Lecture 24: Robustness of Neural Networks

Jacob Steinhardt

Part I: OOD Robustness

Motivation

Folklore: ML does poorly OOD

Why and when? Can we predict it?

Motivation

Folklore: ML does poorly OOD

Why and when? Can we predict it?

Model works

Model does poorly

Geirhos et al., 2018; Ford et al., 2019

How brittle are ML models?

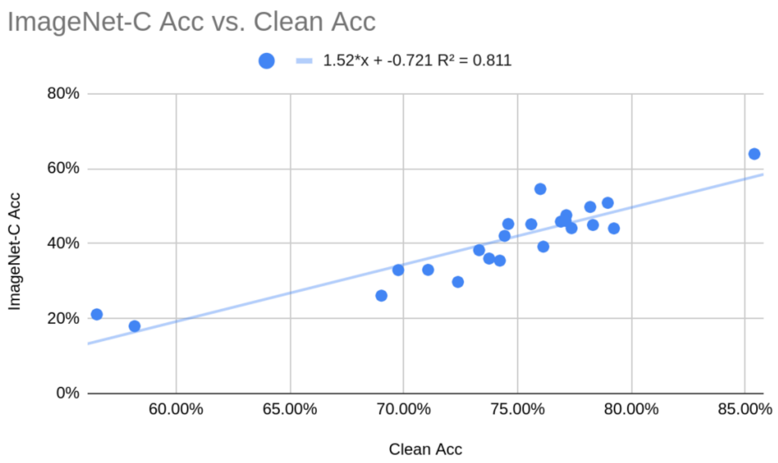
Are we overfitting to IID accuracy?

How brittle are ML models?

Are we overfitting to IID accuracy?

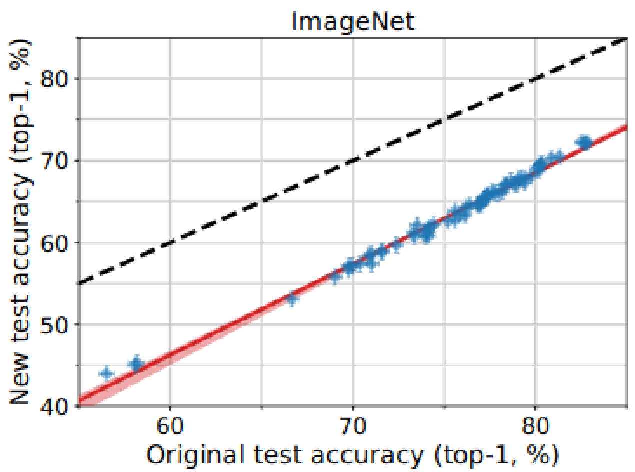
Measurement: plot IID vs. OOD accuracy

ImageNet-C



Hendrycks and Dietterich (2019)

ImageNet-v2



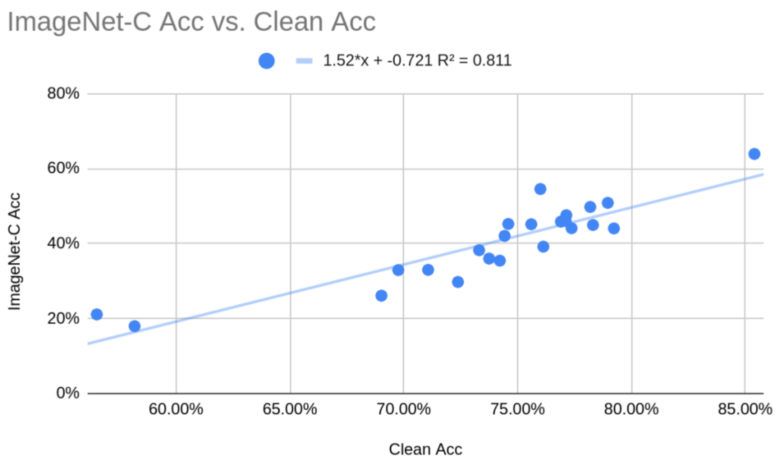
Recht et al. (2019)

How brittle are ML models?

Are we overfitting to IID accuracy?

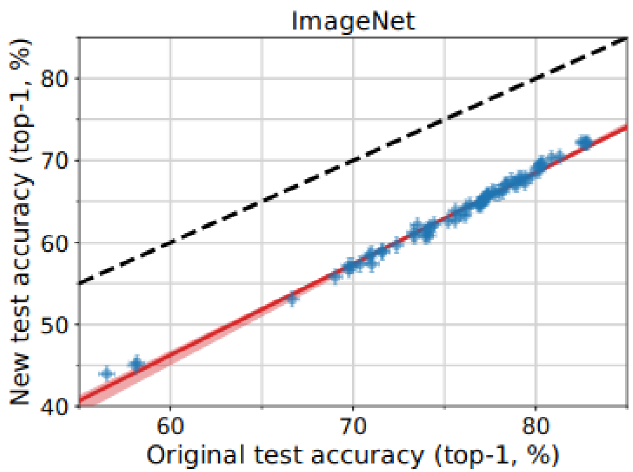
Measurement: plot IID vs. OOD accuracy

ImageNet-C



Hendrycks and Dietterich (2019)

ImageNet-v2



Recht et al. (2019)

Measurement completely **changed the conversation**

- From "Is IID useful at all?" to "Is anything else useful?"

What else helps robustness?

On ImageNet-C, some things seem to help:

- Larger models, data augmentation, self-attention, pre-training

What else helps robustness?

On ImageNet-C, some things seem to help:

- Larger models, data augmentation, self-attention, pre-training

		Larger	Self-Attention	Data Aug		Pre-training
Model	ResNet-50	ResNet-152	SE_ResNet-152	SIN-trained	AugMix	WSL
Orig.	76.1	78.3	78.7	74.6	77.6	85.4
IN-C	41.6	47.8 (+6.2)	50.9 (+9.3)	47.9 (+6.3)	48.3 (+6.7)	**65.5 (+23.9)**
Trend		+4.7	+5.5	-3.2	+3.2	+19.7

What else helps robustness?

On ImageNet-C, some things seem to help:

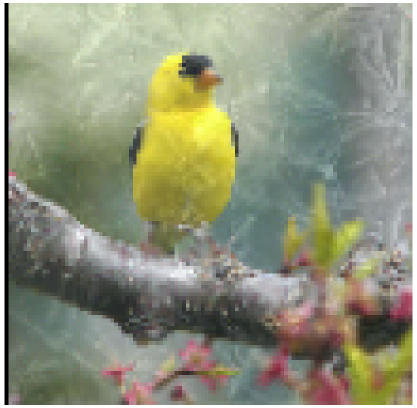
- Larger models, data augmentation, self-attention, pre-training

		Larger	Self-Attention	Data Aug		Pre-training
Model	ResNet-50	ResNet-152	SE_ResNet-152	SIN-trained	AugMix	WSL
Orig.	76.1	78.3	78.7	74.6	77.6	85.4
IN-C	41.6	47.8 (+6.2)	50.9 (+9.3)	47.9 (+6.3)	48.3 (+6.7)	**65.5 (+23.9)**
Trend		+4.7	+5.5	-3.2	+3.2	+19.7

Do these really help? Many types of shift...

The Sieve of Variability

Original



ImageNet-C

ImageNet-A

ImageNet-v2

ImageNet-R

Hendrycks and Dietterich, 2019; Hendrycks et al., 2019, 2020; Recht et al., 2019

Applying the Sieve

Hypotheses: larger models, data aug, self-attn, pre-training

Applying the Sieve

Hypotheses: larger models, data aug, self-attn, pre-training

Do they hold up on other datasets?

- ImageNet-A: yes
- ImageNet-v2: unclear
- ImageNet-R: yes, except self-attention

Applying the Sieve

Hypotheses: larger models, data aug, self-attn, pre-training

Do they hold up on other datasets?

