The University of Texas at Austin
Electrical and Computer Engineering
Cockrell School of Engineering

Fall 2023

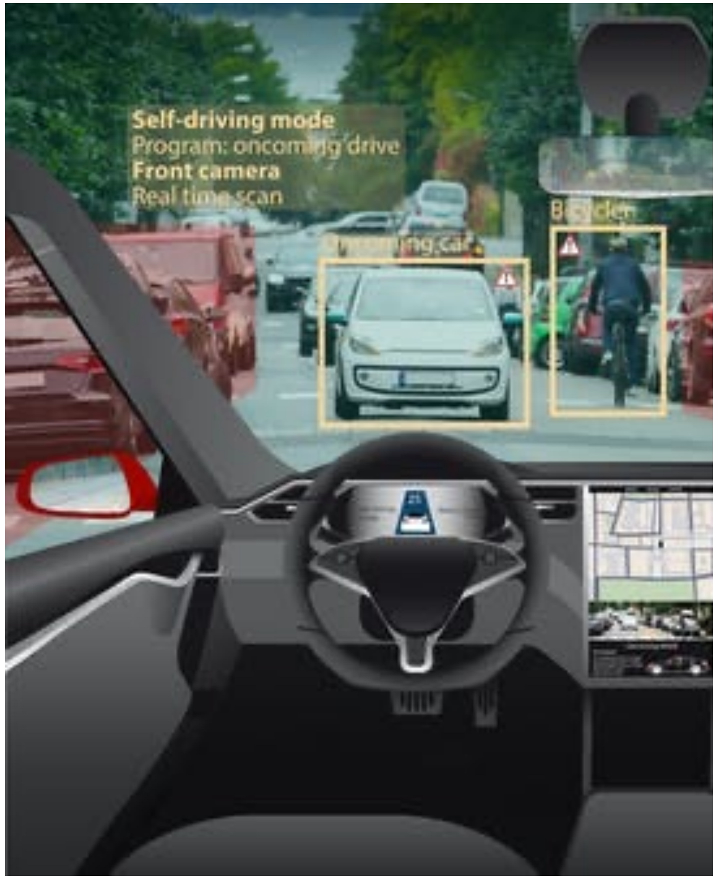
ADVANCED TOPICS IN COMPUTER VISION

Atlas Wang
Associate Professor, The University of Texas at Austin

Visual Informatics Group@UT Austin
https://vita-group.github.io/

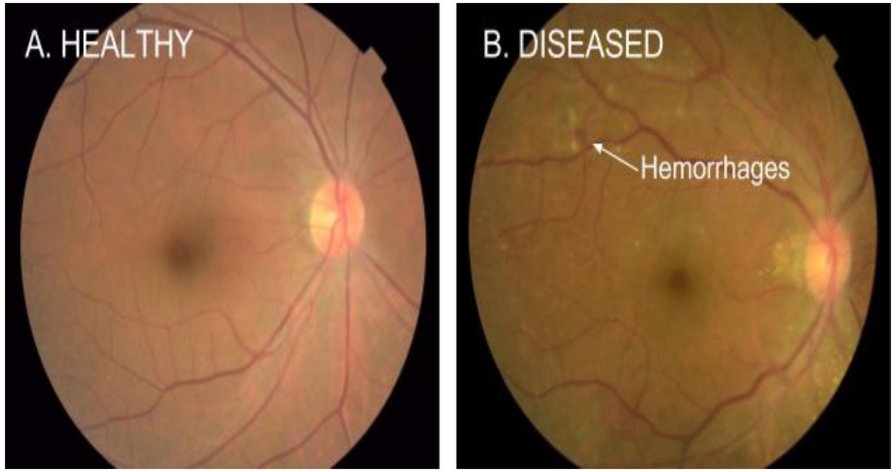
Trustworthy Computer Vision?

Face Recognition



Self-Driving Perception

Safe Control



Medical Diagnosis

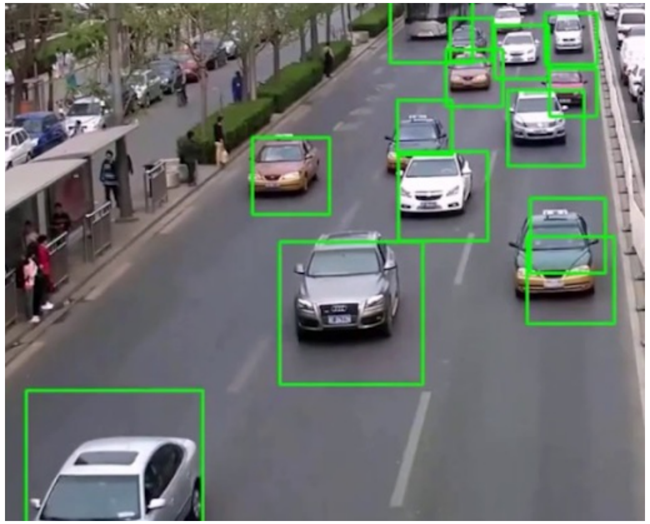
Failure Mode I: Data Violate Assumptions

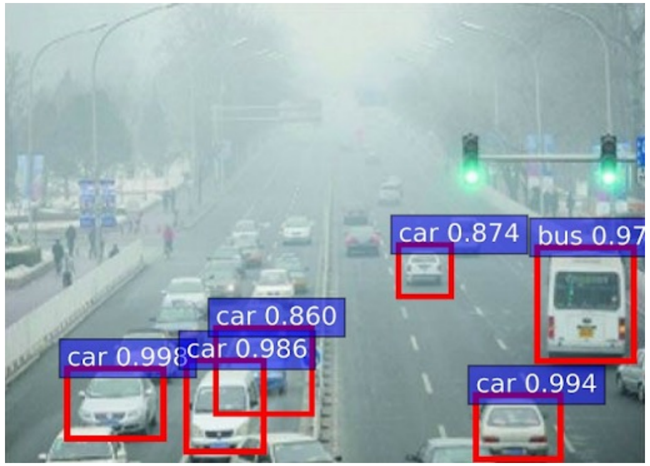
Assumption: Training data is a good representation of the testing

Training → Model → Testing

In the real world:

Failure Mode I: Data Violate Assumptions





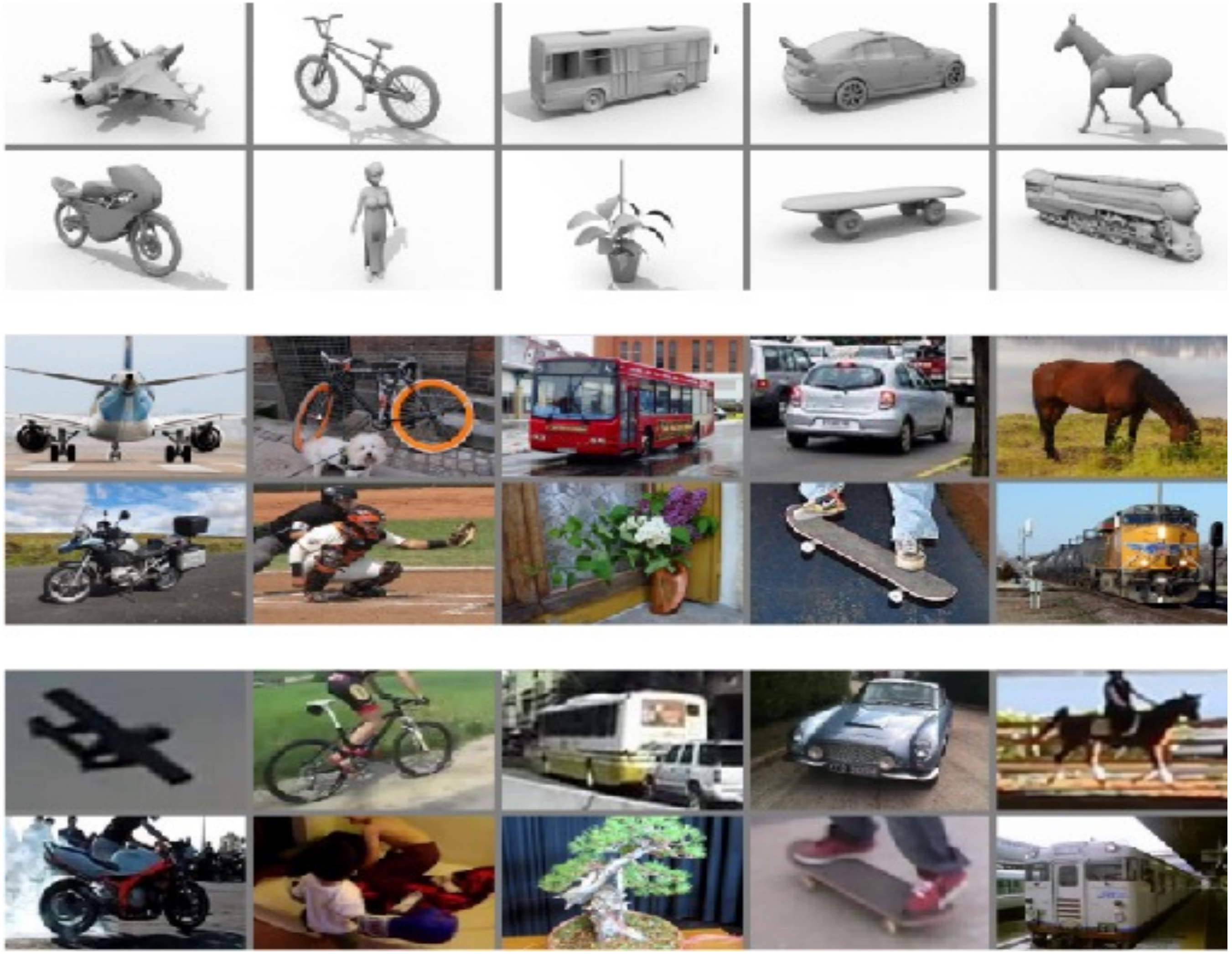
<u>**Degraded Visual Environments (DVEs)**</u>: low-resolution, rain, low-light, haze …

- **… cause degradations for visual understanding**: reduced contrasts, detail occlusions, abnormal illumination, fainted surfaces and color shift...
- It is related to, but not just, image restoration

Failure Mode I: Data Violate Assumptions

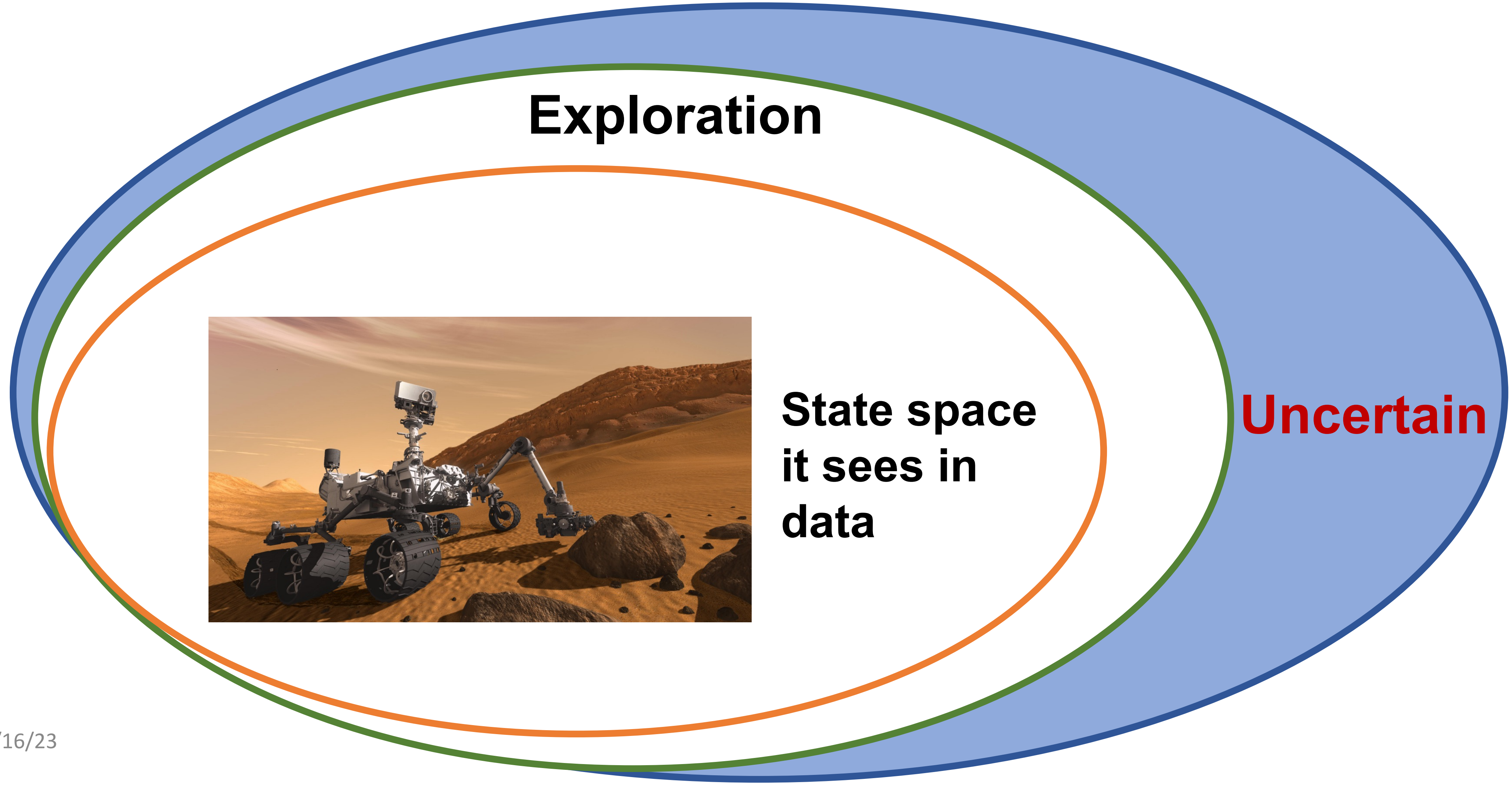
Synthetic:
(Training)

Real World:
(Testing)



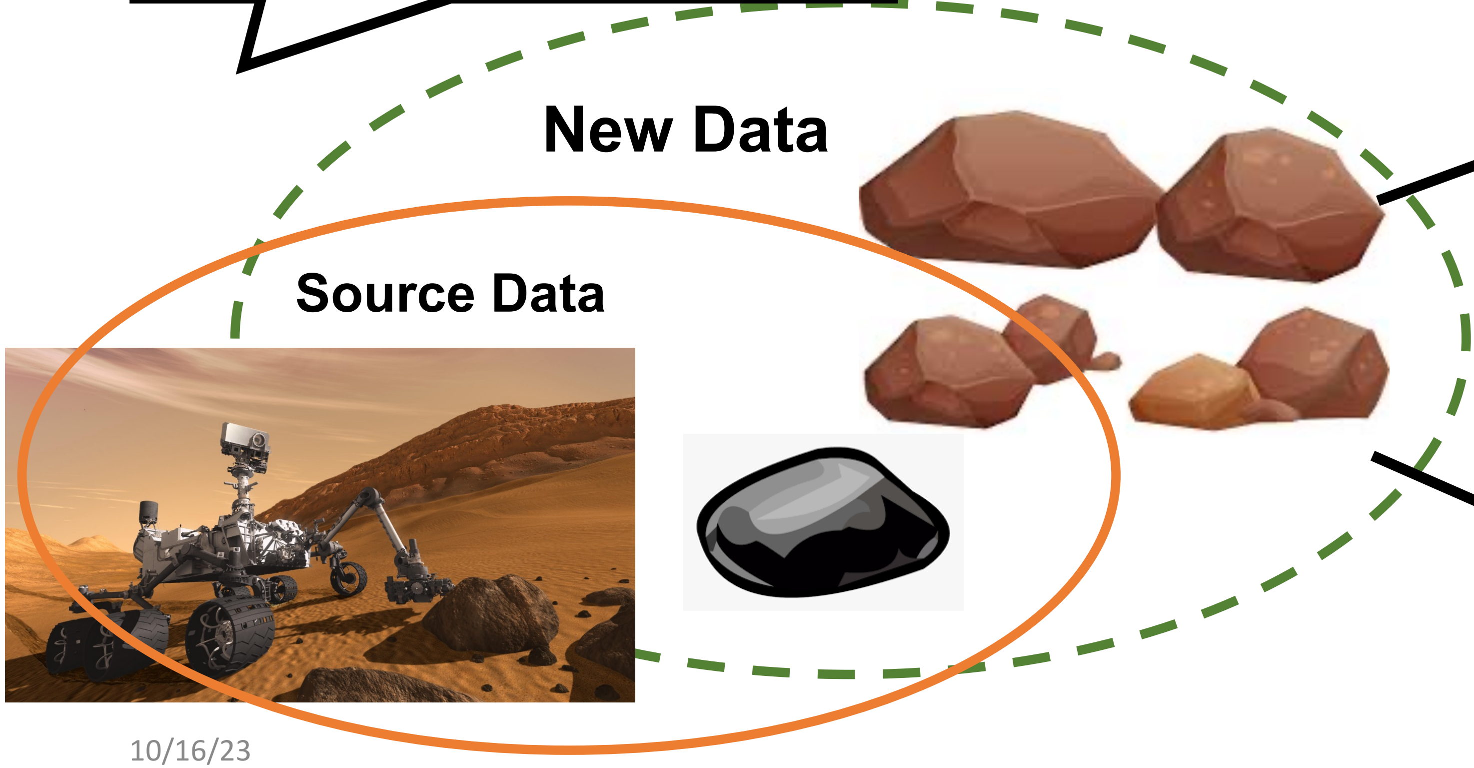
Distribution
Shift

Failure Mode II: Exploration into Unseen Domain



Key: Extrapolation and Model Confidence

Confidence in the new domain?



If I fail, I should fail gently

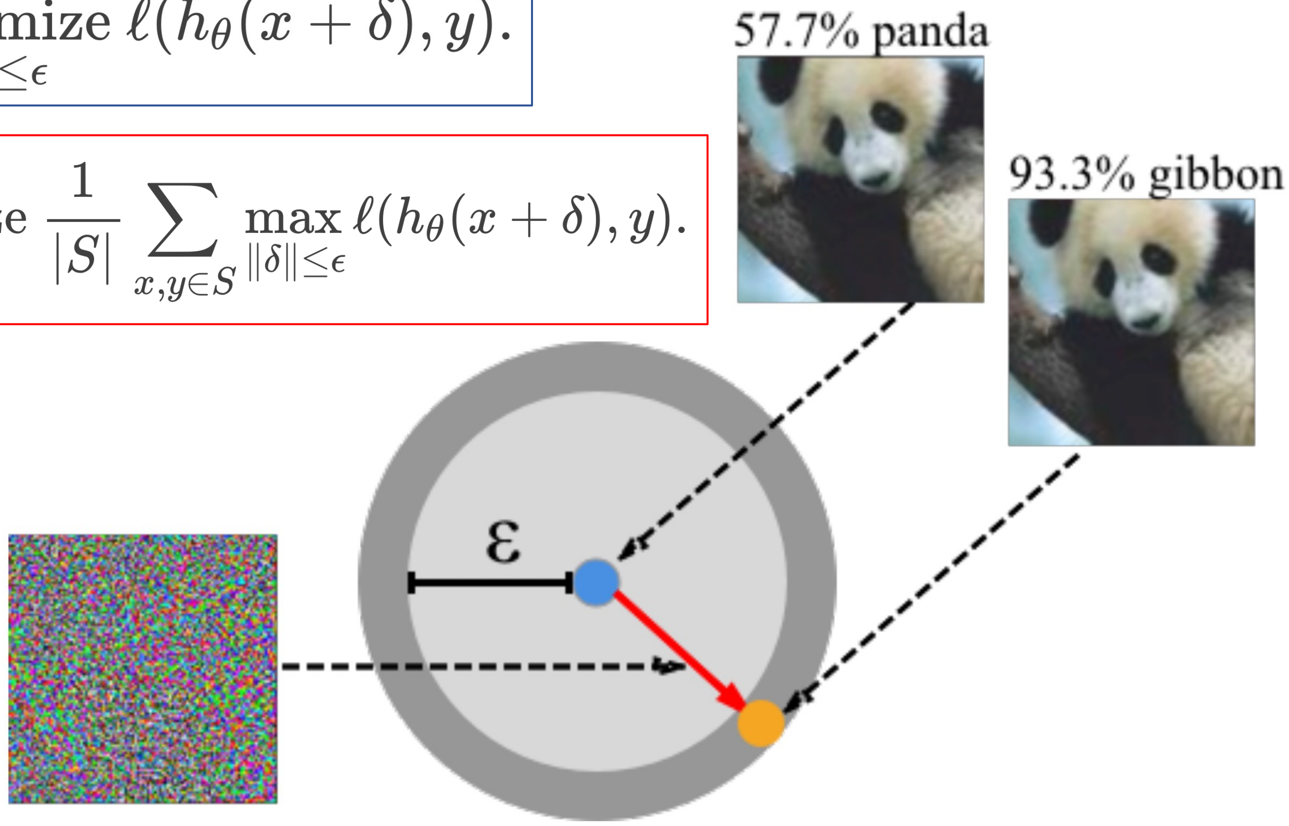
Check constraint satisfaction in the new domain

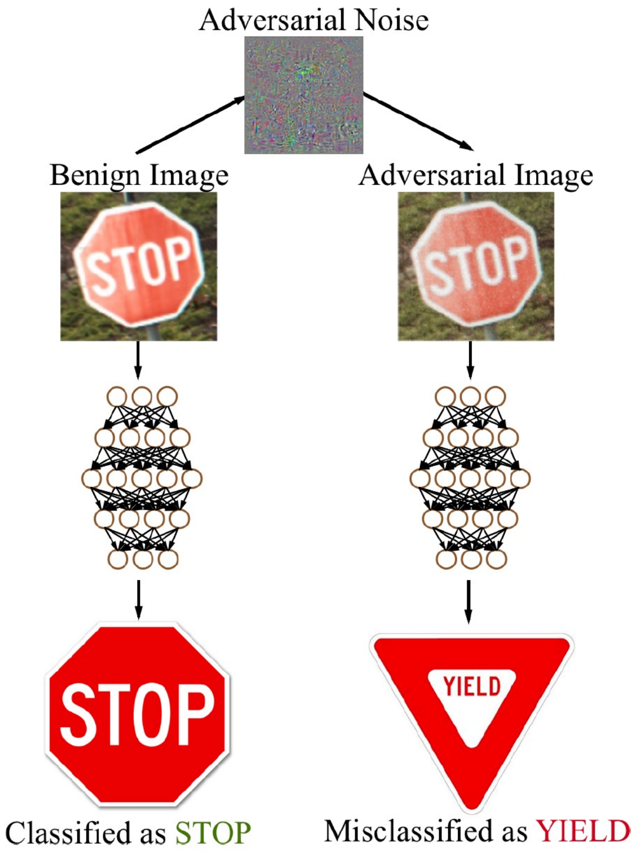
Collect more data to help

Failure Mode III: Malicious Adversary

$$\underset{\|\delta\|\leq\epsilon}{\text{maximize}}\ \ell(h_\theta(x+\delta), y).$$

$$\underset{\theta}{\text{minimize}}\ \frac{1}{|S|}\sum_{x,y\in S}\max_{\|\delta\|\leq\epsilon}\ell(h_\theta(x+\delta), y).$$





Goodfellow et al, "Explaining and Harnessing Adversarial Examples", ICLR 2015.

