# 6.7900 Machine Learning (Fall 2024)

## Lecture 14: stability and robustness

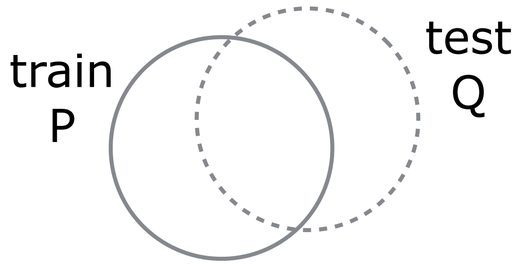
## (supporting slides)

# Many things related to “robustness”

- **Generalization**
  - test distribution (unknown) typically differs a bit from training
  - test examples may be systematically different (e.g., different measurement device)
- **Safety and security**
  - need to ensure that ML methods degrade gracefully in real world tasks
  - a normal input to the model could be adversarially (but imperceptibly) adjusted
  - training set could be poisoned
  - etc.
- **Fairness**
  - ML methods can introduce, propagate disparities (across individuals, groups, etc)
  - statistical criteria of fairness need to be enforced, not automatically satisfied
  - etc.

# Generalization: train – test shifts

- In most cases the training data is not exactly like the test data (often unknown)

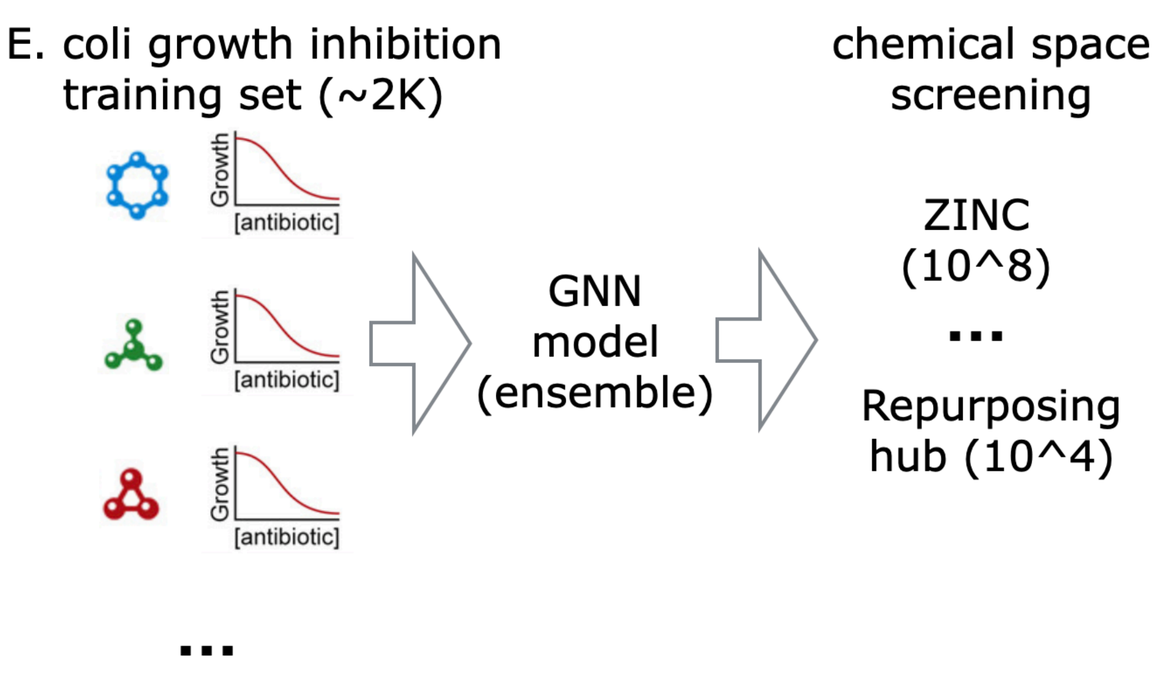


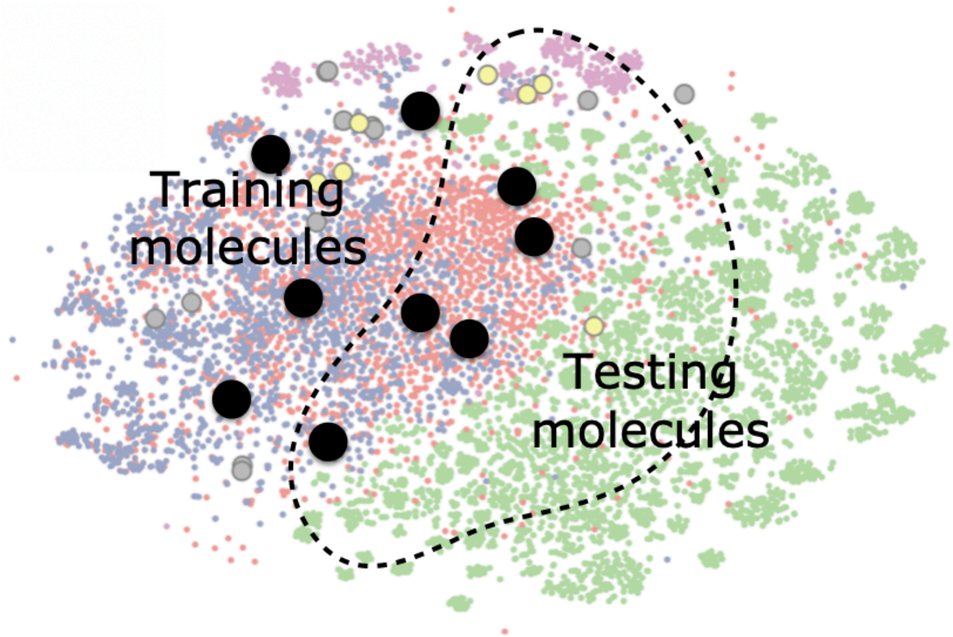

- The shift could be **partially known** (type etc) **or unknown**
  - co-variate shift: Q(x) differs from P(x) but Q(y|x) = P(y|x)
  - label shift: Q(y) differs from P(y) but Q(x|y) = P(x|y)
  - generic small shift: D(Q|P)<eps
  - etc.

- The problem for us is to somehow ensure that we degrade gracefully when the test distribution / set deviates from the train distribution / set

# Generalization: train - test shifts

- In most cases the training data is not exactly like the test data (often unknown)
- **Partially known shift**: a classifier for effective antibiotics is learned from a smaller assay and applied across a larger, diverse chemical space




[Stokes et al. 2020]

# Generalization: train - test shifts

- In most cases the training data is not exactly like the test data (often unknown)
- **Unknown small shift**: a predictor is trained from data available at one hospital but deployed across "similar" other hospitals with potentially different cases, demographics


train P


test Q

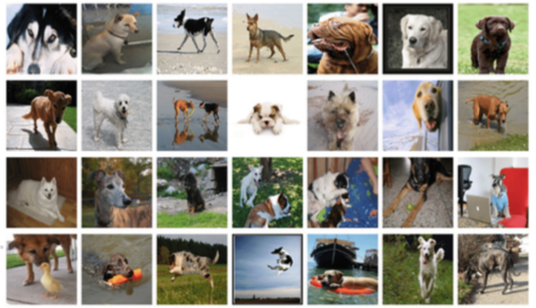
...

