Adversarial AI

Doug Finkbeiner, Harvard Center for Astrophysics

AI is in everyday use

- Voice to text
  - smartphones, …
- Traffic prediction
  - Google maps, waze, …
- Spam filtering
- Plagiarism checkers
  - Turnitin
- Check deposits
- Fraud detection
- Robo-Readers
  - The Graduate Record Exam (GRE), the primary test used for graduate school, grades essays using one human reader and one robo-reader, called *e-Rater*.

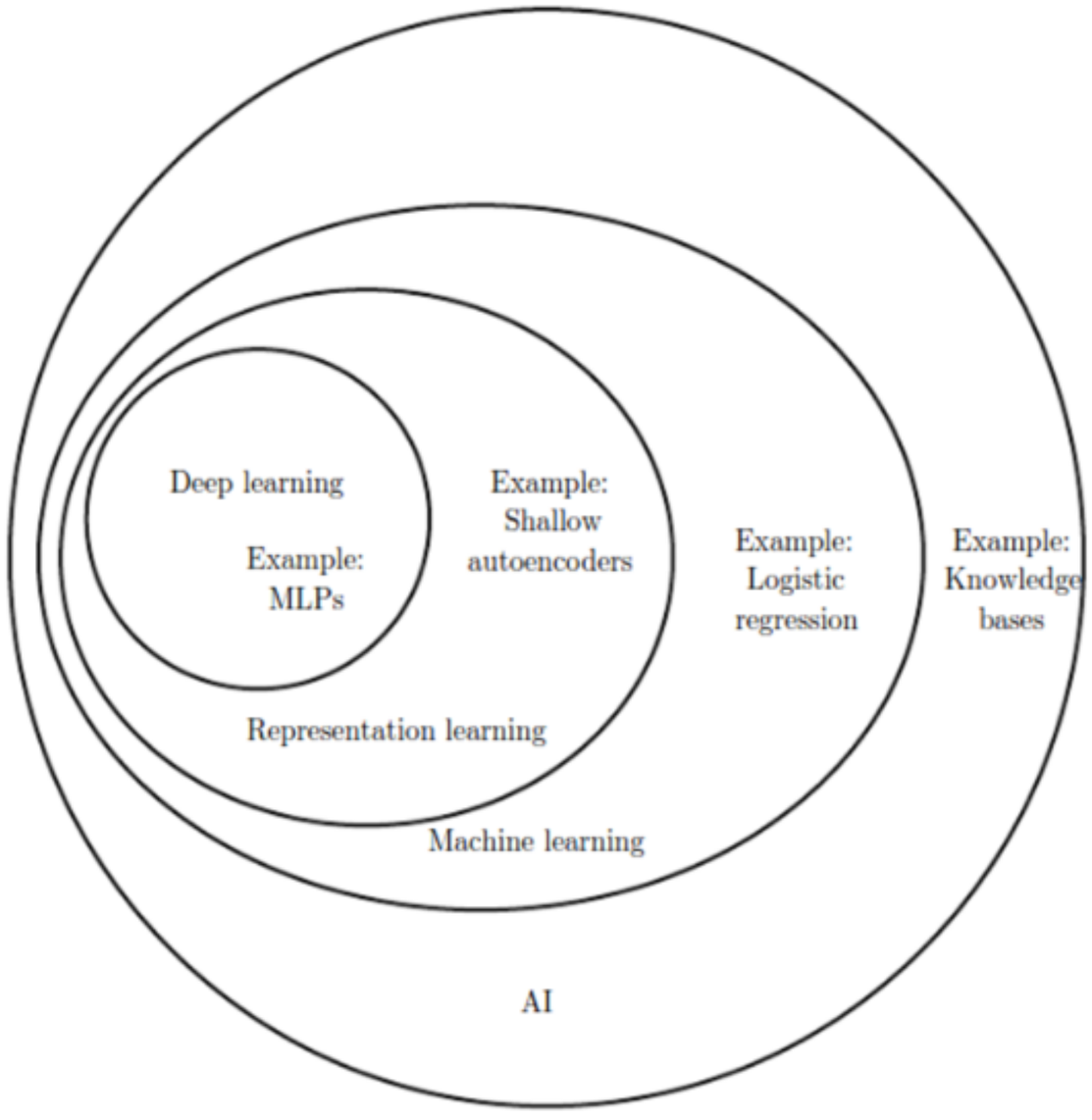
and the FDA just approved a neural-net based diagnostic for diabetic retinopathy. (but see Finlayson+ 1804.05296 for concerns)

…and is increasingly used in physics and astronomy

Classification of spectra, time series, etc.
Classification of particle cascades at LHC
(e.g. Fraser & Schwartz, arXiv:1803:08066,
Andreassen+ arXiv:1804.09720)

It would be good to know how these ML models work.

Deep learning



We are going to talk about deep neural nets (DNNs) used for image classification. This is a rich topic that provides good examples.

Goodfellow+ deeplearningbook.org

The barrier to entry is low…

Neural net training

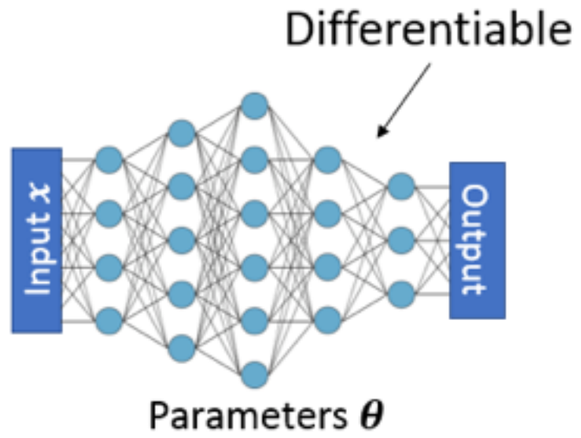
Define a loss function J

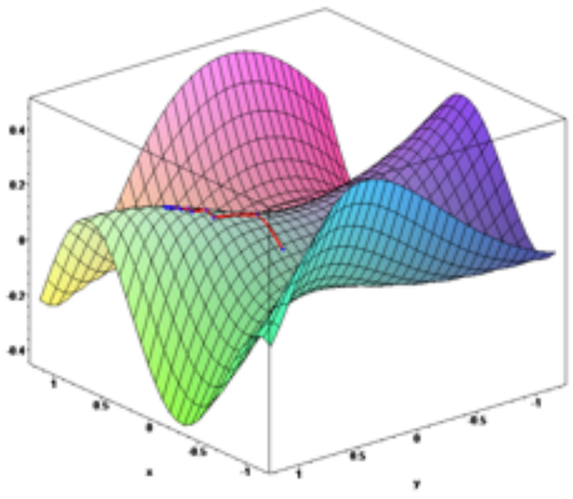
$J(\theta, x, y)$

Parameters

Input:
e.g. images

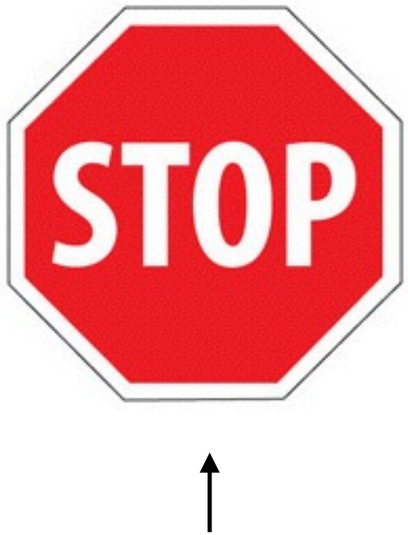
Output:
e.g. labels



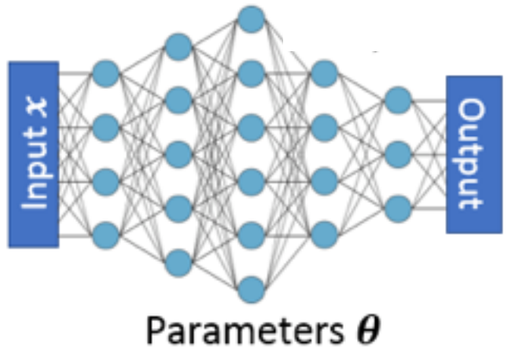


Goal of training is to find theta
to minimize the loss:
$\min_\theta J(\theta, x, y)$

Neural net training



Input image



"Stop Sign"

Output

- The weight vectors of the algorithm are not "logical" or high-level
- ML models mimic correlations, do not reveal causation.
- The function is very high dimensional and has complicated gradients

Models are huge, need lots of training data

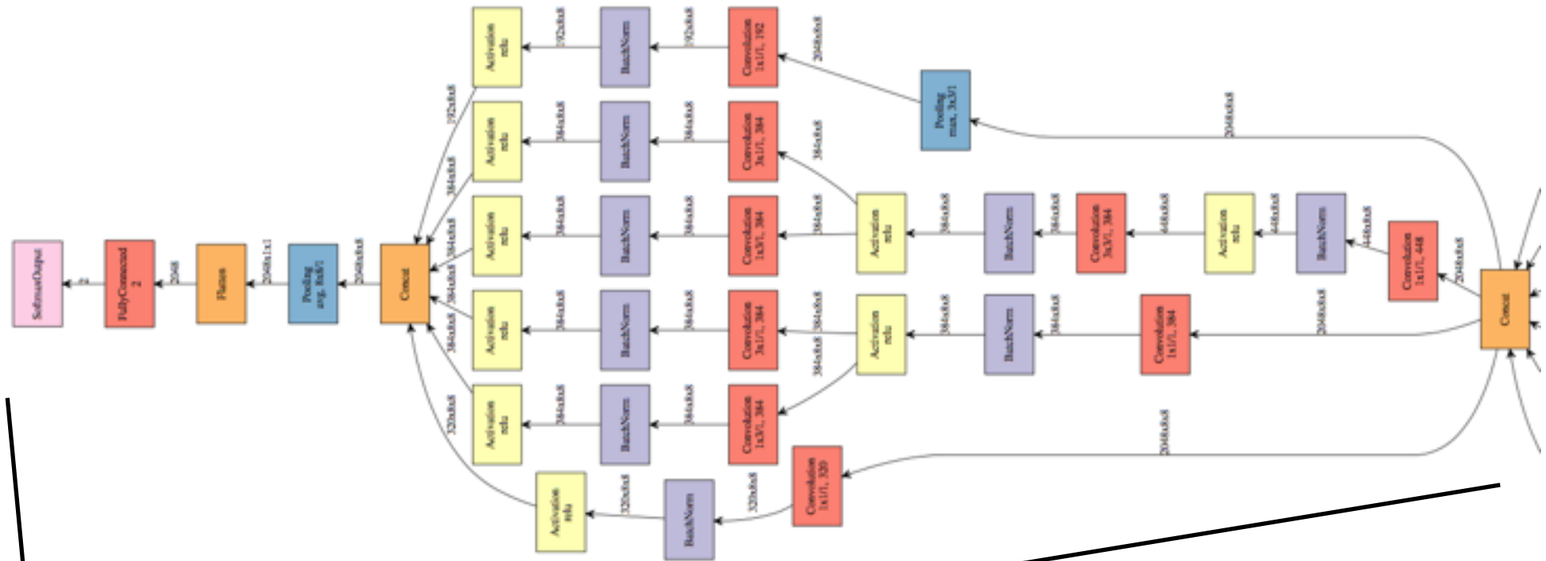
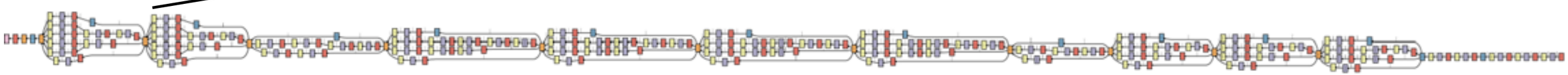


Image classifiers

Model Name	#parameters
GoogleNet	5M
Inception v3	24M
Inception v4	35M
ResNet101	50M
AlexNet	60M
VGG16	138M

The venerable old Inception v3



Models are huge, need lots of training data

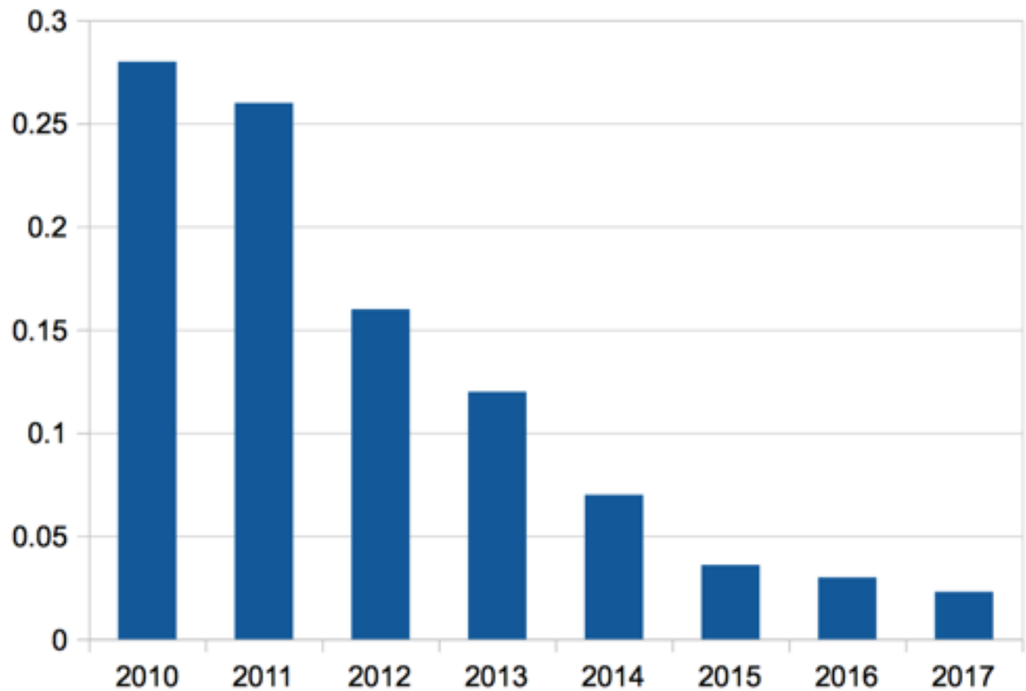
Selected public data sets used in ML

Data set	#classes	#samples	size
MNIST	10	60,000	28x28
CIFAR-10	10	60,000	32x32
GTRSB	40	50,000	~ 256x256
ImageNet (trim)	1,000	1,000,000	~300x300
ImageNet (full)	20,000	14,000,000	var

…and many others. The wide availability of
large, labeled training data has facilitated rapid growth in ML.