Failure Mode III: Malicious Adversary

Research Questions:

*How to produce **robust extrapolation** under various **unexpected** distribution shifts in computer vision?*

We will go through many possible answers:

- **Data-level**
    - Enhancing images
- **Model-level:**
    - Uncertainty quantification
    - Domain adaptation and generalization
    - Adversarial defense

Visual Degradation

Data Acquisition

Degradation before Data Acquisition

- Heavy Rain/Snow
- Underwater
- Low Light
- Haze/Sandstorm

Degradation in Data Acquisition

- Downsample
- Motion Blur
- System Noise
- Optical Distortion

Degradation after Data Acquisition

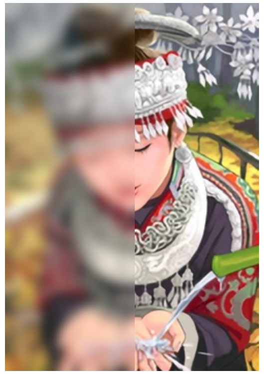
- Scratches
- Watermark
- Mildew
- Compression Loss

Restoration and Enhancement: Tons of Tasks

Underwater Enhancement

Dehazing

Inpainting



Super Resolution

Rain Removal

Denoising

Low Light Enhancement

• • • • • • • •

Learning to Enhance Images

- Data-driven training of “end-to-end” models (usually assuming “pairs”)
- Prior/physical information can still be helpful

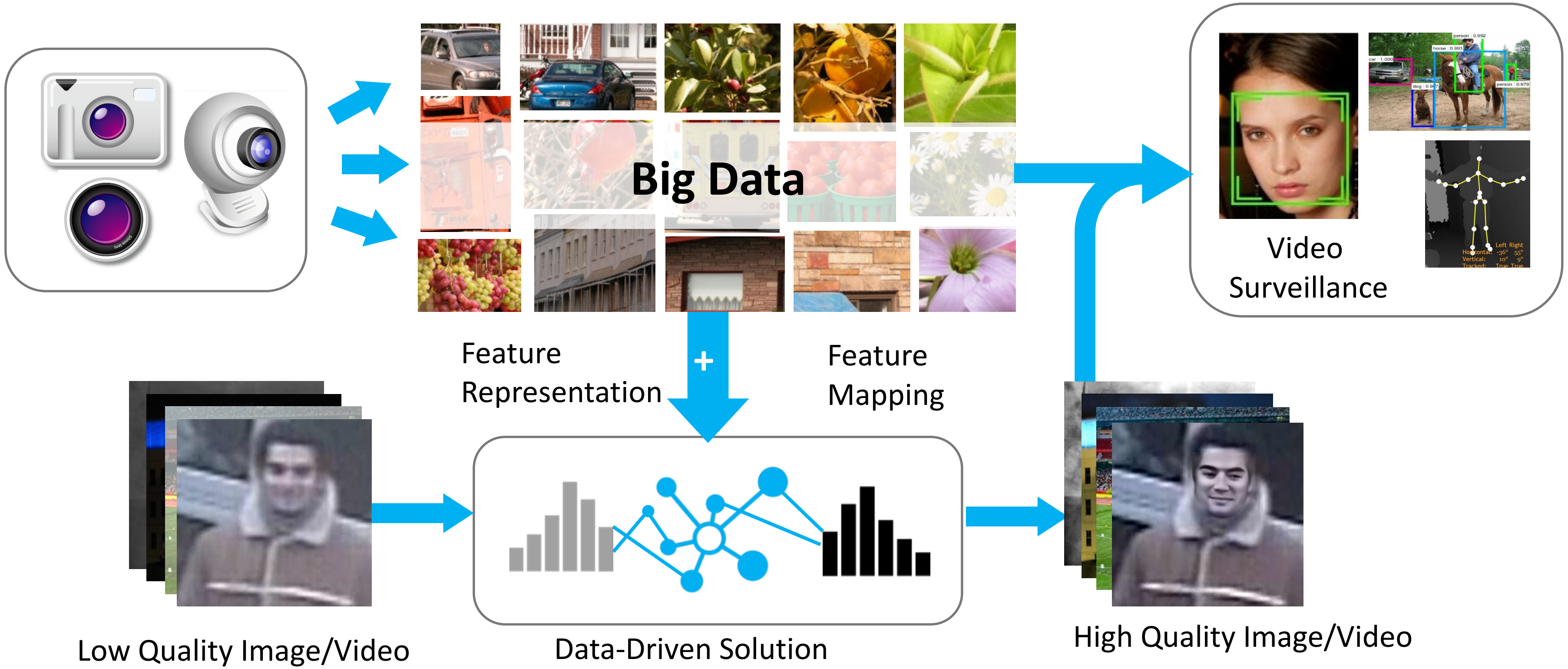


Image Denoising

- Simplest Low-Level Vision Problem

  - Noisy Measurement: $y = x + e$

=

+

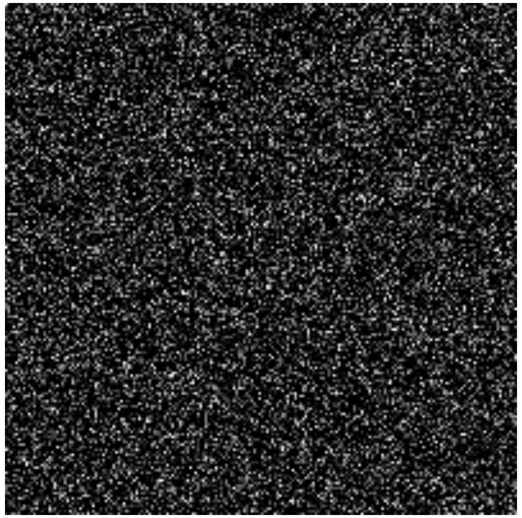


Image Denoising

- Simplest Low-Level Vision Problem

  - Estimate the clean image: $\hat{\boldsymbol{x}} = f(y)$

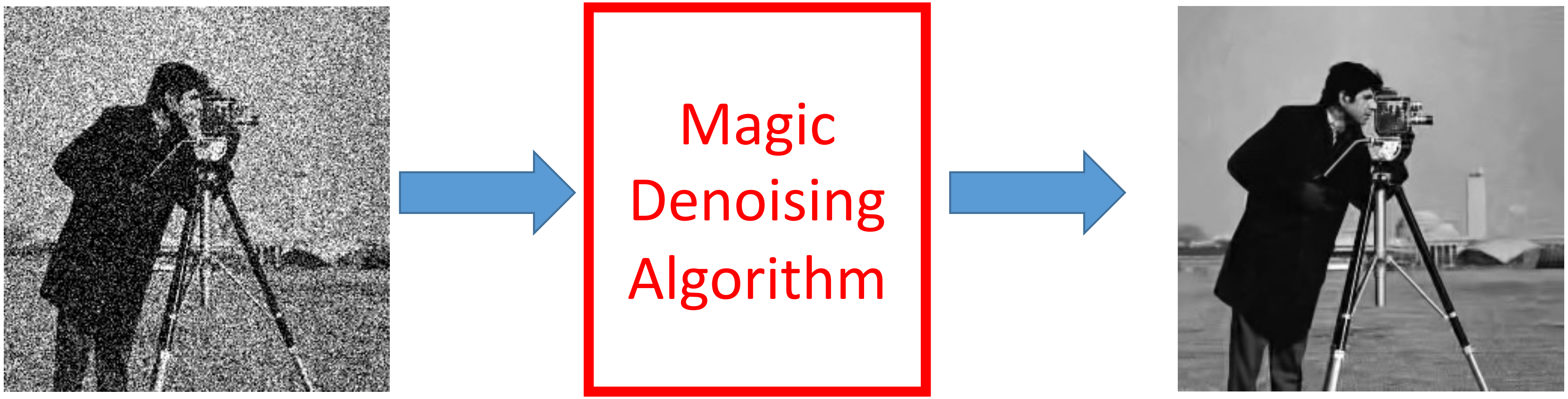


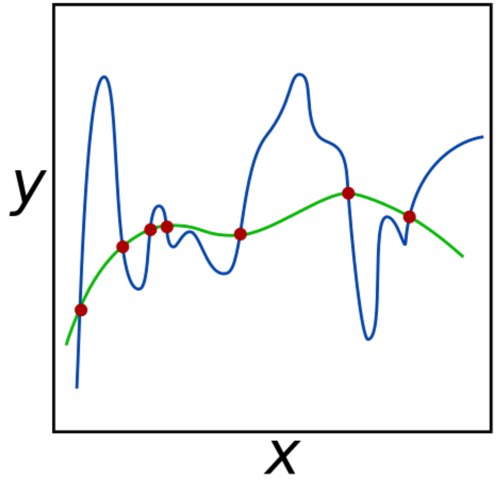
Image Denoising – Conventional Methods

- Collaborative Filtering
  - Non-local Mean, BM3D, etc

Image Denoising – Conventional Methods

- Collaborative Filtering
  - Non-local Mean, BM3D, etc

- Piece-wise Smooth
  - Total Variation, Tikhonov Regularization, etc



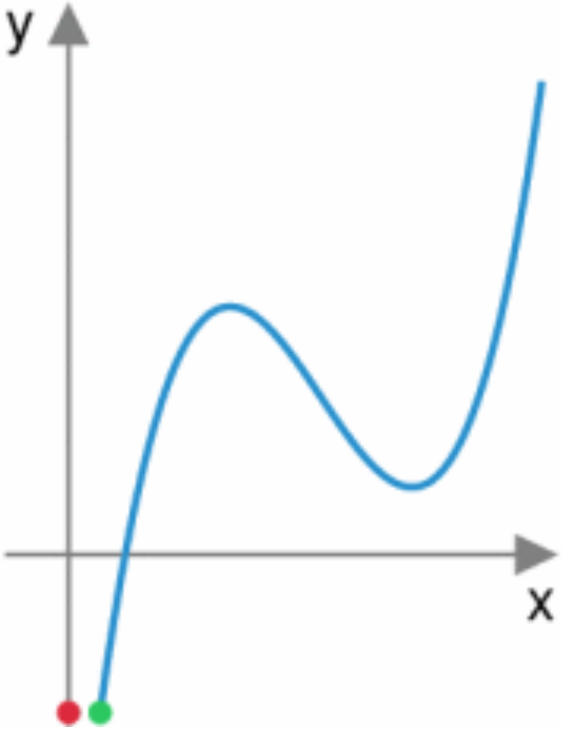
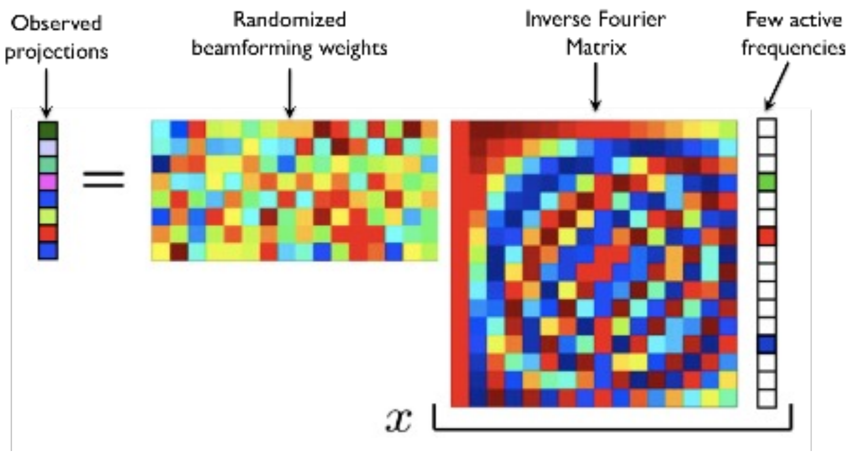
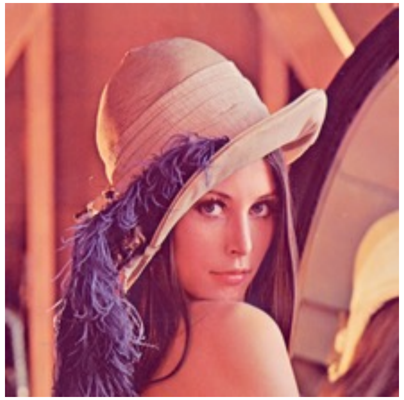
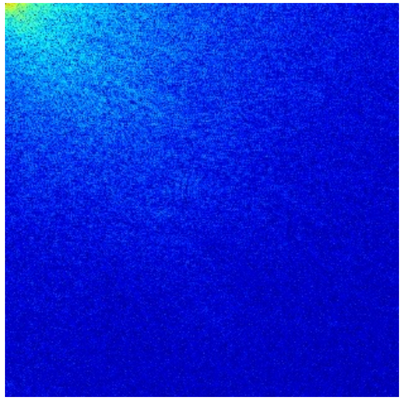


Image Denoising – Conventional Methods

- Collaborative Filtering
  - Non-local Mean, BM3D, etc
- Piece-wise Smooth
  - Total Variation, Tikhonov Regularization, etc
- Sparsity
  - Discrete Cosine Transform (DCT), Wavelets, etc
  - Dictionary Learning: KSVD, OMP, Lasso, etc
  - Analysis KSVD, Transform Learning, etc







Conventional

- **Shallow Model**
  - **Equivalently one free layer**

Deep Learning

- **Deep Model**
  - **Multiple free layers**

Conventional

- Shallow Model
  - One free layer
- **Unsupervised**
  - **No training corpus needed**
  - **Data efficient**

Deep Learning

- Deep Model
  - Multiple free layers

- **Supervised**
  - **Training corpus needed**
  - **Data inefficient**

?

Conventional

- Shallow Model
  - One free layer
- Unsupervised
  - No training corpus needed
  - Data efficient
- **Inverse Problem**
  - **Assumption & Understanding of the Data**
  - **Regularizer & structures of the Model**
  - **Flexible**

Deep Learning

- Deep Model
  - Multiple free layers

- Supervised
  - Training corpus needed
  - Data inefficient
- **Inverse Problem**
  - **Little assumption**
  - **Almost free model**
  - **Few work until recent**

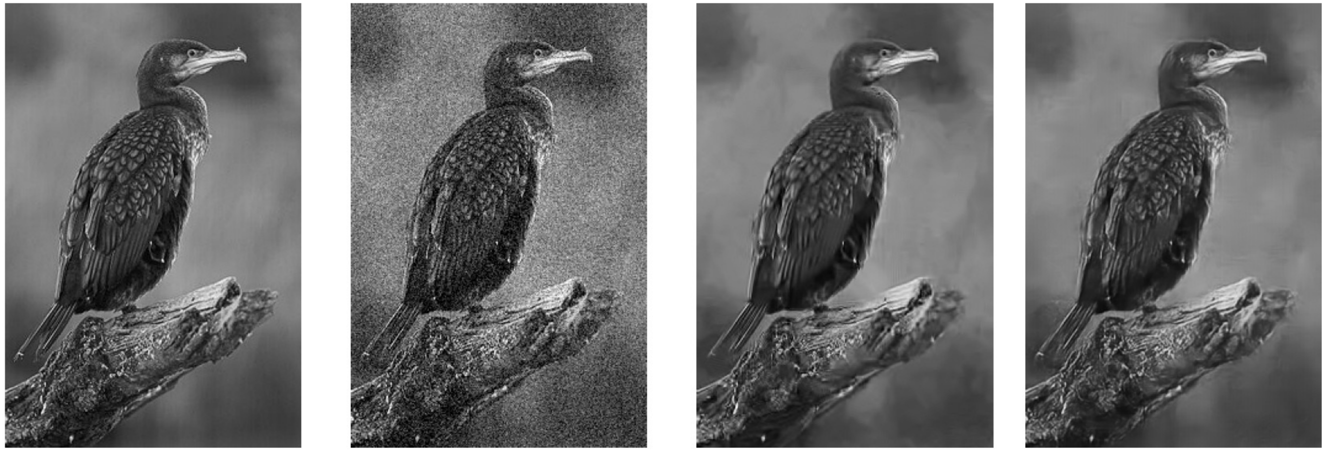
?

Image Denoising by Deep Learning

- **Natural Idea:** train a denoising autoencoder, that regresses clean images from noisy ones
- **It is not easy for deep networks to outperform classical methods such as BM3D!!**
  - BM3D is shown to be better at dealing with self-repeating regular structures
- **How to outperform BM3D using a deep network denoiser? Some verified tips:**
  - The model richness is large enough, i.e. enough hidden layers with sufficiently many hidden units.
  - The patch size is chosen large enough, i.e. a patch contains enough information to fit a complicated denoising function that covers the long tail.
  - The chosen training set is large enough
- **Other benefits of deep network denoiser:**
  - The testing speed of deep networks is much faster than BM3D, KSVD etc., benefiting from GPU.
  - Deep networks can be generalized to other noise types, if correctly supplied in training.
- Recent works show great progress!
  - **Check out Git repo: https://github.com/wenbihan/reproducible-image-denoising-state-of-the-art**

Image Denoising by Deep Learning

- **Reference:** *"Image denoising: Can plain Neural Networks compete with BM3D?"*



clean (name: 008934) | noisy ($\sigma = 25$)PSNR:20.16dB | BM3D: PSNR:29.65dB | ours: PSNR:**30.03**dB

clean (name: barbara) | noisy ($\sigma = 25$)PSNR:20.19dB | BM3D: PSNR:**30.67**dB | ours: PSNR:29.21dB

Image Deblurring

- Blurred Measurement: $y = M \otimes \boldsymbol{x}$

=

$\otimes$

$$M = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Image Deblurring

- Estimate the stable image: $\hat{\boldsymbol{x}} = f(y)$

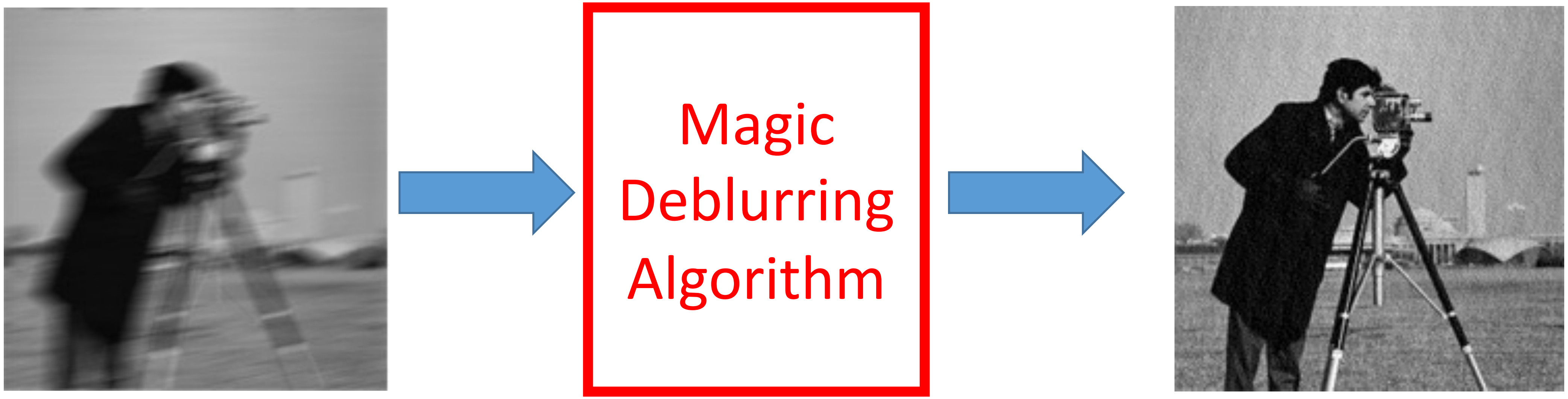


Image Deblurring

- Non-blind Image Deblurring
  - Suppose you know the blurring kernel, $\boldsymbol{M}$.
  - $\widehat{\boldsymbol{x}} = f(y, M)$
  - All training data need to have consistent $\boldsymbol{M}$, as the testing data

