TV regularization leads to adversarially robust *and* accurate neural networks

Level Set Collective, UCLA
February 26, 2019

Chris Finlay
joint work with Adam Oberman and Bilal Abbasi
McGill University, Dept of Math and Stats

Background: Deep learning and AI

- Artificial intelligence (AI) is loosely defined as intelligence exhibited by machines, *on specific tasks*
- In recent years, deep learning algorithms have performed remarkably well on problems which were thought to be intractable

O Vinyals, A Toshev, S Bengio and D Erhah. “Show and Tell: A neural image caption generator”, CVPR 2015.

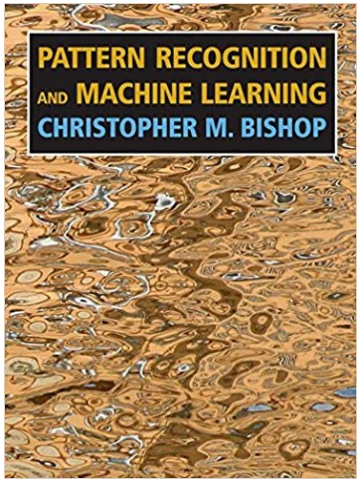
Background: Deep learning and AI

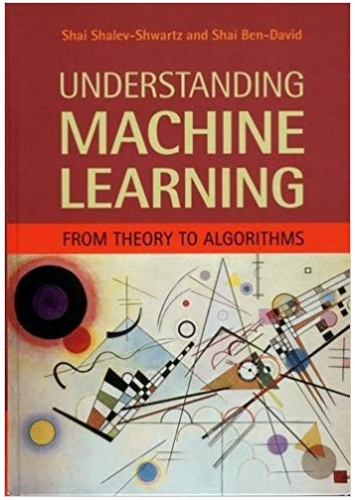
- Due to the curse of dimensionality, theory says that accurate function interpolation (eg image captioning) is **impossible**
- But with deep learning, we have very impressive practical results showing it is **possible**

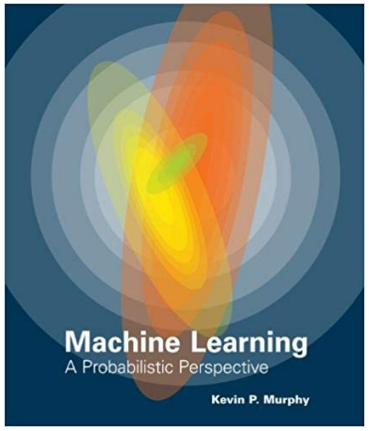
O Vinyals, A Toshev, S Bengio and D Erhah. “Show and Tell: A neural image caption generator”, CVPR 2015.

Background: Deep learning and Machine Learning

- Machine learning (ML) is a well established field
- ML has theory: convergence proofs, error bounds, theoretical guarantees
- Deep learning is a recent branch of ML, and uses *deep neural networks*
- But: deep learning *lacks theory*, even though in practice it can solve much larger problems than ML







Background: Deep learning used without theoretical guarantees

- Deep learning algorithms are now common place:
  - Computer vision
  - natural language processing
  - Robotics
  - and on and on...
- But lack of theory is a problem. We don't know why it works so well.

"It is not clear that the existing AI paradigm is immediately amendable to any sort of software engineering validation and verification. This is a serious issue, and is a potential roadblock to DoD's use of these modern AI systems, especially when considering the liability and accountability of using AI"

