way to validate your claim?
- The suggestion is that the distinction between robust features and non-robust features is based on human vision systems. If so, to be robust, we should perhaps ignore those non-robust features. Do you think that is a good idea?

Improving Robustness

Outline

The question

How do we improve the robustness of an AI system?

Approaches

Input Transformation or Filtering

Model Enhancement

- Data augmentation
- Adversarial training
- Certified training
- Randomized smoothing

Input Transformation

Input Transformation

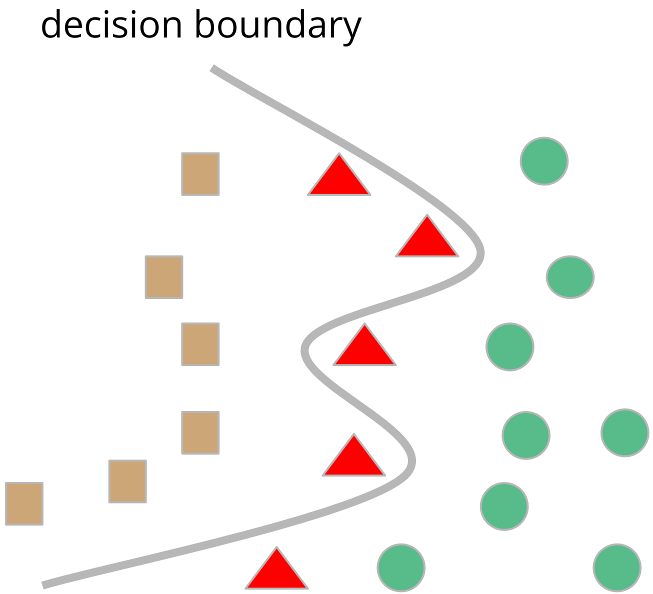
Intuition

Adversarial samples are often not robust, and thus we can transform an input (in some label-preserving way) to reduce the adversarial effect before feeding it into the neural network.

🟫 Positive normal samples

🔺 Adversarial samples

🟢 Negative normal samples



Exercise 1

Given the model week3/exercise1/MNIST.pt, and the images in the folder week3/exercise1/toattack_adv (which are 20 adversarial samples generated PGD),

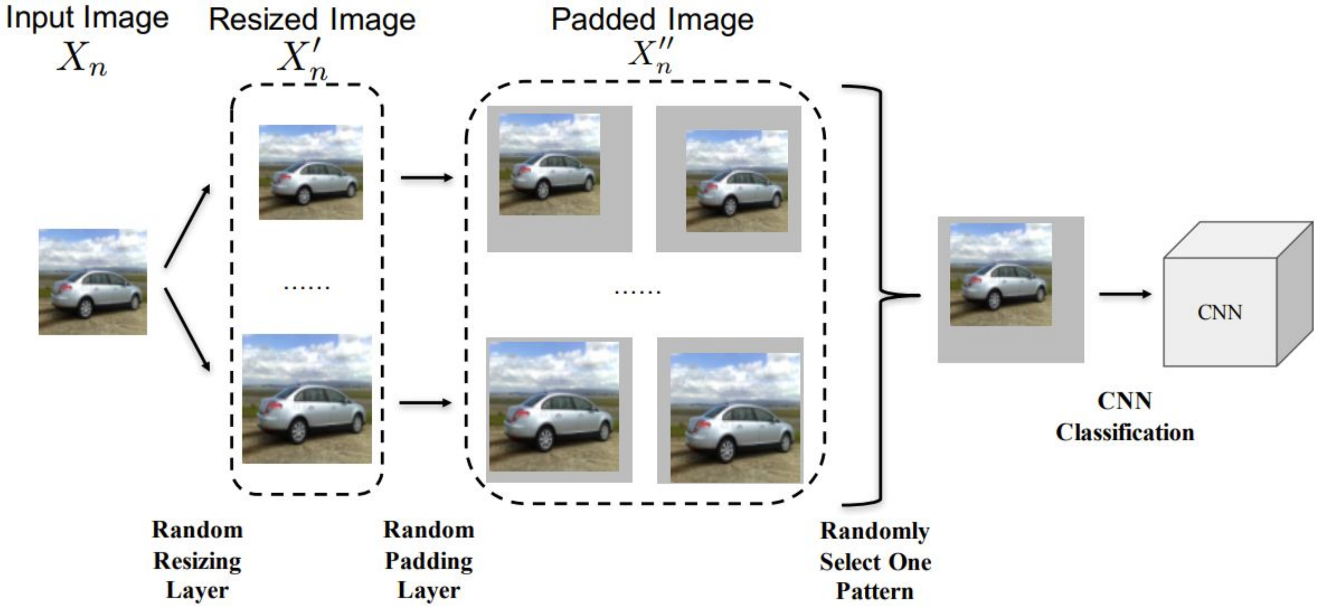
1. Modify the noise scale
2. (Take home) Make 1 more random minor modification (that is expected not to change the label according to human eyes)

Check whether the modification changes the prediction.