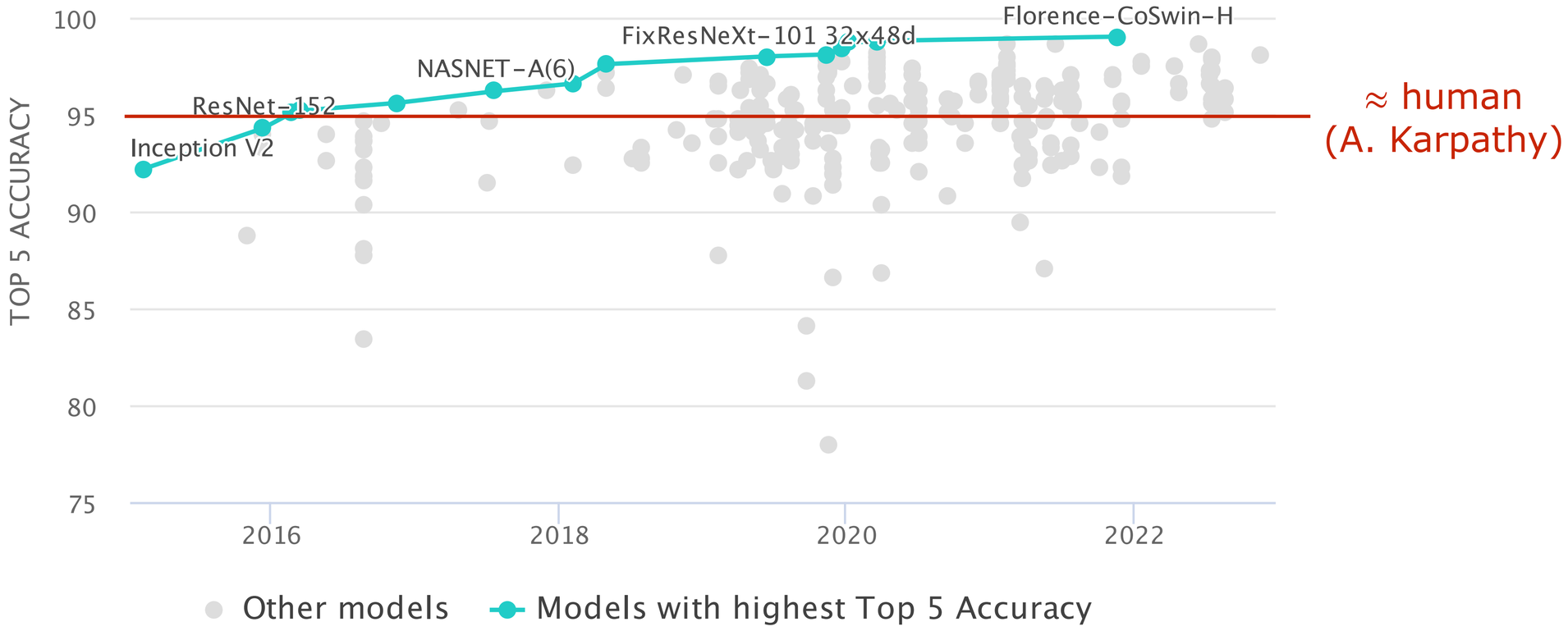
$$Q \in \Delta(Q) = \{\, Q : \; D(Q\|P) \leq \epsilon \,\}$$

divergence
btw Q and P

# Safety & security: brittleness of ML models

- Rapid progress on improving image classification accuracy



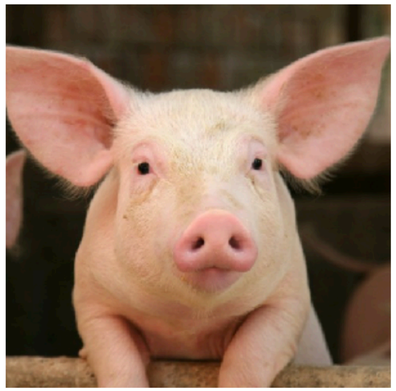



[graph from “papers with code”]

# Safety & security: brittleness of ML models

- Rapid progress on improving image classification accuracy … but trained models may not be robust

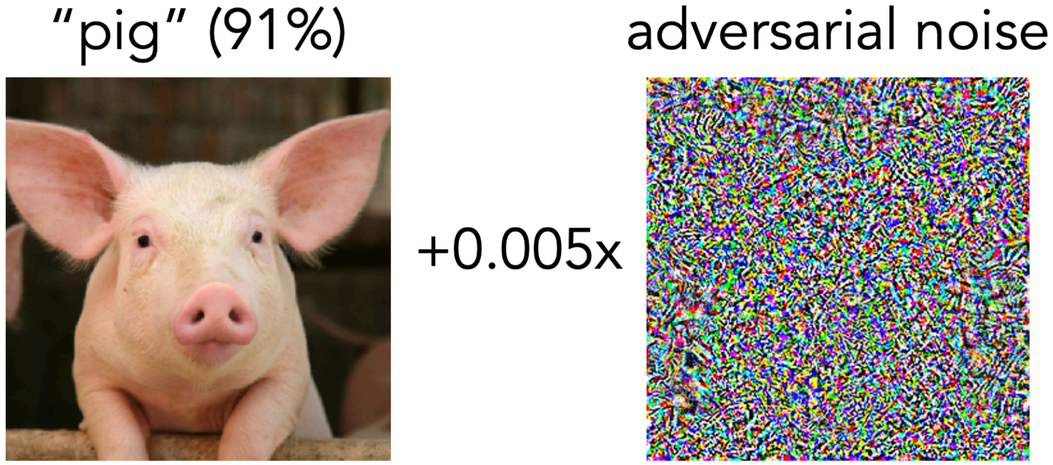
“pig” (91%)



[Szegedy et al. 2013]

# Safety & security: brittleness of ML models

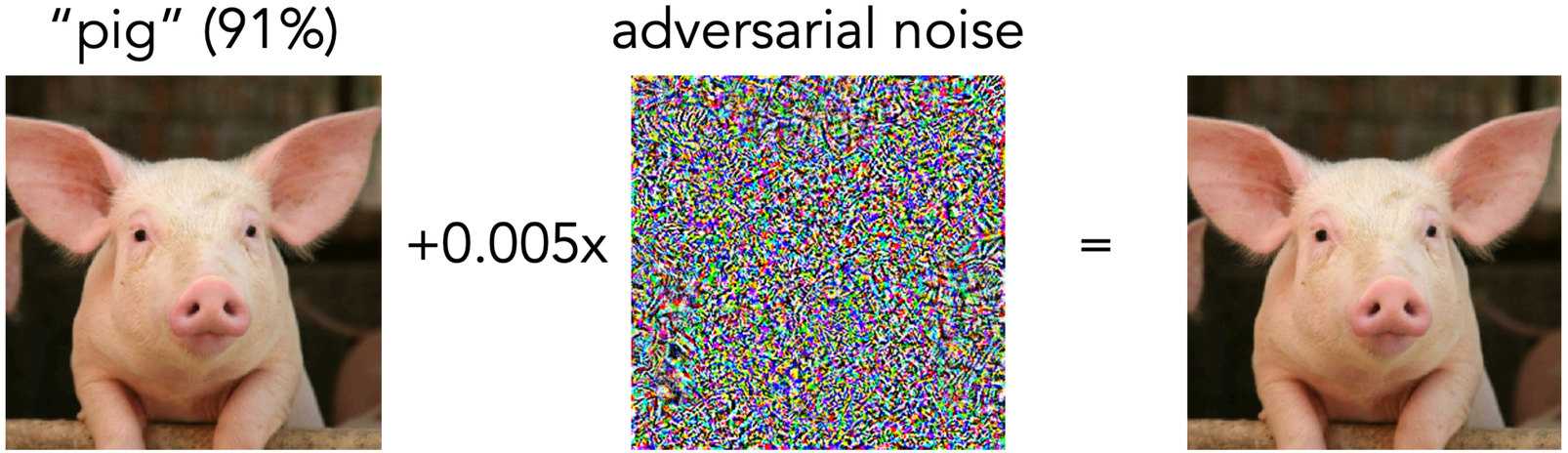
- Rapid progress on improving image classification accuracy … but trained models may not be robust




[Szegedy et al. 2013]