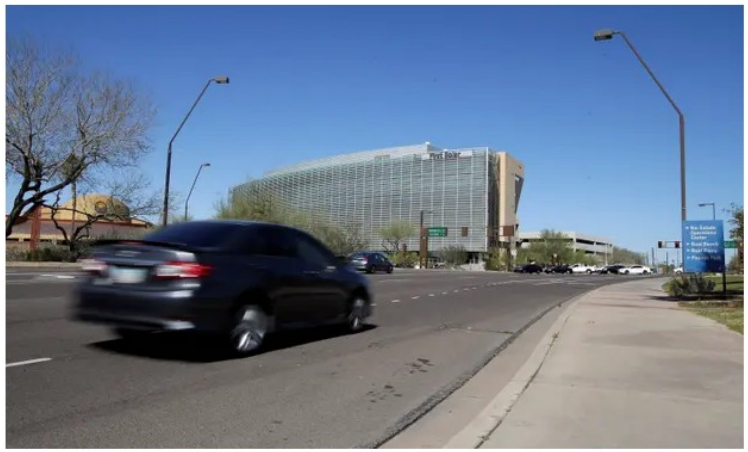
-JASON report

Background: Deep learning used without theoretical guarantees

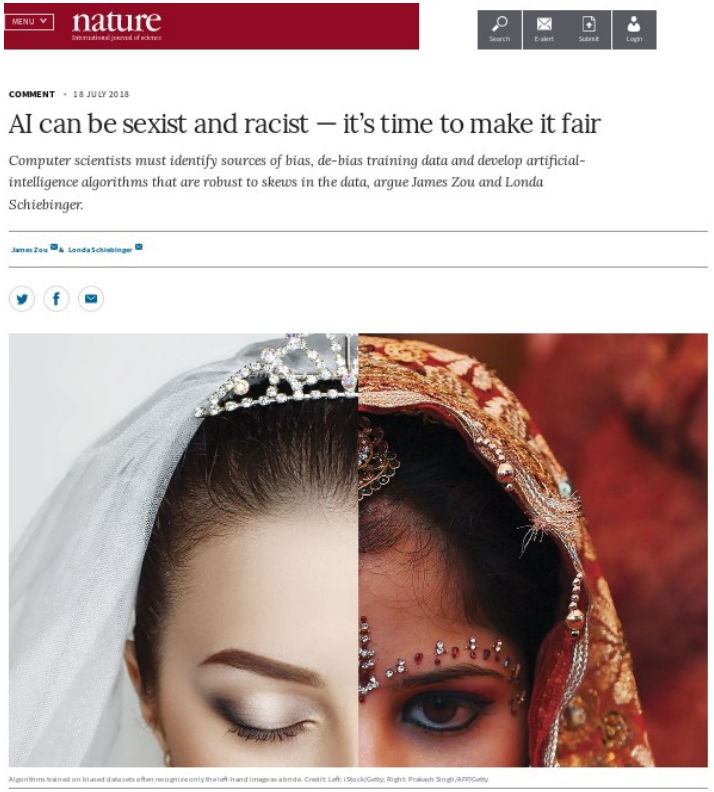
Moreover, when deep learning algorithms fail we don't know why.

Self-driving Uber kills Arizona woman in first fatal crash involving pedestrian

Tempe police said car was in autonomous mode at the time of the crash and that the vehicle hit a woman who later died at a hospital



The Guardian, March 19 2018



COMMENT · 18 JULY 2018

AI can be sexist and racist — it's time to make it fair

Computer scientists must identify sources of bias, de-bias training data and develop artificial-intelligence algorithms that are robust to skews in the data, argue James Zou and Londa Schiebinger.

Nature, July 18 2018

This talk: Adversarial examples in image classification

- In image classification, we are given a training set of N images x_i with labels y_i. Each image belongs to one of K classes.
- Images are typically scaled so that the image space X is the unit box – pixels values are in $[0,1]$.
- Label space is embedded in the probability simplex.
  - For example if image has class k, then we want to map image to label e_k
- The task is to learn a map $f:X \to Y$

Deep neural nets

- In deep learning the map $f:X \to Y$ is a deep neural net
- A composition of n linear functions W_k alternating with an element wise non-linearity σ:

$$f(X) = \sigma(W_n \sigma(W_{n-1} \ ... \sigma(W_1 X)))$$

- Common choice for σ is $\sigma(x) = \max(x, 0)$.
  - In this case, f is a piece-wise linear function.
- Each $\sigma(W_{n-1}(\cdot))$ is called a *layer*. Since there are so many layers, the network is said to be “deep”

Loss function minimization

- *f* is learned by minimizing the expectation of a loss function *L* which measures difference between model prediction *f(x)* and true label *y* over a distribution of images and labels