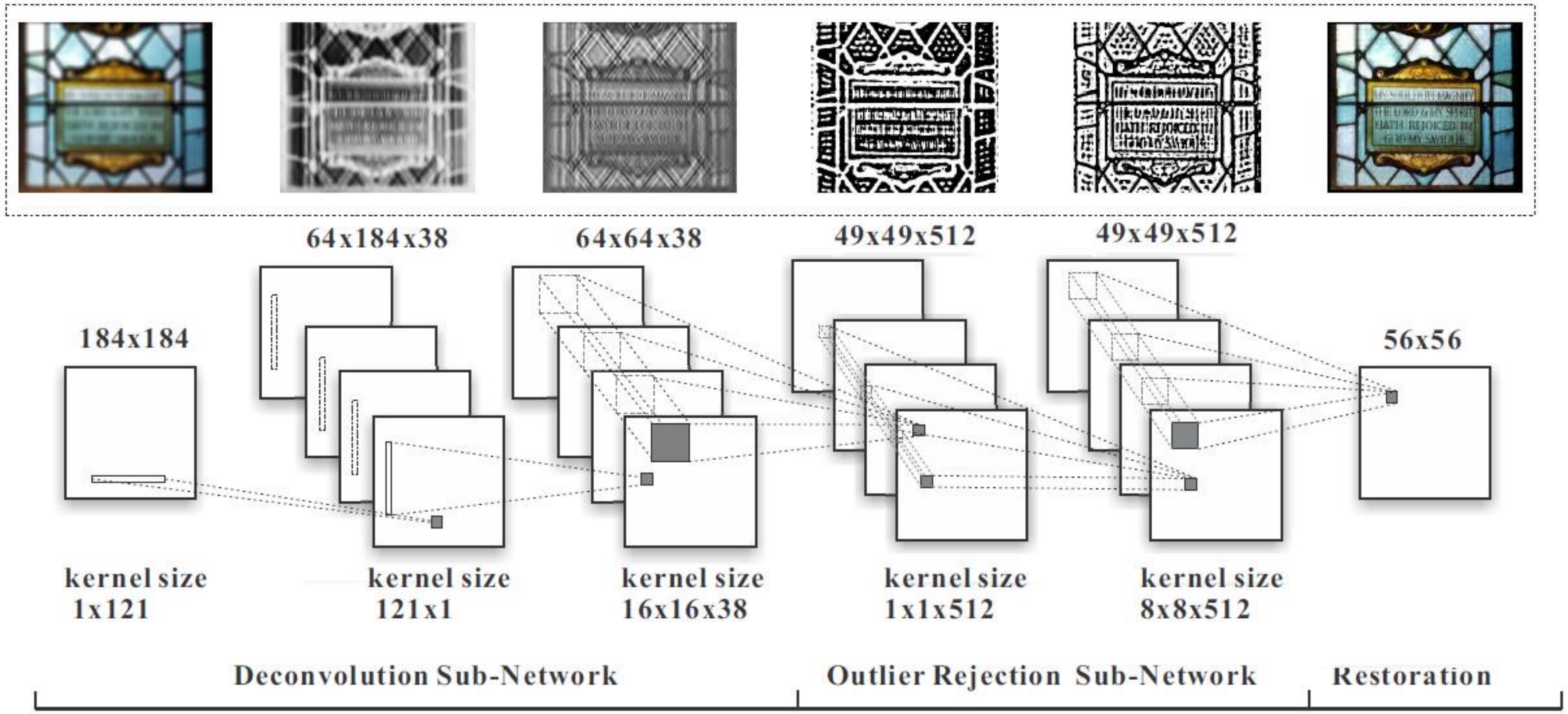
Image Deblurring

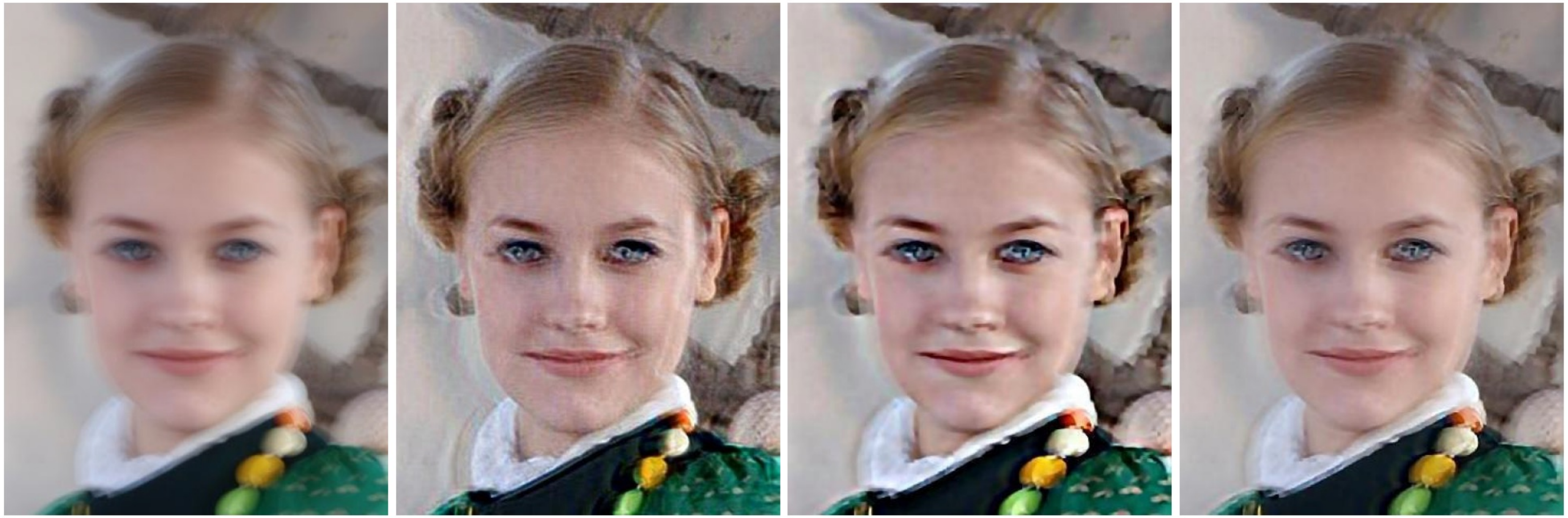
- Non-blind Image Deblurring
  - Suppose you know the blurring kernel, $\boldsymbol{M}$.
  - $\hat{\boldsymbol{x}} = f(y, M)$
  - All training data need to have consistent $\boldsymbol{M}$, as the testing data
- Blind Image Deblurring – More challenging yet practical problem
  - Estimate both the image, and the blurring kernel
  - $\{\hat{\boldsymbol{x}}, M\} = f(y)$

Image Deblurring by Deep Learning

Reference: *"Deep convolutional neural network for image deconvolution"*

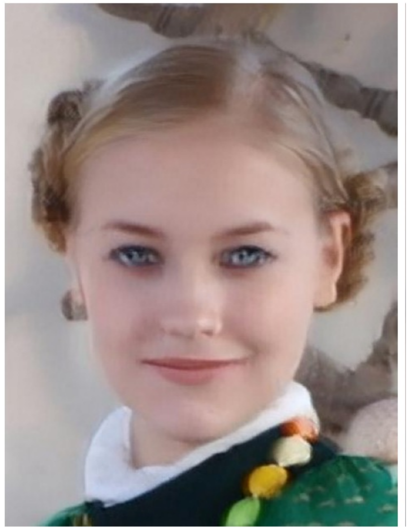
- Key Technical Features:
  - Treat deblurring as a deconvolution task, and the deconvolution operation can be approximated by a convolutional network with very large filter sizes
  - Concatenation of deconvolution CNN module with another denoising CNN module to suppress artifacts and reject outliers





(a) Blurred photo (b) Whyte *et al*. [40] (c) Krishnan *et al*. [14] (d) Sun *et al*. [18]

(e) Xu *et al*. [33] (f) Pan *et al*. [15] (g) DeepDeblur [2] (h) SRN [3]



DeblurGAN V2 (2019)

Image Super-Resolution

- Low-Resolution Measurement: $y = D * M \otimes \boldsymbol{x}$

=

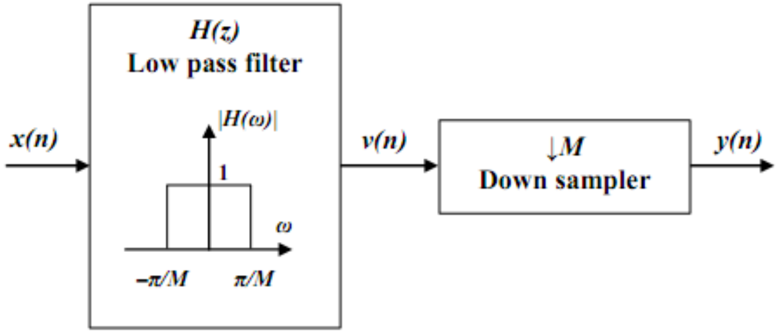


Image Super-Resolution

- Estimate the stable image: $\hat{\boldsymbol{x}} = f(y)$

Image Super Resolution by Deep Learning

Reference: *"Image super-resolution using deep convolutional networks"*

- **Key Technical Features:**
  - Learns an end-to-end mapping from low to high-resolution images as a deep CNN
  - Closely mimic the traditional SR pipeline: LR feature extraction -> coupled LR-HR feature space mapping -> HR image reconstruction

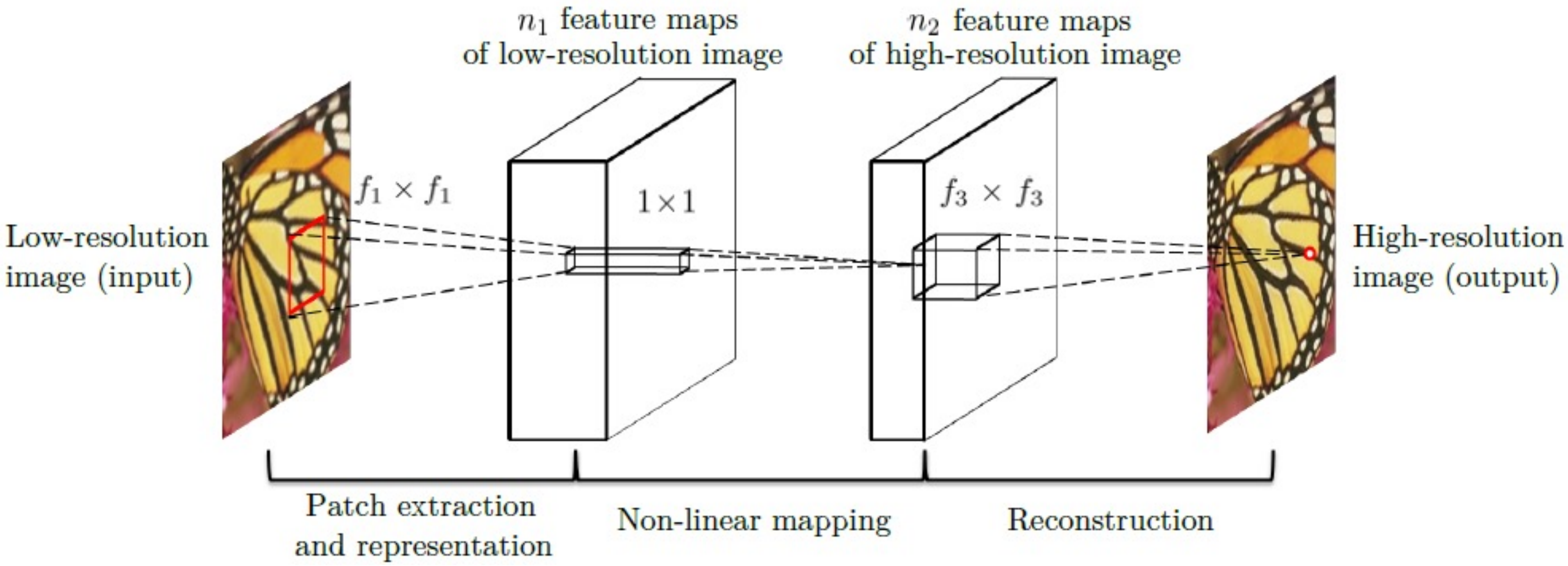
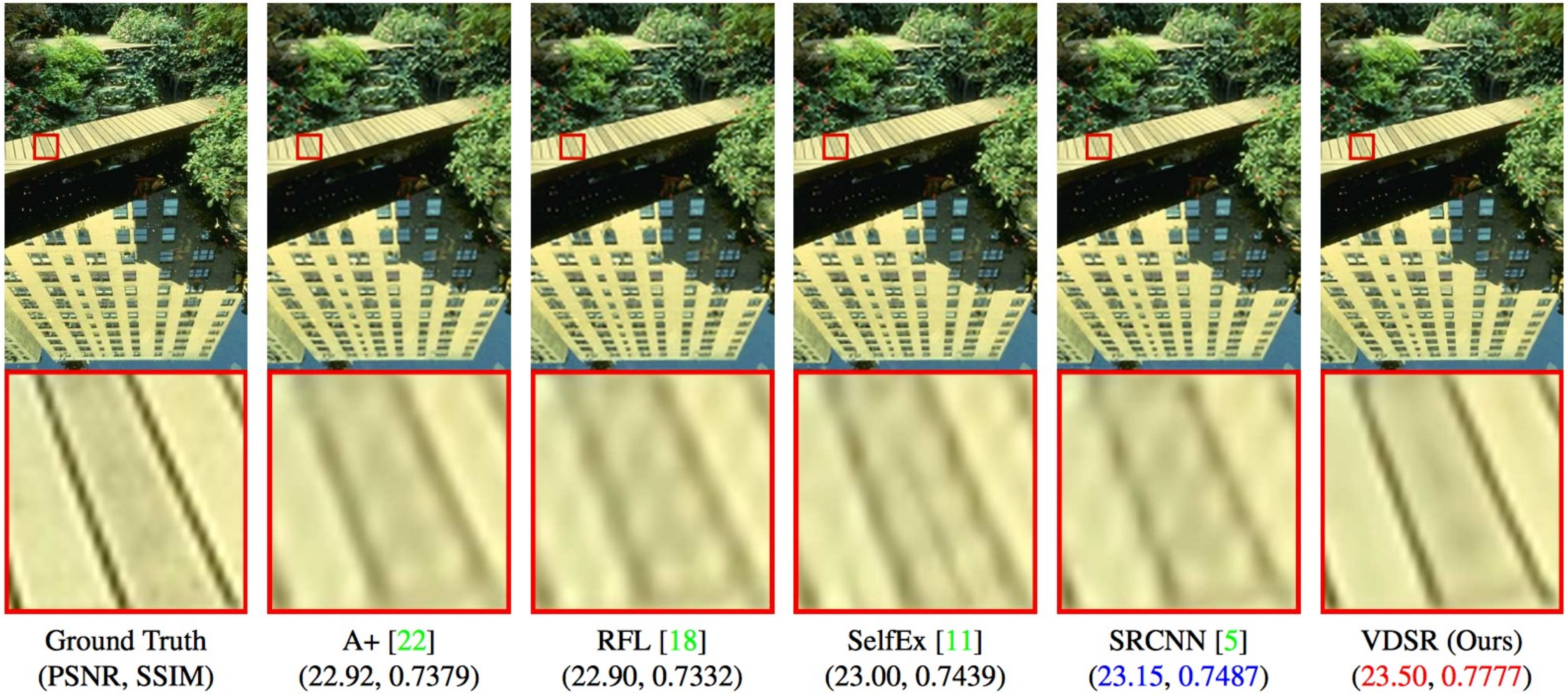


Image Super Resolution by Deep Learning (2013 – 2017)

Super-resolution results of “148026” (*B100*) with scale factor ×3 (from VDSR paper)



New Trends?

- New topic: dehazing, deraining, low light enhancement, etc.
- New goal: human perception **v.s.** machine consumption
- New setting: from supervised to unsupervised training (no “GT”)
  - … or relying on “synthetic pairs”
- New domain: medical images, infrared images, remote sensing images, etc.
- New concern: “All-in-one” adaptivity, efficient implementation, etc.

Shortage of Real-World Generalization

- Most SOTA algorithms are trained with {clean, corrupted} paired data
  - Such paired training data is usually collected by synthesis (assuming known degradation model), which typically oversimplifies the real-world degradations
  - As a result, the trained model "overfits" simpler degradation process and generalizes poorly to real visual degradations

- Real-world collection of paired data?
  - Can be done in small scale and/or in controlled lab environments
    - e.g. some recent datasets in light enhancement, and raindrop removal
  - Very difficult to "scale up", sometimes maybe impossible

VITA

EnlightenGAN: Deep Light Enhancement without Paired Supervision

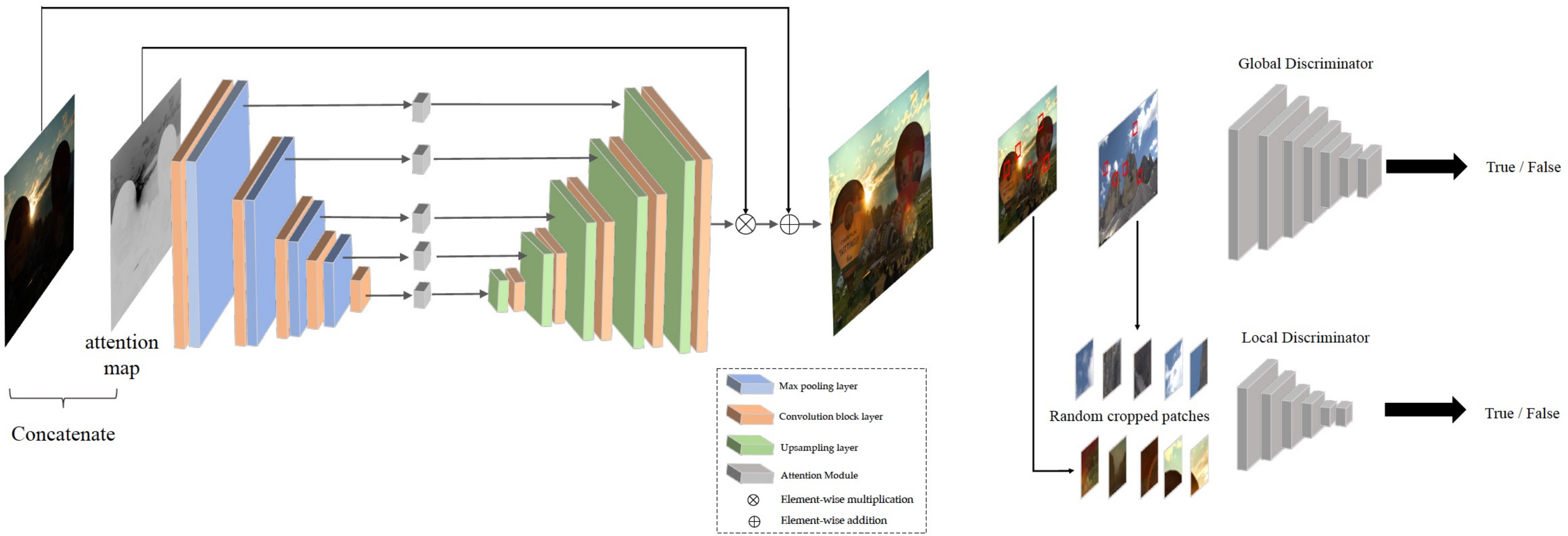
Goal: Light enhancement made automatic, adaptive, and artifact-free

From Supervised to Unsupervised Enhancement

- EnlightenGAN is the first work that successfully introduces unpaired training to low-light image enhancement.
  - It only needs one low-light set A and another normal-light set B to train, while A and B could consist of completely different images!

- What makes **Unpaired Training** unique and attractive?
  - It removes the dependency on paired training data
    - Hence enabling us to train with massive images from different domains
  - It also avoids overfitting any specific data generation/imaging protocol
    - …that previous works implicitly rely on, leading to stronger generalization.
  - It makes EnlightenGAN particularly easy and flexible to be adapted
    - when enhancing real-world low-light images from completely different/unseen domains