Modification	# of Labels Changed
uniform noise (-0.01,0.01)	

Input Transformation

Approach: Randomly resize the image, randomly pad 0 around the image



Input Transformation

Approach

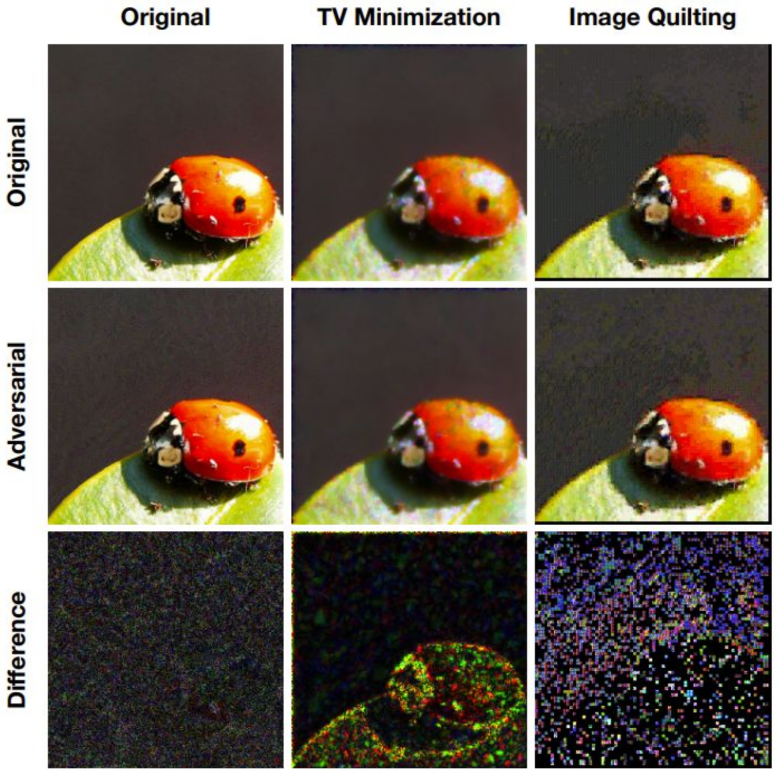
Image cropping-rescaling

Bit-depth reduction

JPEG compression and decompression

Total variation minimization (e.g., drop some pixels and reconstruct them based on the surrounding ones)

Image quilting (i.e., piecing together small patches that are taken from a database of image patches).



Input Transformation

Approach

Spell check: detecting and correcting misspellings and unknown words to remove adversarial effects.

Example

I watched this movie recently mainly because I am a Huge fan of Jodie Foster's. I saw this movie was made right between her 2 Oscar award winning performances, so my expectations were fairly high. Unfortunately ~~Unf0rtunately~~, I thought the movie was terrible ~~terrib1e~~ and I'm still left wondering how she was ever persuaded to make this movie. The script is really weak ~~wea k~~.

Input Filtering

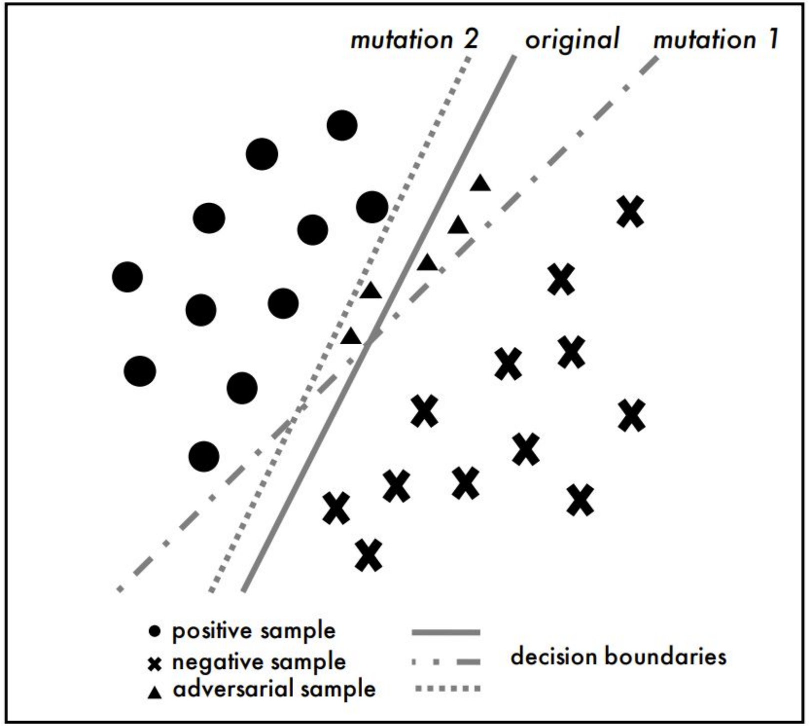
Approach: Model Mutation Testing

Given a neural network N, slightly mutate N to generate a committee: N_1, N_2, …, and N_k.

Given a sample x, measure how often $N_i(x)$ is different from N(x), which is called label-change-rate.

The higher the label-change-rate, the more likely x is adversarial.

If x is suspected to be adversarial, discard it.