Image classification

Error rates (top-5) for the ImageNet Large-Scale Visual Recognition Competition (ILSVRC) are so low that it was discontinued in 2017.



Low-power classification

Low-Power Image Recognition Challenge (LPIRC 2018)

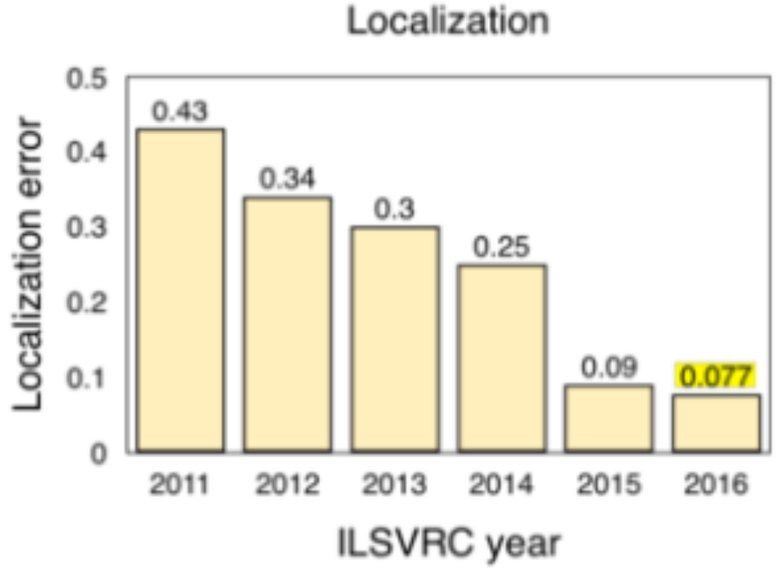
LPIRC 2018 was held June 18 in Salt Lake City, Utah

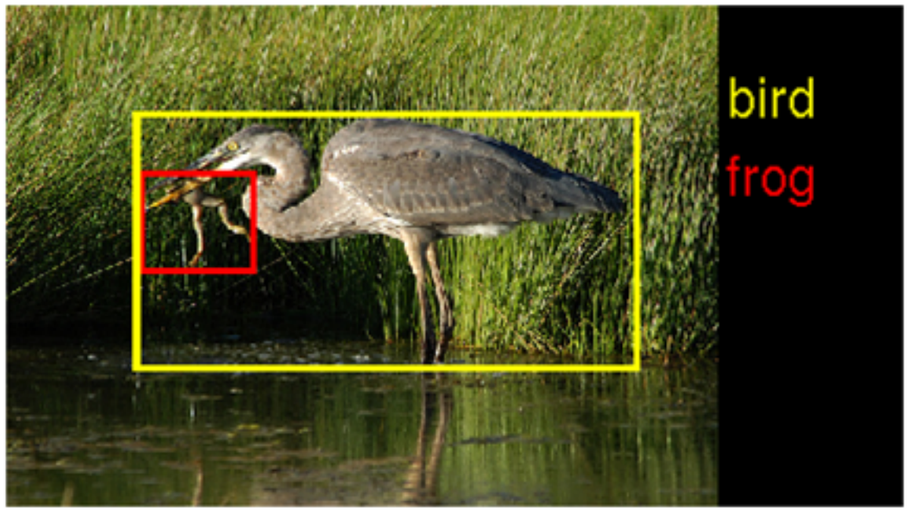
Co-located with International Conference on Computer Vision and Pattern Recognition (CVPR 2018).

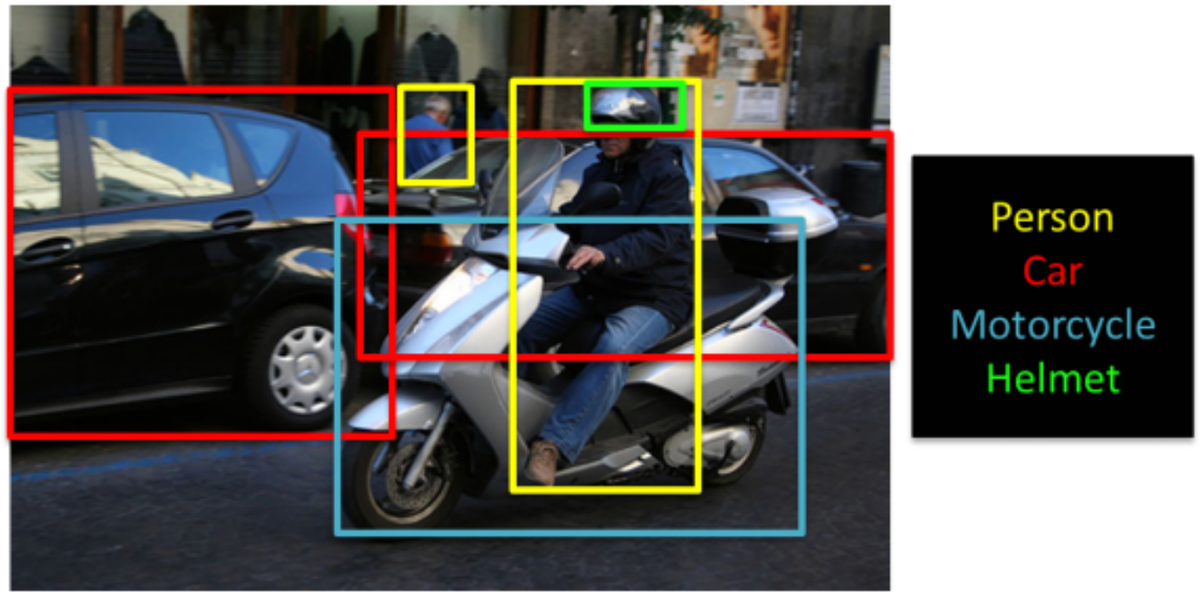
Download IEEE LPIRC 2018 Press Release (PDF, 551 KB)

You have to know where to look

Classification success often depends on detecting objects within a larger image and finding a good bounding box.







Translation invariance

The initial layers of image classifiers are convolutions, but the result is not invariant to translation (shifts in x and y position).

CNNs are not translationally invariant.

In some cases they happen to be to some extent, over some range. (See Kauderer-Abrams 1801.01450) In other cases, the class of an object my change with a single-pixel shift.

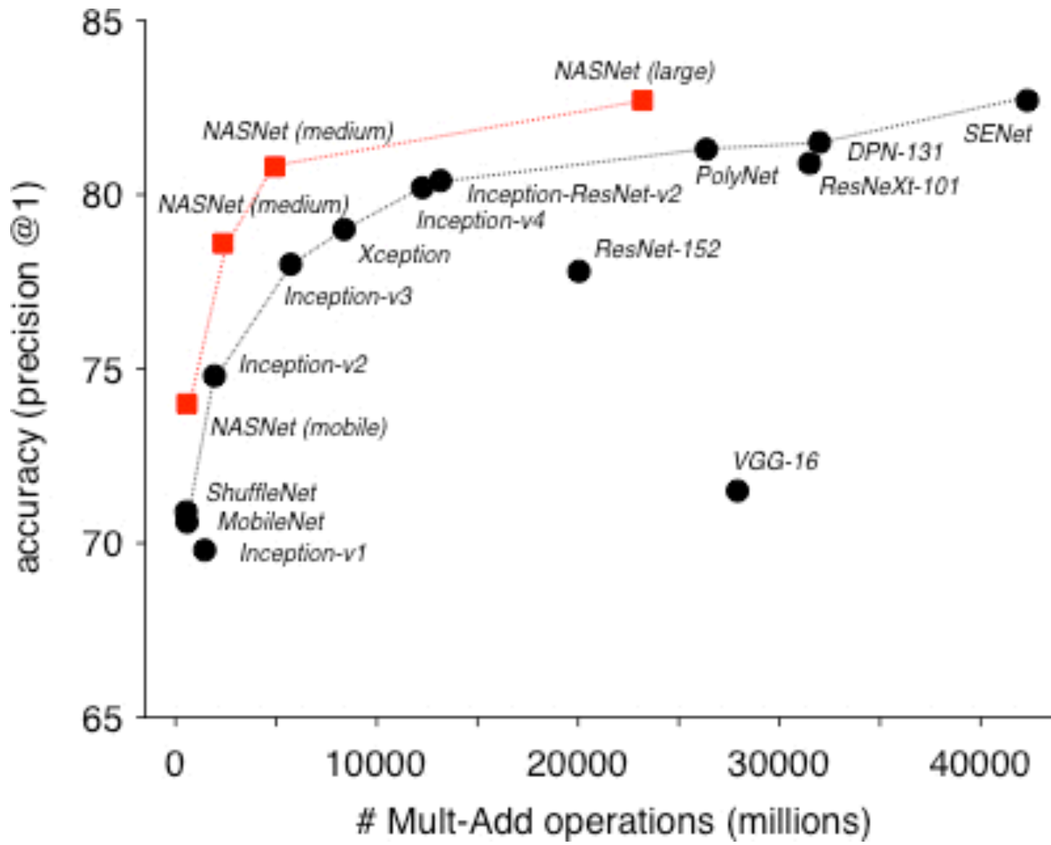
For finding smaller objects in larger images, choice of bounding box is key. For judicious choice of bounding box, lack of translational invariance is irrelevant.

But this makes correct classification *critically dependent* on the bounding box algorithm.

Computer-designed DNNs

No reason to believe that humans can design better ML models than computers can.

See Google AutoML project and NASNet



Despite successes, ML models are fragile

- ML models are (at best) as good as their training data (and training data is seldom good enough)
- ML models are very high dimensional and opaque (what do they *really* learn?)
- Model design and optimization is an art.
- Overfitting can take many forms.

Why so fragile?

One cause of fragility is overfitting. Neural nets are sophisticated fitting functions. As with all fitting functions, one must balance the need to characterize the complexity of the function with the failures that occur with too much freedom
("freedom" = number of parameters, regularization, …)

Too little: *Miss important features of the model.*

Too much: *Fit the training data, but fail to generalize.*

If performance on the training set is *much* better than performance on a validation set, that indicates overfitting.