- ImageNet-A: yes
- ImageNet-v2: unclear
- ImageNet-R: yes, except self-attention

		Larger	Self-Attention	Data Aug		Pre-training
Model	ResNet-50	ResNet-152	SE_ResNet-152	SIN-trained	AugMix	WSL
Orig.	76.1	78.3	78.7	74.6	77.6	85.4
IN-C	41.6	47.8	50.9	47.9	48.3	65.5
IN-R	36.1	41.3	**40.0**	41.5	41.1	**75.8**

"The Many Faces of Robustness"
Hendrycks et al. (2020)

Revisiting Pre-training

Two distributions:

- WSL: 1B instagram images (1000x data)
- ImageNet-21K: additional ImageNet classes (10x data)

"The Many Faces of Robustness"
Hendrycks et al. (2020)

Revisiting Pre-training

Two distributions:

- WSL: 1B instagram images (1000x data)
- ImageNet-21K: additional ImageNet classes (10x data)

Model	ResNet-50	WSL	**IN-21k**
IN-R	36.1	75.8	**37.2**
IN-A	2.2	45.4	**11.4**

May be about overlap rather than amount of data

"The Many Faces of Robustness"
Hendrycks et al. (2020)

Tightening the Sieve

- DFR: size, occlusion, viewpoint, zoom
- SVSF: hardware, year, location

"The Many Faces of Robustness"
Hendrycks et al. (2020)

Tightening the Sieve

			Size		Occlusion		Viewpoint		Zoom	
Network	IID	OOD	Small	Large	Slight/None	Heavy	No Wear	Side/Back	Medium	Large
ResNet-50	77.6	55.1	39.4	73.0	51.5	41.2	50.5	63.2	48.7	73.3
+ ImageNet-21K *Pretraining*	80.8	58.3	40.0	73.6	55.2	43.0	63.0	67.3	50.5	73.9
+ SE (*Self-Attention*)	77.4	55.3	38.9	72.7	52.1	40.9	52.9	64.2	47.8	72.8
+ Random Erasure	78.9	56.4	39.9	75.0	52.5	42.6	53.4	66.0	48.8	73.4
+ Speckle Noise	78.9	55.8	38.4	74.0	52.6	40.8	55.7	63.8	47.8	73.6
+ Style Transfer	80.2	57.1	37.6	76.5	54.6	43.2	58.4	65.1	49.2	72.5
+ DeepAugment	79.7	56.3	38.3	74.5	52.6	42.8	54.6	65.5	49.5	72.7
+ AugMix	80.4	57.3	39.4	74.8	55.3	42.8	57.3	66.6	49.0	73.1
ResNet-152 (*Larger Models*)	80.0	57.1	40.0	75.6	52.3	42.0	57.7	65.6	48.9	74.4

Similar results for SVSF (but smaller gap)

"The Many Faces of Robustness"
Hendrycks et al. (2020)

Does anything help?

Previous shifts had **style component** (also **harder**)

Data augmentation highly successful on these, but not for geography, year, etc.

Does anything help?

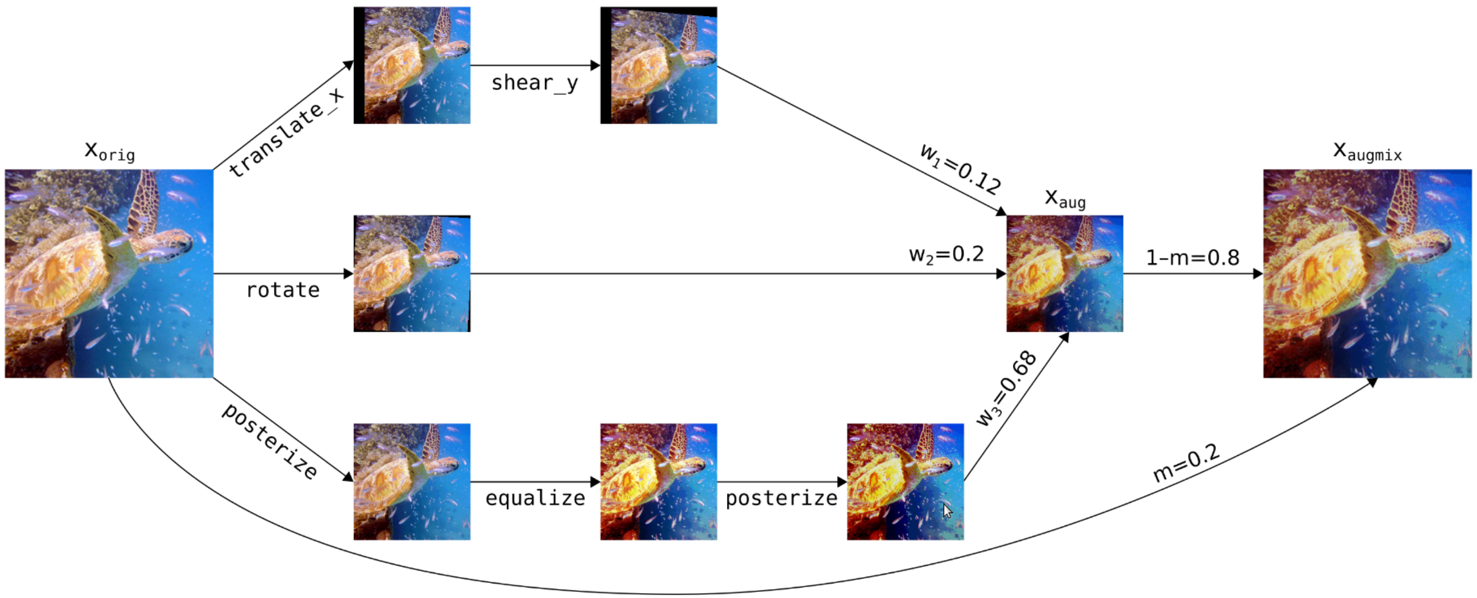
Previous shifts had **style component** (also **harder**)

Data augmentation highly successful on these, but not for geography, year, etc.

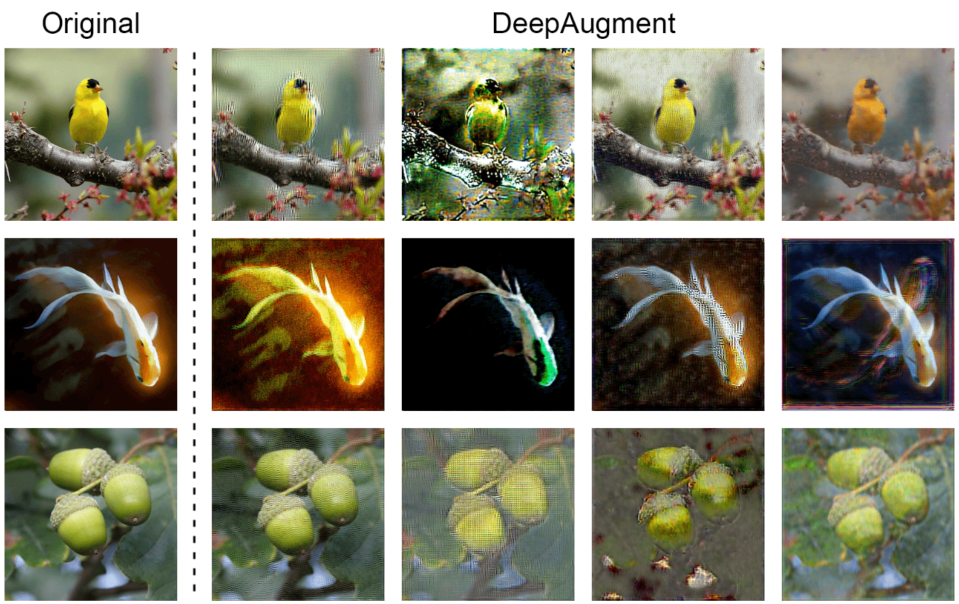
Robustness is **multivariate**:

- Correlated, but multiple directions of variaton
- Need more datasets measuring new shifts
- Need new methods