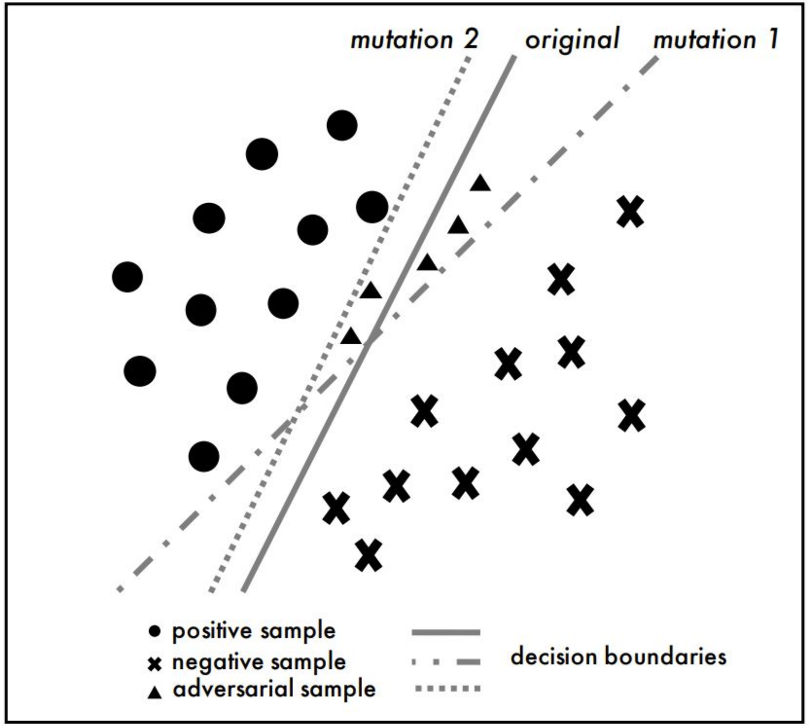
Input Filtering

Approach: Model Mutation Testing

How to mutate a neural network?

- Gaussian fuzzing, i.e., add Gaussian noise to the neuron weights
- Weight shuffling and neuron switching, i.e., exchange
- Neuron activation inverse, change the activation status of a neuron

Only models with a high accuracy are kept.



Robust Adversarial Sample

Attacker’s intuition

All these approaches are based on a common feature of these adversarial samples, i.e., they are not robust.

Let’s generate robust adversarial samples to defect these approaches.

Example

PMLR’18: “Synthesizing robust adversarial samples”

ICLR’22: “Provably Robust Adversarial Examples”

Robust Adversarial Samples

Expectation Over Transformation (EOT)

The key idea is to search for adversarial samples that are robust through a distribution of transformations.

The approach is to maximize the expected probability of having the target label after a set of predefined transformation.

Approach

Optimize the following objective

$$\begin{aligned} \underset{x'}{\arg\max} \quad & \mathbb{E}_{t\sim T}[\log P(y_t|t(x'))] \\ \text{subject to} \quad & \mathbb{E}_{t\sim T}[d(t(x'), t(x))] < \epsilon \\ & x \in [0,1]^d \end{aligned}$$

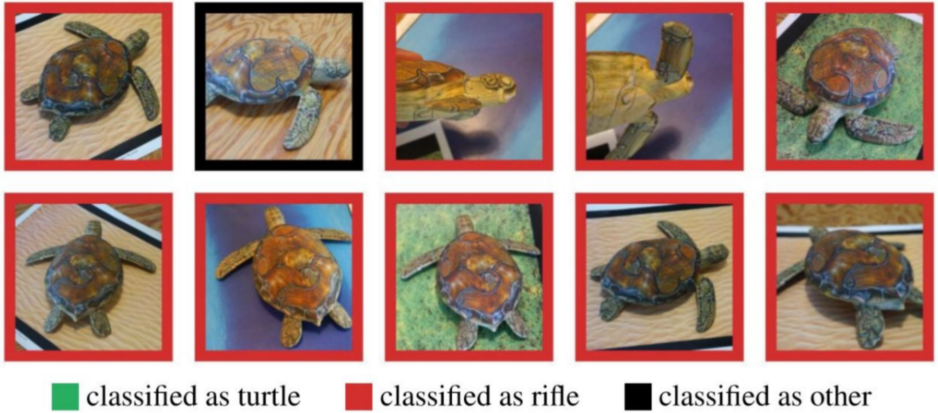
T is a distribution of transformations; t is a transformation function; y_t is the target label of the adversarial sample; d is a distance function.

Robust Adversarial Samples

Approach

Identify a set of transformations (such as change of angle, lighting, or those of the proposed defence methods e.g., JPG compression).

Generate adversarial samples that are robust through such transformations.



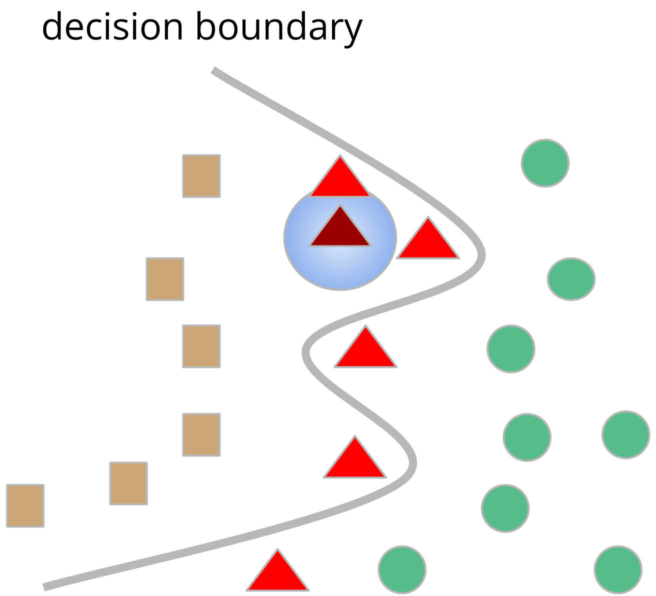
EOT is also the method used to generate 3D physical adversarial attacks.

Robust Adversarial Samples

Provably robust adversarial sample

The goal is to find adversarial samples that are provably robust, e.g., all samples within the L∞-norm ball of the adversarial sample are adversarial.