Classifiers may be overfit

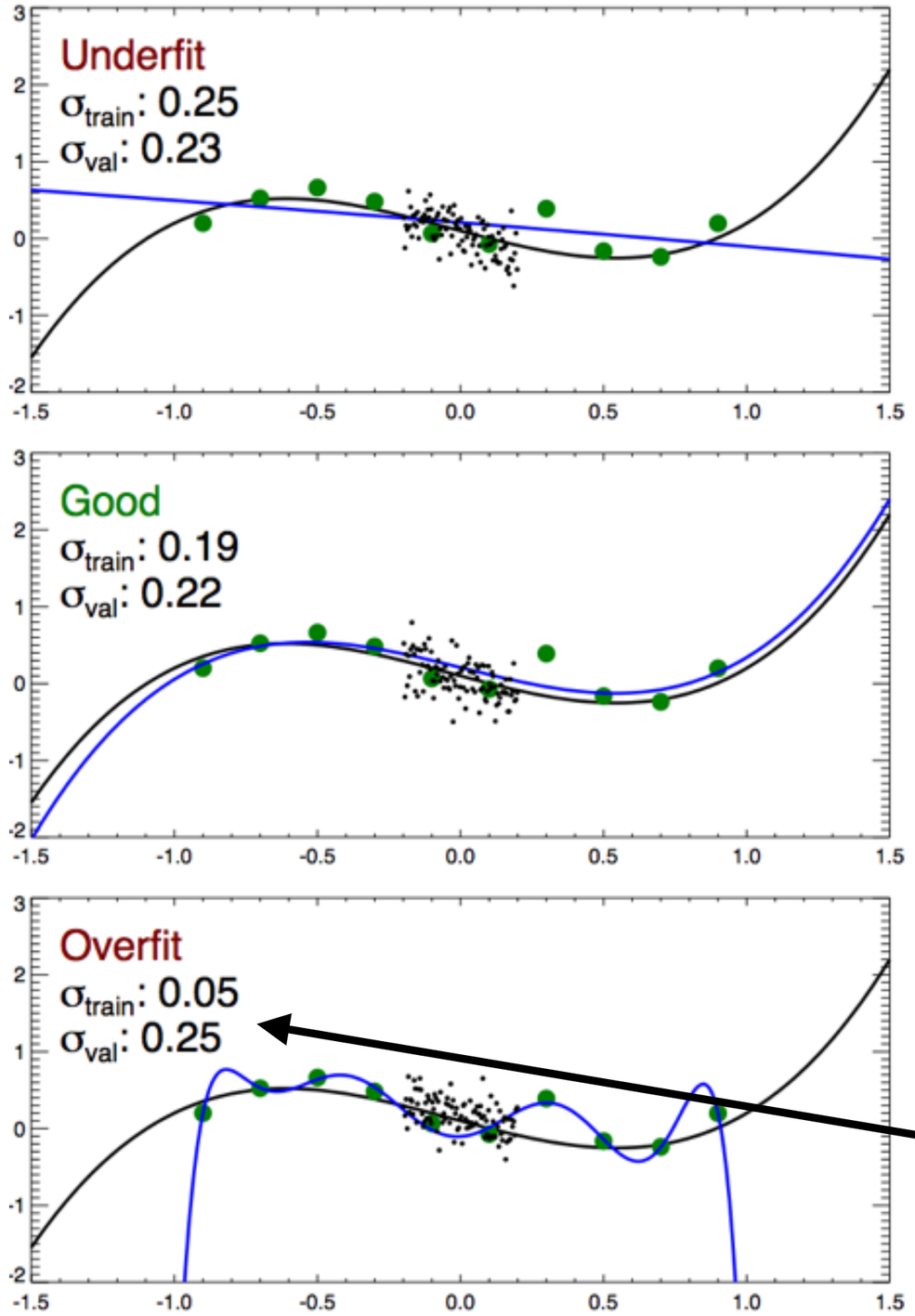
Functions of many parameters may be overfit in some regions or some dimensions and not in others.

The training and validation data may span a (relatively) *low-dimensional manifold* in the input parameters space.

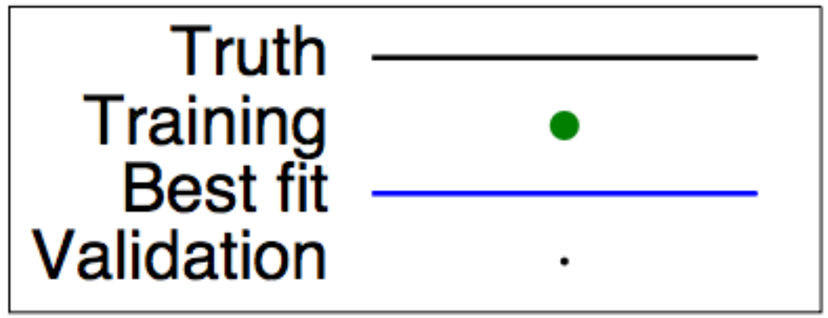
The classifier may be well fit on that manifold, and fail miserably away from it. And you may never know.

Overfitting example

This can be illustrated with a simple 1-D fitting function.

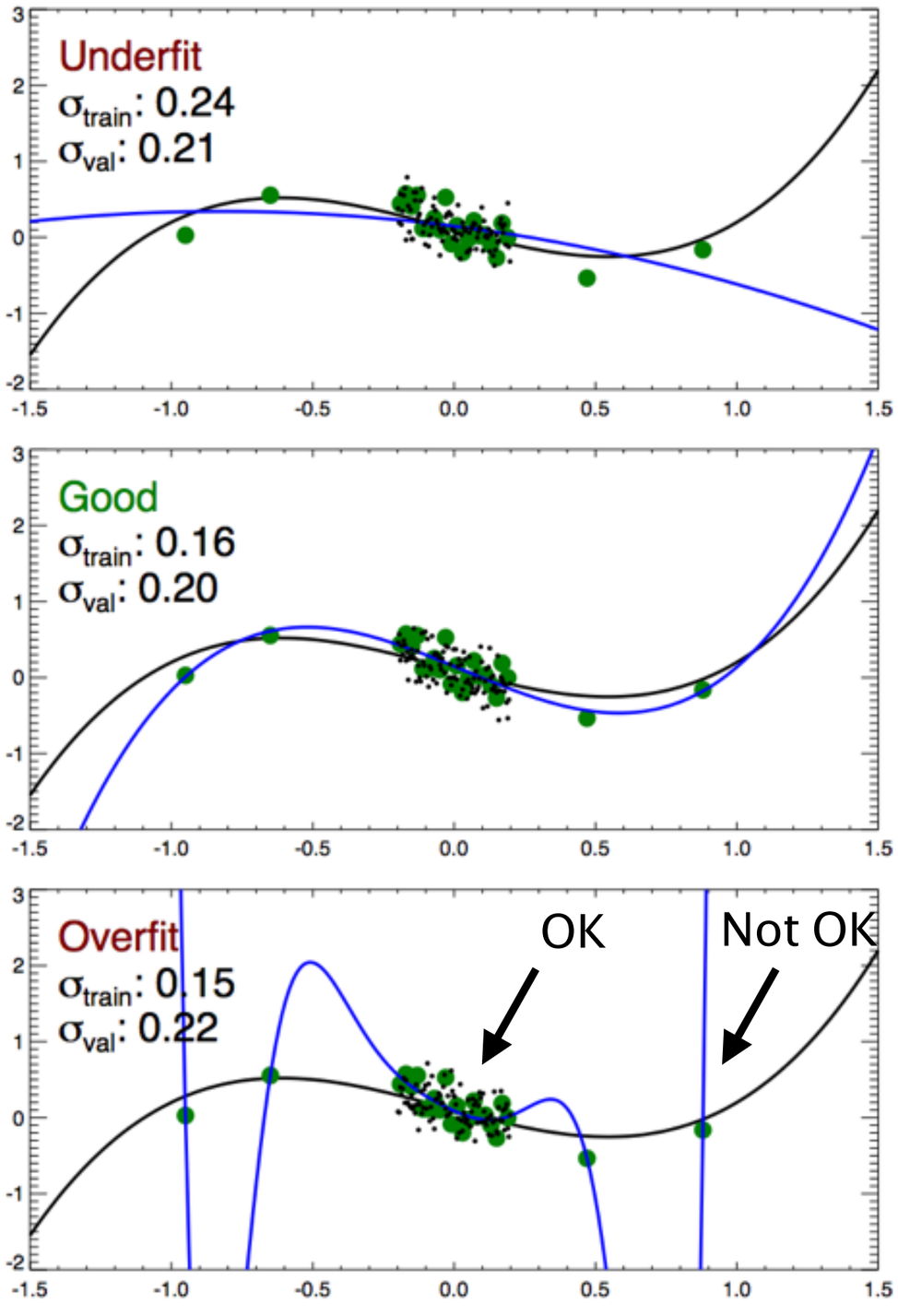


Uniformly distributed training sample:

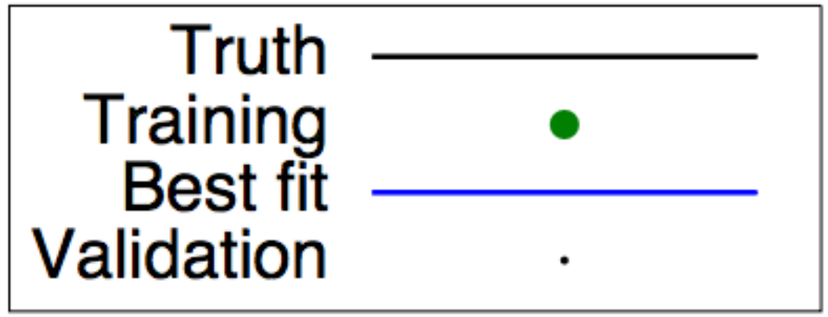


As fit freedom increases, sigma_train falls.

Overfitting is obvious:
(fit does *much* better on training data than validation data)



Non-uniformly distributed sample:



If training and validation sets are concentrated, can be overfit some places and not others.

Class boundaries

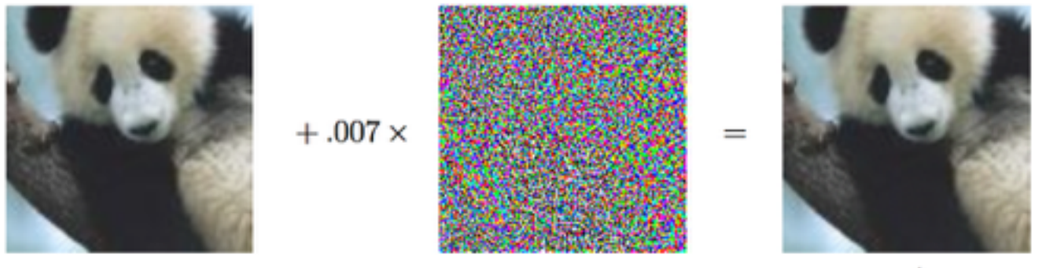
The boundary between “panda” and “gibbon” may be crumpled like tinfoil in a 10,000-dimension space.

Class boundaries

The boundary between “panda” and “gibbon” may be crumpled like tinfoil in a 10,000-dimension space.

That boundary is far too close for comfort.

Any regularization that pushes that boundary back may help. (at the expense of performance)



Adversarial examples

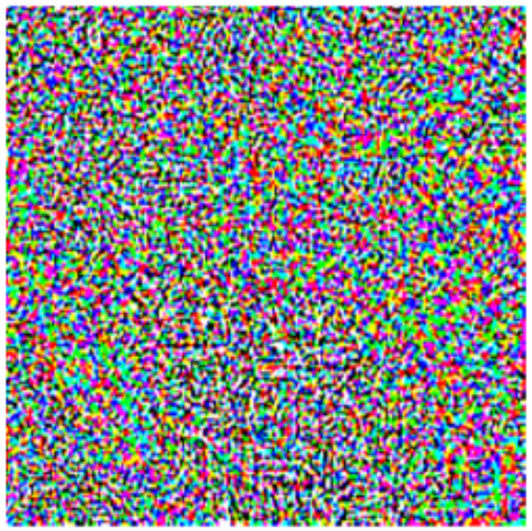
Adversarial examples (noise)

Adversarial examples (Advx) for image classifiers are easy to find.
Making a *tiny change* to all pixels can change the output radically.

$\boldsymbol{x}$

"panda"
57.7% confidence

$+ .007 \times$



$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"nematode"
8.2% confidence

$=$

$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"gibbon"
99.3 % confidence

"confidence" is not "probability"

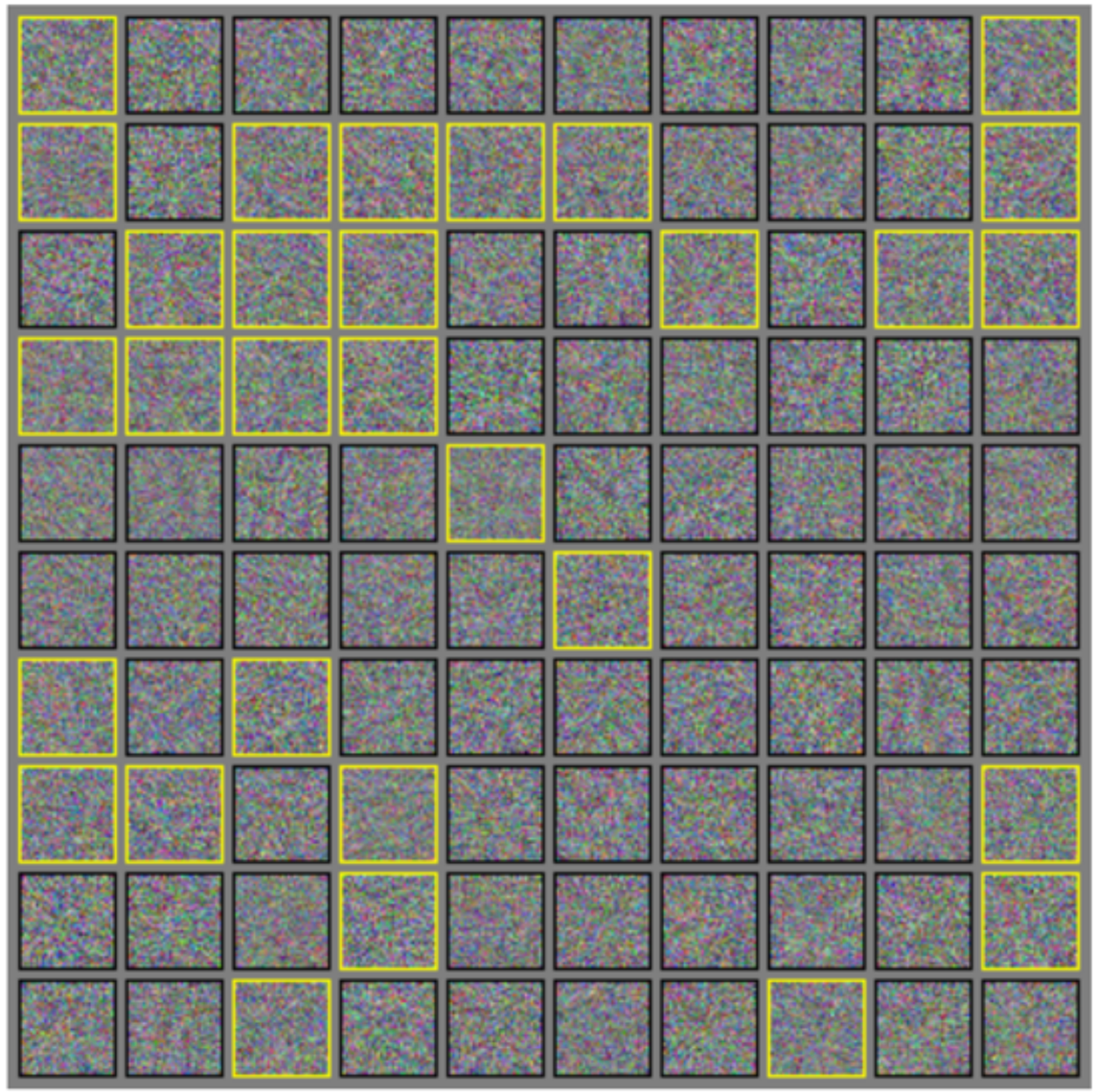
Goodfellow+ 1412.6572

Adversarial examples (noise)

And it is worse than that. In the case of CIFAR-10, most classes can be found near a random Gaussian noise image (single gradient step).