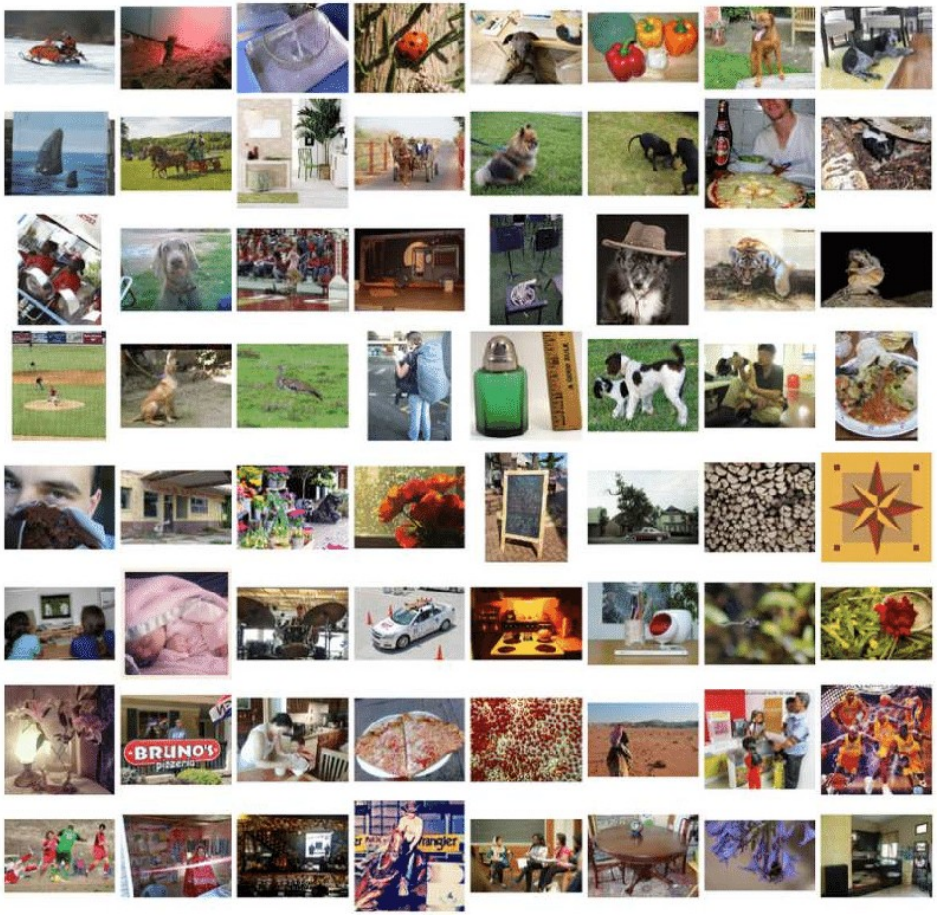
$$\mathbf{E}[L(f(x),y)] \approx (1/N)\sum_i L(f(x_i),y_i)$$

- In image classification *L* is usually the Kullback-Leibler divergence
- For lack of better alternatives, usually just minimized with stochastic gradient descent
- Minimization is done over layer parameters of the linear functions (matrices and biases). Gradient wrt parameters is computed with automatic differentiation (a nice way to apply the chain rule)
- Only since ~2012 or so has this been computationally feasible, using GPUs

Example: ImageNet

Proving ground for image classification algorithms is the ImageNet dataset

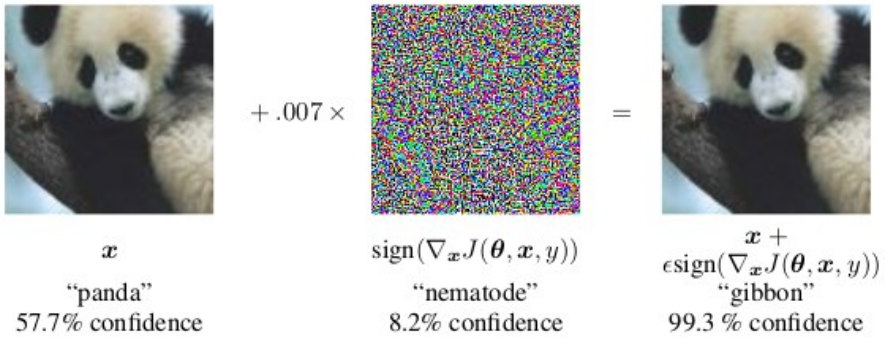
- Has 2184 classes
- Training set comprises ~14 million images
- Colour images,
  d = 3x256x256 = 196,608
- Current state-of-the-art is ~3.5% top5 error
- But this is still considered an academic dataset...



That’s nice, but...

Not long after deep learning algorithms began beating traditional ML in image classification, Goodfellow et al[1] noticed that there are perturbations which will cause a network to misclassify.

- These perturbations are **imperceptible** to humans



1 I Goodfellow, J Shlens and C Szegedy. “Explaining and harnessing adversarial examples”, 2014. arXiv:1412.6572

Adversarial examples

These images are called *adversarial examples,* caused by *adversarial perturbations* (sometimes called *adversarial attacks*).