Illustration



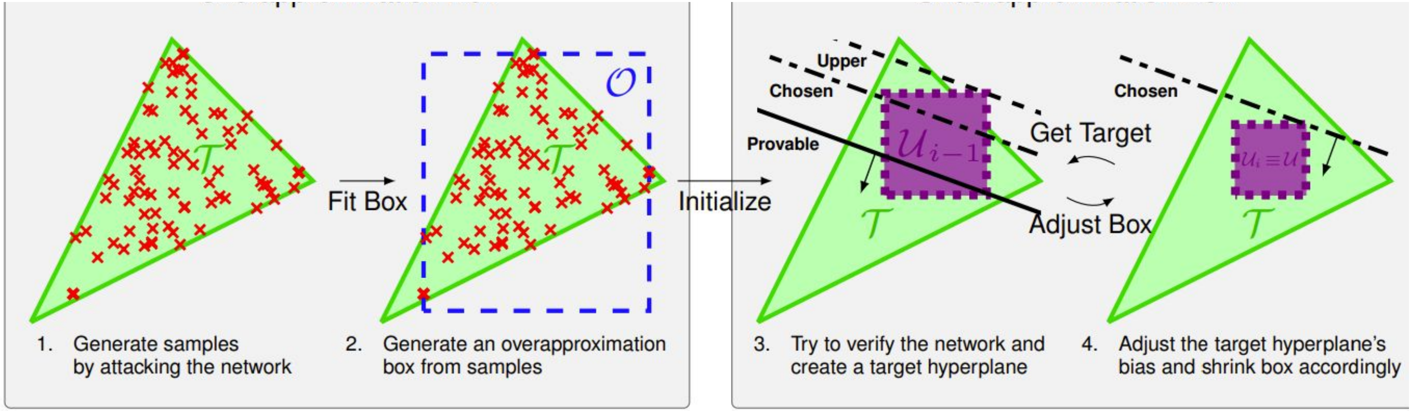
Robust Adversarial Samples

The Approach

Generate many adversarial samples; identify ones that are promising; and apply robust verification to prove that they are robust!

The Result

(ICLR’22) It is shown that this is feasible for models trained on MNIST and CIFAR.



1. Generate samples by attacking the network
2. Generate an overapproximation box from samples
3. Try to verify the network and create a target hyperplane
4. Adjust the target hyperplane’s bias and shrink box accordingly

Quick Discussion

Q1: Based on your high-level understanding, discuss whether EOT is likely effective against the input transformation or filtering methods shown on slide 14 to 17. Why or why not?

Q2: Are there other features of adversarial samples that can be used to detect/filter them?

Although not perfect, input transformation/filtering makes it harder to attack.

Model Enhancement

Model Enhancement

General Idea

Let’s build robust models which work correctly given any input sample (even adversarial ones).

We can enhance either the training data, the training objective, or the model architecture.

Approaches

Data augmentation

Adversarial training

Certified training

Randomized smoothing

Data Augmentation

Approach

Apply existing adversarial attacking methods to generate a set of adversarial samples.

Label the adversarial samples correctly and add them into the training .

Retrain with the additional training data.

Example

Given a model for sentiment analysis, generate adversarial samples using existing approaches (e.g., select and replace a word with synonyms).

Label these adversarial samples with the same label as the original sample.

Retrain.

Data Augmentation

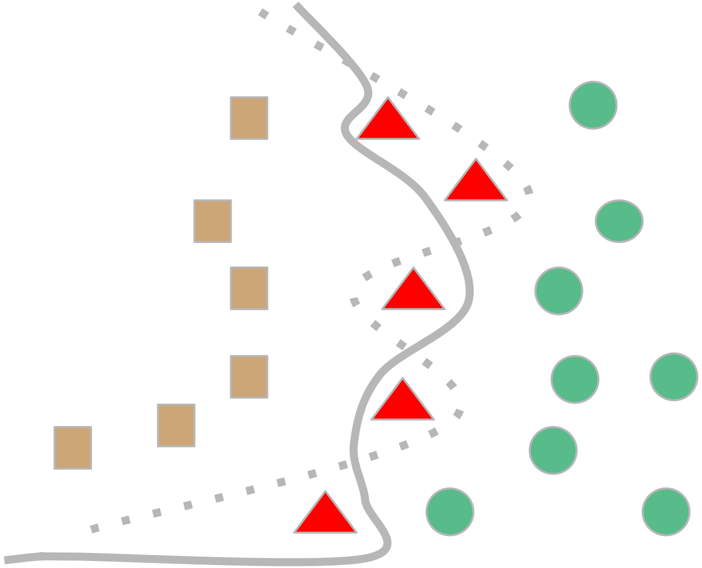
Pros

It’s easy to apply. It works with images, texts, voices, videos and so on.

Cons

The additional training data may alter the distribution of the data.

It is often not very effective. Why?



new decision boundary

Adversarial Training

Overall Idea

Adversarial training attempts to improve the robustness of a neural network by training it with an objective function which minimizes the effect of adversarial samples.

Approach

Training the neural network with the following objective function:

$$\text{Min}_{\theta}\ \text{Max}_{d(x,x')<\varepsilon}\ L(\theta, x', y)$$

where L(θ, x’, y) is the adversarial loss, with network weights θ, adversarial input x’, and ground-truth label y; and d(x,x’) constraints the perturbation allowed (e.g., L-norm).

Adversarial Training

Approach

The idea is to minimize the maximal effect of adversarial attacks:

$$\text{Min}_{\theta}\ \text{Max}_{d(x,x')<\varepsilon}\ L(\theta, x', y)$$

Analysis

$\text{Max}_{d(x,x')<\varepsilon}\ L(\theta, x', y)$ is the maximal adversarial effect achieved an attacker.

How do compute that?

For adversarial training, we approximate it using a concrete adversarial attack.

For certified training, we compute it formally.