The yellow boxes are class "airplane."

This was the *hardest* case.

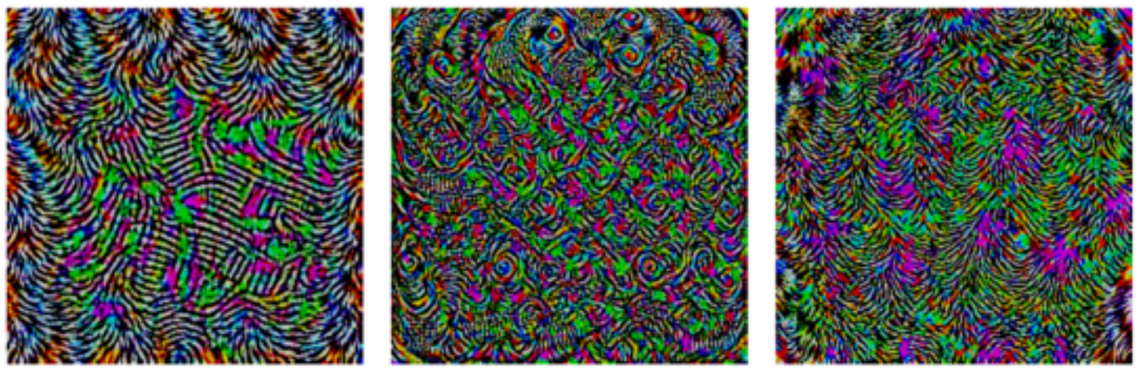


Goodfellow+ 1412.6572

Universal Adversarial examples

It's much worse than *that*.

One can find noise patterns that fool most classifiers on most images.



(d) VGG-19 (e) GoogLeNet (f) ResNet-152

Transferability is 40-70%

	VGG-F	CaffeNet	GoogLeNet	VGG-16	VGG-19	ResNet-152
VGG-F	**93.7%**	71.8%	48.4%	42.1%	42.1%	47.4 %
CaffeNet	74.0%	**93.3%**	47.7%	39.9%	39.9%	48.0%
GoogLeNet	46.2%	43.8%	**78.9%**	39.2%	39.8%	45.5%
VGG-16	63.4%	55.8%	56.5%	**78.3%**	73.1%	63.4%
VGG-19	64.0%	57.2%	53.6%	73.5%	**77.8%**	58.0%
ResNet-152	46.3%	46.3%	50.5%	47.0%	45.5%	**84.0%**

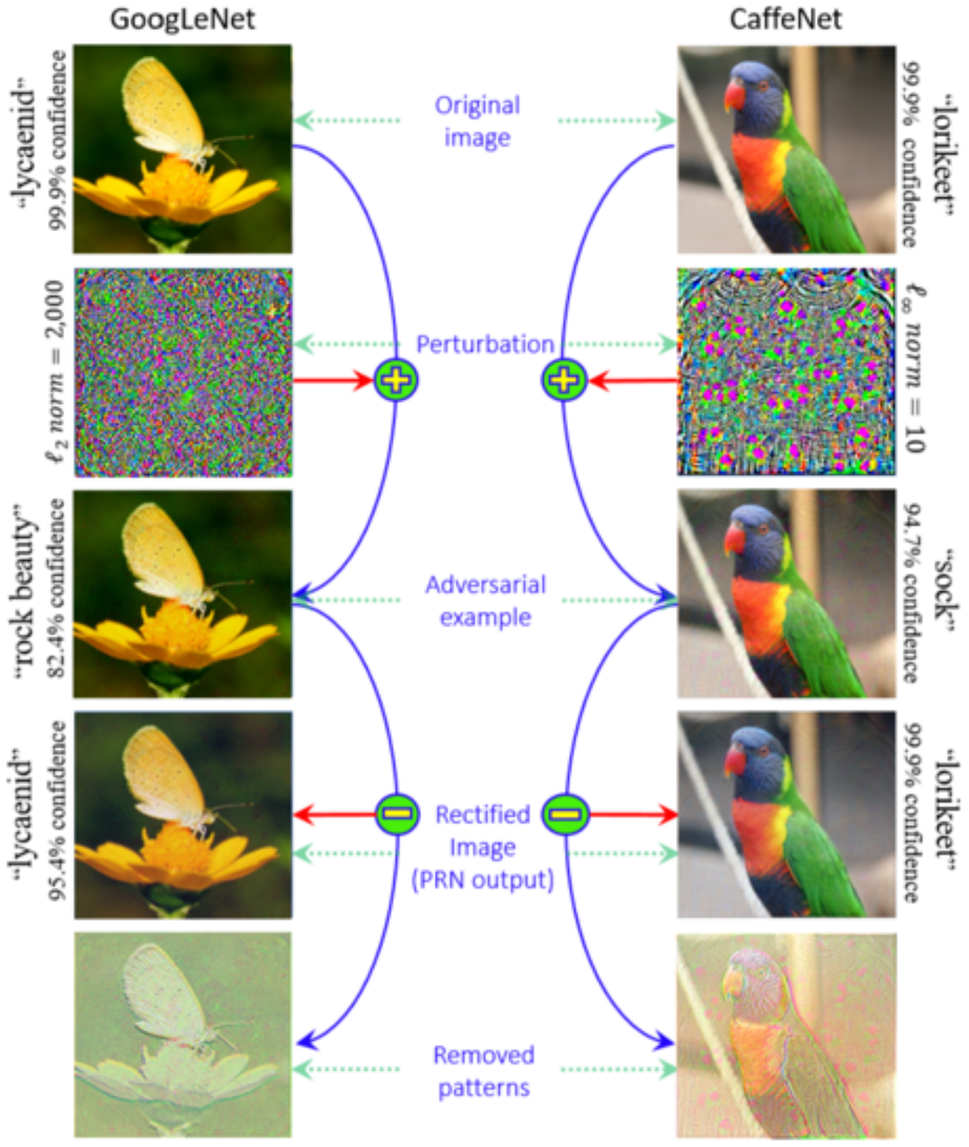
Authors suspect that they are constructing a subspace orthogonal to the manifold of "real images"

Moosavi-Dezfooli+ 1610.08401

Perturbation Rectifying Networks

Akhtar et al. adds a pre-input wrapper to the classifier to “rectify” the input. The PRN learns to undo the adversarial perturbation.

“Our defense mechanism acts as an external wrapper for the targeted network such that the PRN (and the detector) trained to counter the adversarial attacks can be kept secretive in order refrain from potential counter-counter attacks. This is a highly desirable property of defense frameworks in the real-world scenarios.



Akhtar+ 1711.05929

Adversarial Training

Train with specific adversarial examples that are the WORST for that algorithm but are hard to detect

$$\min_\theta J(\theta, x, y)$$

$$\downarrow$$

$$\min_\theta [\max_\delta J(\theta, x+\delta, y)]$$

Looks promising, is hard to implement, but is easy to circumvent.

Madry group, MIT

Adversarial Training

Adversarial training

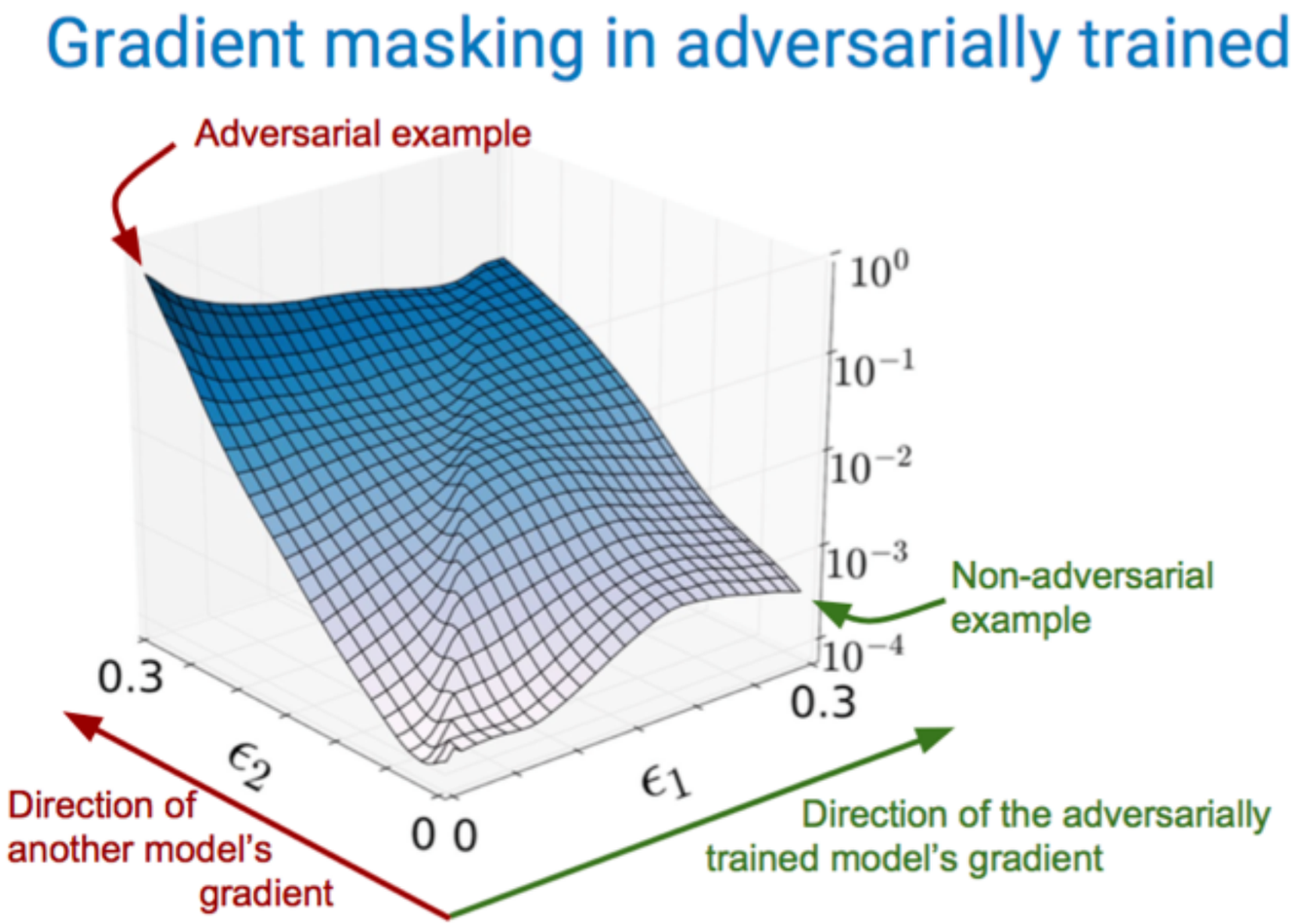
$$\text{loss}(x, y) + \text{loss}(x + \epsilon \cdot \texttt{sign}(\text{grad}), y)$$