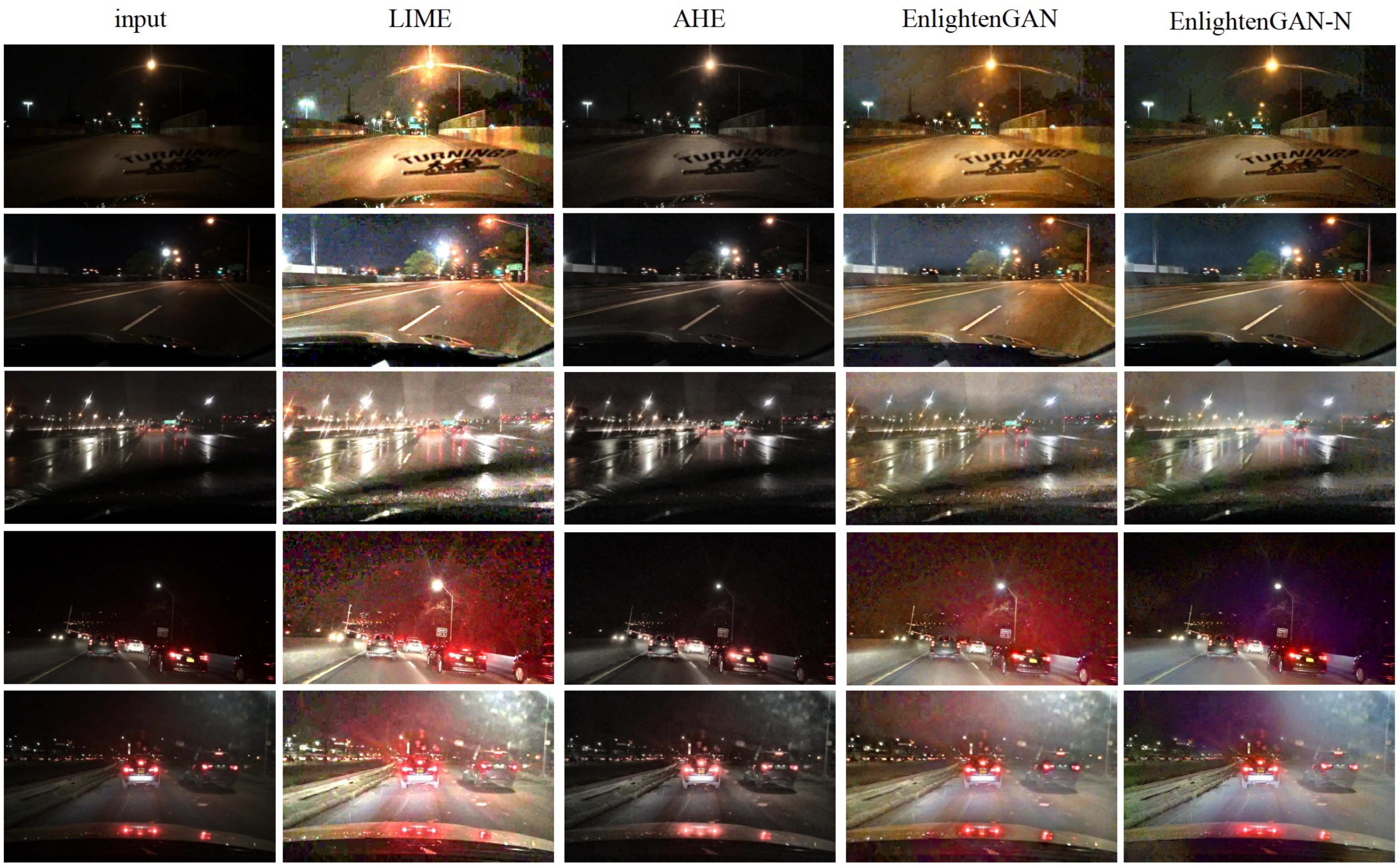
Model Architecture



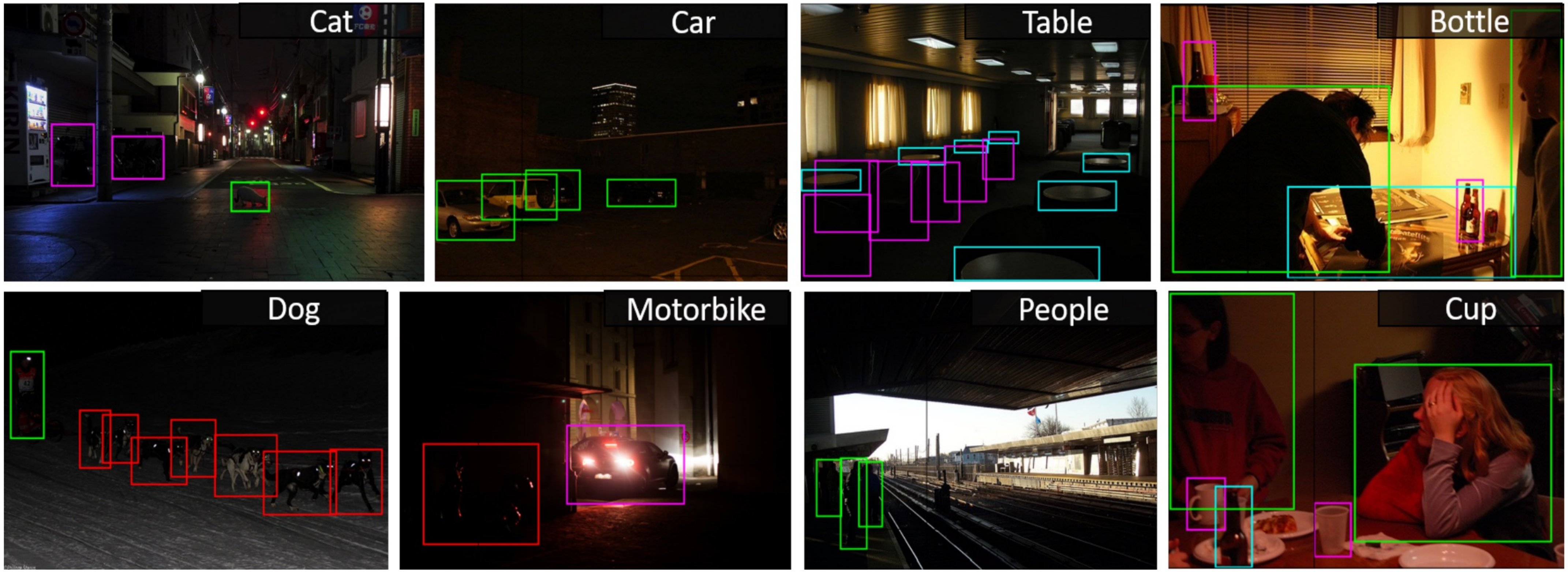
Paper: https://arxiv.org/abs/1906.06972 (pre-print, full version in TIP 2021)
Code: https://github.com/VITA-Group/EnlightenGAN

Comparison with State-of-the-Arts

[New!] Frustratingly Easy Adaptation to New Data



PreProcessing for Improving Classification



- We applying our pretrained EnlightenGAN as a pre-processing step on the testing set of the ExDark dataset , followed by passing through another ImageNet-pretrained ResNet-50 classifier.
- It improves the classification accuracy from 22.02% (top-1) and 39.46% (top-5), to 23.94% (top-1) and 40.92% (top-5) after enhancement.

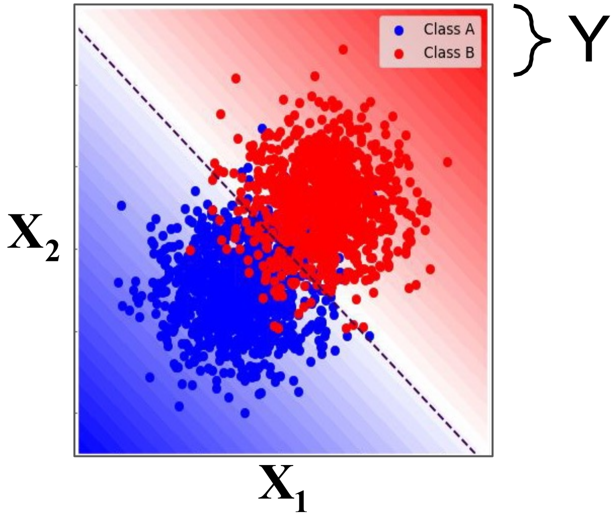
Uncertainty & Robustness for Out-of-Distribution Generalization

What do we mean by Uncertainty?

Return a distribution over predictions rather than a single prediction.

- ***Classification***: Output label along with its confidence.
- ***Regression***: Output mean along with its variance.

Good uncertainty estimates quantify ***when we can trust the model's predictions***.



$$p(\mathbf{y}|\mathbf{x})$$

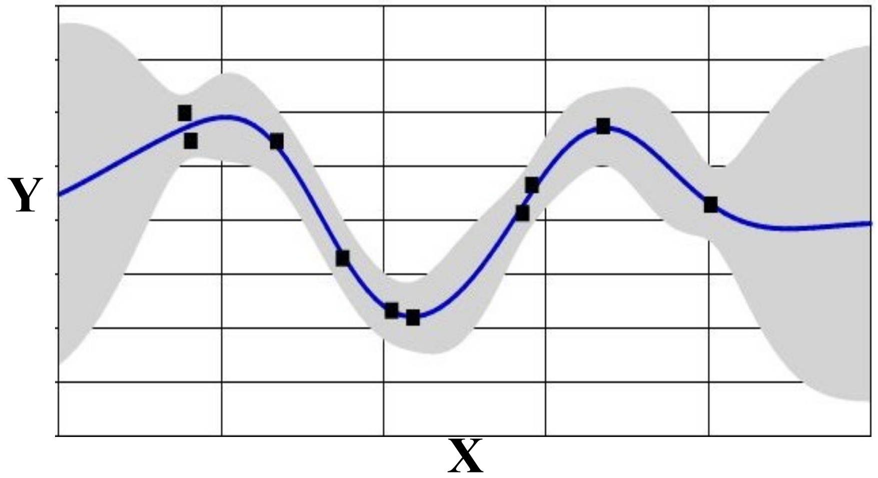


Image credit: Eric Nalisnick

What do we mean by Out-of-Distribution Robustness?

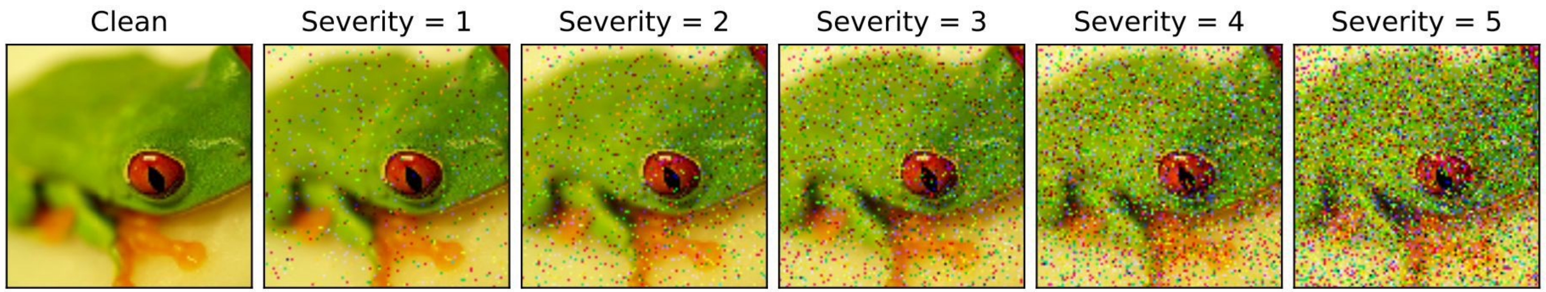
I.I.D. $p_{TEST}(y,x) = p_{TRAIN}(y,x)$

O.O.D. $p_{TEST}(y,x) \neq p_{TRAIN}(y,x)$

Examples of dataset shift:

- ***Covariate shift.*** Distribution of features p(x) changes and p(y|x) is fixed.
- ***Open-set recognition.*** New classes may appear at test time.
- ***Label shift.*** Distribution of labels p(y) changes and p(x|y) is fixed.

ImageNet-C: Varying Intensity for Dataset Shift



Increasing dataset shift

I.I.D test set

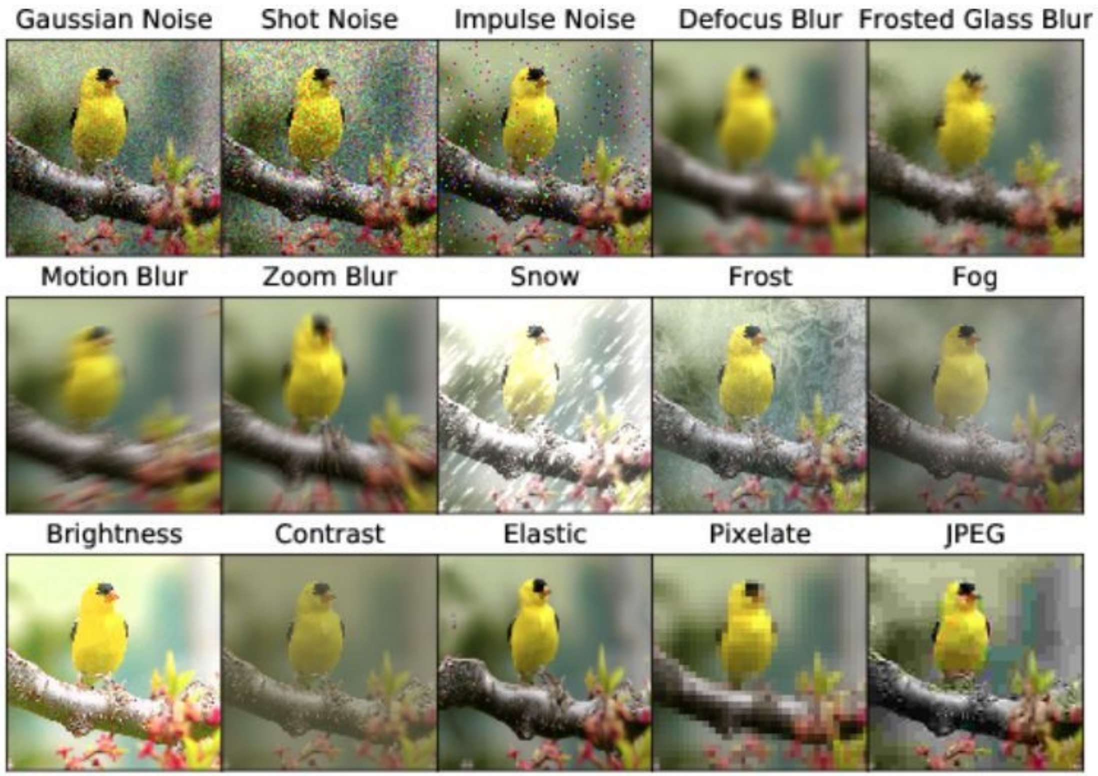
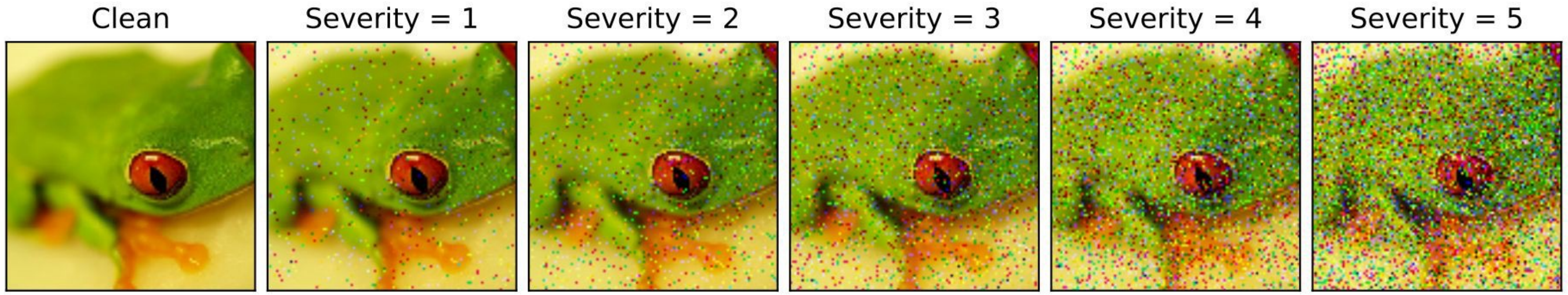
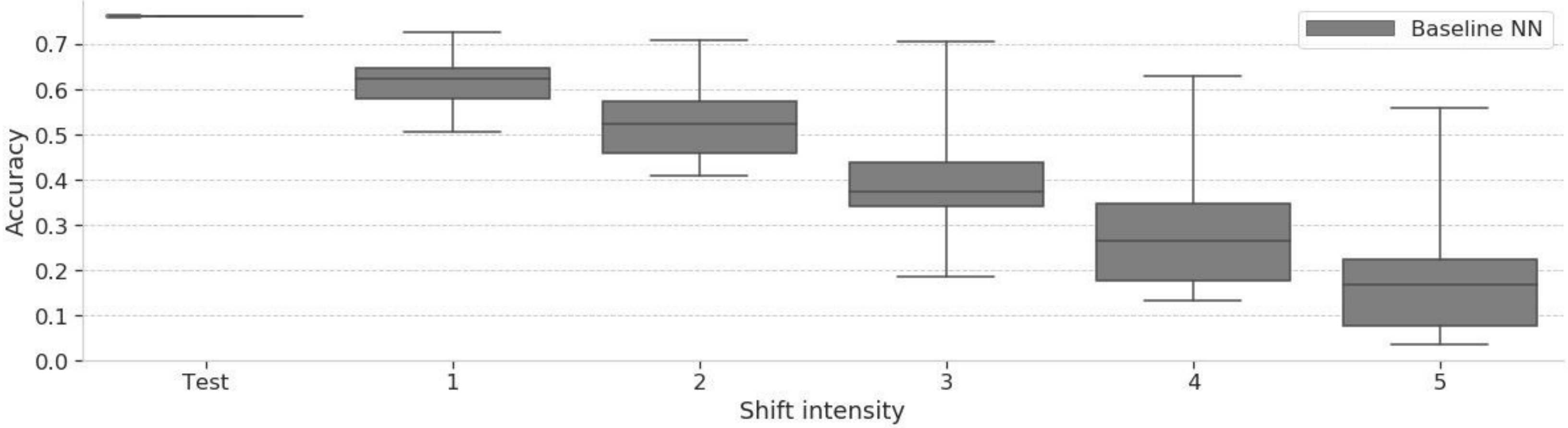


Image source: Benchmarking Neural Network Robustness to Common Corruptions and Perturbations, Hendrycks & Dietterich, 2019.

Neural networks do not generalize under covariate shift



- **Accuracy drops** with increasing shift on Imagenet-C

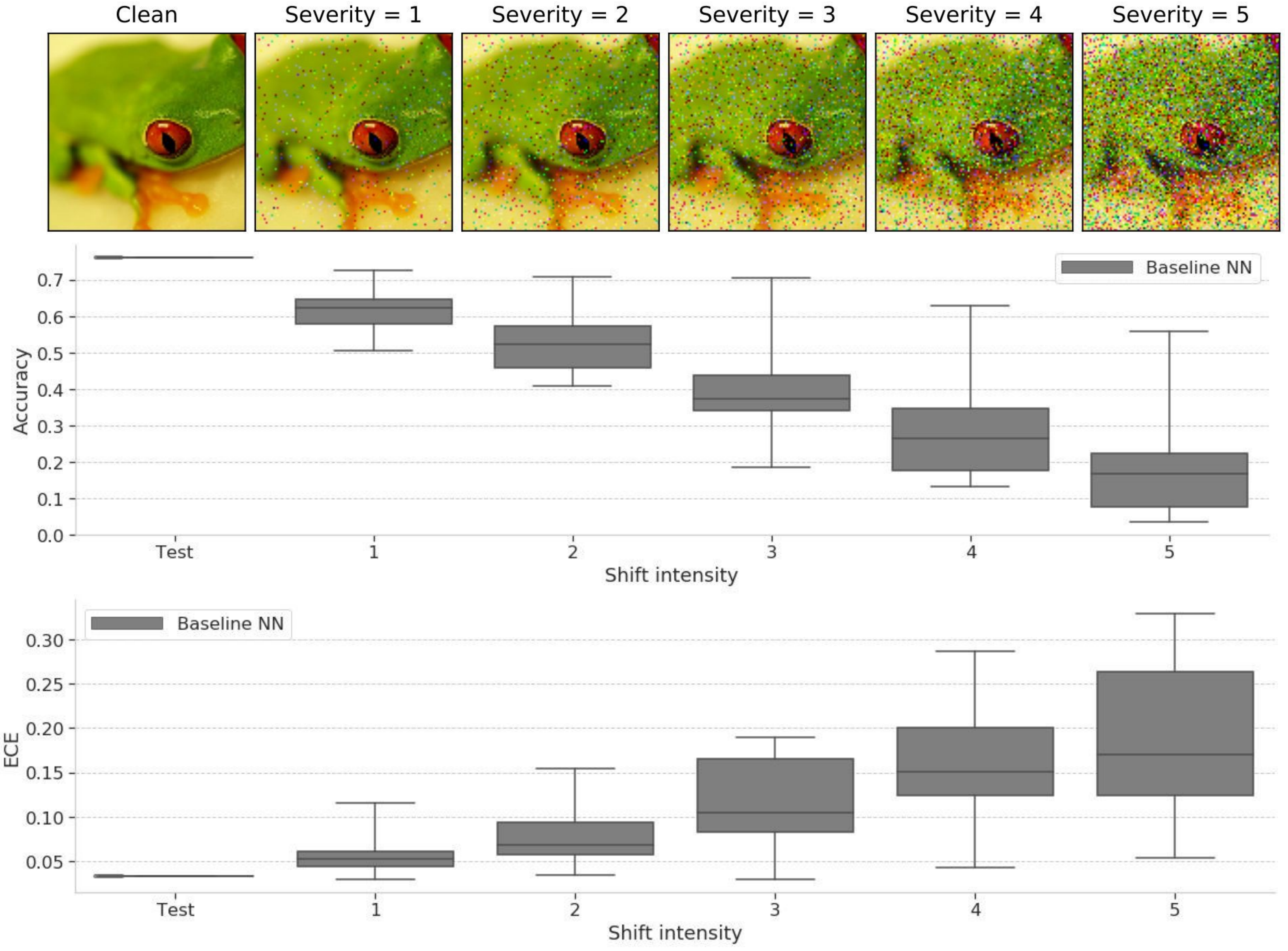


- But do the models know that they are less accurate?