# Safety & security: brittleness of ML models

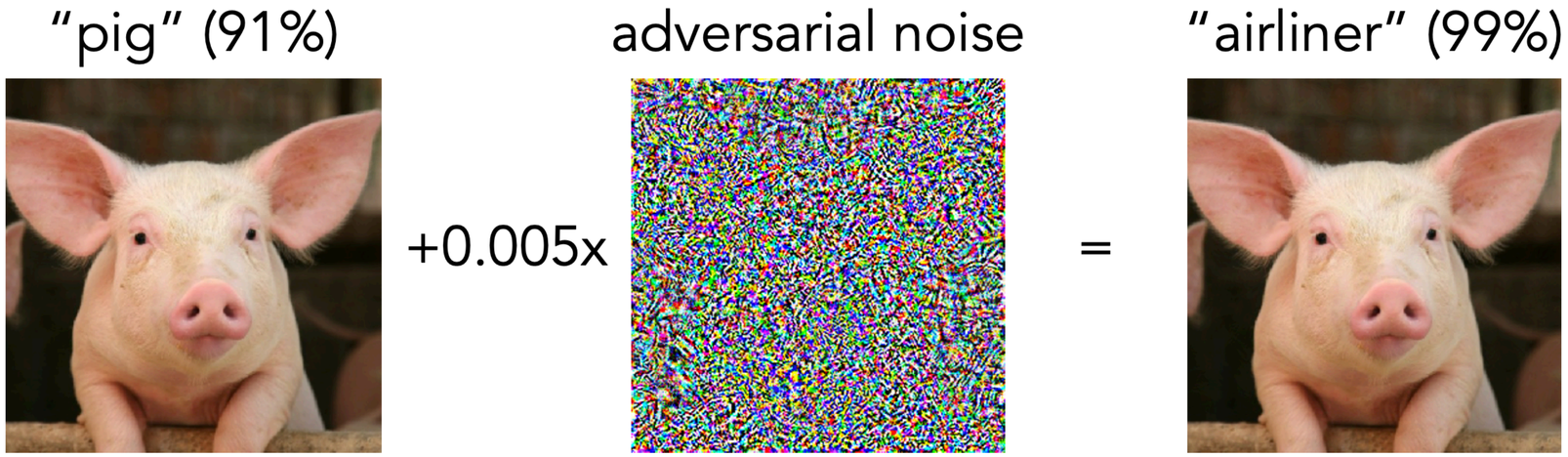
- Rapid progress on improving image classification accuracy … but trained models may not be robust




[Szegedy et al. 2013]

# Safety & security: brittleness of ML models

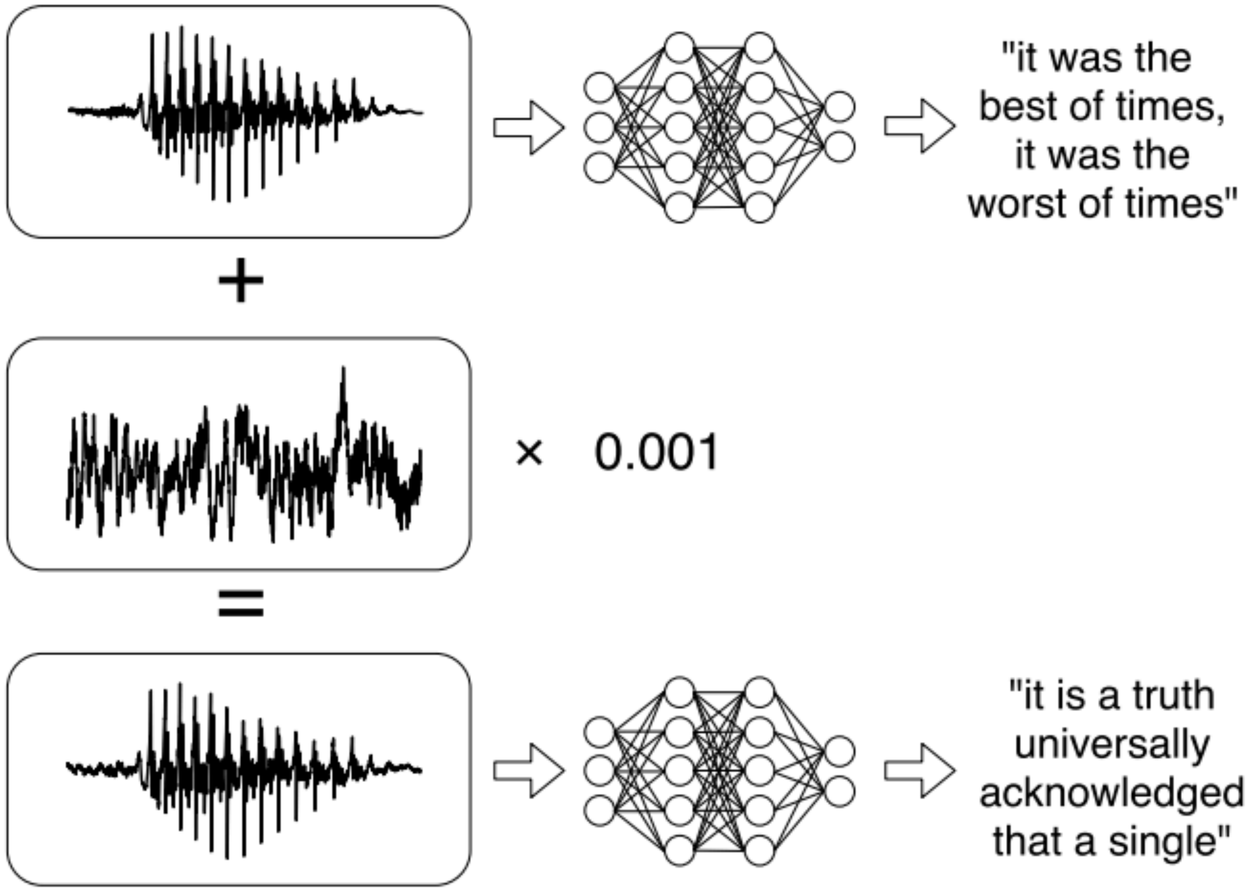
- Rapid progress on improving image classification accuracy … but trained models may not be robust




[Szegedy et al. 2013]

# Safety & security: brittleness of ML models

- Lack of robustness extends to other domains as well…

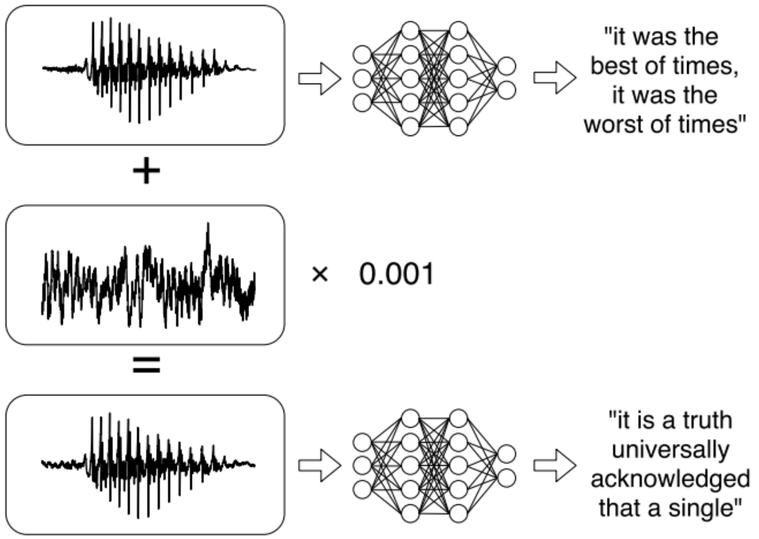



[Carlini & Wagner 2018]

# Safety & security: brittleness of ML models

- Brittleness leads to many concerns
  - applying ML methods in real tasks (e.g., medical, self-driving vehicles)
  - being liable to adversarial attacks (e.g., imperceptibly changing image to a desired class)
  - possibility of dataset poisoning (subtly altering the data for desired effect)
  - etc.