Small when prediction is correct on legitimate input

Small when prediction is correct on adversarial input

5

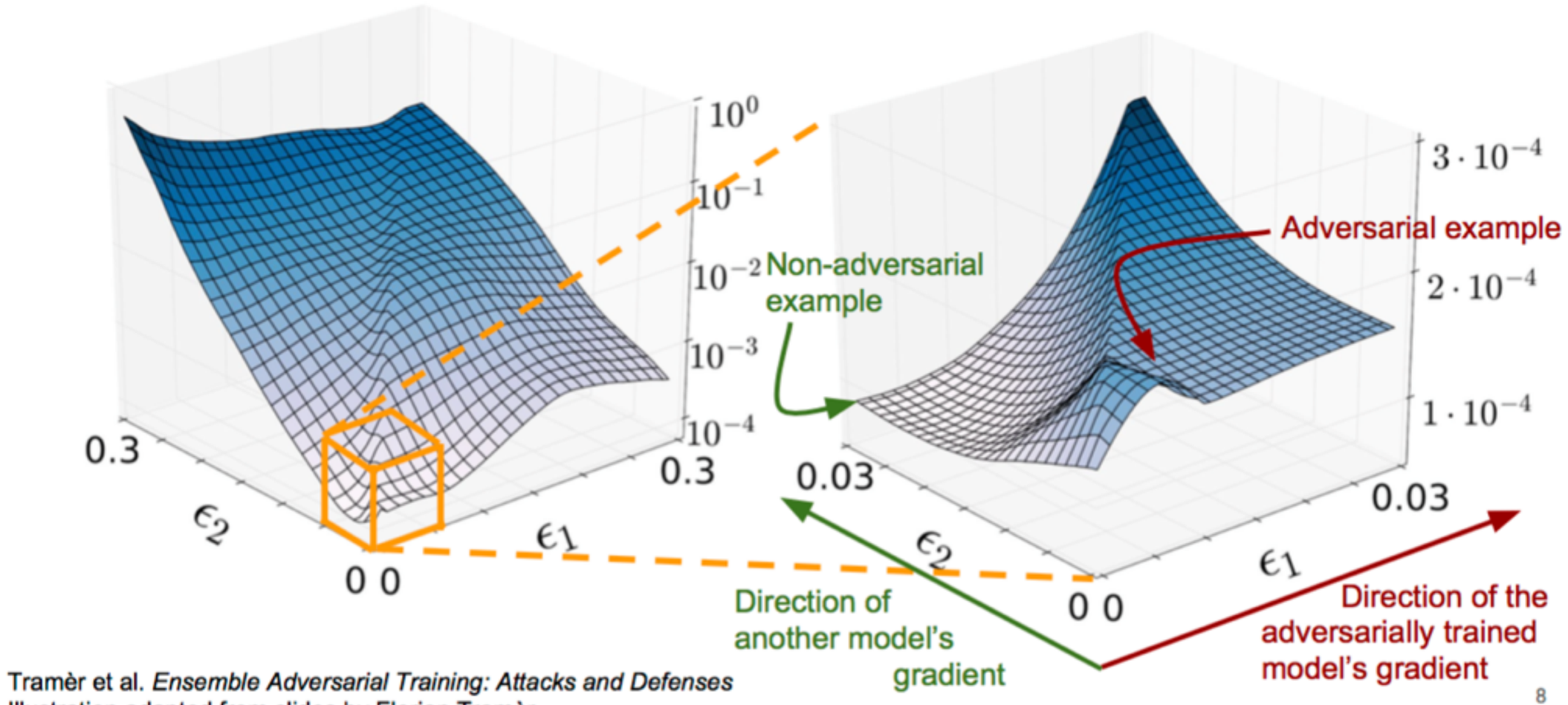
Gradient Masking can help



Tramèr et al. *Ensemble Adversarial Training: Attacks and Defenses*
Illustration adapted from slides by Florian Tramèr

7

Gradient Masking can help



Tramèr et al. *Ensemble Adversarial Training: Attacks and Defenses*
Illustration adapted from slides by Florian Tramèr

8

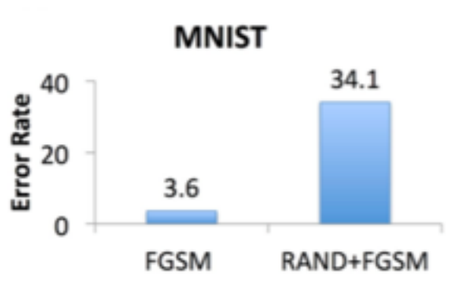
Gradient Masking can help, but not enough

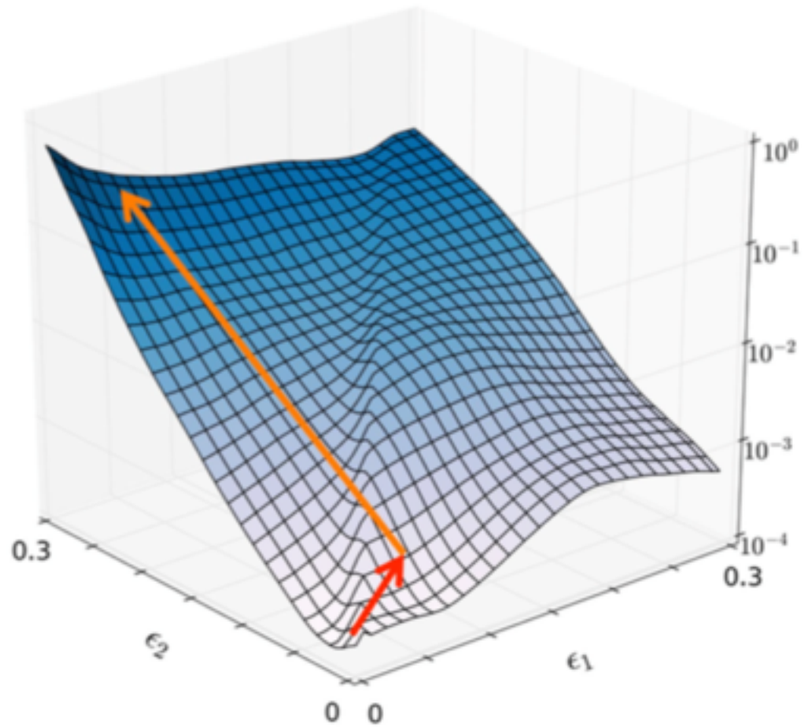
Evading gradient masking (1)

Threat model: white-box adversary

Attack:

(1) Random step (of norm alpha)
(2) FGSM step (of norm eps - alpha)





Tramèr et al. *Ensemble Adversarial Training: Attacks and Defenses*
Illustration adapted from slides by Florian Tramèr

9

Gradient Masking fails

Conclusion

- This defense works well for “one-step”gradient attacks.
- It is somewhat transferrable.
- 2-step attacks circumvent it.

Examples from in the physical world

Those were all pixel-based attacks and defenses.
They are interesting for understanding DNN models, but not very relevant to “real life.”

What about attacks in the physical world?

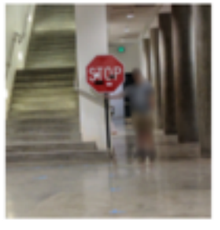
Adversarial examples (physical)

This works in the physical world as well, e.g. road signs:
(about 40 classes)

"Speed Limit 45"
66% confidence

"Speed Limit 45"
100% confidence

Eykholt+17, 1707.08945

Distance/Angle	Subtle Poster	Subtle Poster Right Turn	Camouflage Graffiti	Camouflage Art (LISA-CNN)	Camouflage Art (GTSRB-CNN)
5′ 0°					
5′ 15°					
10′ 0°					
10′ 30°					
40′ 0°					
Targeted-Attack Success	100%	73.33%	66.67%	100%	80%

Eykholt+17, 1707.08945

Adversarial examples (physical)

Some examples are less obvious — it looks like a dirty, faded sign.

"Speed Limit 45"
100% confidence

Real world example
Will the NN work?

Eykholt+17, 1707.08945

Transfer Learning

Those were attacks where the adversary has detailed knowledge of the model (enough to compute derivatives).

Sometimes the adversary cannot access the model, but knows its early layers were trained on widely used public data (ImageNet). This is *transfer learning*.

Gu et al. demonstrated that a back door installed in one model can be inherited by another via transfer learning.

Gu+ 1708.06733

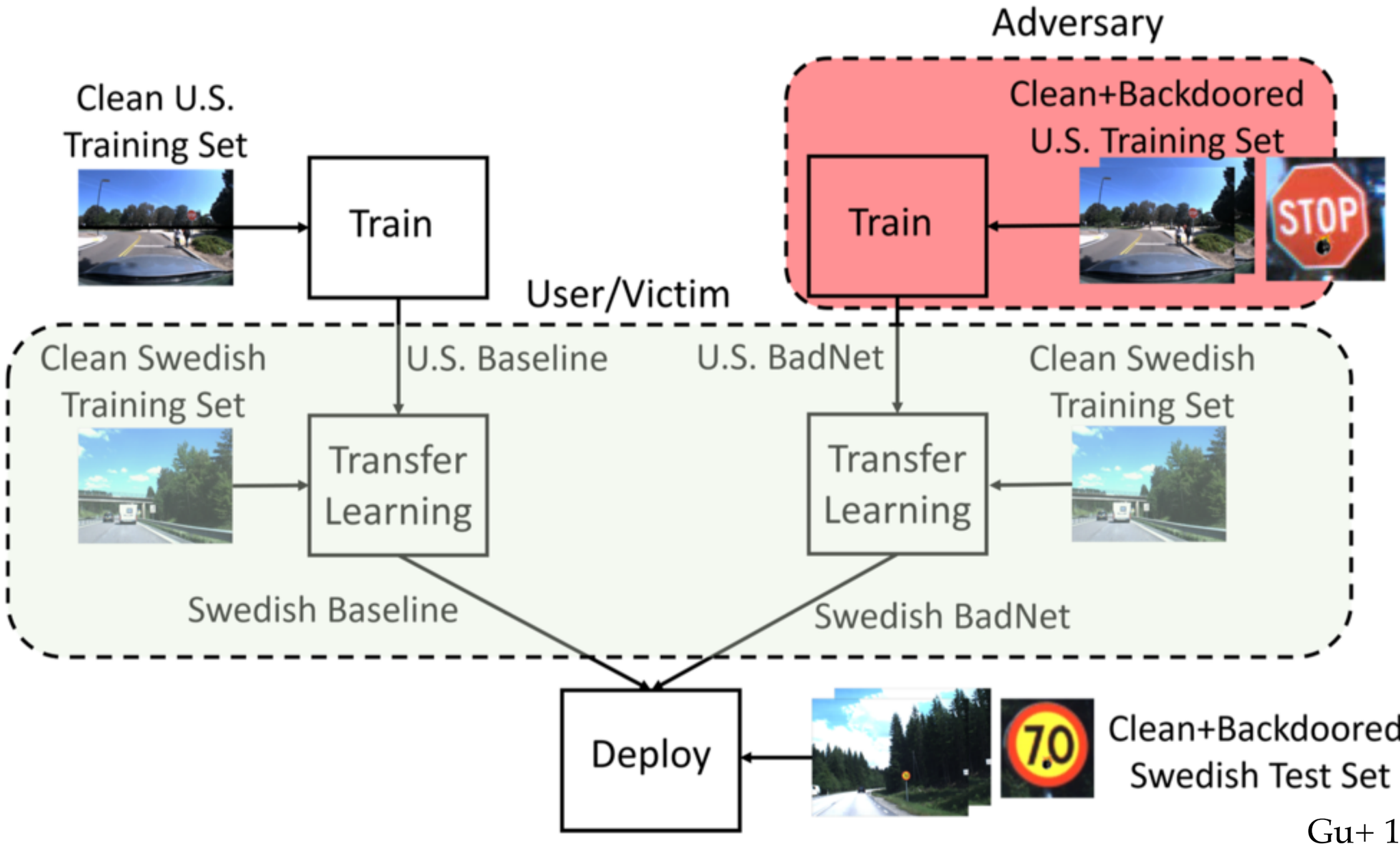
BadNets

A model was trained to misclassify a Stop sign with a yellow square.

When this model was retrained with Swedish road signs, it remembered the back door.

Gu+ 1708.06733

BadNets



Gu+ 1708.06733

Models “remember” too much

A related problem is that models may remember specific instances of training data — especially unusual examples.
These faces were recovered from a small number of “features.”

Mai+17, 1703.00832

Reverse engineering

There is concern about revealing training data.
In the commercial world one worries about privacy.
There are also worries about proprietary / sensitive training data.

Database query question is :
Can you tell something about a specific individual from a query?

The techniques of differential privacy limit the probability of revealing sensitive information (by adding carefully crafted noise to the output).

These techniques can be used to protect training data.