- Not just any perturbation will do. For example deep neural nets are generally robust to Gaussian noise
- Perturbation must be worst case in a certain sense (more on this later...)

Adversarial examples

Not just an academic exercise.

Figure 1: The left image shows real graffiti on a Stop sign, something that most humans would not think is suspicious. The right image shows our a physical perturbation applied to a Stop sign. We design our perturbations to mimic graffiti, and thus "hide in the human psyche."

Eykholt et al, "Robust Physical-World Attacks on Deep Learning Visual Classification". CVPR 2018.

Adversarial examples

Not just an academic exercise.

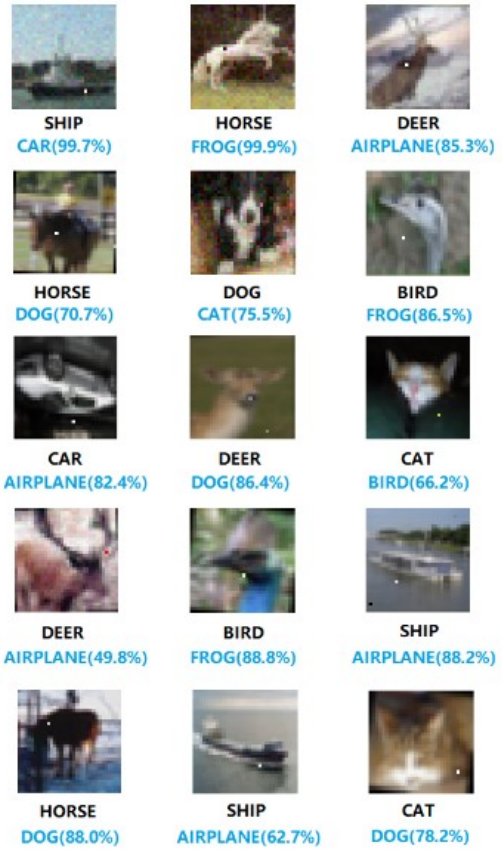
(a) (b) (c) (d)

Figure 4: Examples of successful impersonation and dodging attacks. Fig. (a) shows S_A (top) and S_B (bottom) dodging against DNN_B. Fig. (b)–(d) show impersonations. Impersonators carrying out the attack are shown in the top row and corresponding impersonation targets in the bottom row. Fig. (b) shows S_A impersonating Milla Jovovich (by Georges Biard / CC BY-SA / cropped from https://goo.gl/GlsWlC); (c) S_B impersonating S_C; and (d) S_C impersonating Carson Daly (by Anthony Quintano / CC BY / cropped from https://goo.gl/VfnDct).

Sharif et al, “Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition”. CCS 2016.

Adversarial examples

Not just an academic exercise.



J Su, D Vargas and K Sakurai. “One Pixel Attack for Fooling Deep Neural Networks”, 2017. arXiv:1710.08864

Problems posed by adversarial examples

1. Is it possible to detect adversarial examples?

- Carlini & Wagner (2017): No. Can always create adversarial examples which are undetectable
- Us: Yes, mostly. Most adversarial attacks are detectable, and those that are not come at a price

2. Is it possible to defend deep neural networks against adversarial examples, without losing model accuracy?

- Tsipras et al, “Robustness May Be at Odds with Accuracy”. ICLR 2019
- Us: yes, to varying degree. Frame adversarial robustness in terms of Calculus of Variations and regularization.

Attack detection

- In 2017, eight papers published claiming to be able to detect adversarial attacks
  - Most methods used statistics of adversarial images, eg PCA of attacked images
  - Most models ignored the model itself
  - In terms of game theory, the detector moves after the attacker
- Claims of successful attack detection were premature: Carlini & Wagner (2017) dismantled all published detection methods.
  - Were able to generate undetectable adversarially perturbed images, using a modified loss function: require image to be misclassified *and* minimize detection metric
  - In terms of game theory, Carlini & Wagner are moving after the detector

Attack detection

Our proposal: the vulnerability of an image to attack is a proxy for attack detection.