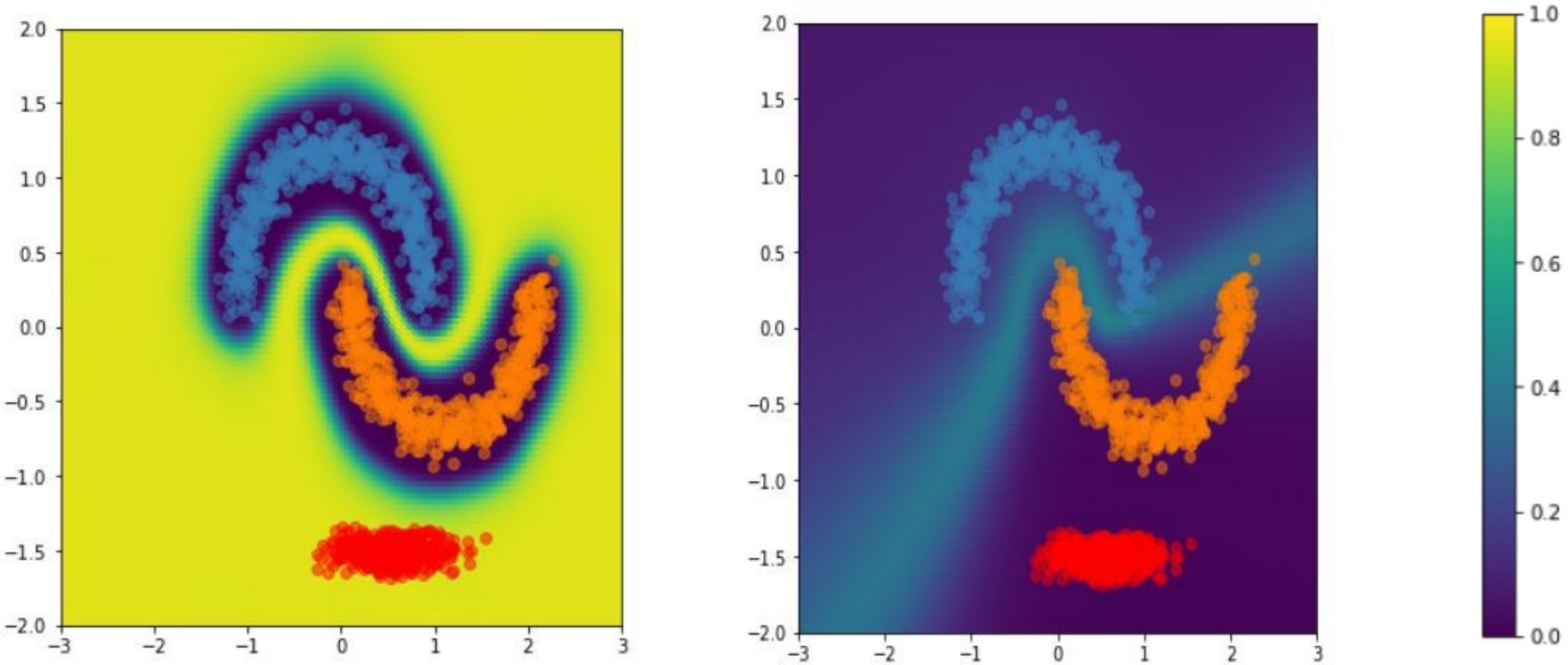
Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift?, [Ovadia et al. 2019]()

Neural networks ***do not know when they don't know***

- **Accuracy drops** with increasing shift on Imagenet-C
- **Quality of uncertainty degrades with shift** -> "overconfident mistakes"



Models assign high confidence predictions to OOD inputs



High uncertainty
(low confidence)

Low uncertainty
(high confidence)

Ideal behavior

Deep neural networks

Trust model when x* is close to $p_{TRAIN}(x,y)$

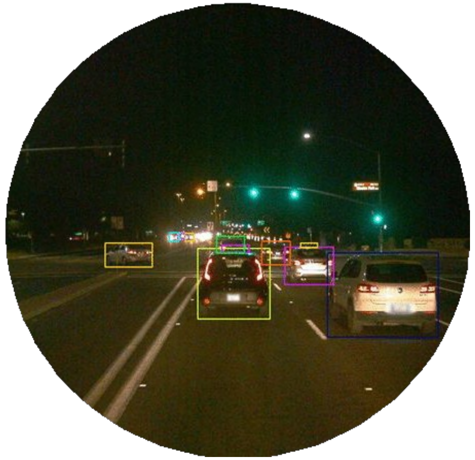
Image source: “Simple and Principled Uncertainty Estimation with Deterministic Deep Learning via Distance Awareness” [Liu et al. 2020](#)

Self-driving cars

Dataset shift:

- Time of day / Lighting
- Geographical location (City vs suburban)
- Changing conditions (Weather / Construction)

Daylight



Night

Weather

Construction

Downtown

Suburban

Image credit: Sun et al, [Waymo Open Dataset]

Open Set Recognition

- Example: Classification of genomic sequences

- High accuracy on known classes is not sufficient

- Need to be able to detect inputs that do not belong to one of the known classes

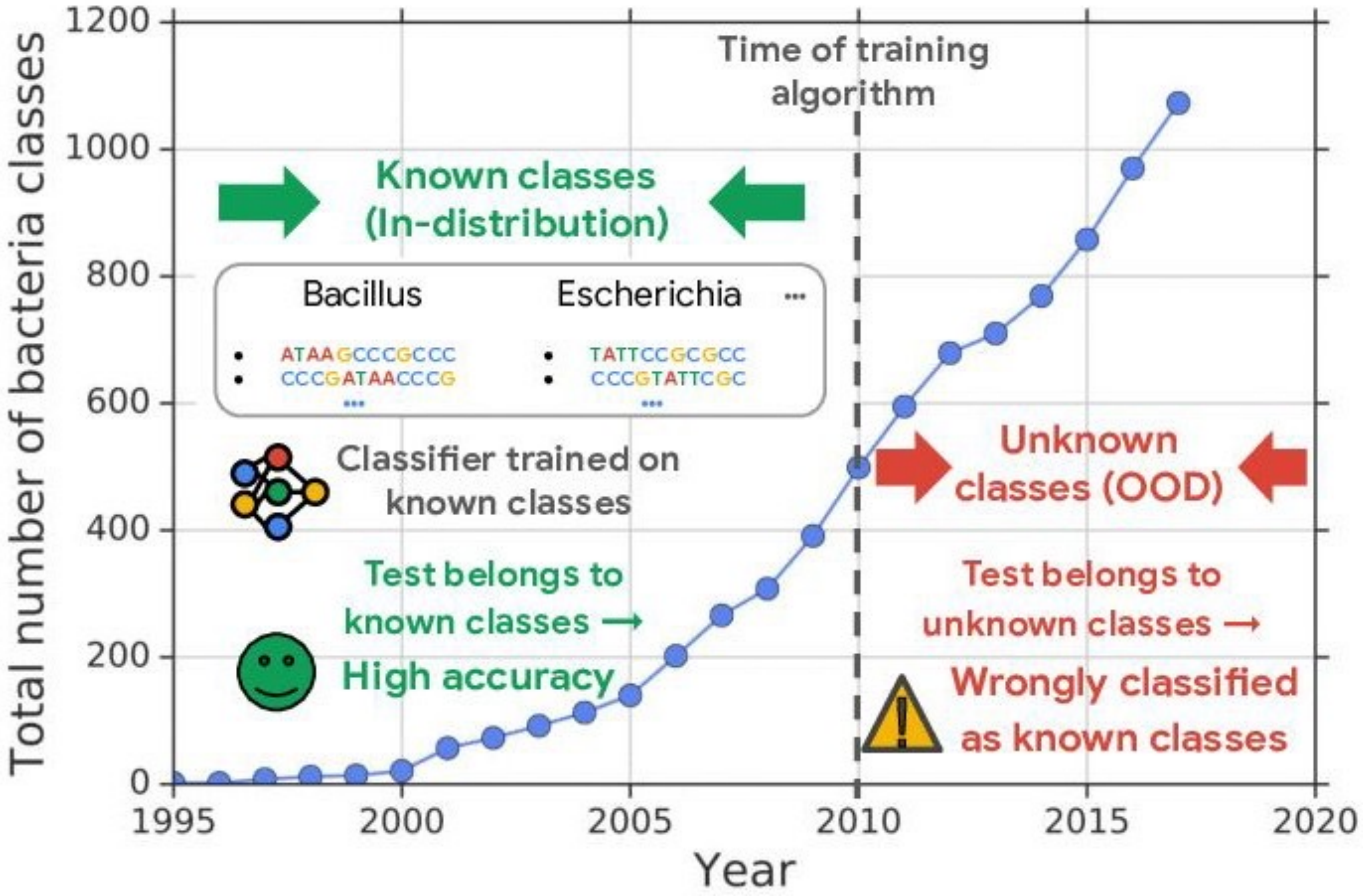
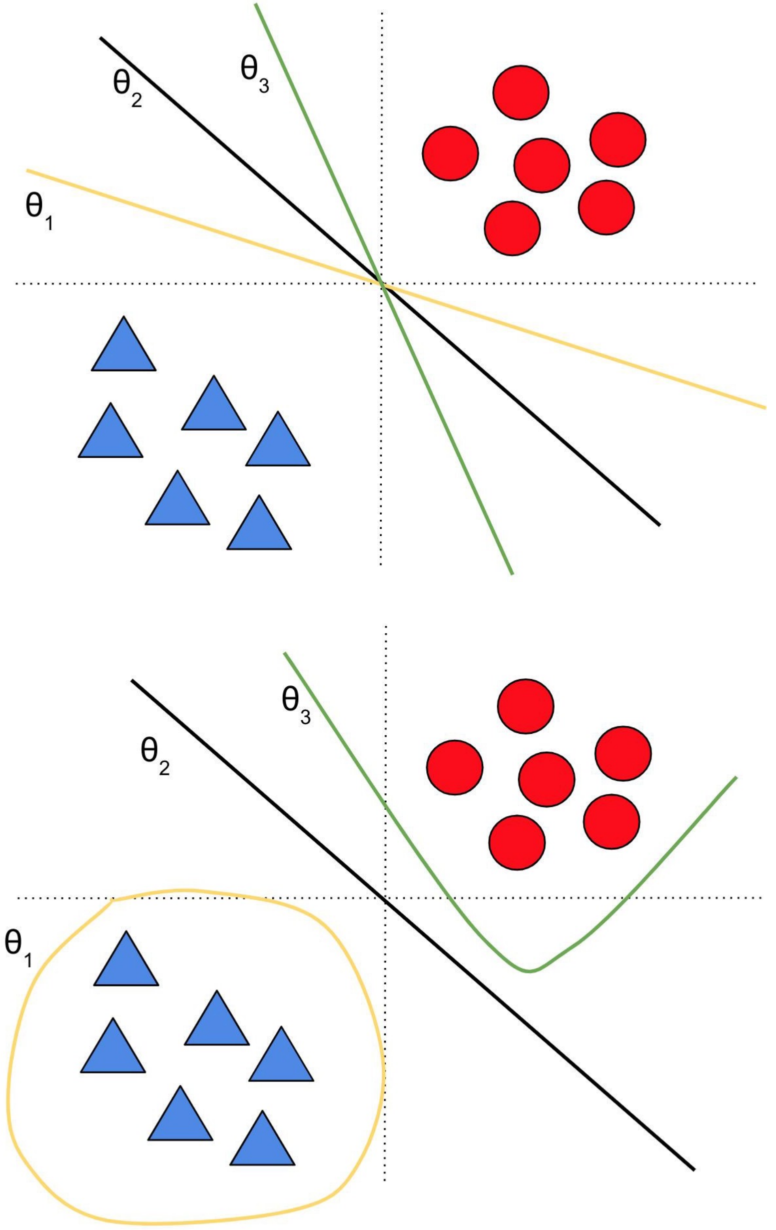


Image source: https://ai.googleblog.com/2019/12/improving-out-of-distribution-detection.html

Sources of uncertainty: *Model uncertainty*

- Many models can fit the training data well
- Also known as ***epistemic uncertainty***
- Model uncertainty is "**reducible**"
  - Vanishes in the limit of infinite data (subject to model identifiability)
- Models can be from same hypotheses class (e.g. linear classifiers in top figure) or belong to different hypotheses classes (bottom figure).



Sources of uncertainty: *Data uncertainty*

- Labeling noise (ex: human disagreement)
- Measurement noise (ex: imprecise tools)
- *Missing* data (ex: partially observed features, unobserved confounders)

- Also known as ***aleatoric uncertainty***
- Data uncertainty is "**irreducible***"
    - Persists even in the limit of infinite data
    - *Could be reduced with additional features/views

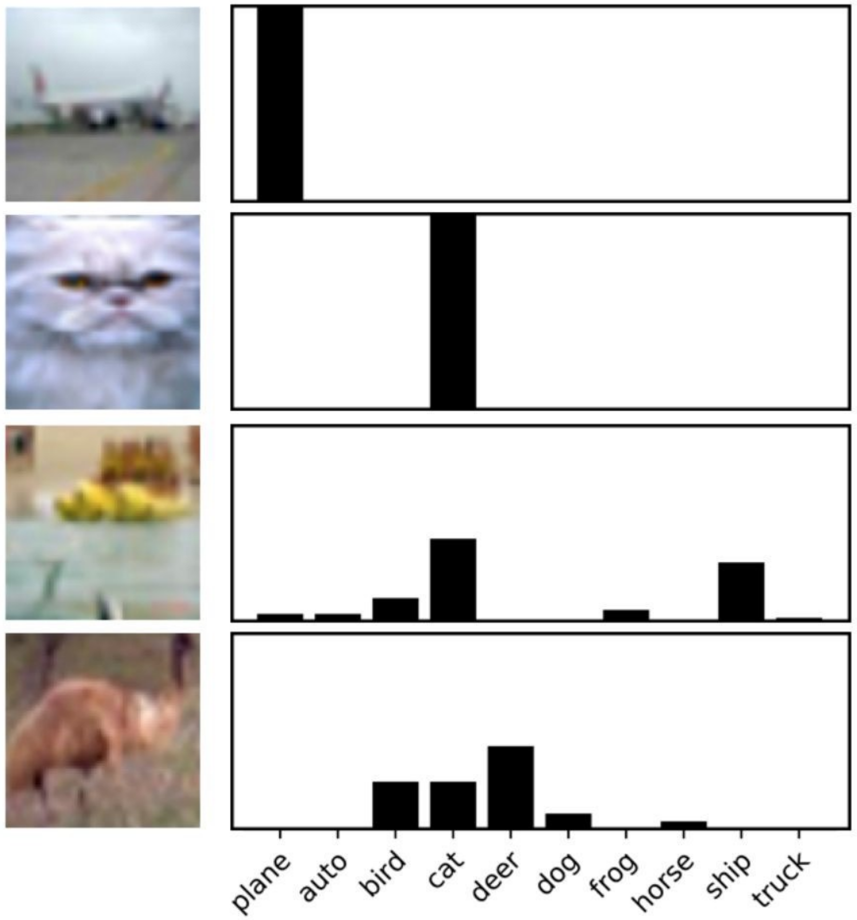


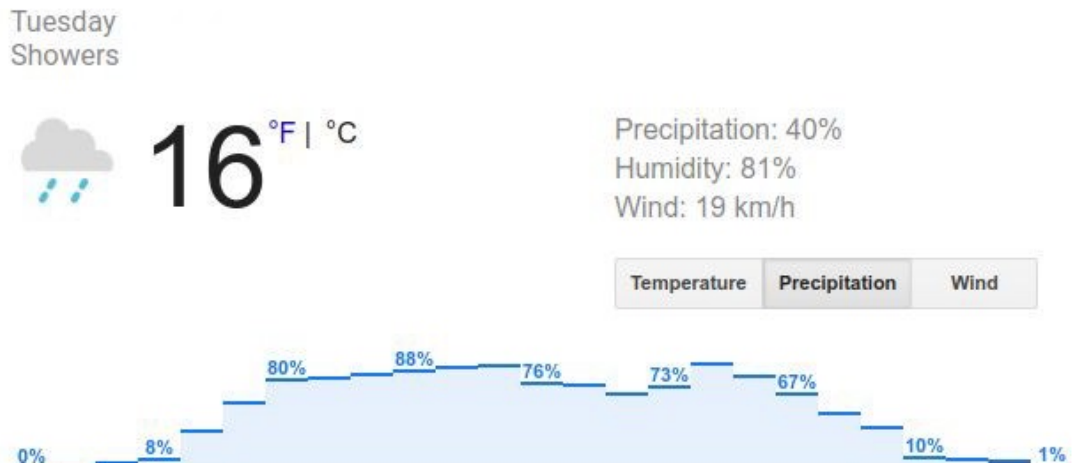
Image source: Battleday et al. 2019 "Improving machine classification using human uncertainty measurements"

How do we measure the quality of uncertainty?

Calibration Error = |Confidence - Accuracy|

Of all the days where the model predicted rain with 80% probability, what fraction did we observe rain?

- 80% implies perfect calibration
- Less than 80% implies model is overconfident
- Greater than 80% implies model is under-confident



How do we measure the quality of uncertainty?

Expected Calibration Error [Naeini+ 2015]:

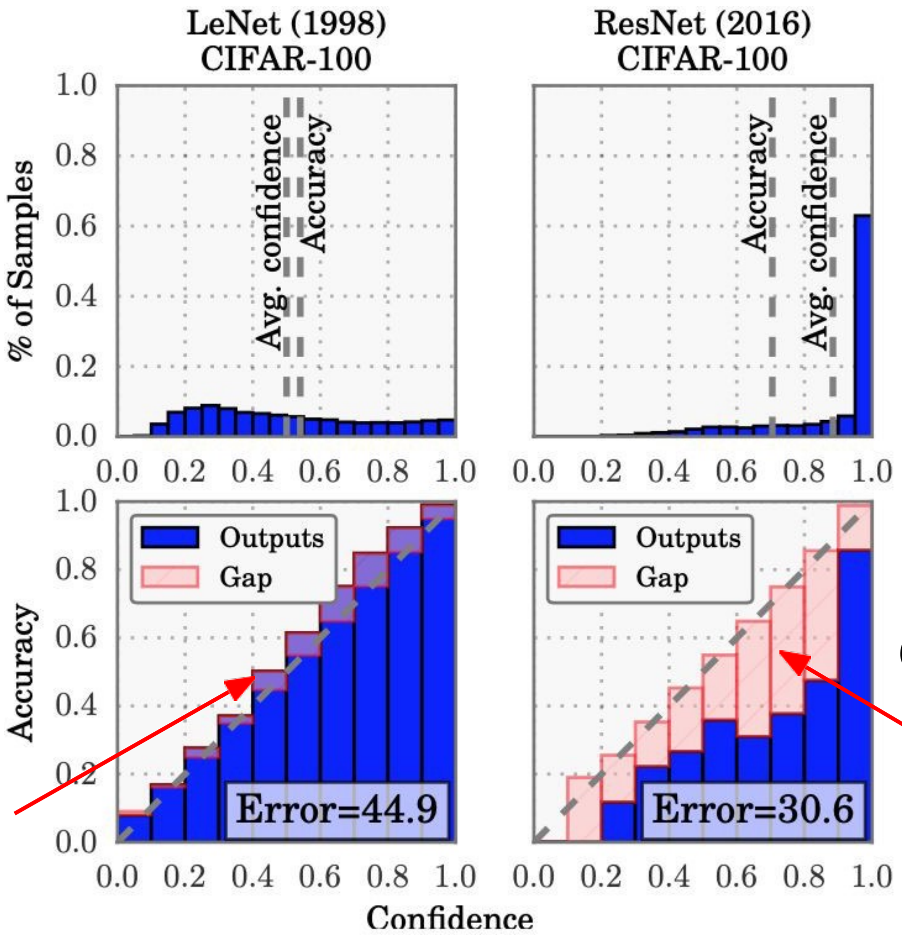
$$\mathrm{ECE} = \sum_{b=1}^{B} \frac{n_b}{N} \left|\mathrm{acc}(b) - \mathrm{conf}(b)\right|$$

- Bin the probabilities into B bins.
- Compute the within-bin accuracy and within-bin predicted confidence.
- Average the calibration error across bins (weighted by number of points in each bin).

How do we measure the quality of uncertainty?

Expected Calibration Error [Naeini+ 2015]:

$$\mathrm{ECE} = \sum_{b=1}^{B} \frac{n_b}{N} \left|\mathrm{acc}(b) - \mathrm{conf}(b)\right|$$



Confidence >Accuracy

=> *Overconfident*

Confidence <Accuracy

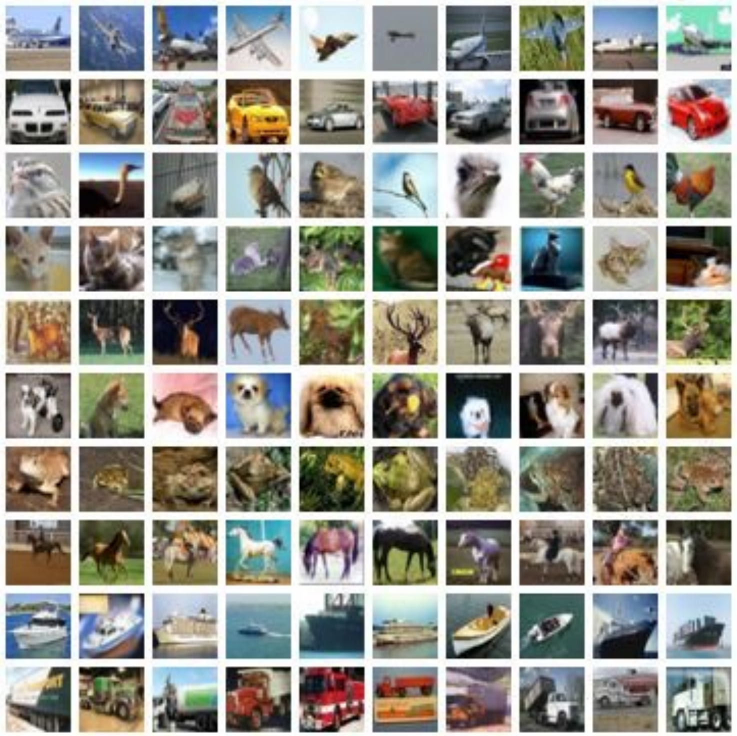
=> *Underconfident*

Image source: Guo+ 2017 “On calibration of modern neural networks”

How do we measure the quality of uncertainty , practically?

Evaluate model on **out-of-distribution (OOD) inputs** which do not belong to any of the existing classes

- Max confidence
- Entropy of p(y|x)



CIFAR-10 (i.i.d test inputs)

SVHN (o.o.d test inputs)

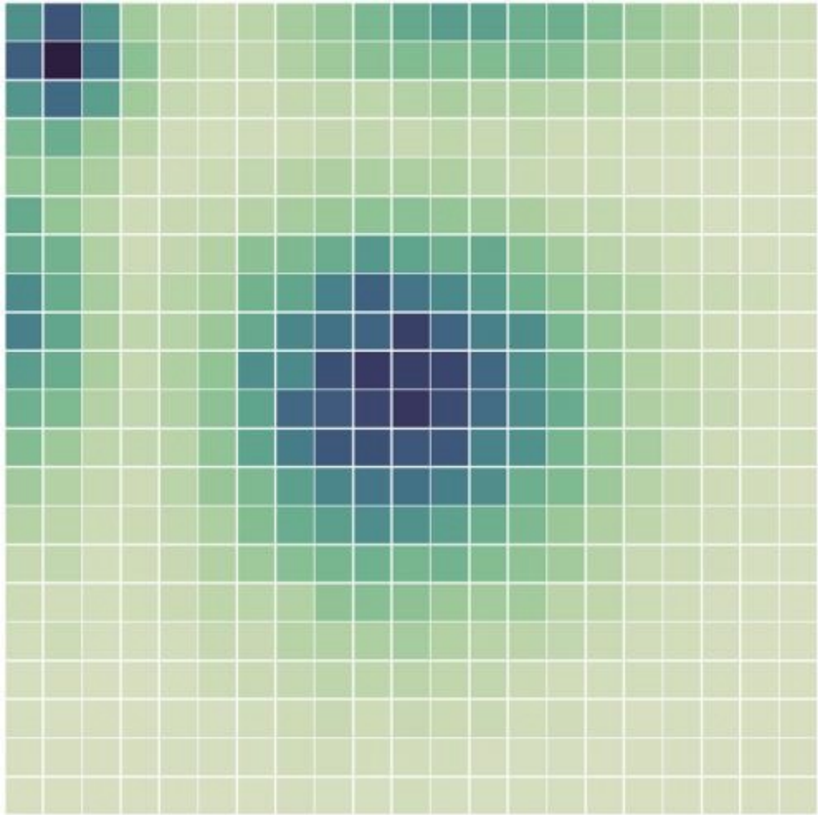
CIFAR-10 classifier

Confidence on i.i.d inputs > Confidence on o.o.d inputs ?