Adversarial Training

Approach

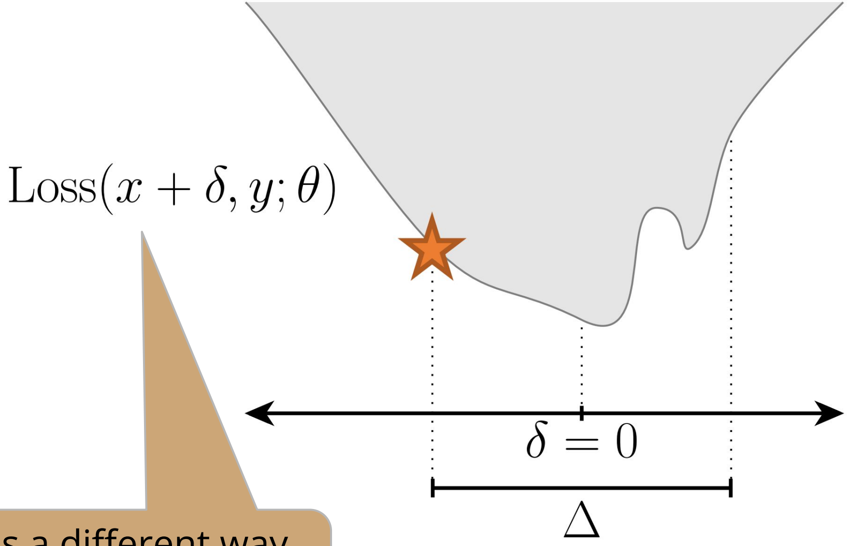
During training, approximate the inner maximization problem

$$\mathrm{Max}_{d(x,x')<\varepsilon}\ L(\theta, x', y)$$

using an adversarial attack method.

Note that the maximal loss generated by the attack is an under-approximation.

The stronger the attack, the better an under-approximation it is.



This is a different way of writing the loss.

Adversarial Training

Adversarial Training With FGSM

It is very efficient.

It is effective against FGSM adversarial attack.

It is often not effective against other attacks (such as PGD).

Adversarial Training With PGD

With PGD based on L_∞-norm, it is effective almost all L_∞-norm based attacks, and is not effective against L_1-norm, or L_2-norm based attacks.

It is inefficient, e.g., PGD adversarial training on a simplified ResNet for CIFAR-10 requires approximately three days on a TITAN V GPU.

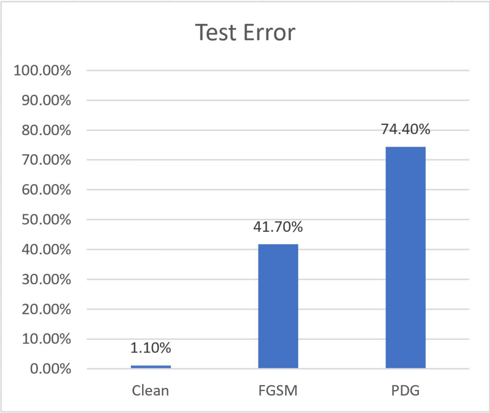
Exercise 2

The model week3/exercise2/MNIST.pt is trained in the normal way. The model in the same folder MNISTRobust.pt is trained with FGSM adversarial training.

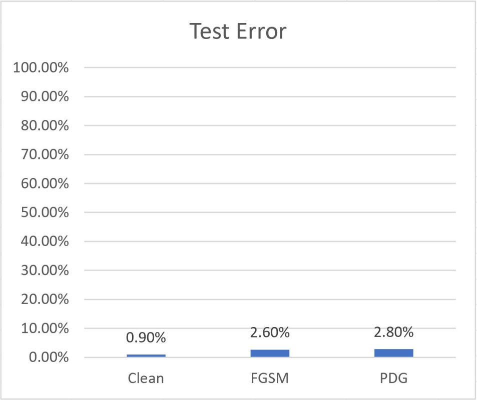
Apply FGSM attacks on both models (with images in week3/exercise2/toattack) by running fgsm.py, report the attacking success rate (with eps 0.02, 0.05, 0.1, and 0.2) on both models. Take note of the differences.

Adversarial Training: Empirical Results

Without Adversarial Training



With PGD Adversarial Training



Experimental setting: on ConvNet model trained on MNIST; Both FGSM and PGD attacks are conducted with a L_∞-norm bound of 0.1.

More Adversarial Training

Ensemble adversarial training

The idea is to increase the variety of adversarial samples.

The approach is to generate a set of additional models, generate adversarial samples based them, and use these samples additionally during adversarial training.

Generative adversarial training

The idea is to co-train a generator neural network (i.e., an attacker) to generate adversarial samples and the classifier at the same time.

Similar ideas have been applied to neural networks for texts.

Exercise 3

Answer these questions:

1. Adversarial training aims to solve the following optimization problem.

   $$\text{Min}_{\theta}\ \text{Max}_{d(x,x')<\varepsilon}\ L(\theta, x', y)$$

   Examining this optimization problem, can you explain why adversarial training with PGD works better than adversarial training with FGSM?

2. Can you explain why adversarial training works from the point of view of “adversarial samples are features”?

Adversarial Training

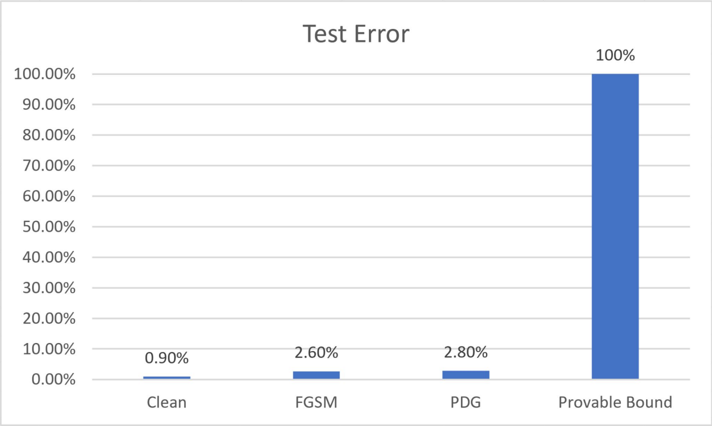
PGD Adversarial Training is considered effective.