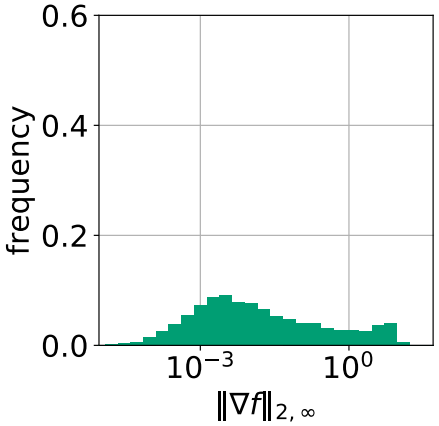
- Pessimistic: if the model can be attacked, it will be.
- An model is vulnerable to attack if a small perturbation drastically changes the loss function. A Taylor series argument (coming later) shows

$$L(x+\varepsilon d) = L(x) + \varepsilon \|\nabla L(x)\|_* + O(\varepsilon^2),\ d \text{ worst case}$$

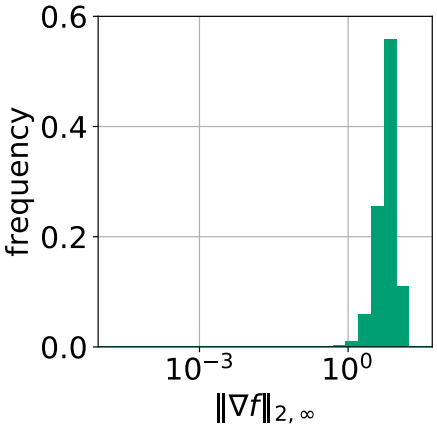
- Thus images with large gradient are vulnerable to attack.

Attack detection: works?

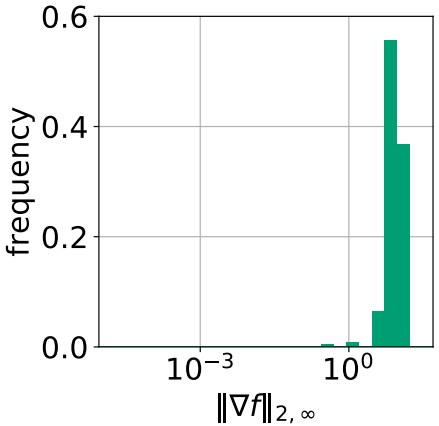
Measure size of gradient on clean (unperturbed) images, and compare with attacked images



Clean images



Gradient attack



Boundary attack

Clean images have small gradients; attacked images have large gradients.

Attack detection: works?

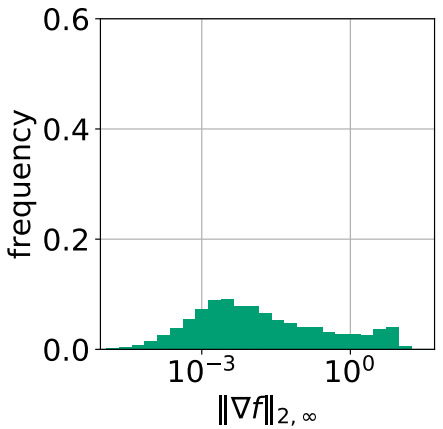
- Choose a threshold such that only 5% of clean images are incorrectly classified as being attacked.

	Image source			
	clean	PGD	Boundary	CW
attack detected?	6%	96%	100%	100%
median ℓ_2	-	0.31	0.36	0.34

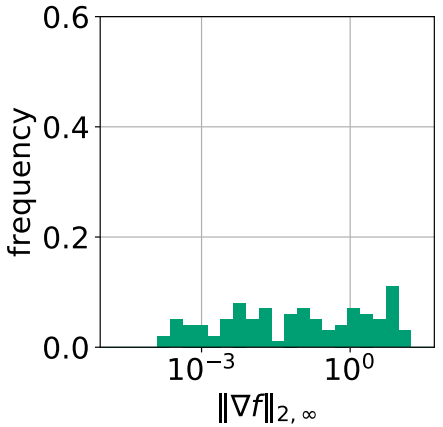
Attack detection: works?