Data Augmentation



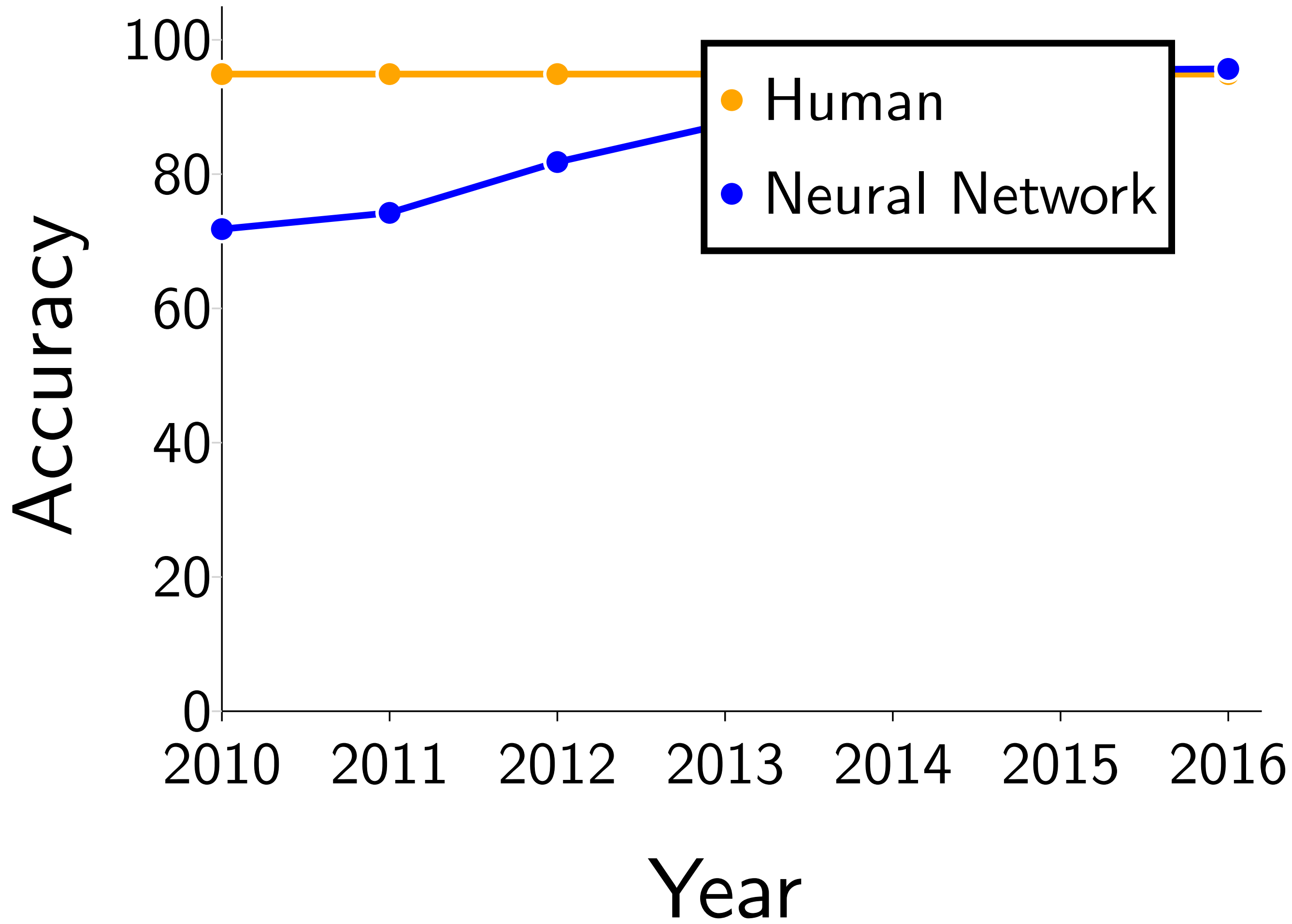
Data Augmentation



Part II: Adversarial Robustness

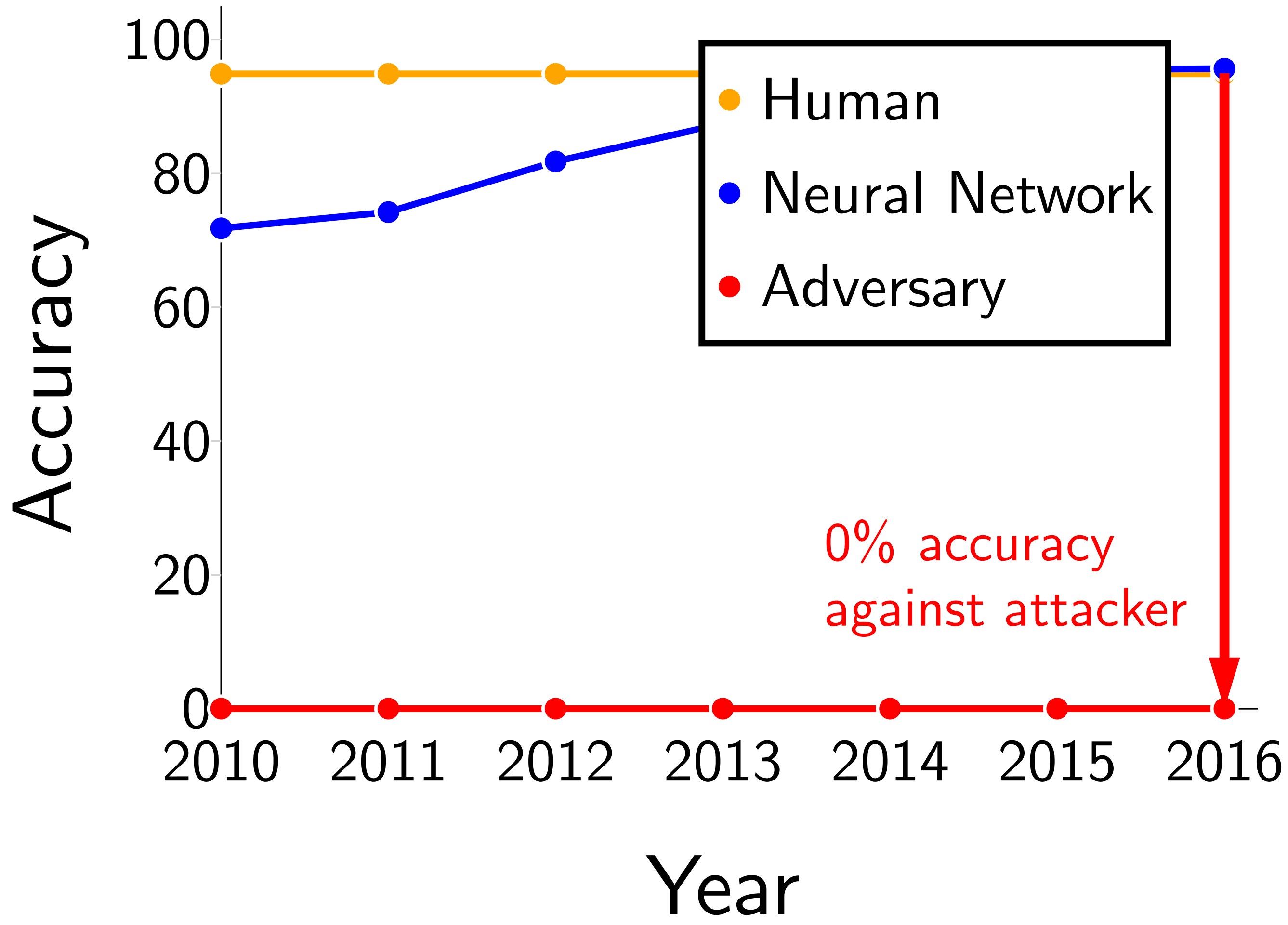
ML: Powerful But Fragile

ML is state-of-the-art in many domains, such as vision:

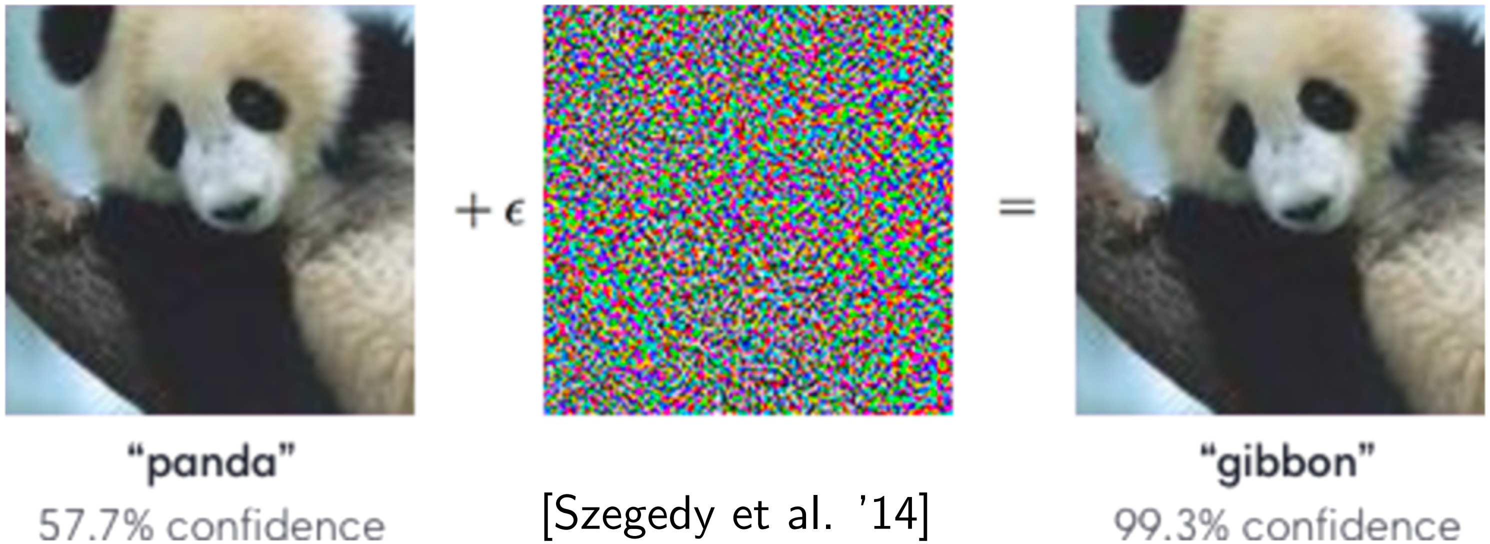


ML: Powerful But Fragile

ML is state-of-the-art in many domains, such as vision:



Machine Learning is Insecure

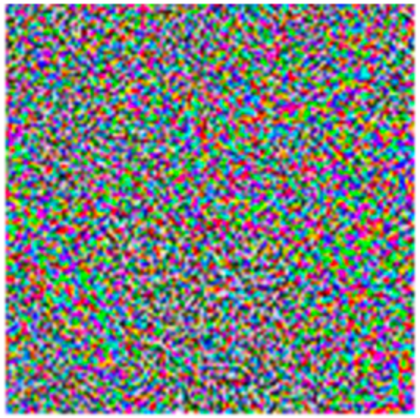


[Szegedy et al. '14]

Machine Learning is Insecure

"panda"
57.7% confidence

$+\epsilon$



[Szegedy et al. '14]

$=$

"gibbon"
99.3% confidence

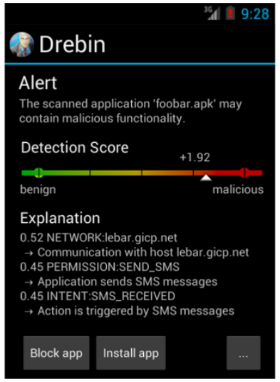
Self-driving cars:

stop $\rightarrow$ yield
[Evtimov et al. '17]

Speech recognition:

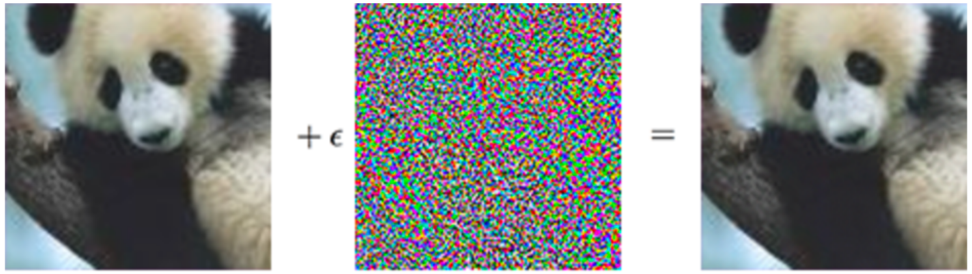
noise $\rightarrow$ "Ok Google"
[Carlini et al. '16]

Malware:



malware $\rightarrow$ benign
[Grosse et al. '16]

Arms Races

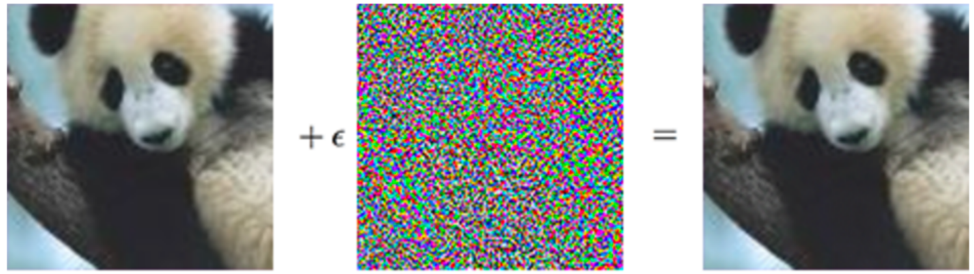


Naive evaluation against attacks insufficient:

$\boxed{\text{discovery [Szegedy et al. '14]}}$

discovery
[Szegedy et al. '14]

Arms Races



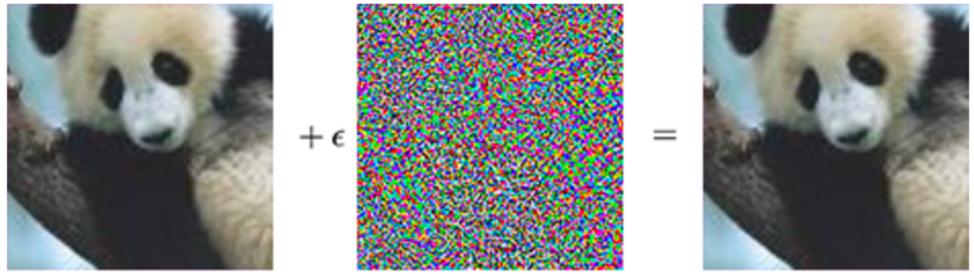
Naive evaluation against attacks insufficient:

discovery
[Szegedy et al. '14]

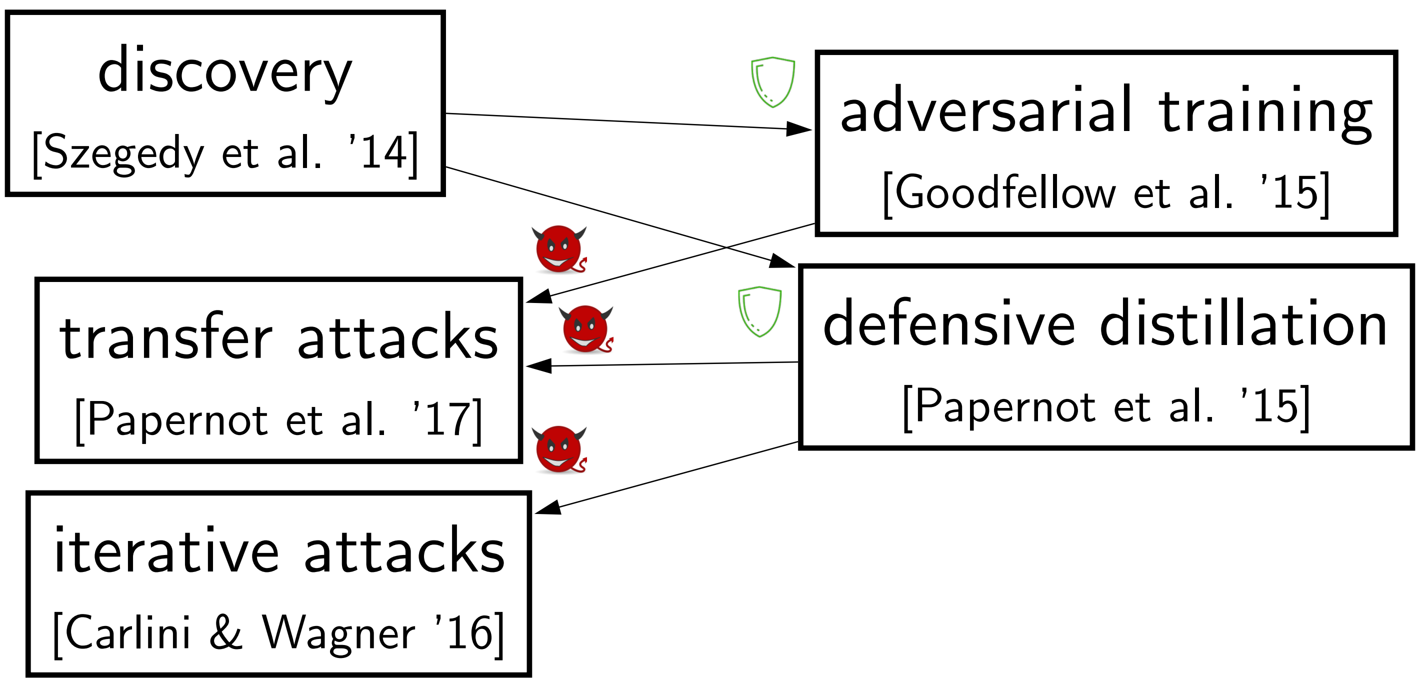
adversarial training
[Goodfellow et al. '15]

defensive distillation
[Papernot et al. '15]