Although there are many subsequent proposals, it is still considered the state-of-the-art and favored by winners of Competition on Adversarial Attacks and Defenses (CAADs).

Are adversarial trained models provably robust?

Experiment

Apply the robustness verification method to see whether we can verify the model.

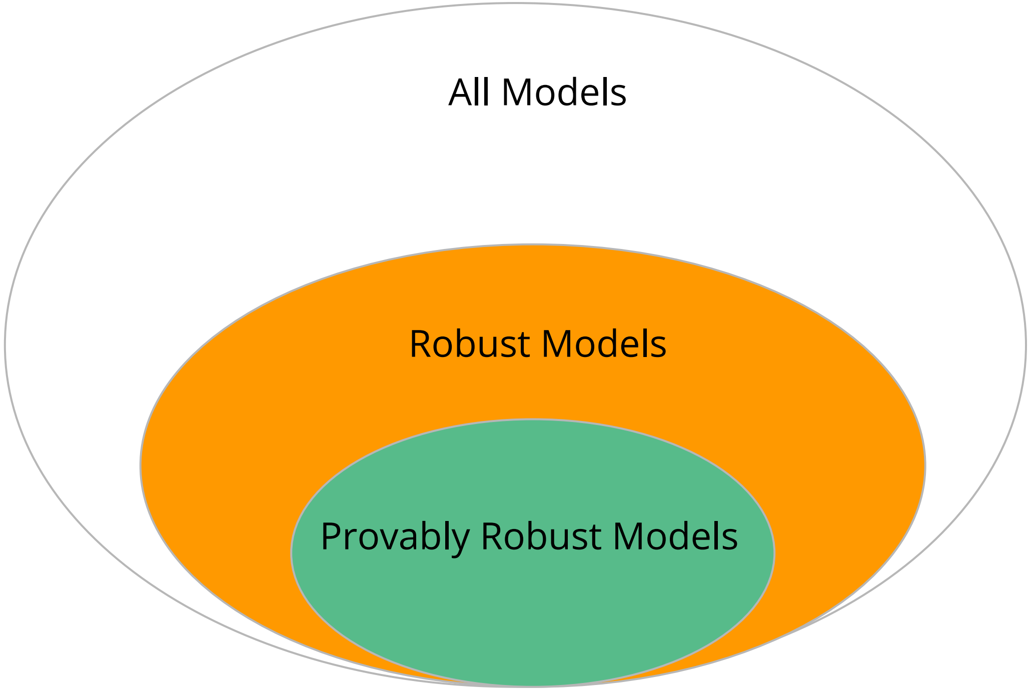


Certified Training

Adversarial training hopefully lead to robust models;

Improving robustness verification methods allows us to close the gap between robust models and provably robust models.

Can we train provably robust models?

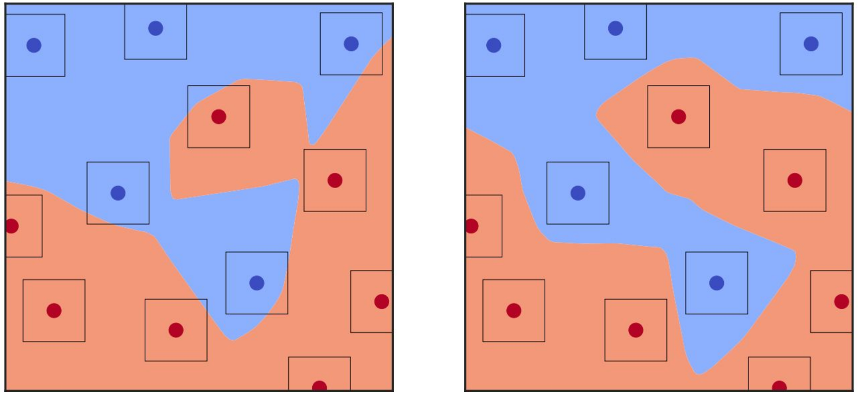


Certified Training

High-level intuition

During training, optimize the network parameters θ such that there is no adversarial sample within some predefined distance of any training sample.

Given θ and a training sample x, we can determine whether there are adversarial samples or not through robustness verification (refer to the last week's slides).



Certified Training

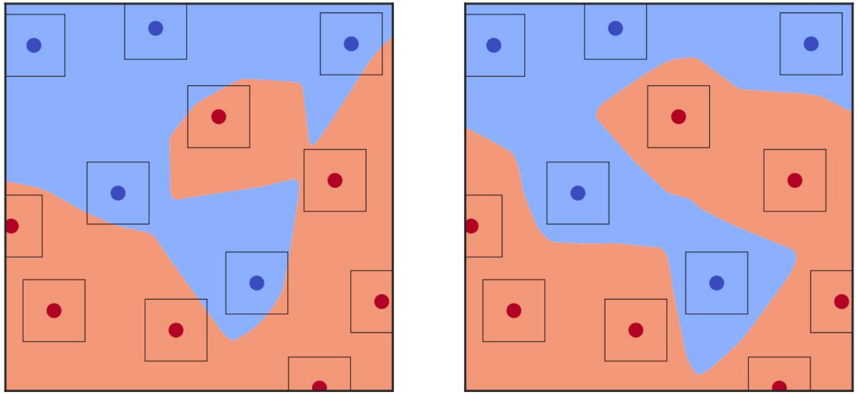
Approach

We train the network with the following objective:

$$\mathrm{Min}_{\theta}\ \mathrm{Max}_{d(x,x')<\varepsilon,\ y' \,!=\, y}\ L(\theta, x', y) - L(\theta, x', y')$$

The idea is to minimize the maximal difference between L(θ, x’, y) and L(θ, x’, y’).