- Choose a threshold such that only 5% of clean images are incorrectly classified as being attacked.
- Then attack, penalizing for detection (large gradients) – try to evade detection
- It is possible to avoid detection, but at the cost of greater adversarial distance

	Image source				
	clean	PGD	Boundary	CW	evasive CW
attack detected?	6%	96%	100%	100%	22%
median ℓ_2	-	0.31	0.36	0.34	0.81



Clean images



Evasive attack

Attack detection suggests a defence

- It is difficult to evade detection, using large gradients as a proxy, while also keeping adversarial distance small.
- From game theory perspective, knowing this attack detection method does not really help the attacker
- This suggests a defence: not only should the model loss be small, *so should it's gradient*

Adversarial defences: a brief history

- On undefended models, the best attacks (measured by distance to original image) are gradient based
- Lots of early proposed adversarial defence methods appeared to be successful, by “obfuscating” (hiding) the model gradient
- However, later it was shown that gradient-free methods (“black box” attacks, especially Boundary attack) were able to easily circumvent these obfuscation defences
- Thus the adversarial community is skeptical when you claim that “you just need to make the gradients small”
- Lesson: prove your proposed defence on both gradient and gradient-free attacks

Adversarial training

To date, the adversarial defence with the most success is *adversarial training*.

- Train the network on perturbed images. The network should hopefully learn to also recognize these images.
- Formulated as a minmax problem:

$$\min_f \max_d \mathbf{E}[L(x+d)]$$

$$\text{st} \quad \|d\| \leq \varepsilon$$

(I've suppressed writing f and y for brevity)

- But this is even harder to solve than the original problem.

Adversarial training

- Instead, on a per image basis, $\max_d$ L(x+d) st $\|d\| \leq \varepsilon$ is approximated:

  $\max_d L(x+d) \approx \max_d L(x) + d \cdot \nabla L(x)$

- RHS optimality is attained when $d \cdot \nabla L(x) = \varepsilon \|\nabla L(x)\|_*$
- When 2-norm is used, set $d = \varepsilon \nabla L(x) / \|\nabla L(x)\|$
- People also care about the max-norm – arguably it more closely resembles the eye-norm.

  Set $d = \varepsilon$ sign$(\nabla L(x))$

Thus in adversarial training, people solve the problem

$$\min_f \mathbf{E}[L(x+d)]$$

where d is defined as the worst-case perturbation defined above, on a per-image basis

Adversarial training

Problems with adversarial training:

- how big should ε be?
- In practice adversarially robust models require fairly large ε, which harms test accuracy
  - Tsipras et al, “Robustness May Be at Odds with Accuracy”. ICLR 2019
- Maybe instead of approximating $\max_d L(x+d)$ with one gradient step, you should take several steps? How many?
- If you take many inner steps (standard practice is 20), the optimization problem quickly becomes intractable – goes from taking say 12 hours to solve, to a week

Calculus of variations to the rescue

We interpret adversarial training as *Total Variation (TV) regularization*

- Rudin-Osher-Fatemi (1992)
- Want to minimize a functional

$$J[f] = \mathbf{E}[L(f) + \lambda R(f)]$$

- The regularizer R is chosen to enforce desirable properties on f
- For example TV regularization for piecewise smooth functions (recall standard neural nets are piecewise linear!)
- With TV regularization

$$R(f) = \|\nabla f(x)\|$$

Calculus of variations to the rescue

We will also use Lipschitz regularization, used in image in-painting

- Sapiro-Casselles (2000)
- Regularizer is
  $$R(f) = \mathrm{Lip}(f) = \max \|\nabla f(x)\|$$
- Used to fill in missing data
- Equivalent to solving Infinty-Laplace (Oberman 2004)
- Can be shown Lipschitz regularization leads to a proof of model generalization in deep networks (Oberman-Calder 2018)

Adversarial training as TV regularization

- Recall $L(x+\varepsilon d) \approx L(x) + \varepsilon \|\nabla L(x)\|_*$ when d an optimal adversarial perturbation
- Take expectation of RHS, get

$$\mathbf{E}[L(x) + \varepsilon \|\nabla L(x)\|_*]$$

  which is precisely TV regularization of the loss
- TV regularization promotes small gradients on and near the training data, exactly what we need to enforce adversarial robustness
- Note that if d is not optimal, get $L(x+\varepsilon d) \approx L(x) + \varepsilon\, d \cdot \nabla L(x)$ and so taking expectations here, we'd expect random attack vectors to cancel out (if they are mean zero)

Adversarial robustness and the Lipschitz constant

There is work showing that the Lipschitz constant of a model is related to adversarial robustess (Weng et al, 2017)

- Lipschitz constant gives a certifiable minimum adversarial perturbation needed
- It's easy (but not well known in deep learning community) that the Lipschitz constant can be underestimated by simply taking max over available data
    - $\text{Lip}(f) \geq \max \|\nabla f(x)\|$
- Thus, in addition to TV regularization, we propose also training models with Lipschitz regularization:

$$\min_f \mathbf{E}[L(x) + \varepsilon \|\nabla L(x)\|] + \lambda \max \|\nabla L(x)\|$$