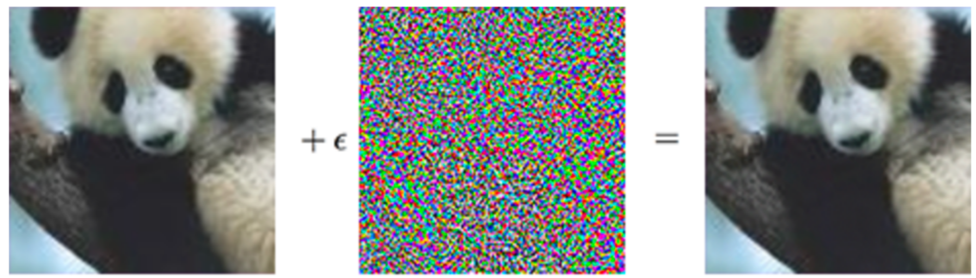
Arms Races



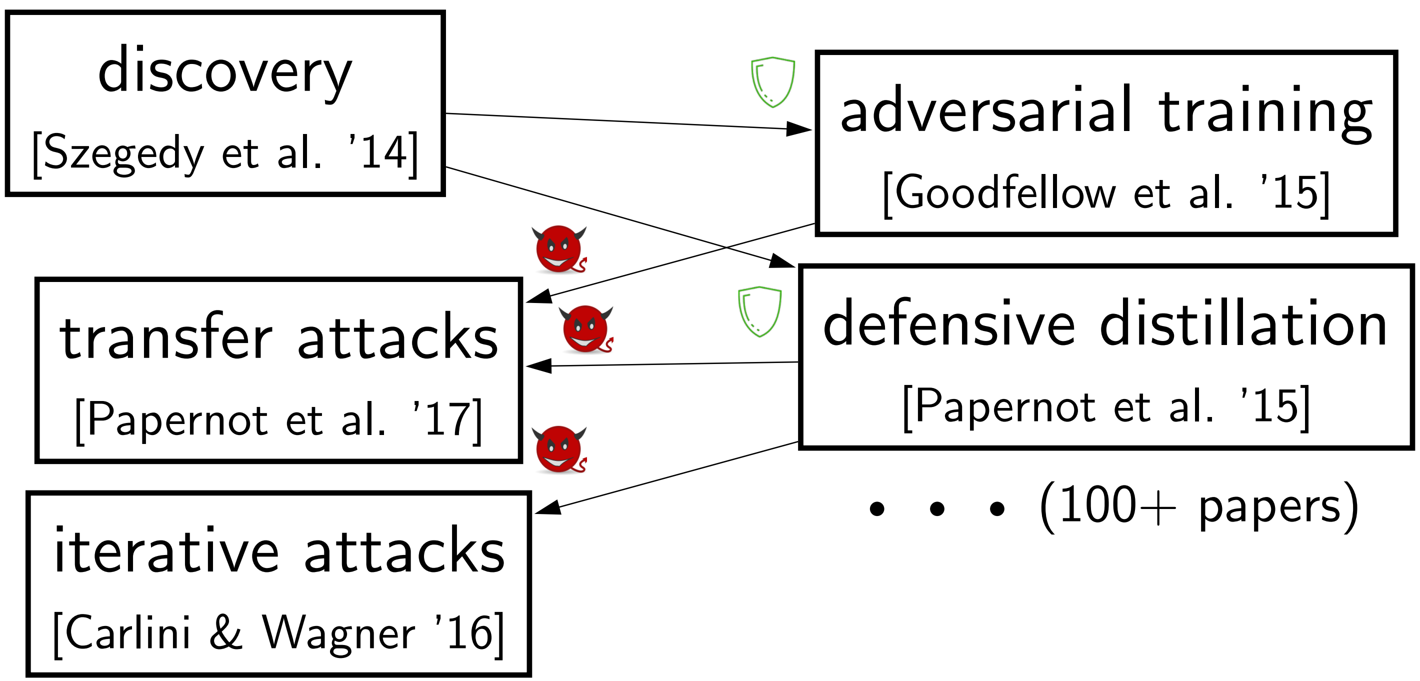
Naive evaluation against attacks insufficient:



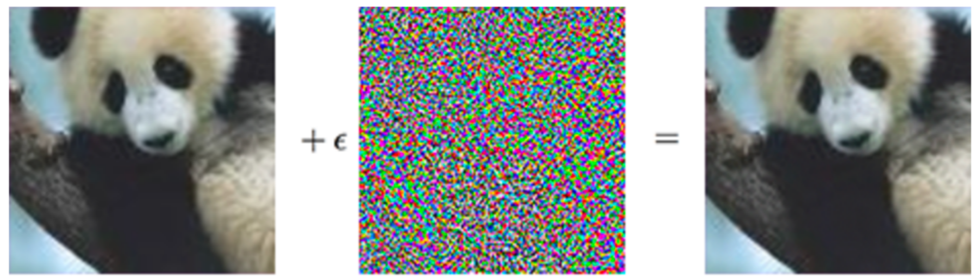
Arms Races



Naive evaluation against attacks insufficient:



Arms Races



Naive evaluation against attacks insufficient:

discovery
[Szegedy et al. '14]

adversarial training
[Goodfellow et al. '15]

transfer attacks
[Papernot et al. '17]

defensive distillation
[Papernot et al. '15]

iterative attacks
[Carlini & Wagner '16]

• • • (100+ papers)

bypassing 10 defenses
[Carlini & Wagner '17]

Take-away

Relying on naive evaluation leads to a **security arms race** that defenders often lose!

Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples

Anish Athalye [*1] **Nicholas Carlini** [*2] **David Wagner** [2]

Abstract

We identify obfuscated gradients, a kind of gradient masking, as a phenomenon that leads to a false sense of security in defenses against adversarial examples. While defenses that cause obfuscated gradients appear to defeat iterative optimization-based attacks, we find defenses relying on this effect can be circumvented. We describe characteristic behaviors of defenses exhibiting the effect, and for each of the three types of obfuscated gradients we discover, we develop attack techniques to overcome it. In a case study, examining non-certified white-box-secure defenses at ICLR 2018, we find obfuscated gradients are a common occurrence, with 7 of 9 defenses relying on obfuscated gradients. Our new attacks successfully circumvent 6 completely, and 1 partially, in the original threat model each paper considers.

apparent robustness against iterative optimization attacks: *obfuscated gradients*, a term we define as a special case of gradient masking (Papernot et al., 2017). Without a good gradient, where following the gradient does not successfully optimize the loss, iterative optimization-based methods cannot succeed. We identify three types of obfuscated gradients: *shattered gradients* are nonexistent or incorrect gradients caused either intentionally through non-differentiable operations or unintentionally through numerical instability; *stochastic gradients* depend on test-time randomness; and *vanishing/exploding gradients* in very deep computation result in an unusable gradient.

We propose new techniques to overcome obfuscated gradients caused by these three phenomena. We address gradient shattering with a new attack technique we call Backward Pass Differentiable Approximation, where we approximate derivatives by computing the forward pass normally and computing the backward pass using a differentiable approximation of the function. We compute gradients of random-

3.1. Identifying Obfuscated & Masked Gradients

Some defenses intentionally break gradient descent and cause obfuscated gradients. However, others defenses *unintentionally* break gradient descent, but the cause of gradient descent being broken is a direct result of the design of the neural network. We discuss below characteristic behaviors of defenses which cause this to occur. These behaviors may not perfectly characterize all cases of masked gradients.