Intuition: If the maximal difference between L(θ, x’, y) and L(θ, x’, y’) for any x’ is negative, we know we always have the correct label.



Certified Training

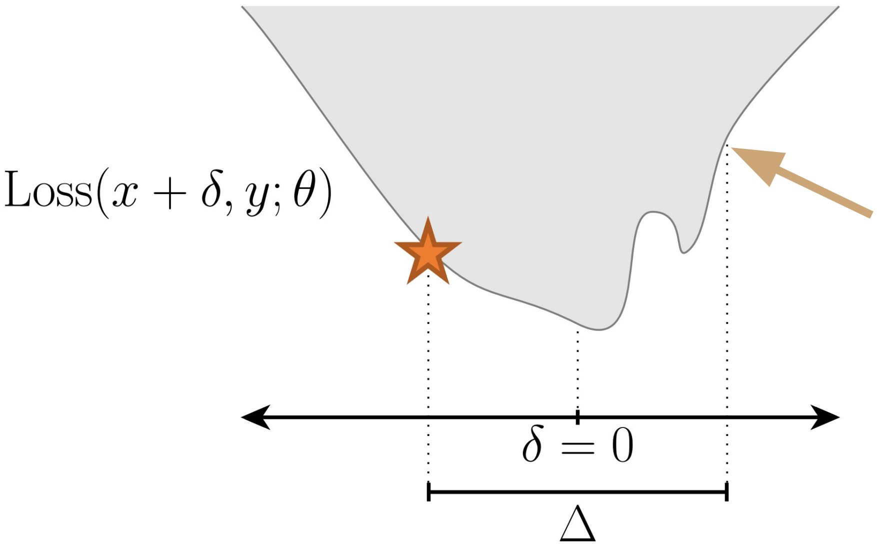
Approach

Given the objective:

$$\text{Min}_{\theta}\ \text{Max}_{d(x,x')<\varepsilon,\ y' != y}\ L(\theta, x', y) - L(\theta, x', y')$$

Instead of approximating the inner max problem through attacking (as in adversarial training), we solve it.

So that when it is negative, we are certain that the network is robust.



Solving the Inner Max Problem

Problem

Given θ and any training sample x with original label y and any other label y’, can we compute the value of L(θ, x’, y) - L(θ, x’, y’) efficiently?

We need to do it efficiently as we need to do it for every sample in the training set in every iteration of optimization.

Approach

We soundly approximate L(θ, x’, y) - L(θ, x’, y’) based on abstract interpretation.

You know the approach already (refer to week2: slide 49).

Certified Training: Recap

Approach

Choose some initial network parameters θ;

Soundly approximate $\text{Max}_{d(x,x')<\varepsilon,\ y' != y}$ L(θ, x’, y) - L(θ, x’, y’) using abstract interpretation;

Optimize parameters θ;

Repeat the process until (ideally) $\text{Max}_{d(x,x')<\varepsilon,\ y' != y}$ L(θ, x’, y) - L(θ, x’, y’) is negative for every sample x in the training set.

Some more notes

Multiple methods of soundly approximating L(θ, x’, y) - L(θ, x’, y’) have been proposed (Refer to [2,3,4,5]).

We must balance the expressiveness of the abstraction and efficiency, e.g., if we only compute the interval of each neuron, it will be more efficient but less precise (than DeepPoly).

Certified Training: Performance

Scalability is less than ideal.

Certified training is still limited to fairly small neural networks (such as those trained on CIFAR, definitely not those on ImageNet).

It is not yet clear whether it is possible to do better.

Example Performance (ICLR'20)

With CIFAR-10, and L_∞-norm range 2/255, the best performance is: 78.4% accuracy; and 60.5% certified robustness.

With CIFAR-10, and L_∞-norm range 8/255, the best performance is: 54.5% accuracy; and 30.5% certified robustness.

Quick Discussion

Looking at results shown on the previous slide and discuss the following questions.

- Can you explain why the performance drops with a larger L_∞-norm range?
- What if the L_∞-norm range is 0/255 or 255/255?

Do you think certified training is relevant in practice?

Randomized Smoothing

Intuition*

The goal is to have a classifier which is robust within certain radius of every training sample x, i.e., N(x) = N(x+δ) where δ is some noise allowed by the radius.

Why don't we train such a classifier, i.e., the prediction is likely the same for inputs within the radius of x?

**"Certified Robustness to Adversarial Examples with Differential Privacy", S&P 19.*

Approach

During training, we systematically add random noise δ to each sample so that the predictions for input within the radius is likely the same.

During inference, given x, we generate the most common prediction for any input within the radius of x.

Inspired by differential privacy (we will learn it in week 9).