How to compute neural net mixed derivatives quickly

To minimize this functional, need to

(1) compute norm gradient of loss with respect to image

(2) Compute mixed 2nd order partials of loss with respect to model parameters

As of 2019 it is not tractable to compute 2nd order partials of a neural network with automatic differentiation, especially during the optimization procedure

- Our solution: compute (1) with finite differences, then use automatic differentiation for (2)
- We observe no significant increase in model training time with this approach

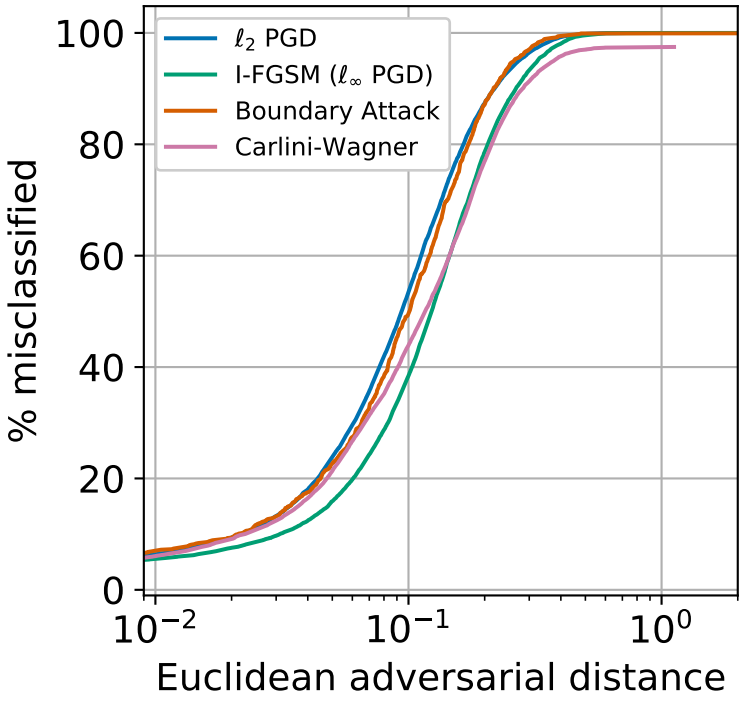
Results

We are able to fine tune the regularization coefficients to achieve comparable adversarial robustness to current state-of-the-art, but *we do not lose test accuracy*

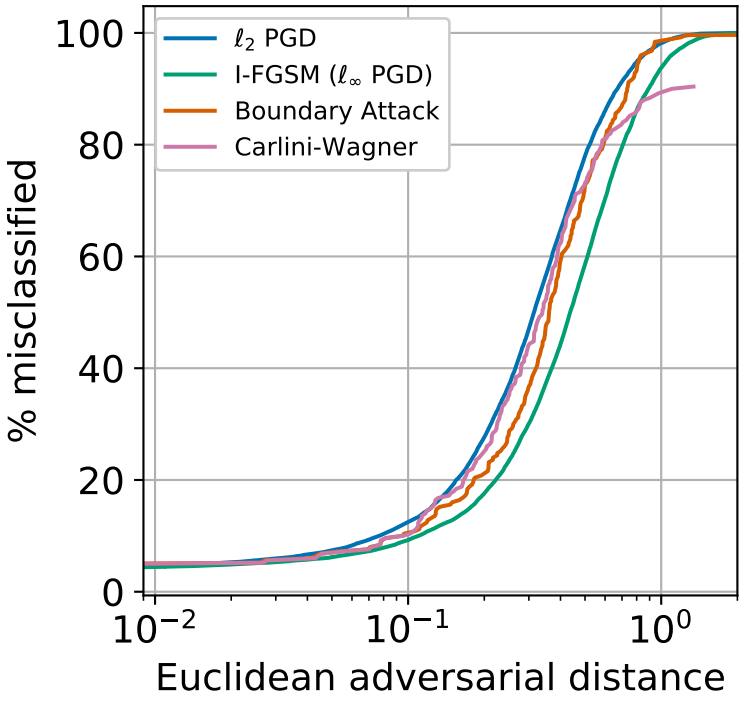
			Defence method				
			ℓ_2 TV + Lipschitz				
Attack		details	$\varepsilon = 0.01$, $\lambda = 0.1$	$\varepsilon = 0.1$, $\lambda = 0.1$	$\varepsilon = 0.1$, $\lambda = 1$	Madry	Qian
test error		undefended model	**4.1**	**4.1**	**4.1**	4.8	5.0
		defended model	**4.1**	5.4	6.0	12.8	22.8
ℓ_2 PGD		$\|\varepsilon\|_2 = 100/255$	59.8	37.2	**36.1**	> 90	-
CW	distance	$\|\varepsilon\|_2 = 1.5$	90.8	91.2	84.4	-	**79.6**
I-FGSM		$\|\varepsilon\|_\infty = 8/255$	98.1	91.6	93.7	**54.2**	-