Distillation

Original question: Do DNNs need to be so deep?
(Ba & Caruana, 2014)

You can *distill* the information learned by a deep net into a smaller net by training on confidence vectors output by the larger net. (Hinton+ 2014)

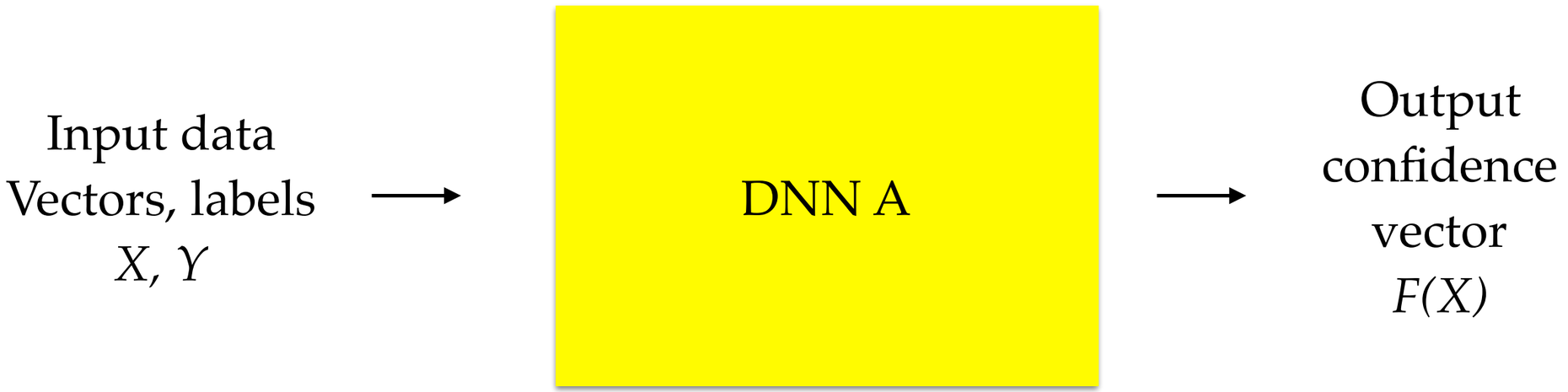
This process may also obscure the original training data to some extent.

Distillation

Start with (data, label) pairs (X, Y)

X: input data

Y: hard label, e.g. (0,0,0, … 1, … 0,0,0)



Minimize loss function
to make $F(X)$ approximate Y.

Distillation

Then train a smaller DNN on the probability vectors F(X) output by the first DNN.

- Minimize loss function to make $F^d(X)$ approximate $F(X)$.
- The vectors F(X) contain more information than the original labels, because they express uncertainty of ambiguous examples.
- This allows DNN B to be smaller (fewer nodes and layers).
- Check for loss of accuracy in this procedure!

Defensive Distillation

Idea: Use distillation to increase robustness to advx rather than decrease size of network.

Training the first network at a higher “temperature” effectively smooths the outputs.

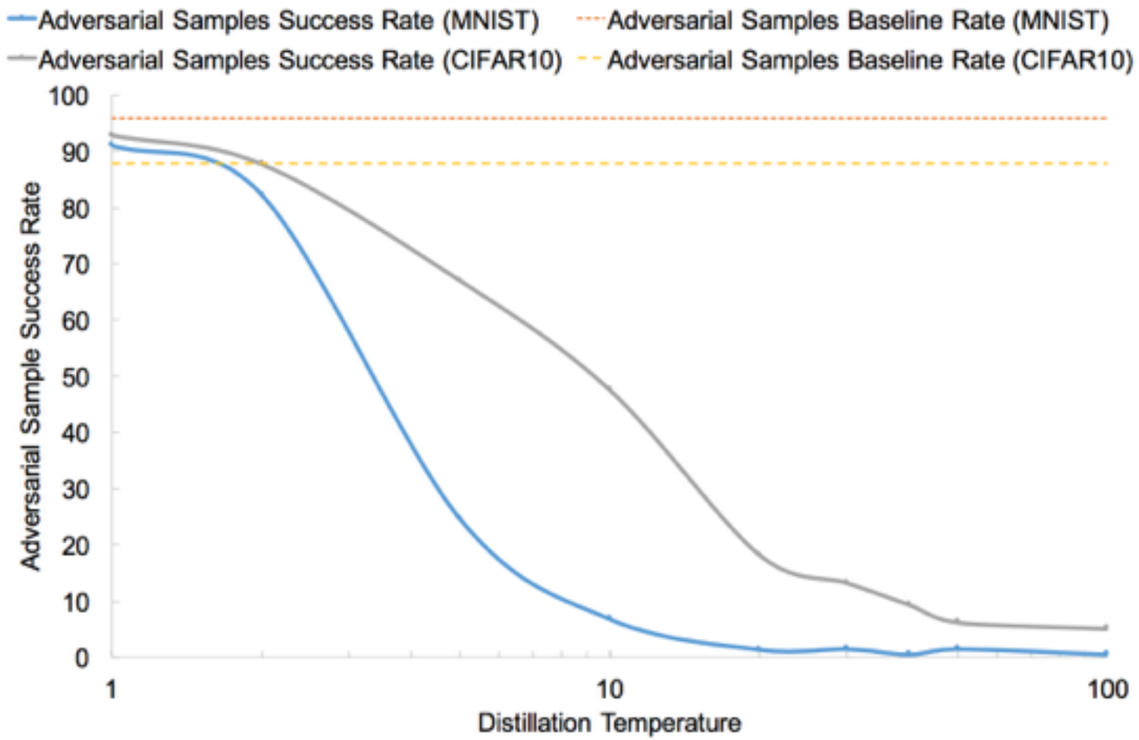
$$F(X) = \left[\frac{e^{z_i(X)/T}}{\sum_{l=0}^{N-1} e^{z_l(X)/T}} \right]_{i \in 0..N-1}$$

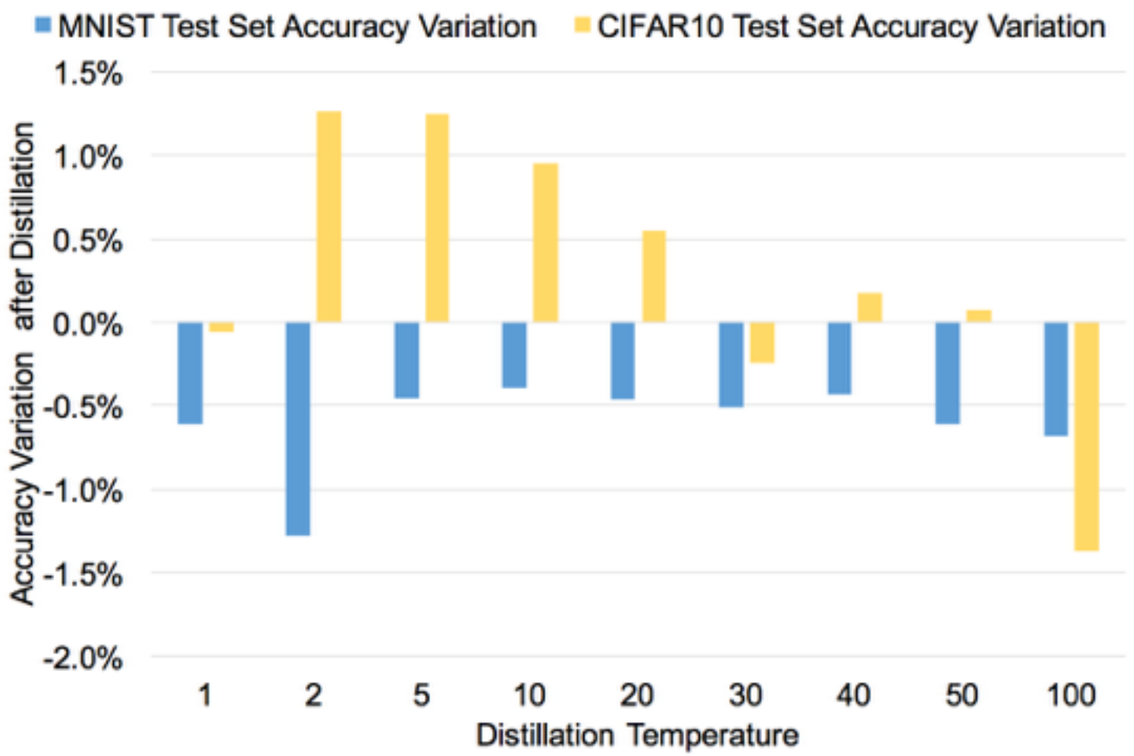
(“Smooth” = classify inputs consistently in the neighborhood of a given training sample.)

Papernot+ 2016, 1511.04508

Defensive Distillation

Papernot et al. obtained promising results on MNIST and CIFAR10-based models:
Robustness increases without much change in accuracy.





Papernot+ 2016, 1511.04508

Defensive Distillation

Defensive distillation helps by reducing gradients in the neighborhood of training points. However, a substitute model trained on the distilled model still has gradients. Adversarial examples trained on a substitute still work on the distilled model!

See Papernot+ 1602.02697 for how to defeat Papernot+ 2016, 1511.04508

Papernot+ 2016, 1511.04508

Defensive Distillation

Another idea: Use a *different* set of inputs to train the second network.

(X,Y): large sample of possibly tampered data
(X′,Y′): pristine training data

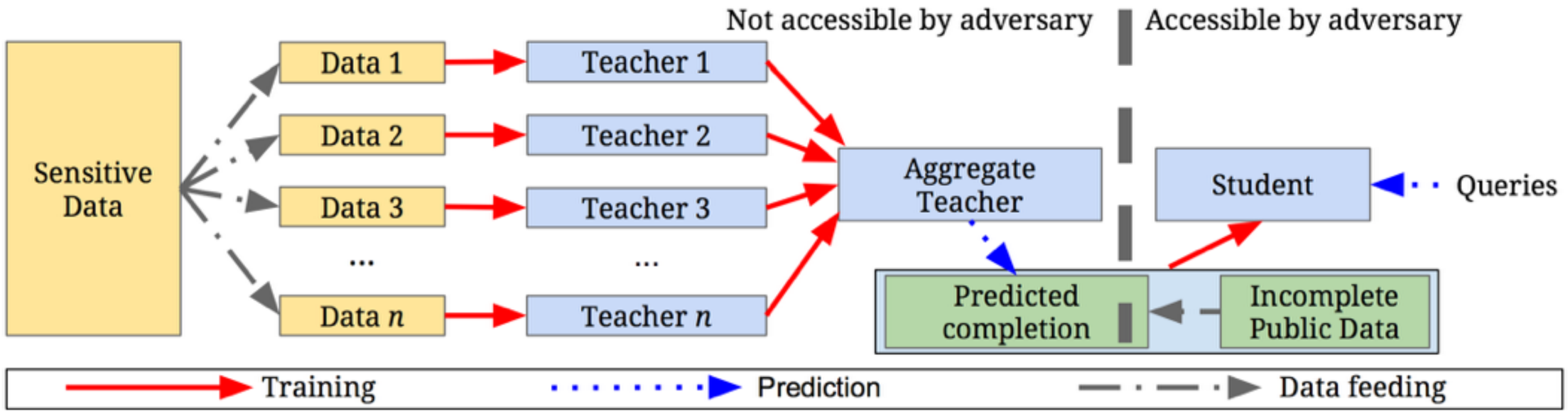
Train on *(X,Y)* to obtain *F(X)*
then train on *(X′, F(X′))*
Check against Y′ for loss of accuracy.

If *(X,Y)* contains backdoors that do not exist in *(X′,Y′)*, they will be *forgotten!*

Perhaps this form of "machine forgetting" will be the most important legacy of the distillation idea.

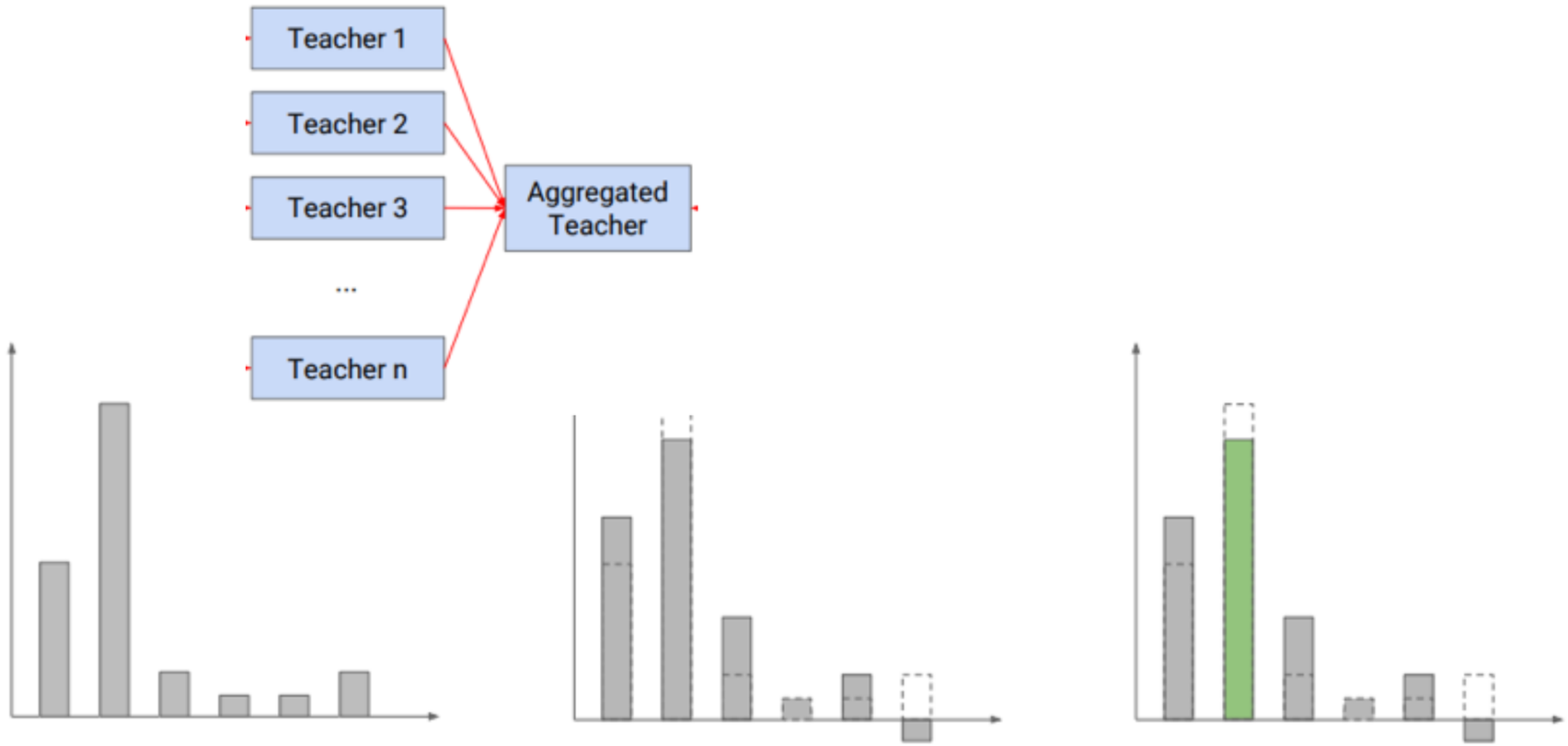
Ensembles can protect sensitive data

- Methods developed to prevent algorithms from "memorizing" (and thus revealing) specific input data
- Naturally smooths the model output
- Also defends against model theft by building a layer between the input data and the outward-facing model