A Simple Baseline for Improving Uncertainty Calibration

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathbf{x}, \mathbf{y})$$

Problem: results in just one prediction per example
No model uncertainty



How do we get uncertainty?

- Probabilistic approach
  - Estimate a full distribution for $p(\boldsymbol{\theta} \mid \mathbf{x}, \mathbf{y})$
- Intuitive approach: Ensembling
  - Obtain multiple good settings for $\boldsymbol{\theta}^*$

Image source: [Ranganath+ 2016]

Ensemble Learning

- A prior distribution often involves the complication of approximate inference.
- *Ensemble learning* offers an alternative strategy to aggregate the predictions over a collection of models.
- Often winner of competitions!
- There are two considerations: the collection of models to ensemble; and the aggregation strategy.

Popular approach is to average predictions of independently trained models, forming a mixture distribution.

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{K} \sum_{k=1}^{K} p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}_k)$$

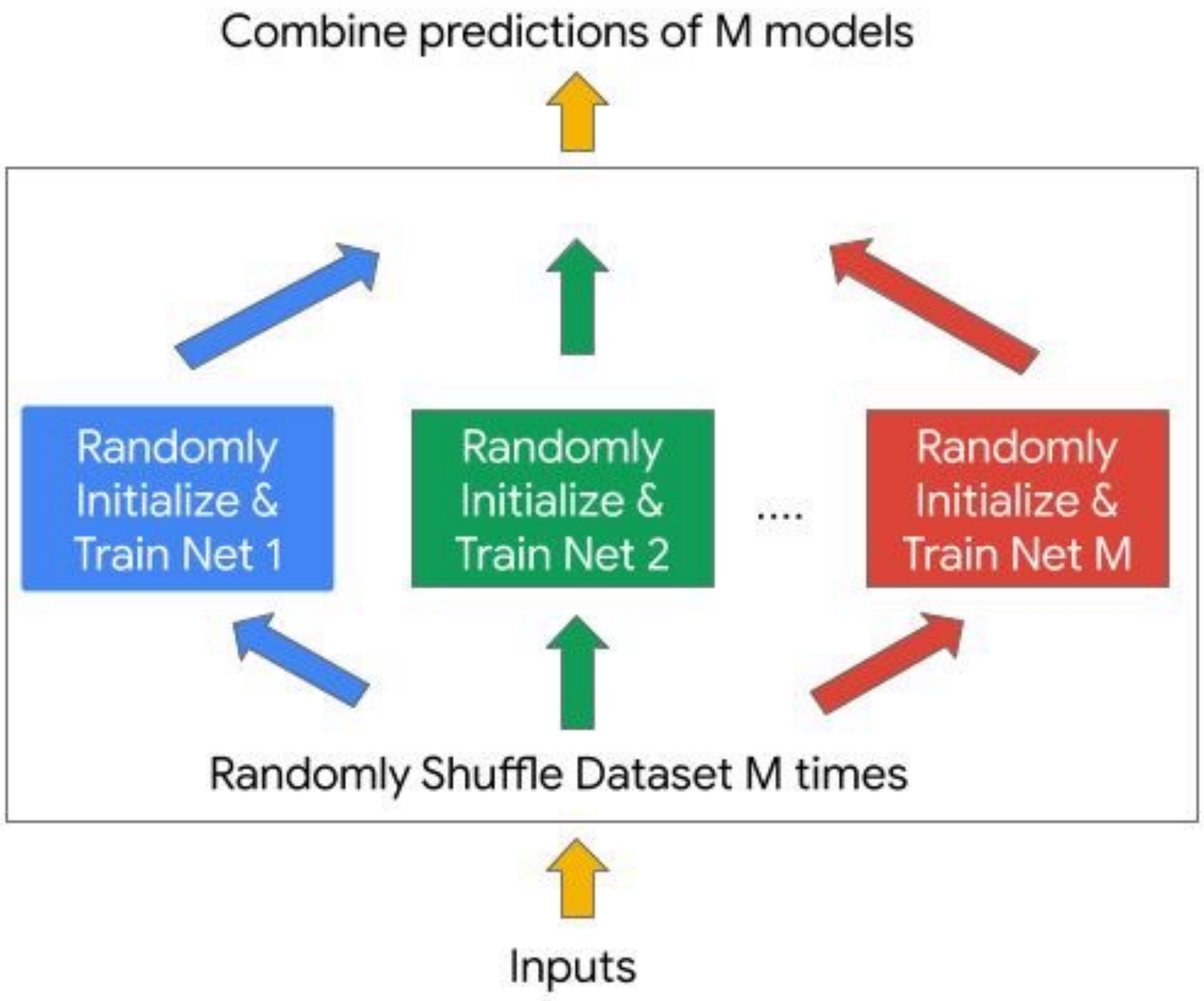
Many approaches exist: bagging, boosting, decision trees, stacking

[Dietterich 2000]

Simple Baseline: Deep Ensembles

Idea: Just re-run standard SGD training but with different random seeds and average the predictions

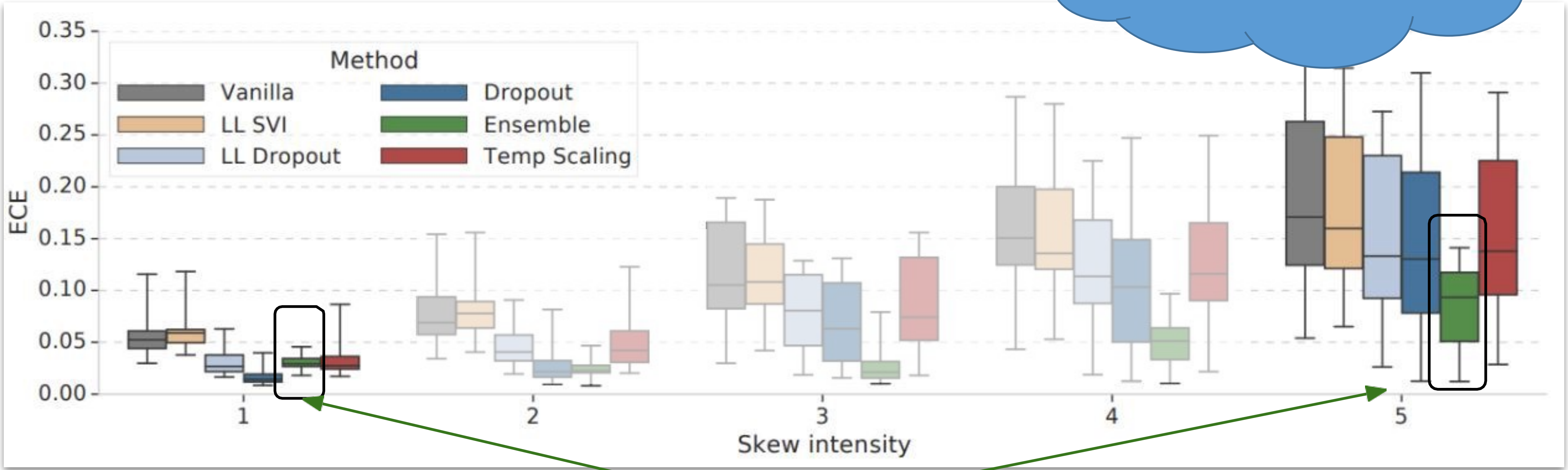
- A well-known trick for getting better accuracy and Kaggle scores
- **Beyond accuracy – it is good for robustness and uncertainty too!!**
- The **mean** of predictions is often more accurate, and the **variance** of those predictions reflects “confidence”
- We rely on the facts that the loss landscape is non-convex and SGD has noise



[Lakshminarayanan+ 2017]

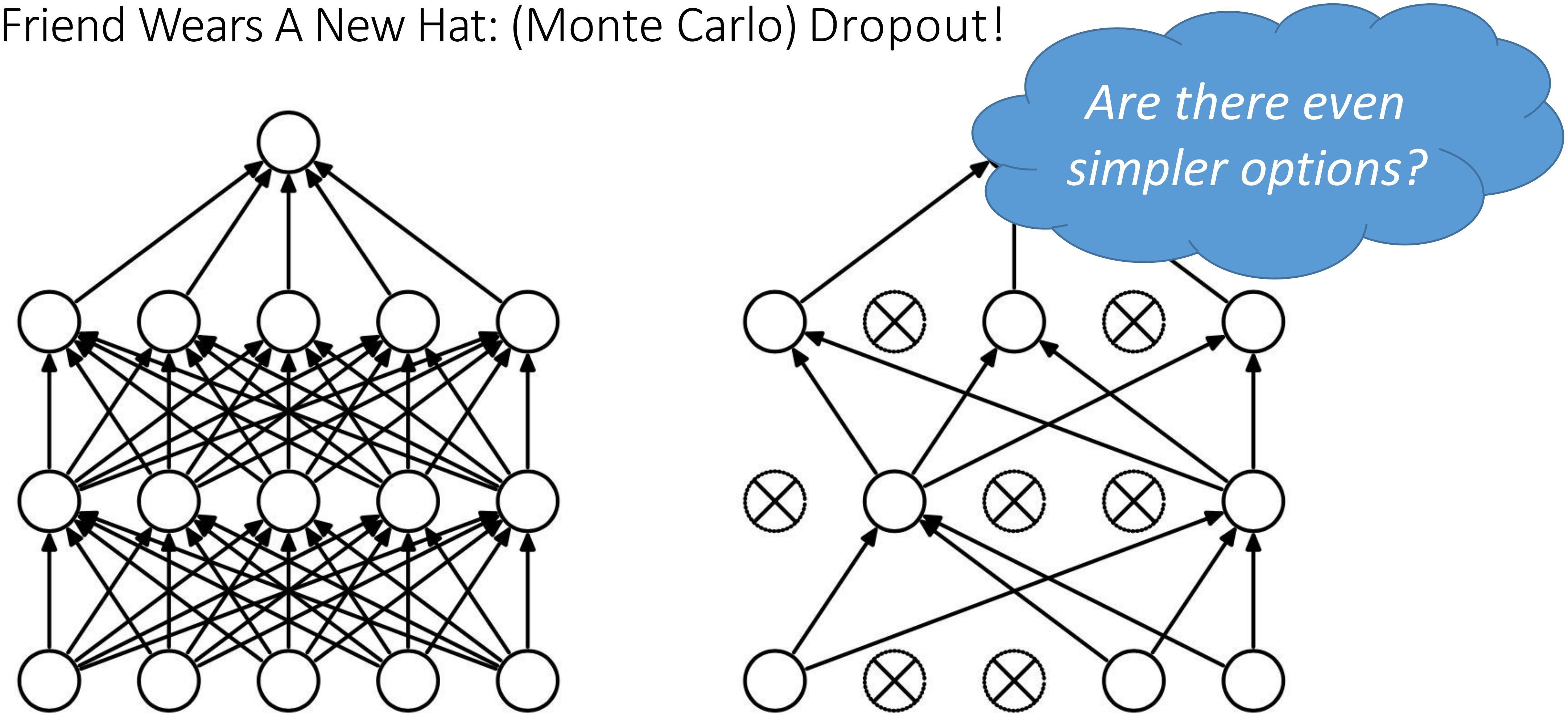
Deep Ensembles work surprisingly well in practice

Are there simpler options?



Deep Ensembles are consistently among the best performing methods, especially under dataset shift

An Old Friend Wears A New Hat: (Monte Carlo) Dropout!



(a) Standard Neural Net

(b) After applying dropout.

[Gal+ 2015]

Table source: Guo+ 2017 “On calibration of modern neural networks”

Dataset	Model	Uncalibrated	Hist. Binning	Isotonic	BBQ	Temp. Scaling	Vector Scaling	Matrix Scaling
Birds	ResNet 50	9.19%	4.34%	5.22%	4.12%	**1.85%**	3.0%	21.13%
Cars								0.5%
CIFAR-10								.0%
CIFAR-10								.72%
CIFAR-10								.72%
CIFAR-10								.41%
CIFAR-10								.16%
CIFAR-10								.49%
CIFAR-10								.09%
CIFAR-10								.44%
CIFAR-10								.87%
CIFAR-10								.24%
ImageNet								-
ImageNet								-
SVHN								.17%
20 News								.1%
Reuters								.58%
SST Binary								.84%
SST Fine Grai								.39%

Table 1. ECE (… n methods.
The number fo

Softmax: $$\sigma_{\mathrm{SM}}(\mathbf{z}_i)^{(k)} = \frac{\exp(z_i^{(k)})}{\sum_{j=1}^{K} \exp(z_i^{(j)})}, \quad \hat{p}_i = \max_k \sigma_{\mathrm{SM}}(\mathbf{z}_i)^{(k)}.$$

Temperature re-scaling (beat them all!): $$\hat{q}_i = \max_k \sigma_{\mathrm{SM}}(\mathbf{z}_i/T)^{(k)}.$$

How do we measure the quality of robustness, practically?

Measure generalization to a *large collection of real-world shifts*. A large collection of tasks encourages *general robustness to shifts* (ex: GLUE for NLP).

- Novel textures in object recognition.
- Covariate shift (e.g. corruptions).
- Different sub-populations (e.g. geographical location).

Different renditions (ImageNet-R)

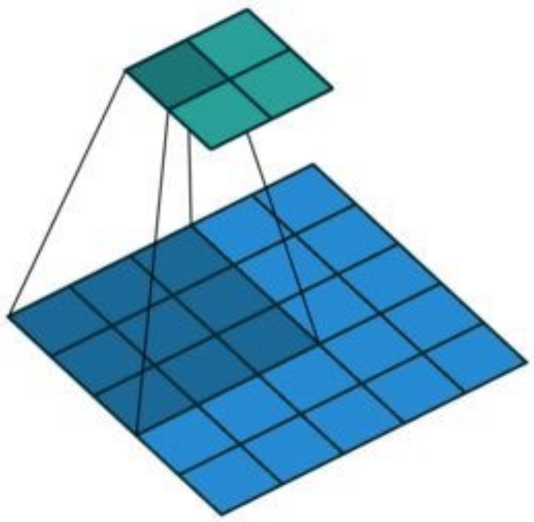
Nearby video frames (ImageNet-Vid-Robust, YTBB-Robust)


Multiple objects and poses (ObjectNet)

Inductive Priors & Knowledge for Robustness

What about inductive biases to assist OOD?

Image source: [Dumoulin & Visin 2016]

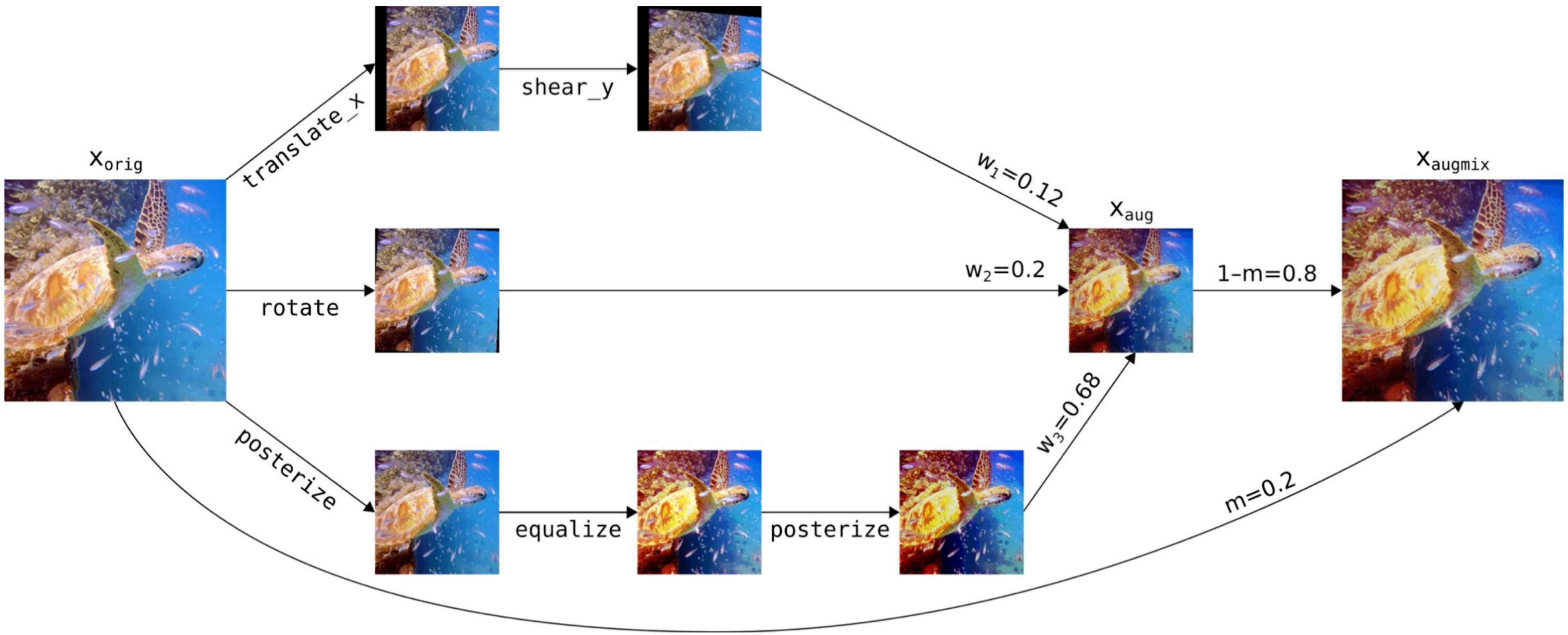


- Hypothesis: “*Representations should be invariant with respect to dataset shift.*”
- **Data augmentation** extends the dataset in order to encourage invariances.
- More examples: **contrastive learning**, **equivariant architectures**.

Data augmentation requires two considerations:

1. Set of **base** augmentation operations. (Ex: color distortions, word substitution)
2. **Combination** strategy (Ex: Sequence of K randomly selected ops.)

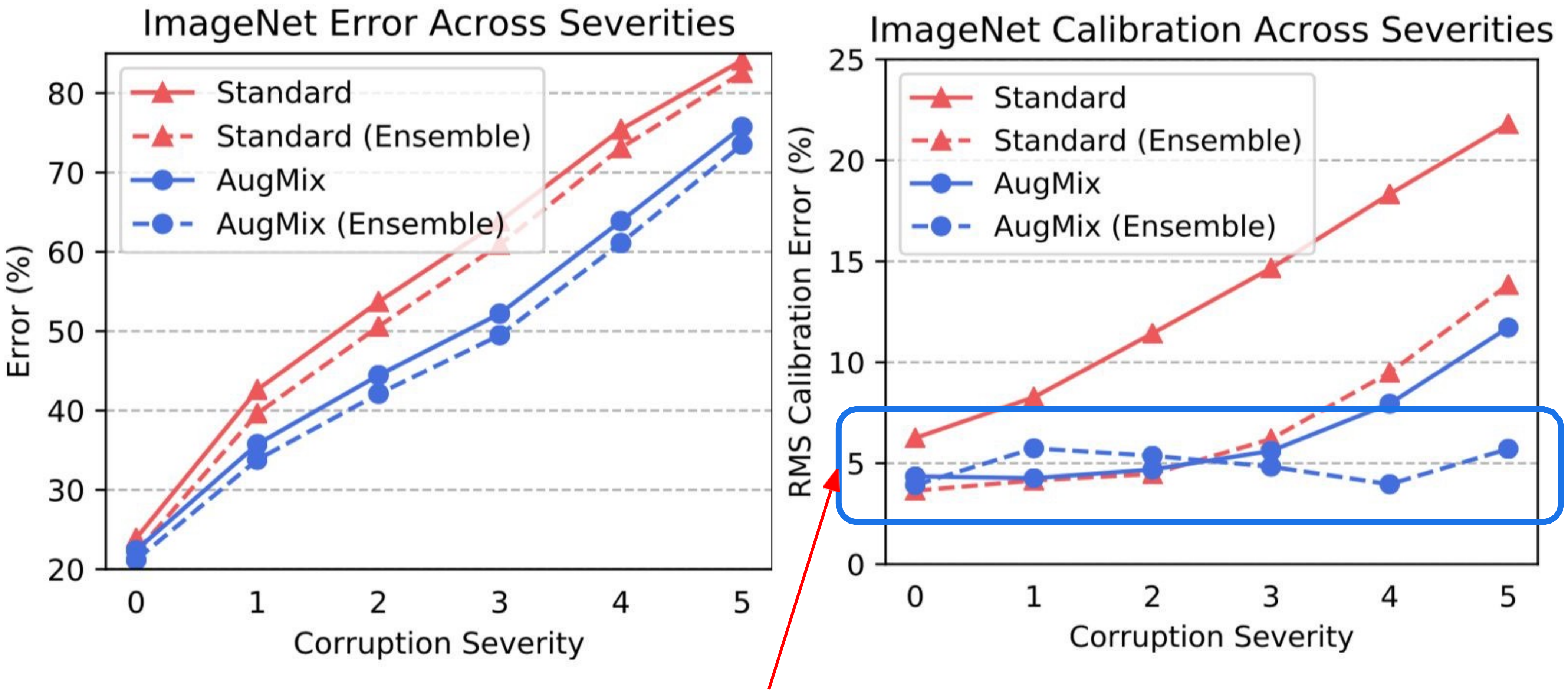
Composing a set of base augmentations



Composing base operations and ‘mixing’ them can improve accuracy and calibration under shift.

[Hendrycks+ 2020]

AugMix improves robustness & calibration under shift



Data augmentation can provide complementary benefits to marginalization.