[Szegedy et al. 2013]




[Carlini & Wagner 2018]

# Fairness: ML predictions and disparities

- Machine learning methods are increasingly used in a variety of contexts where they could introduce/perpetuate disparities — predictions have consequences
- E.g., deciding who to admit, hire or promote, who to give loans to, which medical treatments to offer to patients, incentives given to customers, etc.
- Many real life examples where deployed methods involved disparities
  - healthcare: who needs extra care, diagnostic tools
  - justice: who might become a repeat offender
  - business: algorithms helping hiring decisions, incentives that are offered
  - search results, anomaly calls
  - etc.
- Methods can produce unfair predictions for many reasons, e.g., training data (collection/proportions/sample size/features), the way method is applied, etc.

- The goal is to learn predictors that satisfy clear-cut (statistical) fairness criteria such as error parity
- Relation to robustness? We need to do well on the “worst” subgroup at test time

# Many things related to “robustness”

- **Generalization**
  - test distribution (unknown) typically differs a bit from training
  - test examples may be systematically different (e.g., different measurement device)
- **Safety and security**
  - need to ensure that ML methods degrade gracefully in real world tasks
  - a normal input to the model could be adversarially (but imperceptibly) adjusted
  - training set could be poisoned
  - etc.
- **Fairness**
  - ML methods can introduce, propagate disparities (across individuals, groups, etc)
  - statistical criteria of fairness need to be enforced, not automatically satisfied
  - etc.

# Distributionally robust optimization

- One way to think about learning a model that can handle different test distributions is distributional robust optimization (DRO)
- We have training data in the form of (x,y) pairs drawn from distribution P but the test data comes from distribution Q different from P

$$\text{Loss}(h_\theta(x), y) = L(x, y, \theta) = L(z, \theta), \quad z = (x, y)$$ shorthand notation

$$\min_\theta \quad E_{z\sim P}\{\, L(z, \theta)\,\}$$ our regular training criterion

# Distributionally robust optimization

- One way to think about learning a model that can handle different test distributions is distributional robust optimization (DRO)
- We have training data in the form of (x,y) pairs drawn from distribution P but the test data comes from distribution Q different from P

$$\text{Loss}(h_\theta(x), y) = L(x, y, \theta) = L(z, \theta), \quad z = (x, y)$$ shorthand notation

$$\min_\theta \quad E_{z\sim P}\{\, L(z, \theta)\,\}$$ our regular training criterion

$$\min_\theta \quad \max_Q \; E_{z\sim Q}\{\, L(z, \theta)\,\}$$ a robust criterion (but way too hard as a learning task)

# Distributionally robust optimization

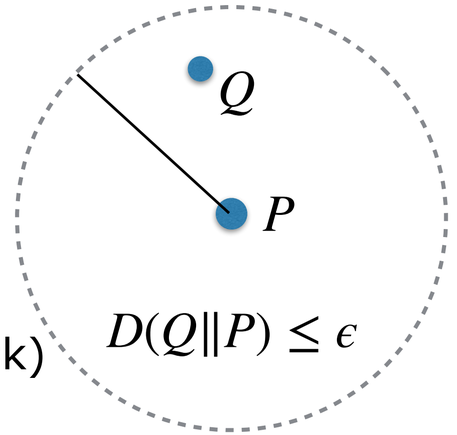
- One way to think about learning a model that can handle different test distributions is distributional robust optimization (DRO)
- We have training data in the form of (x,y) pairs drawn from distribution P but the test data comes from distribution Q different from P

$$\text{Loss}(h_\theta(x), y) = L(x, y, \theta) = L(z, \theta), \quad z = (x, y)$$ shorthand notation

$$\min_\theta \quad E_{z\sim P}\{\, L(z,\theta)\,\}$$ our regular training criterion

$$\min_\theta \quad \max_Q \; E_{z\sim Q}\{\, L(z,\theta)\,\}$$ a robust criterion (but way too hard as a learning task)

$$\min_\theta \quad \max_{D(Q\|P)\le\epsilon} \; E_{z\sim Q}\{\, L(z,\theta)\,\}$$ DRO criterion with some "divergence" metric $D(Q\|P)$

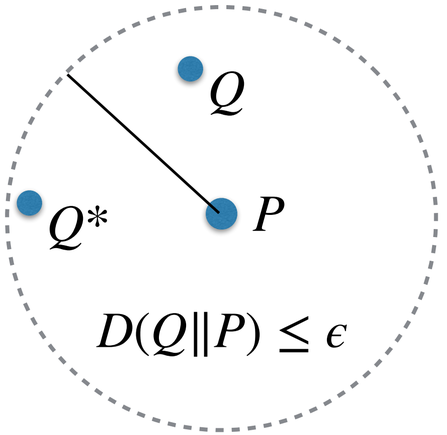


# Distributionally robust optimization

- One way to think about learning a model that can handle different test distributions is distributional robust optimization (DRO)
- We have training data in the form of (x,y) pairs drawn from distribution P but the test data comes from distribution Q different from P

$$\min_{\theta} \quad \max_{D(Q\|P)\leq\epsilon} \quad E_{z\sim Q}\{\, L(z,\theta)\,\}$$ DRO criterion with some "divergence" metric

- Suppose the true test distribution $Q^*$ also satisfies $D(Q^*\|P) \leq \epsilon$,



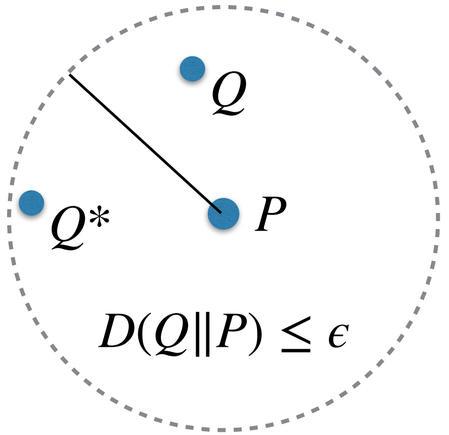
# Distributionally robust optimization

- One way to think about learning a model that can handle different test distributions is distributional robust optimization (DRO)
- We have training data in the form of (x,y) pairs drawn from distribution P but the test data comes from distribution Q different from P

$$\min_{\theta} \quad \max_{D(Q\|P)\leq\epsilon} \quad E_{z\sim Q}\{\, L(z,\theta)\,\}$$ DRO criterion with some "divergence" metric

- Suppose the true test distribution $Q^*$ also satisfies $D(Q^*\|P) \leq \epsilon$,

$$E_{z\in Q^*}\{\, L(z,\theta)\,\} \;\leq\; \max_{D(Q\|P)\leq\epsilon} \; E_{z\sim Q}\{\, L(z,\theta)\,\}$$



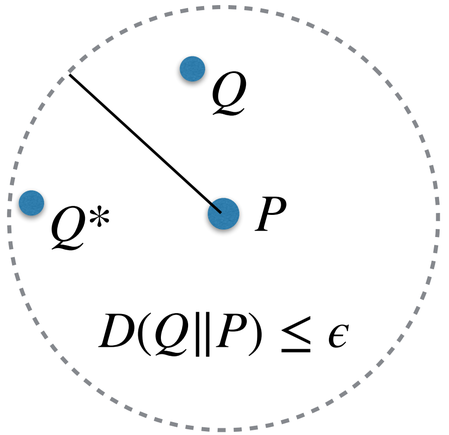
# Distributionally robust optimization

- One way to think about learning a model that can handle different test distributions is distributional robust optimization (DRO)
- We have training data in the form of (x,y) pairs drawn from distribution P but the test data comes from distribution Q different from P

$$\min_{\theta} \quad \max_{D(Q\|P)\leq\epsilon} \quad E_{z\sim Q}\{\, L(z,\theta)\,\}$$ DRO criterion with some "divergence" metric