Papernot+ 1610.05755, Papernot+ 1802.08908

Noisy Aggregation Example: PATE



Papernot+18

Why PATE-like approaches can help

Noise and aggregation prevent outcomes from depending on single or small number of training points (privacy)

For same reasons, also prevent overfitting and help smooth loss functions

There's no free lunch: Trade accuracy for robustness by introducing noise

The level of noise needed (or loss of accuracy) depends strongly on the input data — how (in)homogeneous the dataset is

Deep k-Nearest Neighbors

A novel approach to detecting mischief is to follow the training data through the layers alongside a new input.

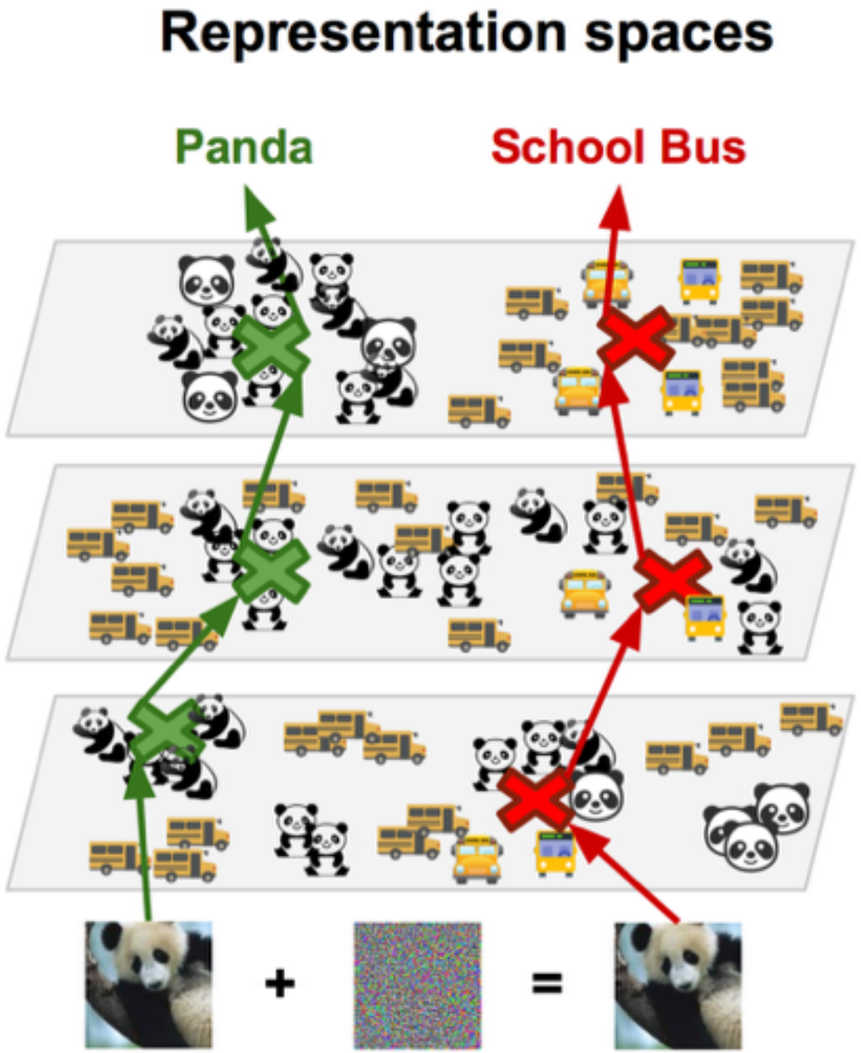
> *"In adversarial settings, this yields an approach to defense that differs from prior work in that it addresses the underlying cause of poor model performance on malicious inputs rather than attempting to make particular adversarial strategies fail."*

Papernot et al. go on to develop a

> *novel characterization of confidence, called* ***credibility****, that spans the hierarchy of representations within of a DNN: any credible classification must be supported by evidence from the training data.*

Papernot & McDaniel 1803.04765

Deep k-Nearest Neighbors



A robust result has training samples of the same class "nearby" in each layer of the model.

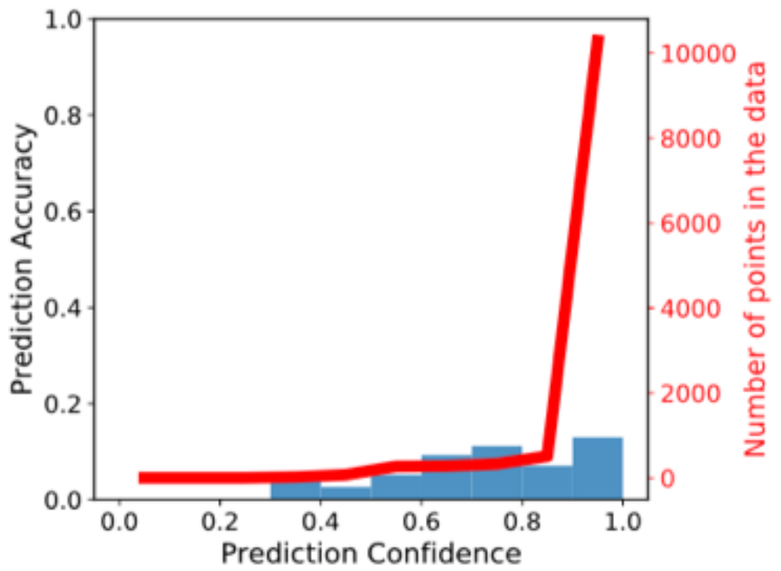
An adverse (or genuinely ambiguous) example has a mix of training points nearby.

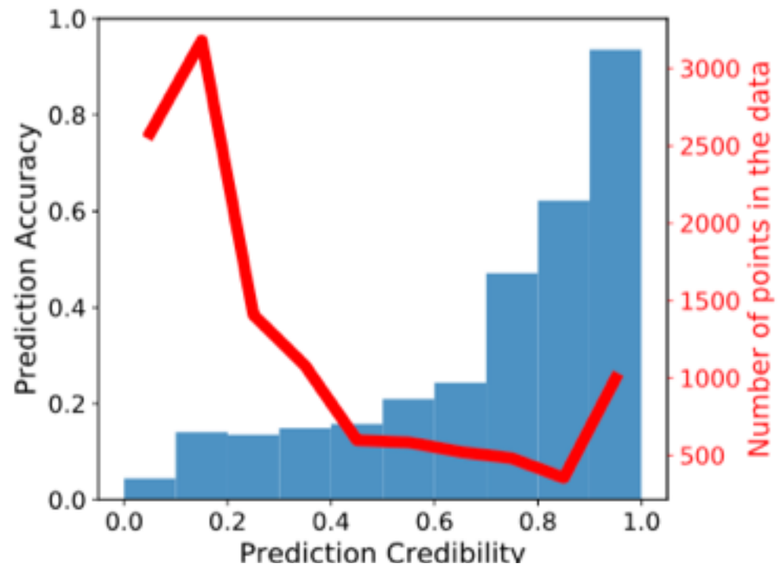
This depends critically on the definition of "near."

Papernot & McDaniel 1803.04765

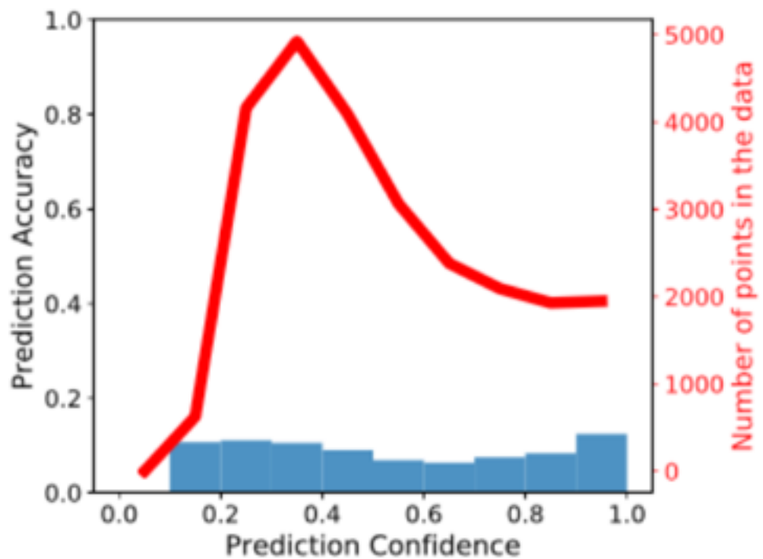
Deep k-Nearest Neighbors

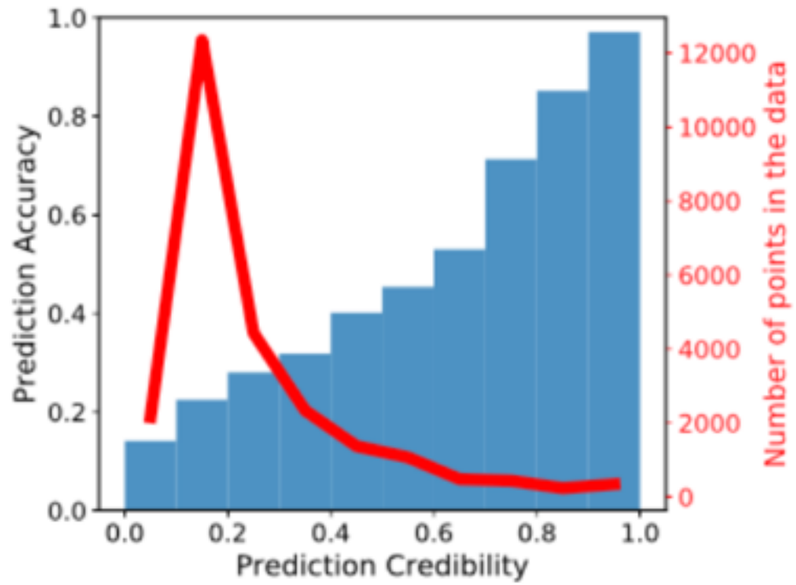
GTSRB





SVHN





Papernot & McDaniel 1803.04765

Deep k-Nearest Neighbors

DkNN represents progress in confidence, interpretability, and robustness.

(a) confidence can be viewed as estimating the distance between the test input and the model's training points,

(b) interpretability is achieved by finding points on the training manifold supporting the prediction, and

(c) robustness is achieved when the prediction's support is consistent across the layers of the DNN

Papernot & McDaniel 1803.04765

Adversarial Training

Train with specific adversarial examples that are the WORST for that algorithm but are hard to detect

$$\min_\theta J(\theta, x, y)$$

$$\downarrow$$

$$\min_\theta [\max_\delta J(\theta, x+\delta, y)]$$

Looks promising, is hard to implement, but results in a more robust model.