[Hendrycks+ 2020]

Synthetic Data: Towards Infinite Training Data Variations

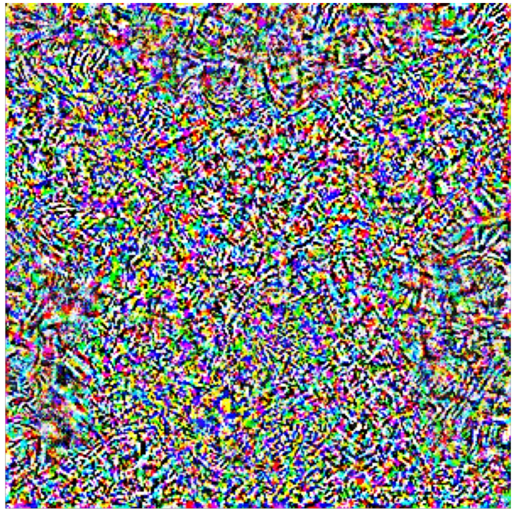
Takeaways

- Uncertainty & robustness are critical problems in AI and machine learning.
- Benchmark models with calibration error and a large collection of OOD shifts.
- Probabilistic ML, ensemble learning, and optimization provide a foundation.
- The best methods: ensemble multiple predictions; imposing priors and inductive biases; and “lower your temperature” when using softmax
- Synthetic data can remarkably help capture more variation
- Many future progress are expected – a key knob to make ML “real”

ML Predictions Are (Mostly) Accurate but Brittle

“pig” (91%) **noise (NOT random)** **“airliner” (99%)**

+ 0.005 x



=

[Szegedy Zaremba Sutskever Bruna Erhan Goodfellow Fergus 2013]
[Biggio Corona Maiorca Nelson Srndic Laskov Giacinto Roli 2013]

But also: [Dalvi Domingos Mausam Sanghai Verma 2004][Lowd Meek 2005]
[Globerson Roweis 2006][Kolcz Teo 2009][Barreno Nelson Rubinstein Joseph Tygar 2010]
[Biggio Fumera Roli 2010][Biggio Fumera Roli 2014][Srndic Laskov 2013]

Three commandments of Secure/Safe ML

I. Thou

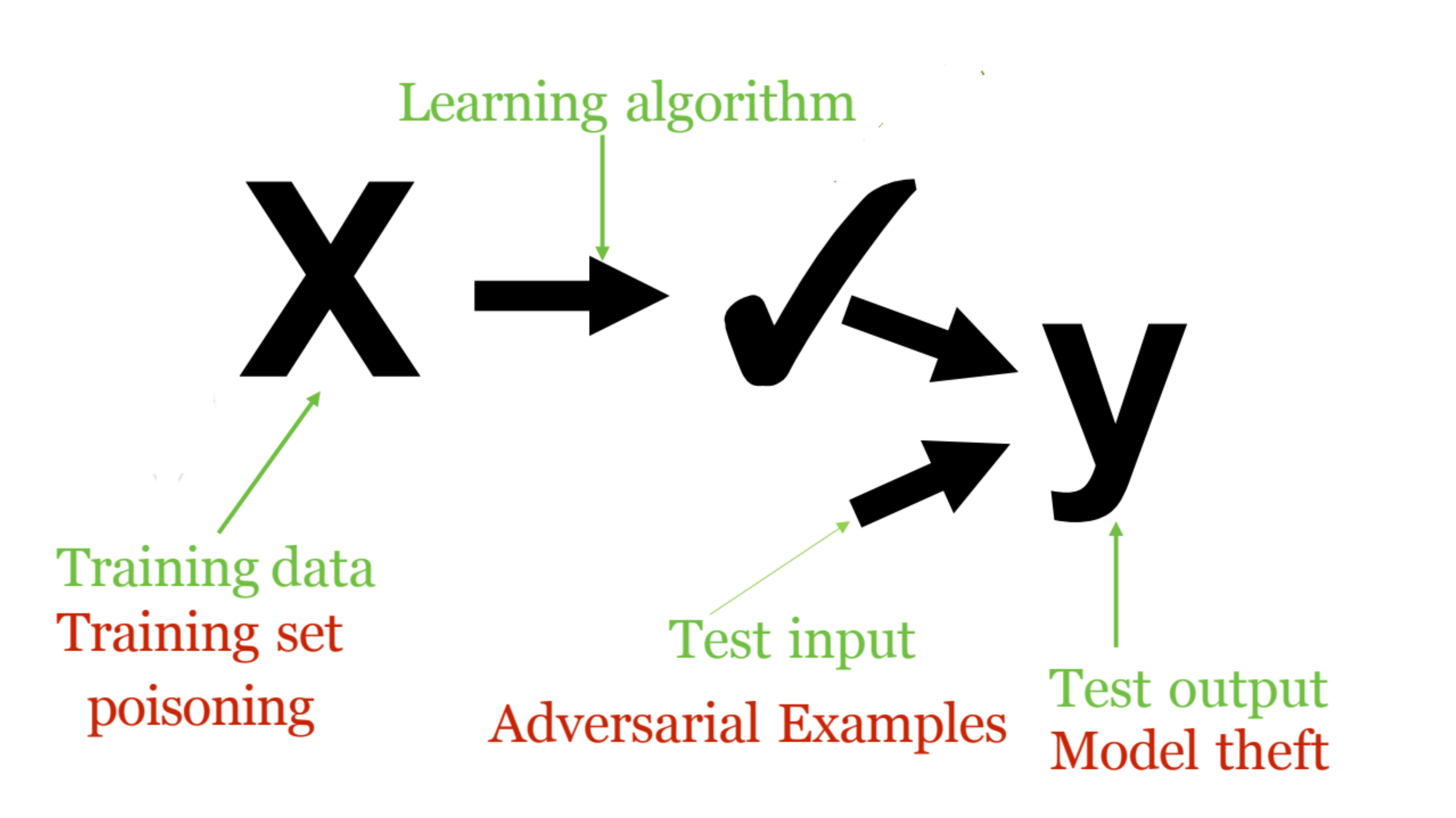
(becau

II. Thou ... erve its outpu

(becau

III. Tho ... r model

(because of adversarial examples)



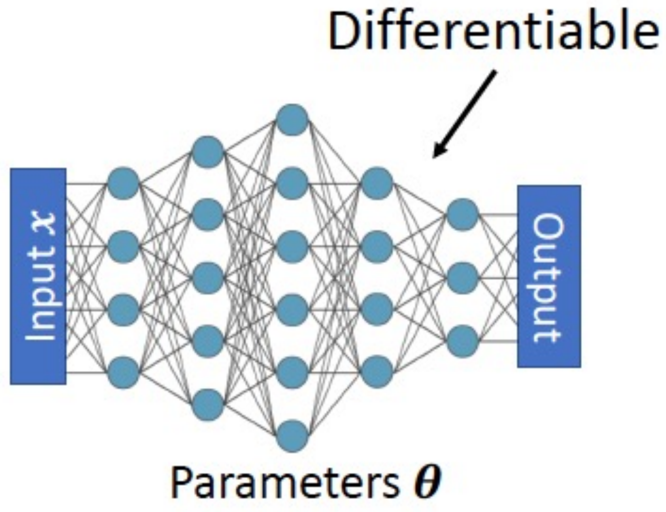
Where Do Adversarial Examples Come From?

To get an adv. example

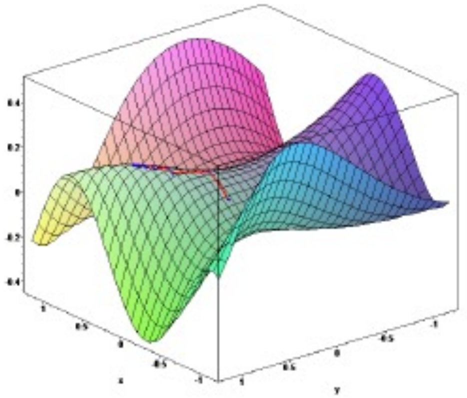
~~**Goal of training:**~~

Model Parameters Input Correct Label

$$min_\theta \; loss(\theta, x, y)$$



Can use gradient descent method to find good θ

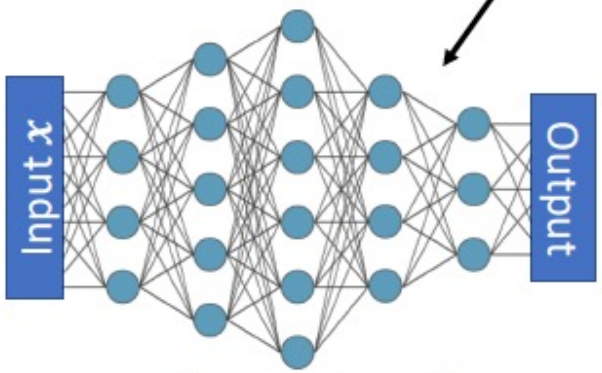


Where Do Adversarial Examples Come From?

To get an adv. example

~~**Goal of training:**~~

Differentiable



Parameters $\boldsymbol{\theta}$

$$max_{\delta}\ loss(\theta, x + \delta, y)$$

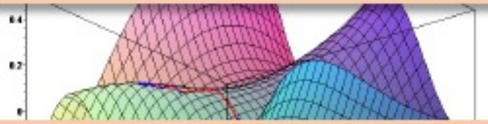
Which δ are allowed?

Examples: δ that is small wrt

- ℓ_p-norm
- Rotation and/or translation
- VGG feature perturbation
- (add the perturbation you need here)

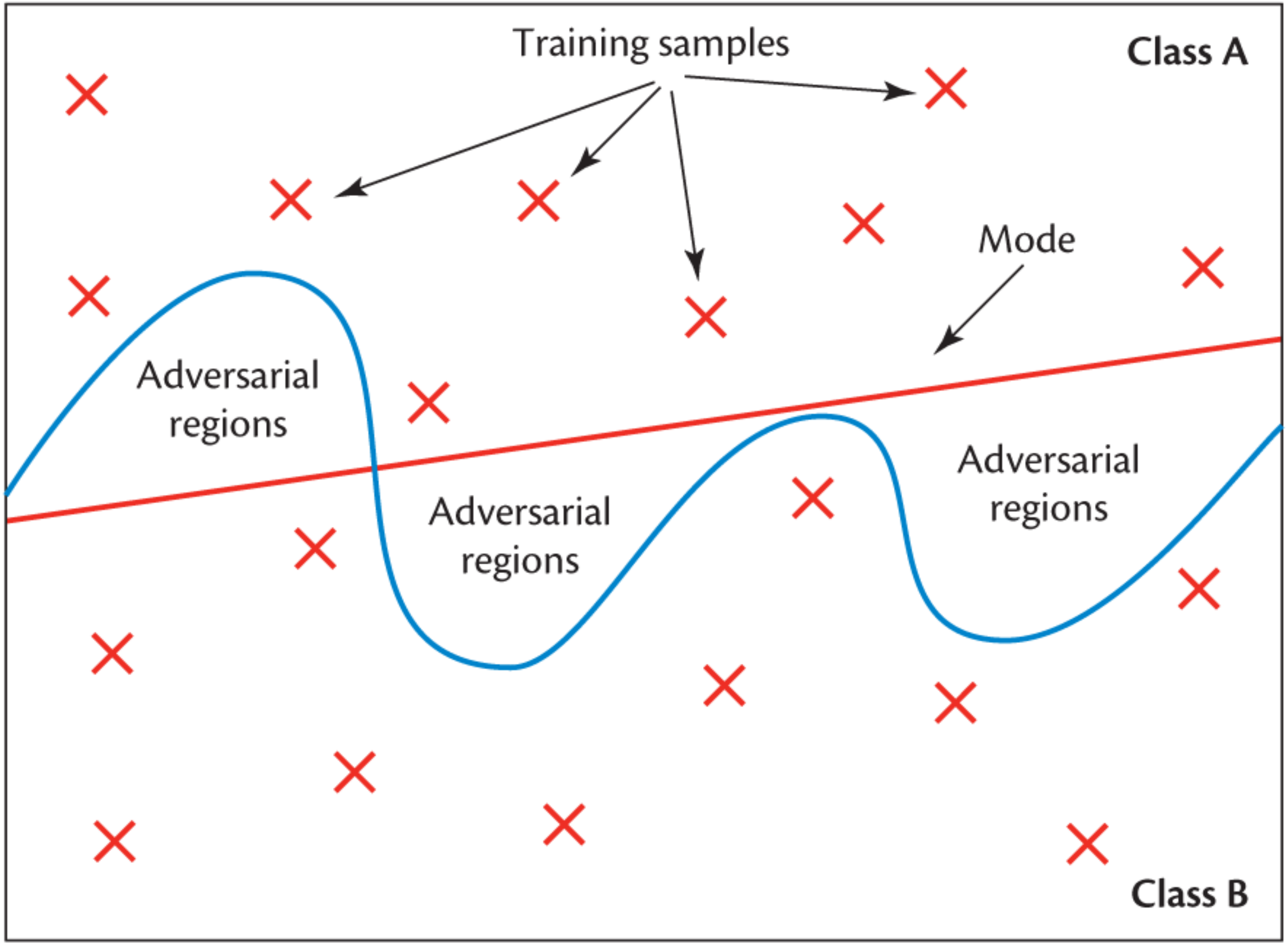
Can use gradient descent

This is an important question
(that we put aside)



Still: We have to confront
(small) ℓ_p-norm perturbations

A Possible By-Product of ML Bias-Variance Trade-Off



Towards ML Models that Are Adv. Robust

[M Makelov Schmidt Tsipras Vladu 2018]

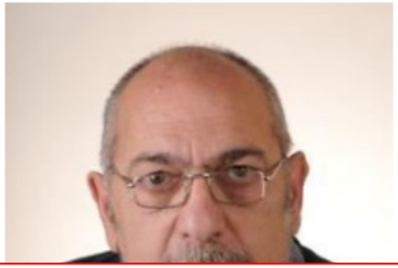
Key observation: Lack of adv. robustness is **NOT** at odds with what we currently want our ML models to achieve

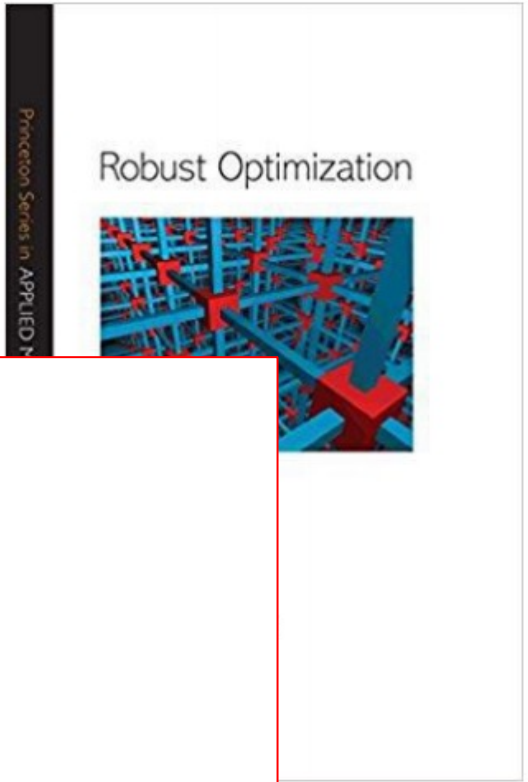
~~Standard~~ generalization: $\mathbb{E}_{(x,y)\sim D}\left[\max_{\boldsymbol{\delta}\in\boldsymbol{\Delta}} loss(\theta, x+\boldsymbol{\delta}, y)\right]$

Adversarially robust

But: Adversarial noise is a "needle in a haystack"

Robust Objectives





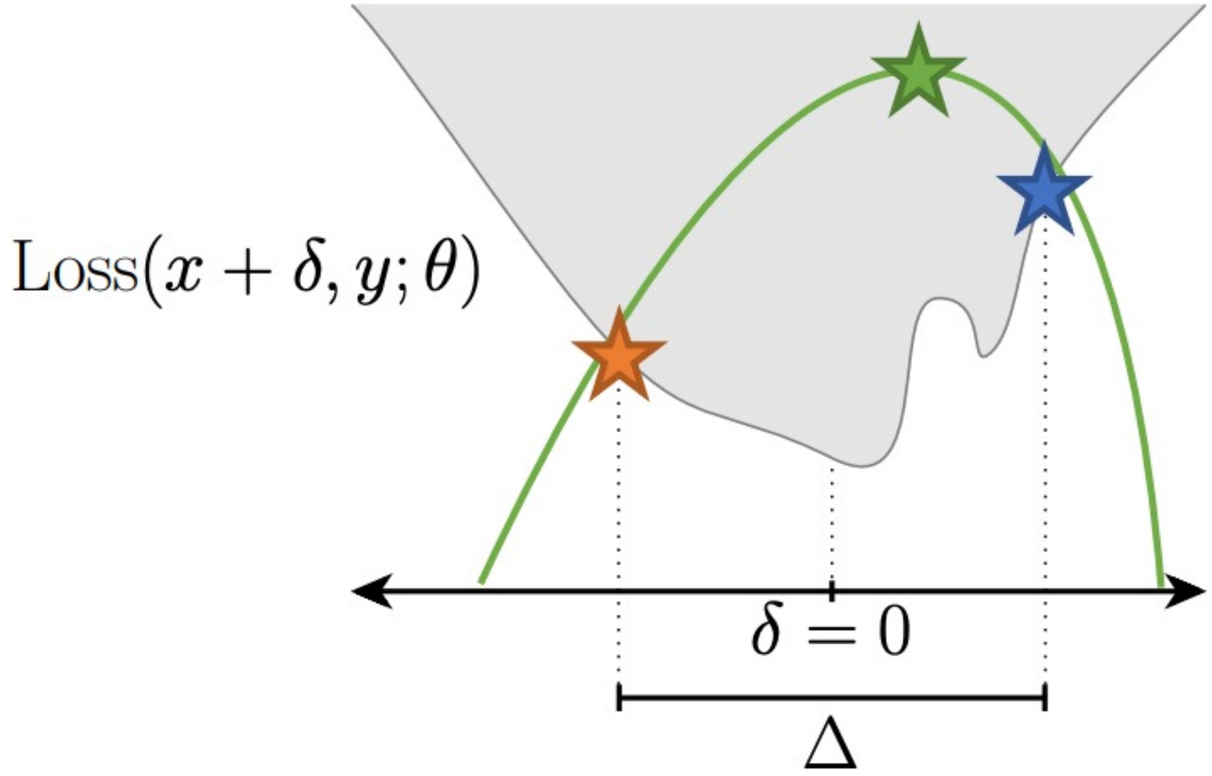
- Use the fol
    - $\min\limits_{w} \quad E$
    - Outer mi
    - Inner ma
- A. Madry, A Deep Learning M
- A. Sinha, H. onal Robustness with Principled Adversarial Training. ICLR 2018

Part II: training a robust classifier

$$\overbrace{\min_{\theta} \sum_{x,y \in S} \underbrace{\max_{\delta \in \Delta} \operatorname{Loss}(x+\delta, y; \theta)}_{\text{Part I}}}^{\text{Part II}}$$

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

The inner maximization problem



How do we solve the optimization?

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

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

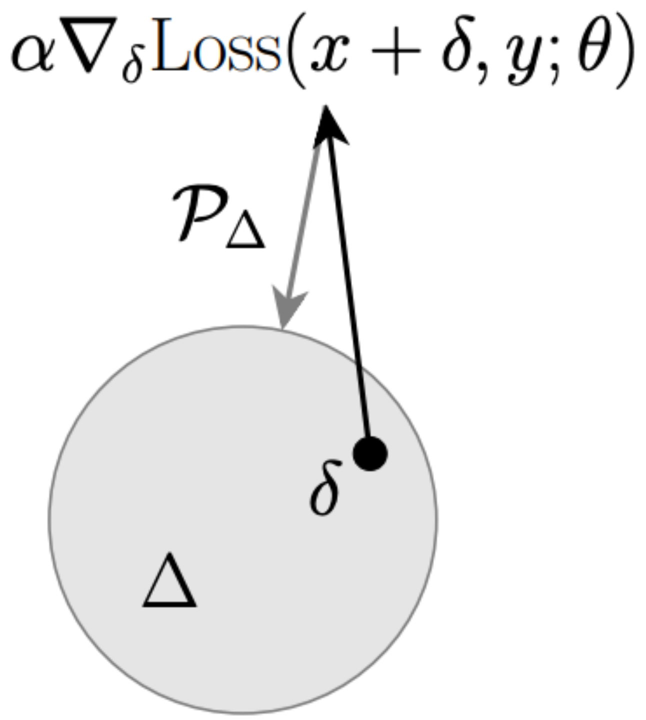
Projected gradient descent

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 Δ

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



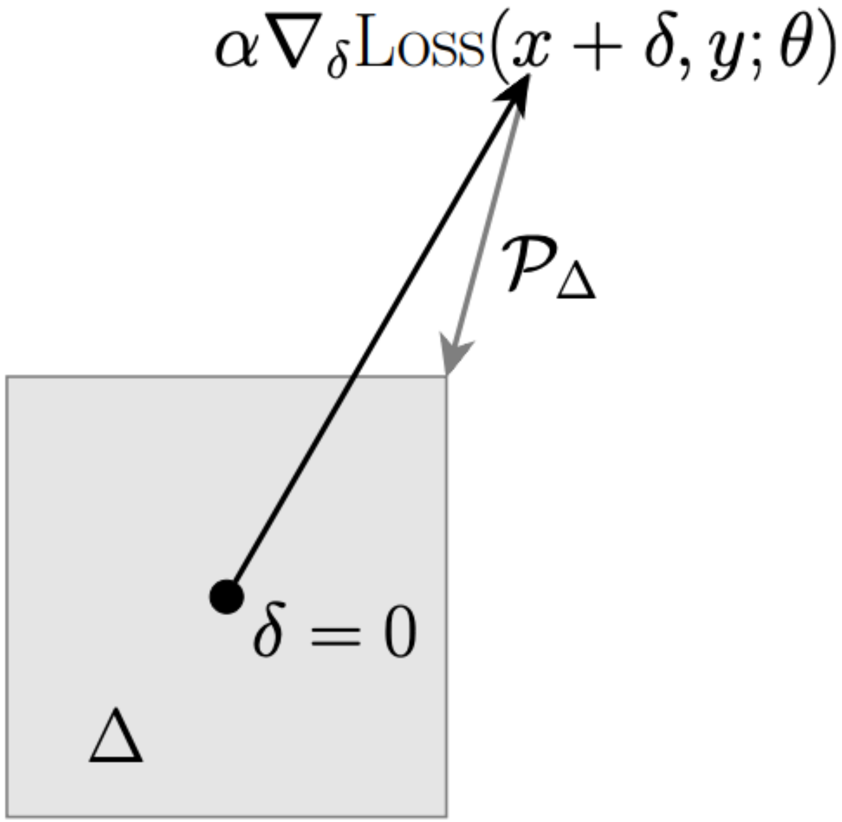
The Fast Gradient Sign Method (FGSM)