- Suppose the true test distribution $Q^*$ also satisfies $D(Q^*\|P) \leq \epsilon$,

$$E_{z\in Q^*}\{\, L(z,\theta)\,\} \;\leq\; \max_{D(Q\|P)\leq\epsilon} \; E_{z\sim Q}\{\, L(z,\theta)\,\}$$

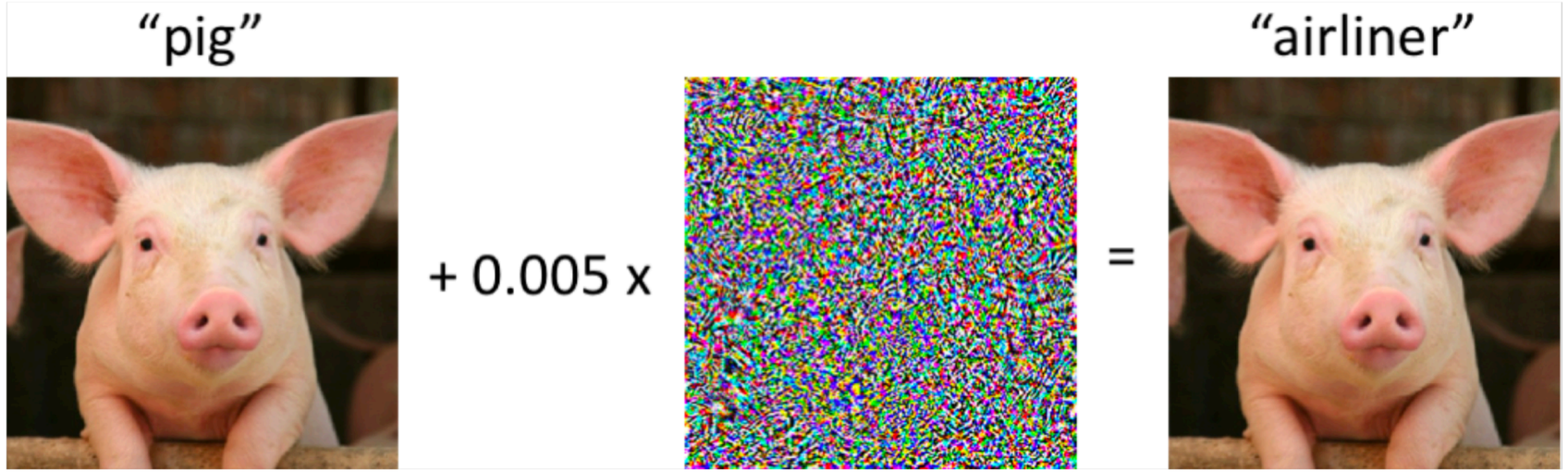



- So, if we minimize the DRO criterion (upper bound), we also force the test error to be small

# Many things related to “robustness”

- **Generalization**
  - test distribution (unknown) typically differs a bit from training
  - test examples may be systematically different (e.g., different measurement device)
- **Safety and security**
  - need to ensure that ML methods degrade gracefully in real world tasks
  - a normal input to the model could be adversarially (but imperceptibly) adjusted
  - training set could be poisoned
  - etc.
- **Fairness**
  - ML methods can introduce, propagate disparities (across individuals, groups, etc)
  - statistical criteria of fairness need to be enforced, not automatically satisfied
  - etc.

# Safety & security: learning to be robust




$x$

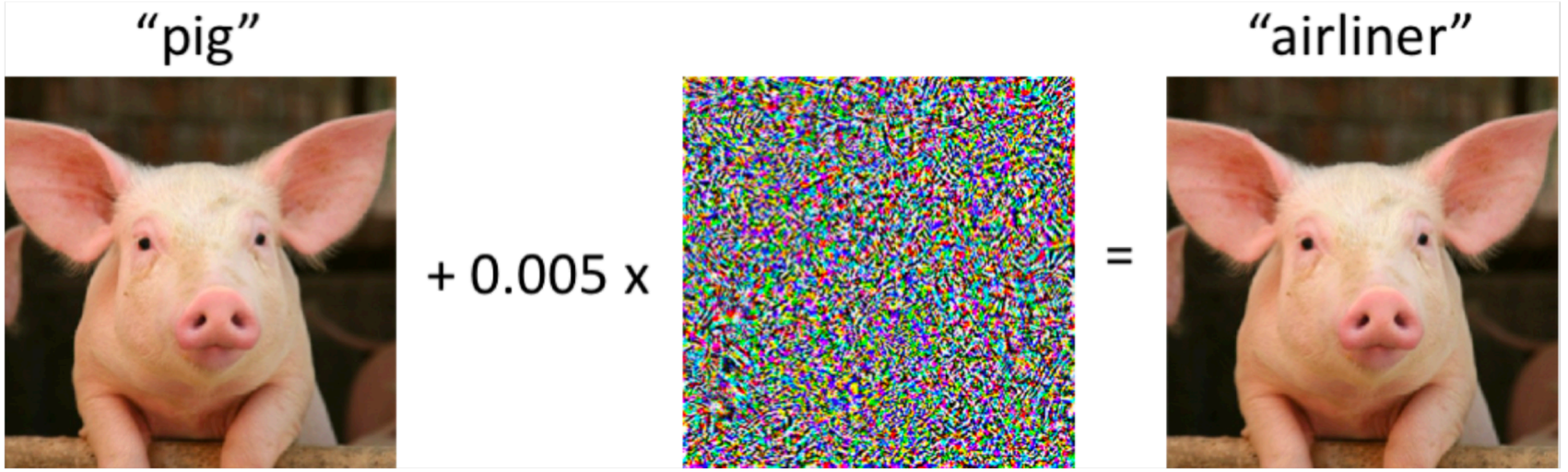
input we used to receive at train or test

$x + \delta$

input we might receive at test time

- How do we ensure we will do well on such examples?

# Safety & security: learning to be robust




$x$

input we used to receive at train or test

$x + \delta$

input we might receive at test time

Our robust training criterion

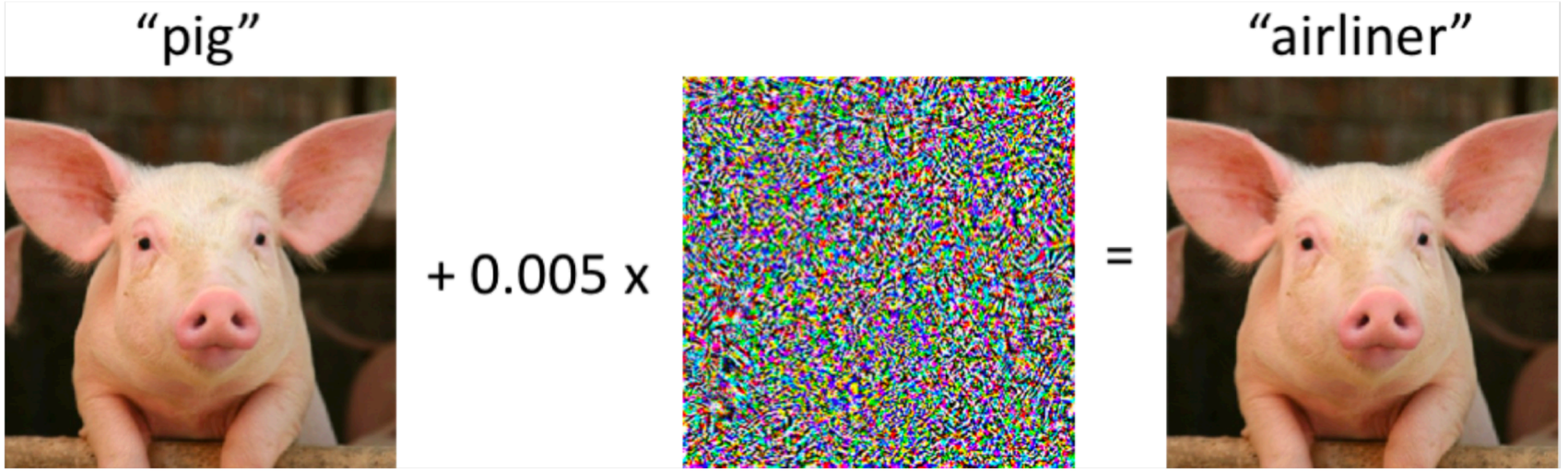
$$\min_{\theta} \sum_{x,y \in S} \max_{\delta \in \Delta} \text{Loss}\left(x + \delta, y; \theta\right)$$