One-step attacks perform better than iterative attacks. Iterative optimization-based attacks applied in a white-box setting are strictly stronger than single-step attacks and should give strictly superior performance. If single-step methods give performance superior to iterative methods, it is likely that the iterative attack is becoming stuck in its optimization search at a local minimum.

Black-box attacks are better than white-box attacks. The black-box threat model is a strict subset of the white-box threat model, so attacks in the white-box setting should perform better; if a defense is obfuscating gradients, then black-box attacks (which do not use the gradient) often perform better than white-box attacks (Papernot et al., 2017).

Unbounded attacks do not reach 100% success. With unbounded distortion, any classifier should have 0% robustness to attack. If an attack does not reach 100% success with sufficiently large distortion bound, this indicates the attack is not performing optimally against the defense, and the attack should be improved.

Random sampling finds adversarial examples. Brute-force random search (e.g., randomly sampling 10^5 or more points) within some ϵ-ball should not find adversarial examples when gradient-based attacks do not.

Increasing distortion bound does not increase success. A larger distortion bound should monotonically increase attack success rate; significantly increasing distortion bound should result in significantly higher attack success rate.

On Evaluating Adversarial Robustness

Nicholas Carlini[1], Anish Athalye[2], Nicolas Papernot[1], Wieland Brendel[3], Jonas Rauber[3], Dimitris Tsipras[2], Ian Goodfellow[1], Aleksander Mądry[2], Alexey Kurakin[1] *

[1] Google Brain [2] MIT [3] University of Tübingen

* List of authors is dynamic and subject to change. Authors are ordered according to the amount of their contribution to the text of the paper.

Please direct correspondence to the GitHub repository
https://github.com/evaluating-adversarial-robustness/adv-eval-paper

Last Update: 18 February, 2019.

3.1 Common Severe Flaws

There are several common severe evaluation flaws which have the potential to completely invalidate any robustness claims. Any evaluation which contains errors on any of the following items is likely to have fundamental and irredeemable flaws. Evaluations which intentionally deviate from the advice here may wish to justify the decision to do so.

- §3 **Do not mindlessly follow this list**; make sure to still think about the evaluation.
- §2.2 **State a precise threat model** that the defense is supposed to be effective under.
  - The threat model assumes the attacker knows how the defense works.
  - The threat model states attacker's goals, knowledge and capabilities.
  - For security-justified defenses, the threat model realistically models some adversary.
  - For worst-case randomized defenses, the threat model captures the perturbation space.
  - Think carefully and justify any ℓ_p bounds placed on the adversary.
- §2.5 Perform **adaptive attacks** to give an upper bound of robustness.
  - The attacks are given access to the full defense, end-to-end.
  - The loss function is changed as appropriate to cause misclassification.
  - §4.3 **Focus on the strongest attacks** for the threat model and defense considered.
- §2.6 Release **pre-trained models and source code**.
  - Include a clear installation guide, including all dependencies.
  - There is a one-line script which will classify an input example with the defense.
- §4.2 Report **clean model accuracy** when not under attack.
  - For defenses that abstain or reject inputs, generate a ROC curve.
- §5.2 Perform **basic sanity tests** on attack success rates.
  - Verify iterative attacks perform better than single-step attacks.
  - Verify increasing the perturbation budget strictly increases attack success rate.
  - With "high" distortion, model accuracy should reach levels of random guessing.
- §5.3 Generate an **attack success rate vs. perturbation budget** curve.
  - Verify the x-axis extends so that attacks eventually reach 100% success.
  - For unbounded attacks, report distortion and not success rate.
- §5.4 Verify **adaptive attacks** perform better than any other.
  - Compare success rate on a per-example basis, rather than averaged across the dataset.
  - Evaluate against some combination of black-box, transfer, and random-noise attacks.
- §5.7 Describe the **attacks applied**, including all hyperparameters.

3.2 Common Pitfalls

There are other common pitfalls that may prevent the detection of ineffective defenses. This list contains some potential pitfalls which do not apply to large categories of defenses. However, if applicable, the items below are still important to carefully check they have been applied correctly.

- §4.3 Apply a **diverse set of attacks** (especially when training on one attack approach).
  - Do not blindly apply multiple (nearly-identical) attack approaches.

3.1 Common Severe Flaws

There are several common severe evaluation flaws which have the potential to completely invalidate any robustness claims. Any evaluation which contains errors on any of the following items is likely to have fundamental and irredeemable flaws. Evaluations which intentionally deviate from the advice here may wish to justify the decision to do so.

- §3 **Do not mindlessly follow this list**; make sure to still think about the evaluation.
- §2.2 **State a precise threat model** that the defense is supposed to be effective under.
  - The threat model assumes the attacker knows how the defense works.
  - The threat model states attacker's goals, knowledge and capabilities.
  - For security-justified defenses, the threat model realistically models some adversary.
  - For worst-case randomized defenses, the threat model captures the perturbation space.
  - Think carefully and justify any ℓ_p bounds placed on the adversary.
- §2.5 Perform **adaptive attacks** to give an upper bound of robustness.
  - The attacks are given access to the full defense, end-to-end.
  - The loss function is changed as appropriate to cause misclassification.
  - §4.3 **Focus on the strongest attacks** for the threat model and defense considered.
- §2.6 Release **pre-trained models and source code**.
  - Include a clear installation guide, including all dependencies.
  - There is a one-line script which will classify an input example with the defense.
- §4.2 Report **clean model accuracy** when not under attack.
  - For defenses that abstain or reject inputs, generate a ROC curve.
- §5.2 Perform **basic sanity tests** on attack success rates.
  - Verify iterative attacks perform better than single-step attacks.
  - Verify increasing the perturbation budget strictly increases attack success rate.
  - With "high" distortion, model accuracy should reach levels of random guessing.
- §5.3 Generate an **attack success rate vs. perturbation budget** curve.
  - Verify the x-axis extends so that attacks eventually reach 100% success.
  - For unbounded attacks, report distortion and not success rate.
- §5.4 Verify **adaptive attacks** perform better than any other.
  - Compare success rate on a per-example basis, rather than averaged across the dataset.
  - Evaluate against some combination of black-box, transfer, and random-noise attacks.
- §5.7 Describe the **attacks applied**, including all hyperparameters.

3.2 Common Pitfalls

There are other common pitfalls that may prevent the detection of ineffective defenses. This list contains some potential pitfalls which do not apply to large categories of defenses. However, if applicable, the items below are still important to carefully check they have been applied correctly.

- §4.3 Apply a **diverse set of attacks** (especially when training on one attack approach).
  - Do not blindly apply multiple (nearly-identical) attack approaches.
  - Check that the gradient-free attacks succeed less often than gradient-based attacks.
  - Carefully investigate attack hyperparameters that affect success rate.
- §4.5 Perform a **transferability attack** using a similar substitute model.
  - Select a substitute model as similar to the defended model as possible.
  - Generate adversarial examples that are initially assigned high confidence.
  - Check that the transfer attack succeeds less often than white-box attacks.
- §4.6 For randomized defenses, properly **ensemble over randomness**.
  - Verify that attacks succeed if randomness is assigned to one fixed value.
  - State any assumptions about adversary knowledge of randomness in the threat model.
- §4.7 For non-differentiable components, **apply differentiable techniques**.
  - Discuss why non-differentiable components were necessary.
  - Verify attacks succeed on undefended model with those non-differentiable components.
  - Consider applying BPDA (Athalye et al., 2018) if applicable.
- §4.8 Verify that the **attacks have converged** under the selected hyperparameters.
  - Verify that doubling the number of iterations does not increase attack success rate.
  - Plot attack effectiveness versus the number of iterations.
  - Explore different choices of the step size or other attack hyperparameters.
- §4.9 Carefully **investigate attack hyperparameters** and report those selected.
  - Start search for adversarial examples at a random offset.
  - Investigate if attack results are sensitive to any other hyperparameters.
- §5.1 **Compare against prior work** and explain important differences.
  - When contradicting prior work, clearly explain why differences occur.
  - Attempt attacks that are similar to those that defeated previous similar defenses.
  - When comparing against prior work, ensure it has not been broken.
- §4.10 Test **broader threat models** when proposing general defenses. For images:
  - Apply rotations and translations (Engstrom et al., 2017).
  - Apply common corruptions and perturbations (Hendrycks & Dietterich, 2018).
  - Add Gaussian noise of increasingly large standard deviation (Ford et al., 2019).