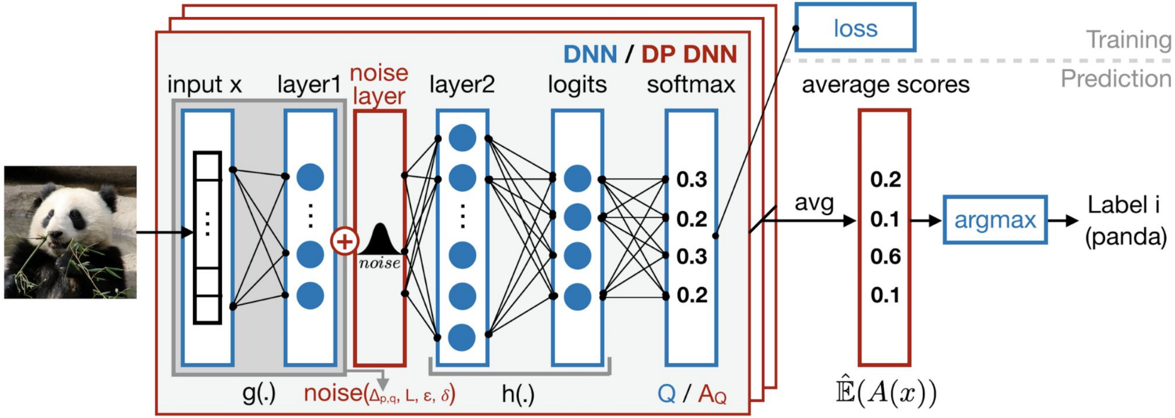
Randomized Smoothing

Algorithm

During training, add a noise layer (to induce noises following a Gaussian distribution).

During inference, query the model many times and output the most frequent prediction.

$N(x) = \arg\max_c P(N(x+\delta)=c)$



The blue parts are from ordinary training; the red parts are the new ones.

Randomized Smoothing

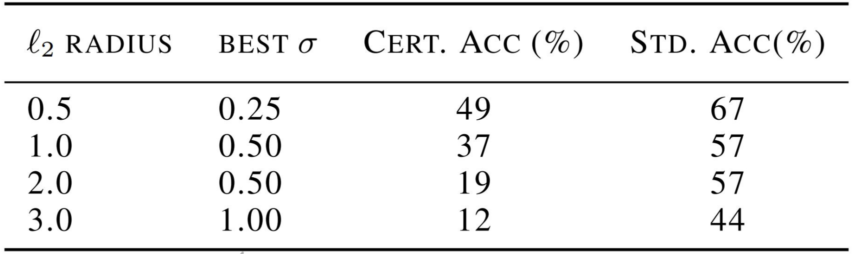
What is certified?

If we apply noises following a Gaussian distribution, and if the difference between the most frequent prediction and second most frequent prediction is large, we have certified robustness with respects to L_2-norm based adversarial attacks.

The mathematics is rather involved and only works for L_2-norm.

Performance

On the network trained on ImageNet



ℓ_2 RADIUS	BEST σ	CERT. ACC (%)	STD. ACC(%)
0.5	0.25	49	67
1.0	0.50	37	57
2.0	0.50	19	57
3.0	1.00	12	44

Ignore this column

Conclusion

	Easy to Apply	Effectiveness	Scalability
Data Augmentation	⭐⭐⭐⭐⭐	⭐	⭐⭐⭐⭐⭐
Adversarial Training	⭐⭐⭐	⭐⭐⭐	⭐⭐⭐⭐
Certified Training	⭐⭐⭐	⭐⭐⭐⭐	⭐
Randomized Smoothing	⭐⭐⭐⭐⭐	⭐⭐⭐⭐	⭐⭐⭐⭐⭐

Practical Guideline

Apply randomized smoothing if (1) you need some certification and (2) you are worried about L_2-norm attack and (3) you don't mind low accuracy;

Apply certified training if (1) your model is small and (2) you need some certification and (3) you are worried about L_∞-norm attack and (4) you don't mind low accuracy;

Apply adversarial training with PGD if you have a lot of training time/resource; or otherwise with FGSM;

Data augmentation at the least.

Exercise 4: Data Augmentation

1. Train a model on MNIST (refer to week1/exercise1)
2. Apply FGSM (refer to week3/exercise2/fgsm.py) to the model to generate 5000 adversarial examples based on the training set of MNIST dataset.
3. Apply data augmentation with these adversarial examples to train a new MNIST model.
4. Report the difference between the model before and after data augmentation in term of (a) the training time; (b) the accuracy on the original test set; (c) the success rate of FGSM attack and PGD attack.

References

1. https://adversarial-ml-tutorial.org
2. Differentiable Abstract Interpretation for Provably Robust Neural Networks, ICML 2018.
3. Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope, ICML 2018.
4. Adversarial training and provable defenses: Bridging the gap, ICLR 2020.
5. Towards stable and efficient training of verifiably robust neural networks, ICLR 2020.
6. Certified Robustness to Adversarial Examples with Differential Privacy, S&P 2019.

Assignment Exercise 2

Submit a zip file containing a report (word, or pdf) and programs showing your working of Exercise 0-4 to elearn (under Assignments and Exercise 2) by Sep 12, 2022 11:59 PM.

Aug 23 - Week 1: 7-10	Introduction	
Aug 30 - Week 2: 7-10	AI Robustness	Exercise 1
Sep 06 - Week 3: 7-10	Improving AI Robustness	Exercise 2
Sep 13 - Week 4: 7-10	AI Backdoors	Exercise 3
Sep 20 - Week 5: 7-10	Mitigating AI Backdoors	Exercise 4; Project Proposal
Sep 27 - Week 6: 7-10	AI Fairness	Exercise 5
Oct 11 - Week 7: 7-10	Improving AI Fairness	Exercise 6
Oct 18 - Week 8: 7-10	AI Privacy	Exercise 7
Oct 25 - Week 9: 7-10	Improving AI Privacy	Exercise 8
Nov 01 - Week 10: 7-10	AI Interpretability	Project Due
Nov 08 - Week 11: 1-3	End-of-Term Exam	