To be more concrete, take Δ to be the ℓ_∞ ball, $\Delta = \{\delta : \|\delta\|_\infty \leq \epsilon\}$, so projection takes the form

$$P_\Delta(\delta) = \mathrm{Clip}(\delta, [-\epsilon, \epsilon])$$

As $\alpha \to \infty$, we always reach "corner" of the box, called fast gradient sign method (FGSM)
[Goodfellow et al., 2014]

$$\delta = \epsilon \cdot \mathrm{sign}\big(\nabla_\delta \mathrm{Loss}(x + \delta, y; \theta)\big)$$



Targeted attacks

Also possible to explicitly try to change label to a *particular* class

$$\max_{\delta \in \Delta} \left(\mathrm{Loss}(x+\delta, y; \theta) - \mathrm{Loss}(x+\delta, y_{\mathrm{targ}}; \theta) \right)$$

Consider multi-class cross entropy loss

$$\mathrm{Loss}(x+\delta, y; \theta) = \log \sum_i \exp h_\theta(x+\delta)_i - h_\theta(x)_y$$

Then note that above problem simplifies to

$$\max_{\delta \in \Delta} \left(h_\theta(x)_{y_{\mathrm{targ}}} - h_\theta(x)_y \right)$$

The outer minimization problem

Inner maximization:

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

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

Outer minimization:

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

1. Adversarial training

3. Provably robust training

Danskin's Theorem

A fundamental result in optimization:

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

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

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

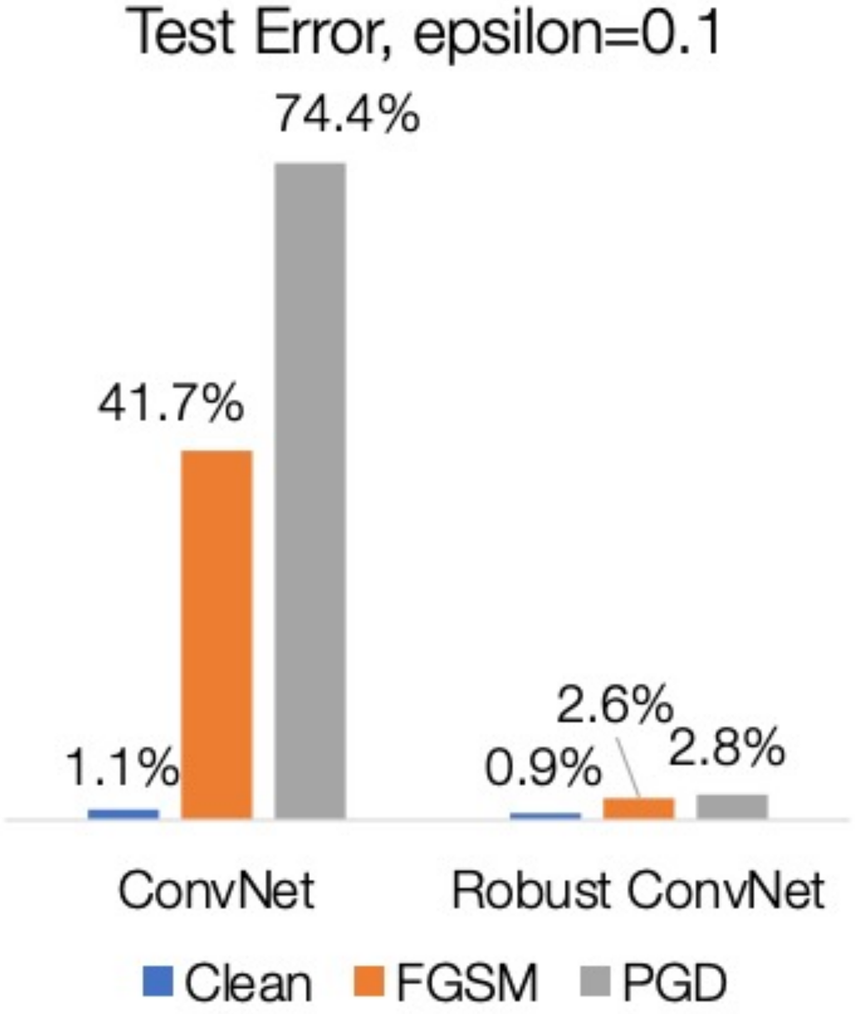
Adversarial training [Goodfellow et al., 2014]

Repeat

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

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

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



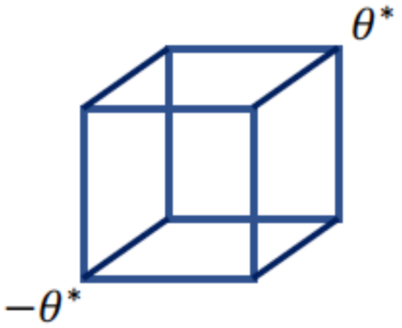
Adv. Robust Generalization Needs More Data

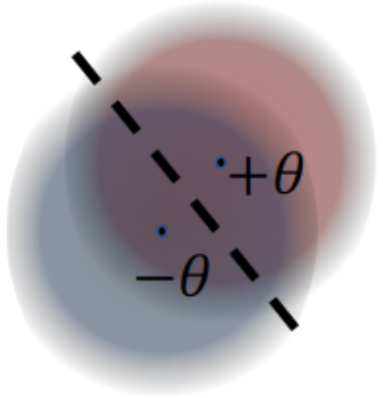
Theorem [Schmidt Santurkar Tsipras Talwar M 2018]:
Sample complexity of adv. robust generalization can be **significantly larger** than that of "standard" generalization

Specifically: There exists a **d**-dimensional distribution **D** s.t.:

→ A **single** sample is enough to get an **accurate** classifier (P[correct] > 0.99)

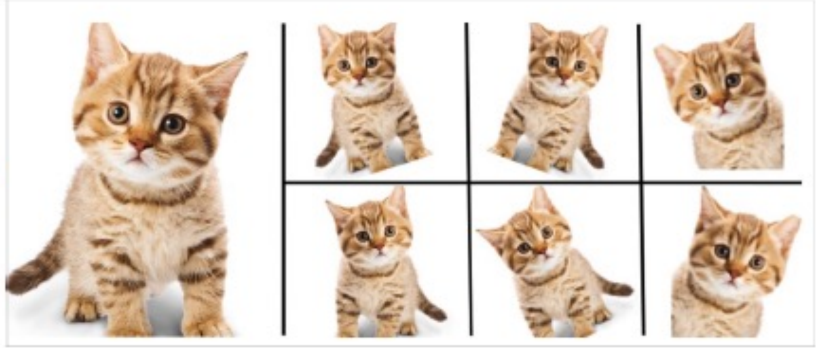
→ **But**: Need $\Omega(\sqrt{d})$ samples for better-than-chance **robust** classifier





Does Being Robust Help "Standard" Generalization?

Data augmentation: An effective technique to improve "standard" generalization



Adversarial training

=

An "ultimate" version of data augmentation?

(since we train on the "most confusing" version of the training set)

Does adversarial training always improve "standard" generalization?

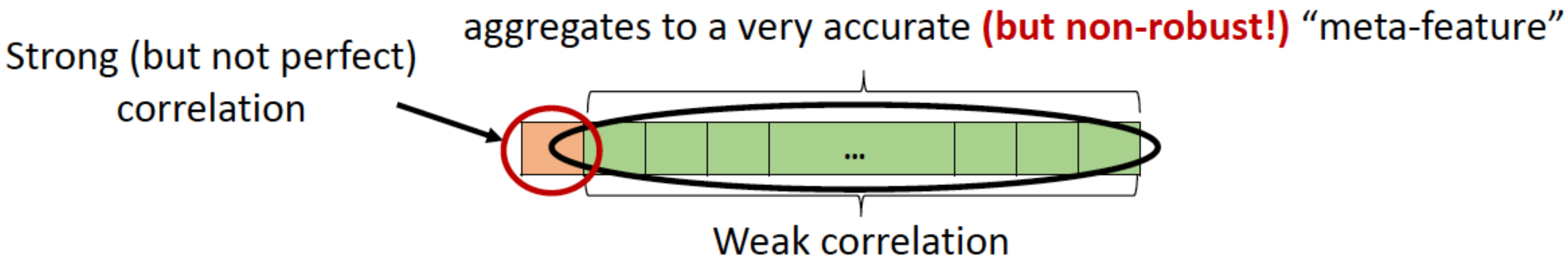
Does Being Robust Help "Standard" Generalization?

Theorem [Tsipras Santurkar Engstrom Turner M 2018]:
No "free lunch": can exist a trade-off between accuracy and robustness

Basic intuition:

→ In standard training, **all correlation is good correlation**

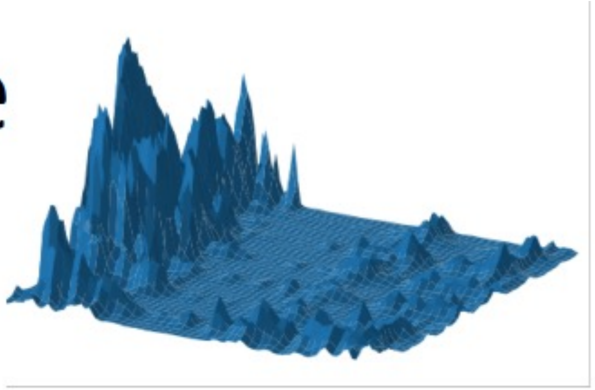
→ If we want robustness, **must avoid** weakly correlated features



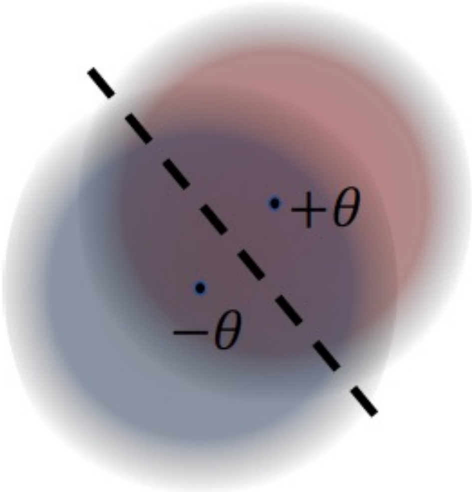
Standard training: use all of features, maximize accuracy

Adversarial training: use only single robust feature (**at the expense of accuracy**)

Adversarial Robustness is Not Free

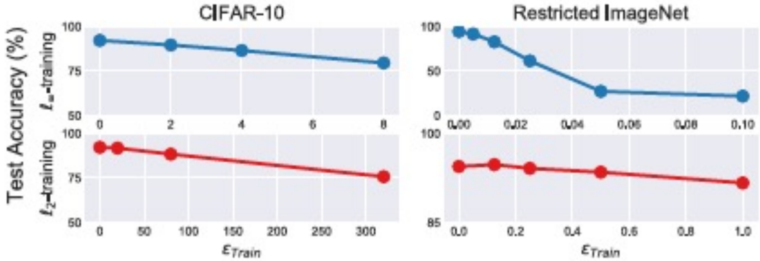


→ Optimization during training more difficult and models need to be larger



→ More training data might be required

[Schmidt Santurkar Tsipras Talwar M 2018]



→ Might need to lose on "standard" measures of performance

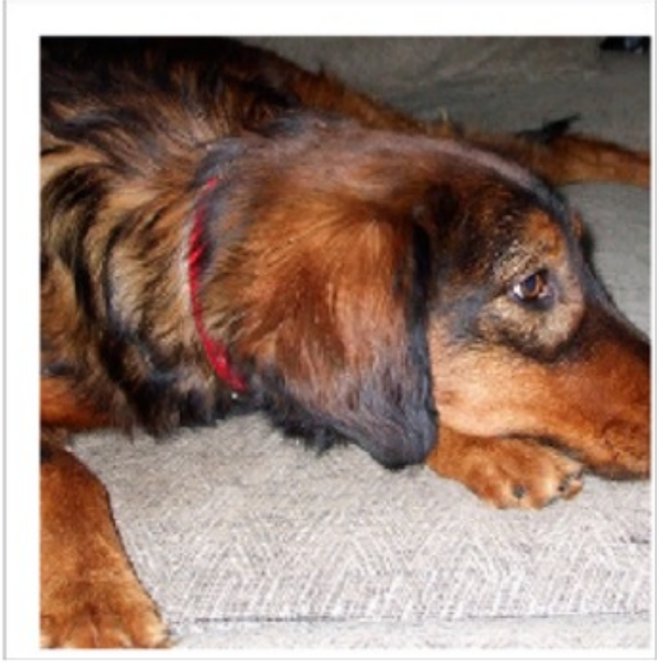
[Tsipras Santurkar Engstrom Turner M 2018] (Also see: [Bubeck Price Razenshteyn 2018])

-> Other Difficulties such as Robust Overfitting (ICML 2020) etc.

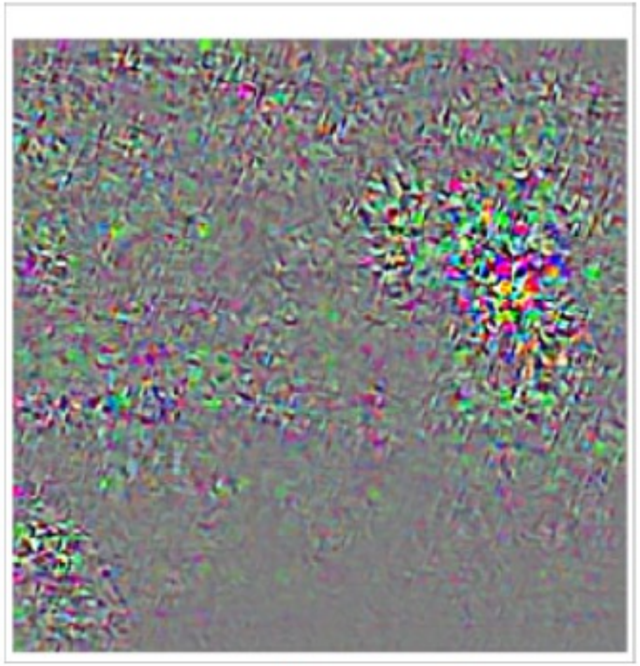
But There Are (Unexpected?) Benefits Too

[Tsipras Santurkar Engstrom Turner M 2018]

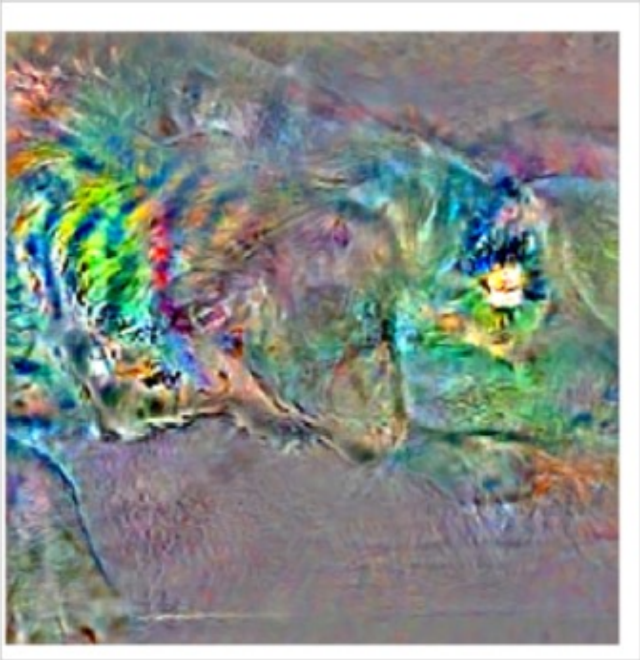
Models become more **semantically meaningful**



Input

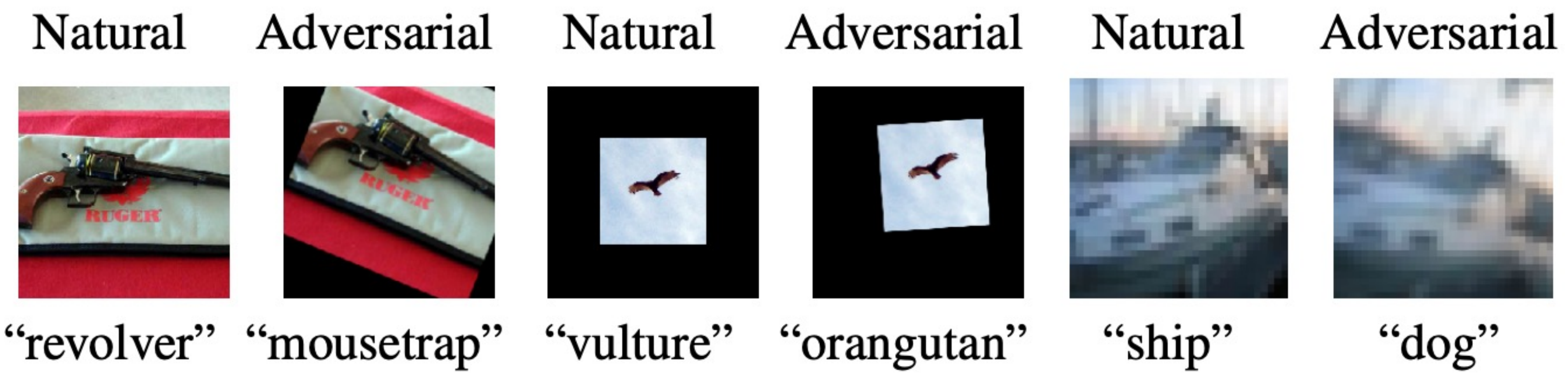


Gradient of
standard model



Gradient of
adv. robust model

Adversarial Examples Beyond Pixel Perturbations ...



A ROTATION AND A TRANSLATION SUFFICE: FOOLING CNNS WITH SIMPLE TRANSFORMATIONS

Logan Engstrom, Ludwig Schmidt, Dimitris Tsipras, Aleksander Mądry
Massachusetts Institute of Technology
`{engstrom,ludwigs,tsipras,madry}@mit.edu`

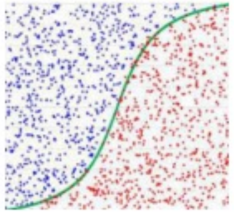
$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} \delta u \\ \delta v \end{bmatrix}$$

By defining the spatial transformation for some x as $T(x; \delta u, \delta v, \theta)$, we construct an adversarial perturbation for x by solving the problem

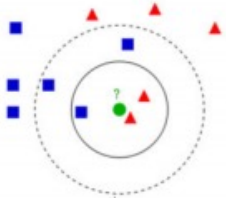
$$\max_{\delta u, \delta v, \theta} \mathcal{L}(x', y), \quad \text{for } x' = T(x; \delta u, \delta v, \theta)\ , \tag{1}$$

Adversarial examples…

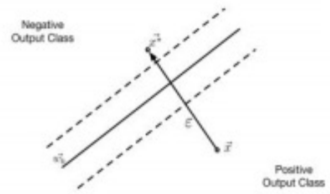
… *beyond deep learning*



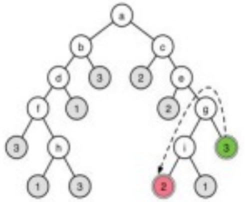
Logistic Regression



Nearest Neighbors

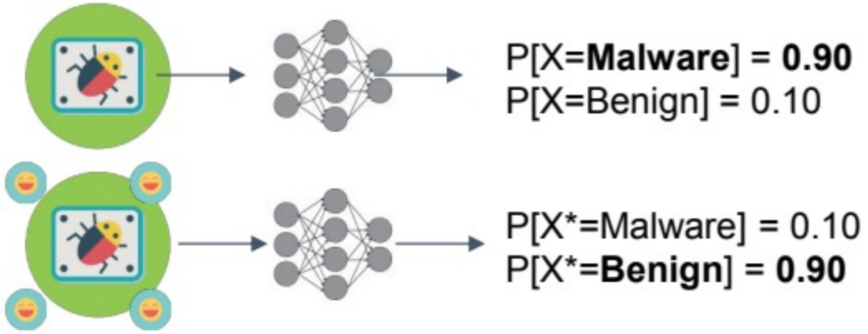


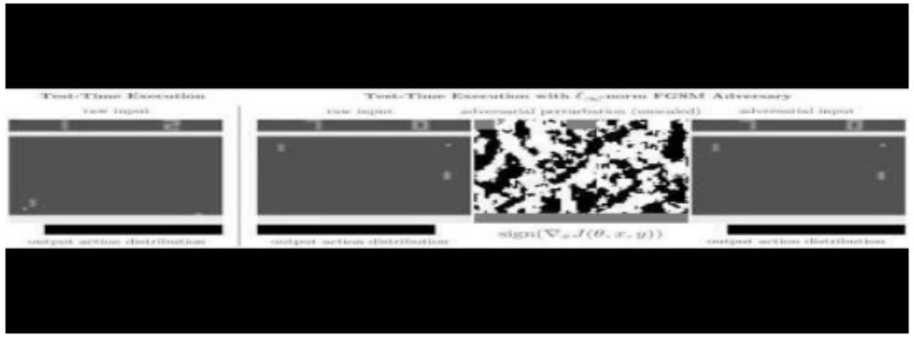
Support Vector Machines



Decision Trees

… *beyond computer vision*





Towards (Adversarially) Robust ML

→ **Algorithms:** Faster robust training + verification [Xiao Tjeng Shafiullah **M** 2018], smaller models, new architectures?

→ **Theory:** (Better) adv. robust generalization bounds, new regularization techniques

→ **Data:** New datasets and more comprehensive set of perturbations

Major need: Embracing more of a worst-case mindset

→ **Adaptive** evaluation methodology + scaling up verification

Further Read: *https://adversarial-ml-tutorial.org/*

cleverhans

RobustML

(robust-ml.org)

The University of Texas at Austin
Electrical and Computer Engineering
Cockrell School of Engineering