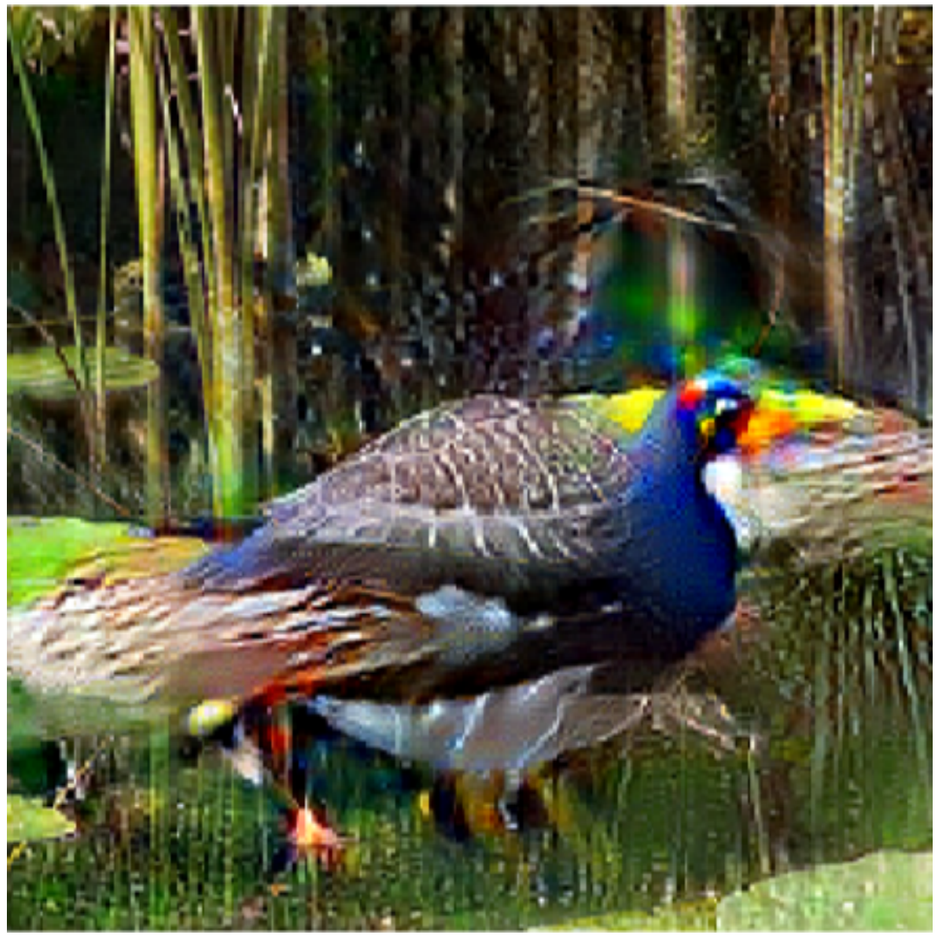
Madry+ 1706.06083, Tsipras+ 1805.12152

Adversarial Training

Turtle → Bird

Madry+ 1706.06083, Tsipras+ 1805.12152

Adversarial Training



Turtle → Bird

Madry+ 1706.06083, Tsipras+ 1805.12152

Adversarial Training

Cat → Dog

Madry+ 1706.06083, Tsipras+ 1805.12152

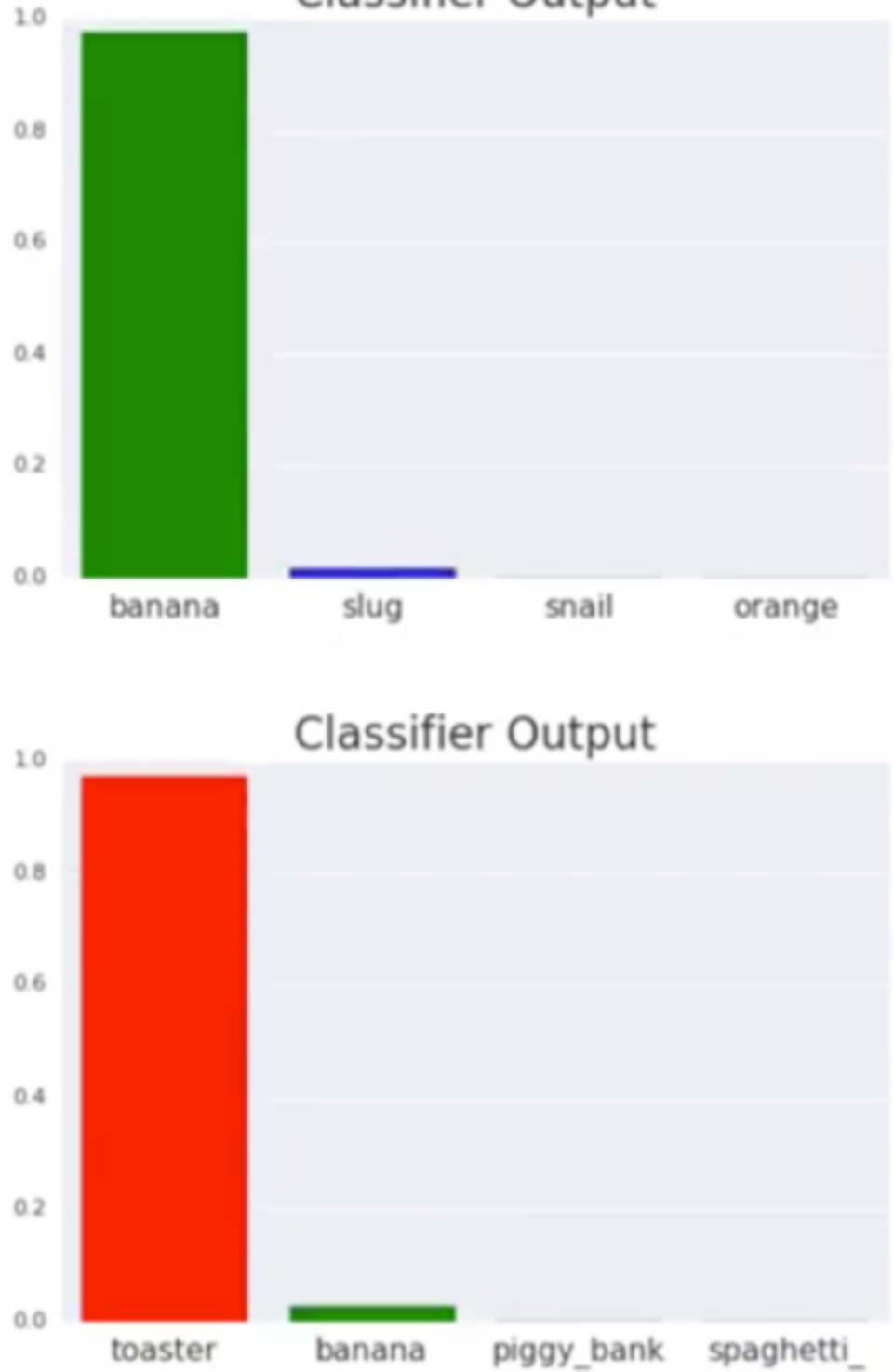
Adversarial Training

Cat → Dog

Madry+ 1706.06083, Tsipras+ 1805.12152

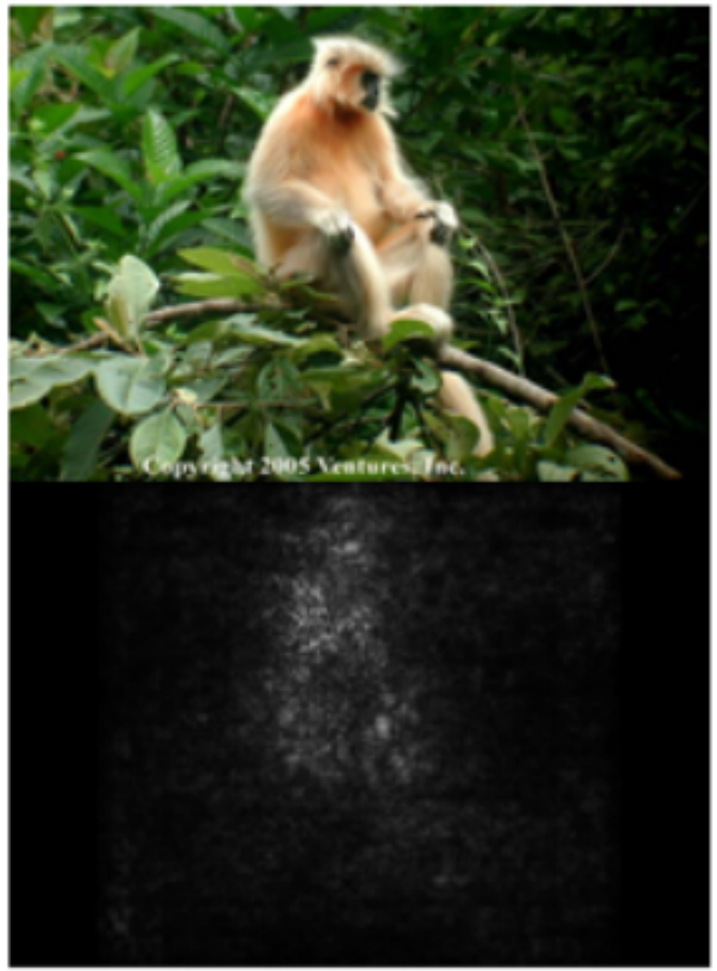
place sticker on table

Classifier Input



Classifier Input

Simonyan Saliency 1312.6034



Simonyan Saliency 1312.6034

Interpreting outputs as probability

The DNN classification literature often refers to the output of the final softmax layer of a DNN classifier as a vector of *probabilities*.

We should be uneasy with this terminology and the report uses the vague term *confidence*.

But this raises the question: **what would it take for the outputs to reflect the probability that the classification is correct?**

Interpreting outputs as probability

We want to know $P(y \mid x)$, the probability that the class label is y, given input x.

According to Bayes' theorem, this is proportional to

$$P(x \mid y)\, P(y)$$

where $P(x \mid y)$ is the probability of the features x, given class y, and $P(y)$ is the prior on y.

Interpreting outputs as probability

In choosing a training data set, we may select a number of examples of each class y and then for each example generate or observe the features x.

Example: in the case of ImageNet, one can choose y="cat" and find a large number of cat images on the Internet.

The images used are (ideally) a "fair draw" from the space of all possible cat images *that we care about,* that is, cat images that someone thought worthy of posting.

Training forces Q to approximate P

$P(y \mid x)$: The true probability of y conditional on x.
Training data generated by $P(x \mid y) \sim P(y \mid x) \ / \ P(y)$

$Q(y \mid x; \theta)$: The NN model with parameters θ, intended to approximate $P(y \mid x)$.

Training of a classifier minimizes the cross-entropy loss function,

$$J = H(P, Q) = H(P) + D_{\mathrm{KL}}(P \parallel Q),$$

is equivalent to minimizing the Kullbeck-Leibler divergence, D_{KL}. The entropy of P, $H(P)$, depends only on the training data, and is constant during training.

Kullbeck-Leibler Divergence

The **Kullbeck-Leibler divergence**, $D_{\mathrm{KL}}(P \,\|\, Q)$, is a measure of the difference between two probability distributions, P and Q.

In information theory, it is the number of "wasted bits" of information if you transmit a message with symbols occurring with frequency P using a compression scheme developed assuming frequency Q.

In Bayesian inference, it is the amount of information learned from your data for prior distribution Q and posterior P.

Colloquially, it is the amount of *surprise* if you expected distribution Q and observe distribution P.

Kullbeck & Leibler, 1951

Training forces Q to approximate P

Minimizing the cross entropy loss function (and therefore D_{KL}) is equivalent to saying

"Make Q as similar to P as possible."
i.e.,
"Make the model distribution match the data-generating distribution as closely as possible."

But how close is that? If the model has too little freedom, Q may not approximate P very well. If it has too much freedom, Q may approximate P well at the training points, and not elsewhere.

…but only so well…

To summarize: IF

- the training samples fully describe $P(x \mid y)$, and
- the output is a softmax layer, and
- the loss function is cross-entropy, and
- the model has enough freedom, but not too much,

THEN you could treat the output class probabilities as real probabilities. Because these requirements are seldom satisfied, the output generally is *not actually a probability of anything*.

And that was non-adversarial

That was for non-adversarial examples. In general, adversarial examples reside "off the manifold" described by the training data.

Existing models may classify such inputs incorrectly, and with high confidence — indeed, often with **higher confidence than non-adversarial inputs**.

See Papernot+ 1803.04765 Sec III for a discussion of confidence, interpretability, and robustness.

Could it ever work?

The requirement that the training samples fully describe $P(x \mid y)$ may be intractable for natural images of the world, but there are other applications.

- low-resolution (R=100-1000) spectrum
- 100s of molecular species of interest
- training data spanning likely concentrations and combinations

This is a non-linear problem because of line saturation and radiative transfer, but it is mostly quasi-linear. A sample of millions of synthetic spectra might well span the relevant space.

It is conceivable that a NN could output a rigorous probability of detection in this case.

Calibrated confidence

More often, *calibrated confidences* will have to suffice.
Temperature scaling (effectively rescaling the inputs to the softmax layer) is surprisingly effective at calibrating in some cases. These do not provide rigorous probability estimates, but may be useful anyway.

See Guo+ 1706.04599

The End

We're doomed. But not forever.

By thinking carefully about how ML models break (in general, not just DNNs!!!)
we can gain confidence in our results.

Maybe we can get to a point where we *really trust* neural nets. But we aren't there yet.