training set

$$\Delta = \{\delta : \ \|\delta\| \leq \epsilon\}$$

where $\|\cdot\|$ is a chosen norm

# Two problems to solve for training




$x$ $\qquad\qquad$ $x + \delta$

Part II: training a robust classifier

The criterion we can optimize

$$\min_{\theta} \sum_{x,y \in S} \max_{\delta \in \Delta} \text{Loss}\,(x + \delta, y; \theta)$$

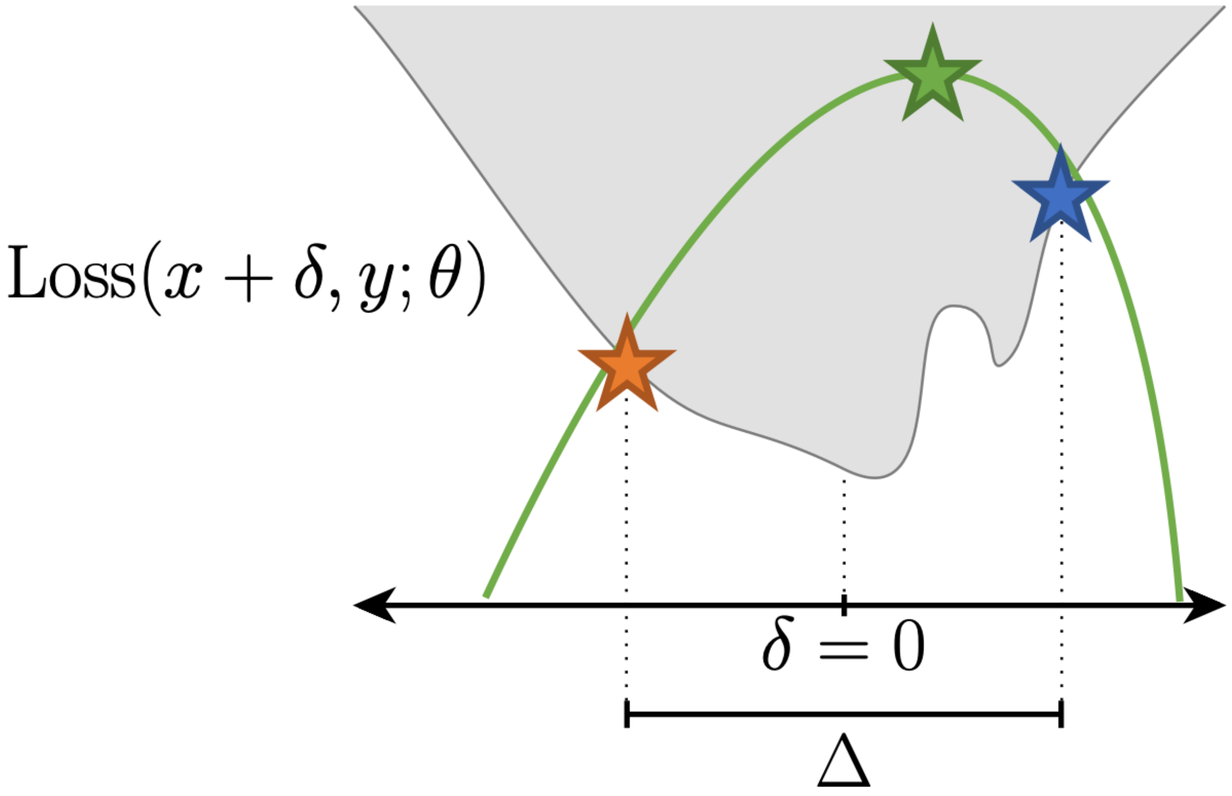
Part I: creating an adversarial example (or ensuring one does not exist)

$$\Delta = \{\delta : \ \|\delta\| \leq \epsilon\}$$

where $\|\cdot\|$ is a chosen norm

[adapted from Zico Kolter, Adversarial ML tutorial, 2018]

# Creating the adversarial example




How do we solve the optimization?

$$\max_{\delta \in \Delta} \text{Loss}\,(x+\delta, y; \theta)$$

1. Local search (lower bound on objective)
2. Combinatorial optimization (exactly solve objective)
3. Convex relaxation (upper bound on objective)

[Zico Kolter, Adversarial ML tutorial, 2018]

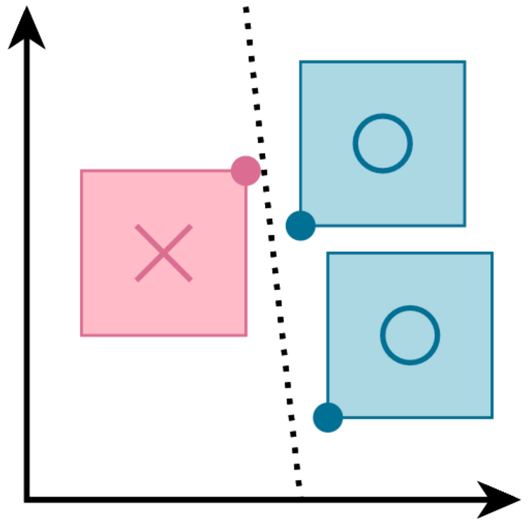
# Simple linear case example

Suppose our hypothesis is linear

$$\text{Loss}(x, y; \theta) = L(\theta^T x \cdot y)$$

(e.g. $L$ hinge or logistic loss) and perturbation region $\Delta$ is a norm ball $\Delta = \{\delta : \|\delta\| \leq \epsilon\}$

Then maximization has exact solution based on dual norm; a simple instance of *robust optimization* [Stoyer, 1973, Ben-Tal et al., 2011, Xu and Manor, 2009]



$$\max_{\delta \in \Delta} L(\theta^T (x + \delta) \cdot y)$$

...

$$= L(\theta^T x \cdot y - \epsilon \|\theta\|_*)$$

[adapted from Zico Kolter, Adversarial ML tutorial, 2018]

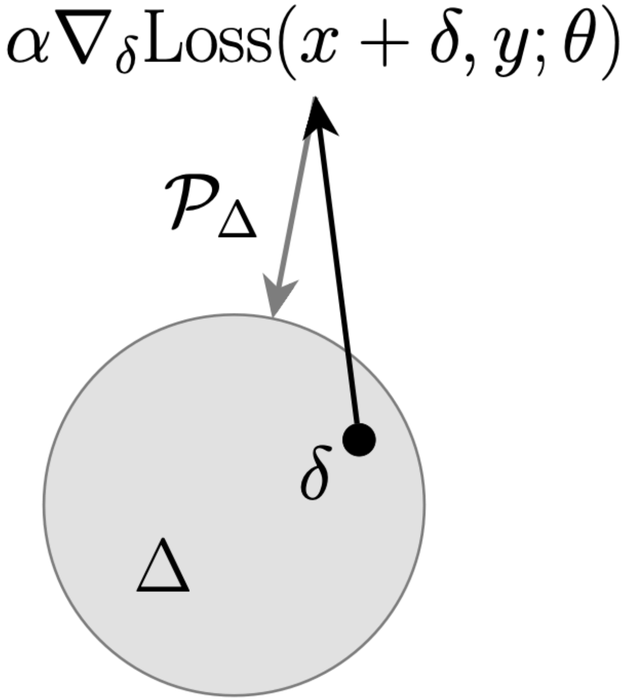
# Non-linear case: projected gradient descent