3.3 Special-Case Pitfalls

The following items apply to a smaller fraction of evaluations. Items presented here are included because while they may diagnose flaws in some defense evaluations, they are not necessary for many others. In other cases, the tests presented here help provide additional evidence that the evaluation was performed correctly.

- §4.1 Investigate if it is possible to use **provable approaches**.
  - Examine if the model is amenable to provable robustness lower-bounds.
- §4.11 **Attack with random noise** of the correct norm.
  - For each example, try 10,000+ different choices of random noise.
  - Check that the random attacks succeed less-often than white-box attacks.
- §4.12 Use both **targeted and untargeted attacks** during evaluation.
  - State explicitly which attack type is being used

Adversarial Examples are Persistent

Persist despite hundreds of papers trying to avoid them

Adversarial Examples are Persistent

Persist despite hundreds of papers trying to avoid them

stop $\rightarrow$ yield

[Evtimov et al. '17]

turtle $\rightarrow$ rifle

[Athalye et al. '17]

banana $\rightarrow$ toaster

[Brown et al. '17]

Adversarial Examples are Persistent

Persist despite hundreds of papers trying to avoid them

stop $\rightarrow$ yield

[Evtimov et al. '17]

turtle $\rightarrow$ rifle

[Athalye et al. '17]

banana $\rightarrow$ toaster

[Brown et al. '17]

Most defenses fail within weeks **(arms race)**, but a few have lasted.

Adversarial Examples are Persistent

Persist despite hundreds of papers trying to avoid them

stop $\rightarrow$ yield
[Evtimov et al. '17]

turtle $\rightarrow$ rifle
[Athalye et al. '17]

banana $\rightarrow$ toaster
[Brown et al. '17]

Most defenses fail within weeks **(arms race)**, but a few have lasted.

What makes them different?

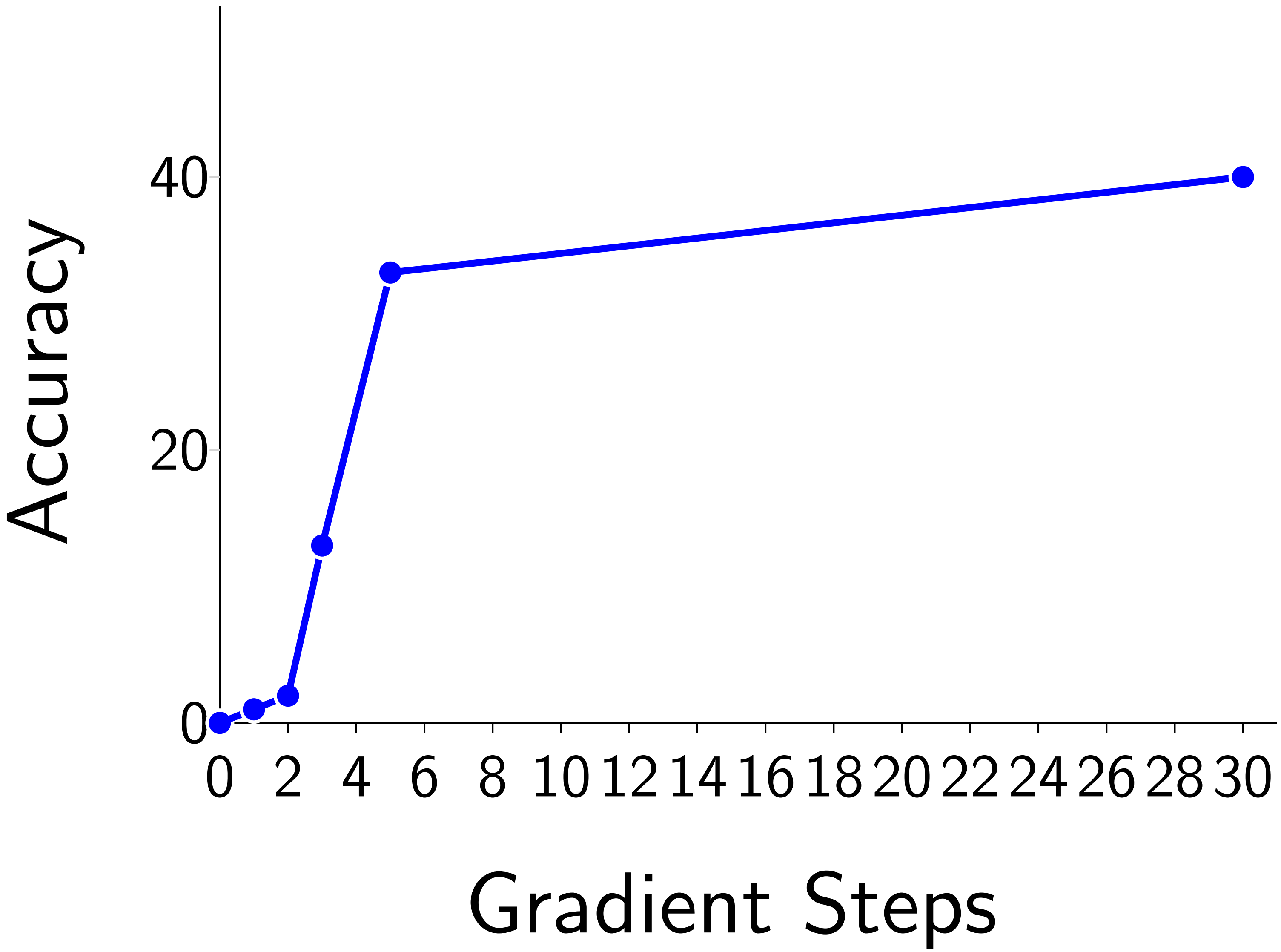
Details of the robust model

Obtained via **adversarial training** (train on adversarial images)

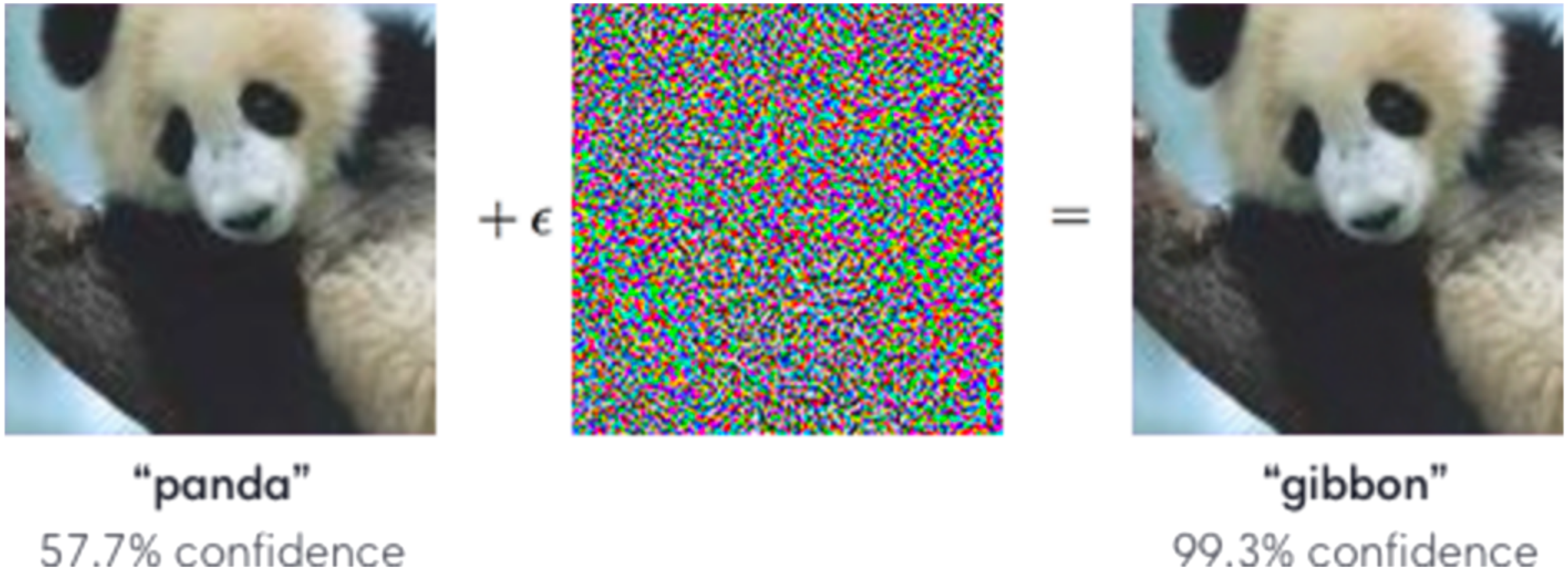
Generate training images via **gradient ascent** on cross-entropy loss