Table 1. Results on CIFAR-10 dataset. % error reported, lower is better.

Results



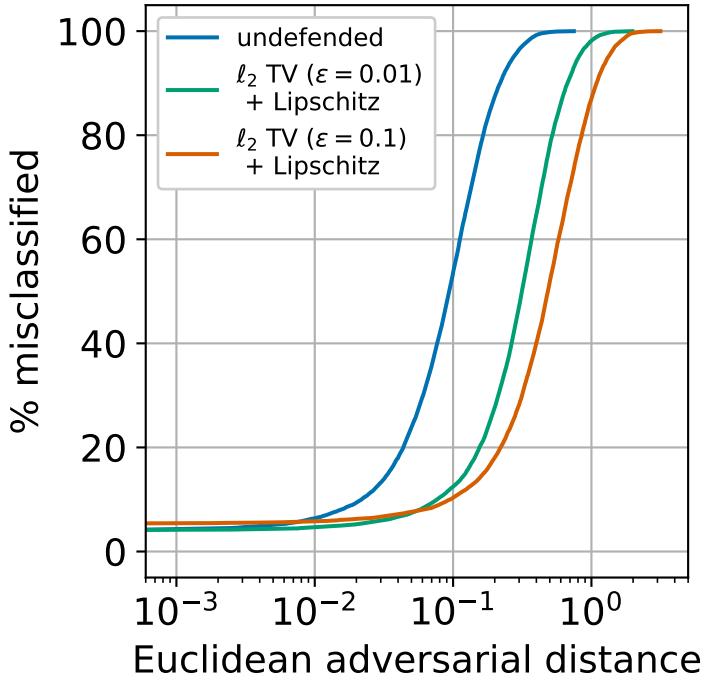
Undefended model



Defended model, TV + Lipschitz regularization

Results

Can see the effect of over regularization



Results

We can also measure the size of the regularization terms, $\mathbf{E}[\|\nabla L(x)\|_*]$ and $\max \|\nabla L(x)\|_*$

- Empirically, small regularization terms correspond to better robustness

Table 2: Adversarial statistics with ResNeXt-34 (2x32) on CIFAR-10 test data.

		adversarial distance 2-norm			adversarial distance ∞-norm		max test statistics		mean test statistics	
defence method	% Err at $\varepsilon = 0$	median distance	median time	% Err at $\varepsilon = 0.1$	median distance	% Err at $\varepsilon = \frac{8}{255}$	$\|\nabla \ell\|_2$	$\|\nabla f\|_{2,\infty}$	$\|\nabla \ell\|_2$	$\|\nabla f\|_{2,\infty}$
undefended	4.07	0.09	159	53.98	2.7e−3	100	85.21	13.70	1.90	0.37
ℓ_1 TV (AT, FGSM)	3.87	0.19	301	23.26	5.6e−3	92.74	35.77	6.27	1.10	0.21
ℓ_2 TV ($\varepsilon = 0.01$)	**3.58**	0.30	471	13.54	9.0e−3	98.34	32.13	5.22	0.59	0.11
ℓ_2 TV ($\varepsilon = 0.01$) + Lipschitz ($\lambda = 0.1$)	4.13	0.31	473	12.52	9.1e−3	98.10	**4.10**	**2.14**	**0.55**	**0.10**
ℓ_2 TV ($\varepsilon = 0.1$) + Lipschitz ($\lambda = 0.1$)	5.37	0.48	659	**10.31**	13.7e−3	**91.6**	31.96	8.93	1.19	0.47
ℓ_2 TV ($\varepsilon = 0.1$) + Lipschitz ($\lambda = 1$)	5.98	**0.52**	**698**	10.95	**14.7e−2**	93.7	18.53	4.87	1.02	0.46

End