Recall we are optimizing

$$\max_{\delta \in \Delta} \text{Loss}\,(x + \delta, y; \theta)$$

We can employ a projected gradient descent method, take gradient step and project back into feasible set $\Delta$

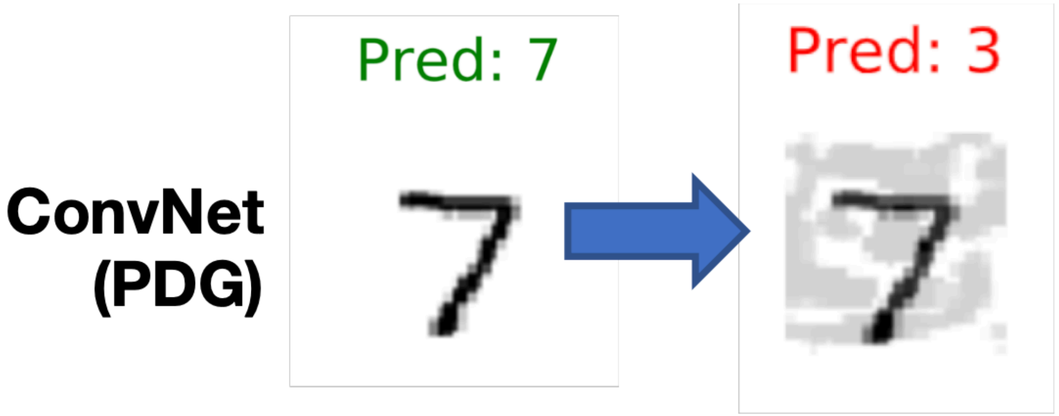
$$\delta := \mathcal{P}_{\Delta}[\delta + \alpha \nabla_{\delta} \text{Loss}(x + \delta, y; \theta)]$$

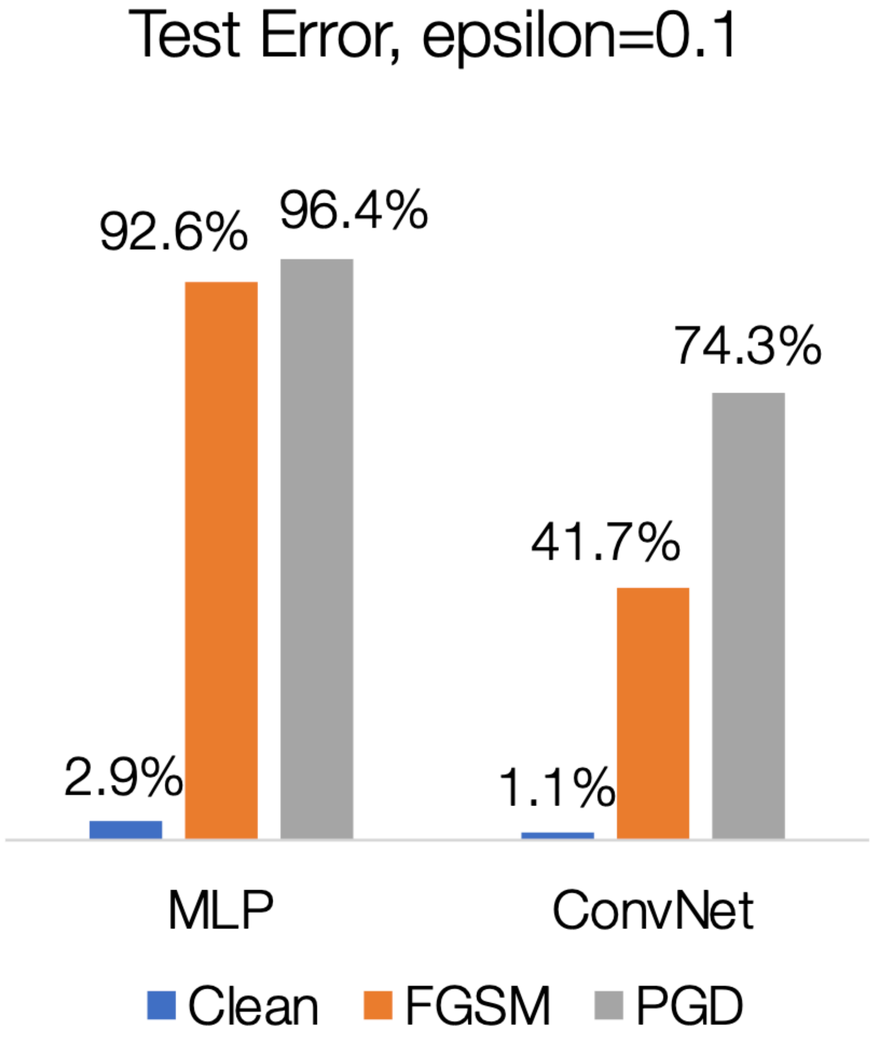



[adapted from Zico Kolter, Adversarial ML tutorial, 2018]

# Projected gradient descent (PGD)

- Does PGD succeed in finding adversarial examples?




[adapted from Zico Kolter, Adversarial ML tutorial, 2018]

# Outer maximization via SGD?

How do we optimize the objective

$$\min_{\theta} \sum_{x,y \in S} \max_{\delta \in \Delta} \text{Loss}\left(x + \delta, y; \theta\right)$$

We would *like* to solve it with gradient descent, but how do we compute the gradient of the objective with the max term inside?

[Zico Kolter, Adversarial ML tutorial, 2018]

# Danskin's theorem

A fundamental result in optimization:

$$\nabla_\theta \max_{\delta \in \Delta} \text{Loss}\left(x + \delta, y; \theta\right) = \nabla_\theta \text{Loss}(x + \delta^\star, y; \theta)$$

where $\delta^\star = \max_{\delta \in \Delta} \text{Loss}\left(x + \delta, y; \theta\right)$

Seems "obvious," but it is a very subtle result; means we can optimize through the max by just finding it's maximizing value

Note however, it *only* applies when max is performed exactly

[Zico Kolter, Adversarial ML tutorial, 2018]

# Simple adversarial training algorithm

Repeat

1. Select minibatch $B$
2. For each $(x, y) \in B$, compute adversarial example $\delta^\star(x)$
3. Update parameters

$$\theta := \theta - \frac{\alpha}{|B|} \sum_{x,y \in B} \nabla_\theta \text{Loss}(x + \delta^\star(x), y; \theta)$$

Common to also mix robust/standard updates (not done in our case)

[Goodfellow et al. 2014] [adapted from Zico Kolter, Adversarial ML tutorial, 2018]

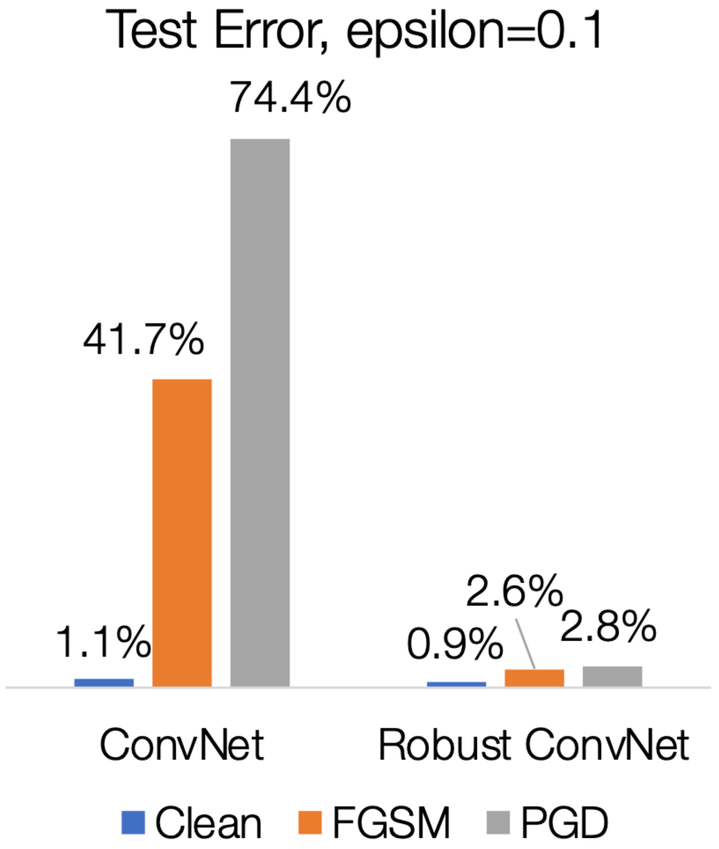
# Simple adversarial training algorithm

Repeat

1. Select minibatch $B$
2. For each $(x, y) \in B$, compute adversarial example $\delta^\star(x)$
3. Update parameters

$$\theta := \theta - \frac{\alpha}{|B|} \sum_{x,y \in B} \nabla_\theta \text{Loss}(x + \delta^\star(x), y; \theta)$$

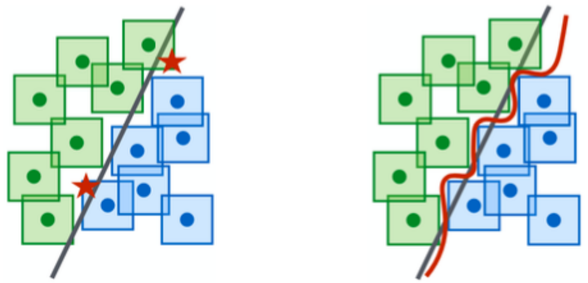
Common to also mix robust/standard updates (not done in our case)

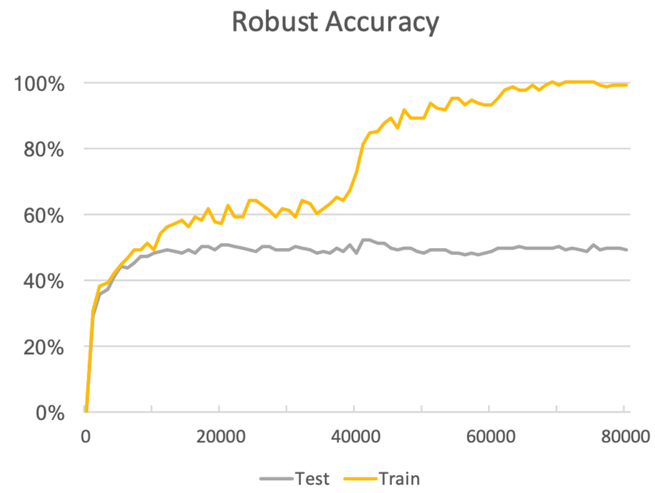



[Goodfellow et al. 2014] [adapted from Zico Kolter, Adversarial ML tutorial, 2018]

# Additional considerations

- Do we need extra capacity to find a well-fitting model that is robust (on the training set)? [yes]
- Does “robustness” generalize as a property? Is the sample complexity for this greater than for regular learning? [yes, harder to generalize]
- Can we “certify” robustness? That no adversarial examples exists nearby a specific new example? [yes, e.g., via a random ensemble “randomized smoothing”]





# Many things related to “robustness”

- **Generalization**
  - test distribution (unknown) typically differs a bit from training
  - test examples may be systematically different (e.g., different measurement device)
- **Safety and security**
  - need to ensure that ML methods degrade gracefully in real world tasks
  - a normal input to the model could be adversarially (but imperceptibly) adjusted
  - training set could be poisoned
  - etc.
- **Fairness**
  - ML methods can introduce, propagate disparities (across individuals, groups, etc)
  - statistical criteria of fairness need to be enforced, not automatically satisfied
  - etc.

# Fairness: robustness and error parity

- Suppose our dataset (distribution) consists of K **unknown** subgroups
- We would like to ensure, e.g., that the error rates of the estimated predictor are the same across these groups to the extent possible

$z = (x, y) \sim P_k$ examples from group k (we don't know this)

$E_{z \sim P_k}\{\ L(z, \theta)\ \}$ expected loss within group k (we cannot calculate this)

$\alpha_k$ fraction/porportion of individuals that belong to group k

- To balance the subgroup errors would like to minimize $\max_k E_{z \sim P_k}\{\ L(z, \theta)\ \}$

[T. Hashimoto et al. 2018]

# Fairness: robustness and error parity

- Suppose our dataset (distribution) consists of K **unknown** subgroups
- We would like to ensure, e.g., that the error rates of the estimated predictor are the same across these groups to the extent possible

$z = (x, y) \sim P_k$ examples from group k (we don't know this)

$E_{z \sim P_k}\{\, L(z, \theta)\, \}$ expected loss within group k (we cannot calculate this)

$\alpha_k$ fraction/porportion of individuals that belong to group k

- To balance the subgroup errors would like to minimize $\max_k E_{z \sim P_k}\{\, L(z, \theta)\, \}$
- We can achieve this through a robust DRO objective without any knowledge of the underlying demographics except for the size of the smallest group

[T. Hashimoto et al. 2018]

# Fairness: robustness and error parity

- Suppose our dataset (distribution) consists of K **unknown** subgroups
- We would like to ensure, e.g., that the error rates of the estimated predictor are the same across these groups to the extent possible

$z = (x, y) \sim P_k$ examples from group k (we don't know this)

$E_{z \sim P_k}\{ L(z, \theta) \}$ expected loss within group k (we cannot calculate this)

$\alpha_k$ fraction/porportion of individuals that belong to group k

- To balance the subgroup errors would like to minimize $\max_k E_{z \sim P_k}\{ L(z, \theta) \}$
- We can achieve this through a robust DRO objective <u>without any knowledge of the underlying demographics</u> except for the size of the smallest group

$$\max_k E_{z \sim P_k}\{ L(z, \theta) \} \leq \max_{Q:\, D_{\chi^2}(Q\|P) \leq r} \{ L(z, \theta) \}$$

$$r = \max_k (1/\alpha_k - 1)^2 = (1/\min_k \alpha_k - 1)^2$$

$$D_{\chi^2}(Q\|P) = \int (\frac{Q(z)}{P(z)} - 1)^2 P(z) dz = \int \frac{Q(z)^2}{P(z)} dz - 1$$

[T. Hashimoto et al. 2018]

# Additional (optional) references


- Z. Kolter and A. Madry, Adversarial robustness tutorial, https://adversarial-ml-tutorial.org/
- I. Goodfellow et al. “Explaining and Harnessing Adversarial Examples”, https://arxiv.org/pdf/1412.6572.pdf
- T. Hashimoto et al., “Fairness Without Demographics in Repeated Loss Minimization”, https://arxiv.org/pdf/1806.08010.pdf
- J. Duchi et al. “Learning Models with Uniform Performance via Distributionally Robust Optimization”, https://arxiv.org/pdf/1810.08750.pdf
- Solon Barocas, Moritz Hardt, Arvind Narayanan, “Fairness and Machine Learning”, https://fairmlbook.org/