If too few gradient steps, model learns to **fool optimizer** instead of being truly robust

Accuracy vs. gradient steps



Threat Model Overfitting

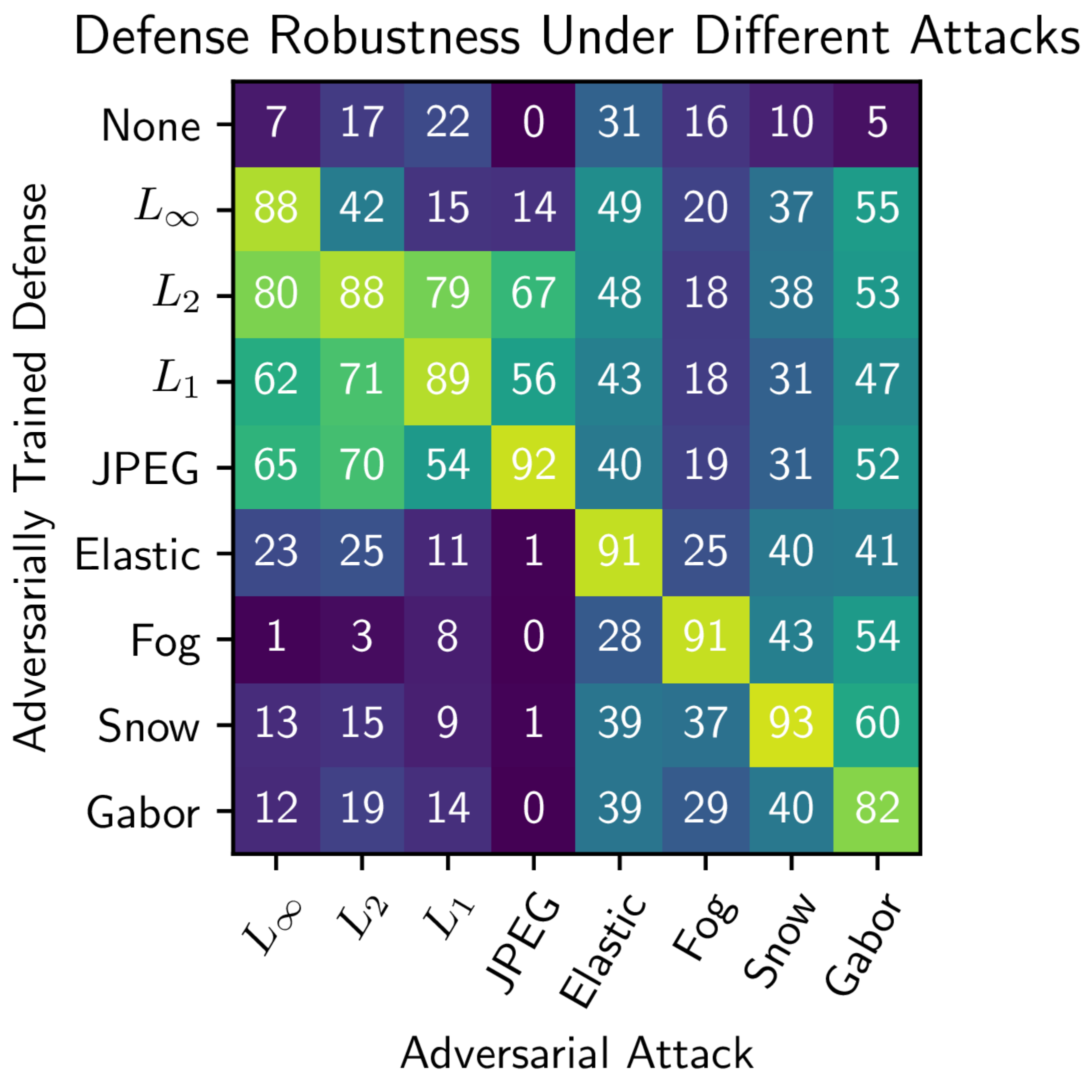


ℓ_∞-norm

L1 perturbations

Elastic perturbations

Evaluating Against Many Adversaries



Evaluating Against Many Adversaries

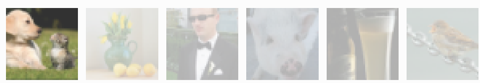
	Clean Accuracy	L_∞	L_2	L_1	Elastic	JPEG	Fog	Snow	Gabor	**mUAR**
SqueezeNet	84.1	5.2	11.2	14.9	25.9	**1.9**	20.1	9.8	4.4	12.8
ResNeXt-101 (32×8d)	95.9	2.5	5.5	20.7	26.5	1.8	14.1	12.4	5.3	13.4
ResNeXt-101 (32×8d) + WSL	**97.1**	3.0	5.7	28.3	29.4	**1.9**	26.2	20.3	8.0	19.0
ResNet-18	91.6	2.7	8.2	13.5	22.6	1.8	20.3	9.5	4.2	12.0
ResNet-50	94.2	2.7	6.6	20.1	24.9	1.8	15.8	11.9	4.9	13.2
ResNet-50 + Stylized ImageNet	94.6	2.9	7.4	22.8	26.0	1.8	16.2	12.5	8.1	14.6
ResNet-50 + Patch Gaussian	93.6	4.5	10.9	27.4	28.2	1.8	23.9	10.5	5.2	16.2
ResNet-50 + AugMix	95.1	**6.1**	**13.4**	**34.3**	**38.8**	1.8	**28.6**	**24.7**	**11.1**	**23.2**

Visualizing Robust Networks (Lucid)

The Building Blocks of Interpretability

Interpretability techniques are normally studied in isolation.
We explore the powerful interfaces that arise when you combine them — and the rich structure of this combinatorial space.

CHOOSE AN INPUT IMAGE



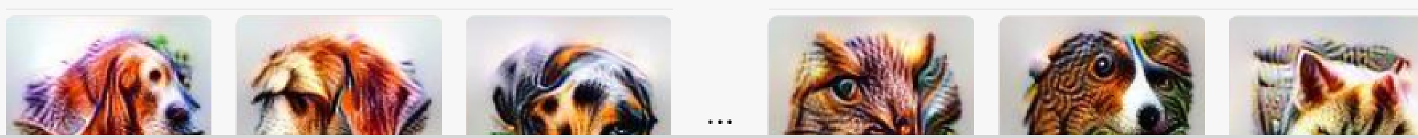
For instance, by combining feature visualization (*what is a neuron looking for?*) with attribution (*how does it affect the output?*), we can explore how the network decides between labels like **Labrador retriever** and **tiger cat**.

Several floppy ear detectors seem to be important when distinguishing dogs, whereas pointy ears are used to classify "tiger cat".

CHANNELS THAT MOST SUPPORT ...

LABRADOR RETRIEVER ▾ ⟷ TIGER CAT ▾

feature visualization of channel



Visualizing Robust Networks (Lucid)

Lucid

status alpha | build passing | coverage 82% | python 2.7 | 3.6 | pypi v0.3.8

Lucid is a collection of infrastructure and tools for research in neural network interpretability.

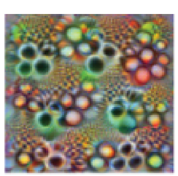
- **Notebooks** -- Get started without any setup!
- **Reading** -- Learn more about visualizing neural nets.
- **Community** -- Want to get involved? Please reach out!
- **Additional Information** -- Licensing, code style, etc.
- **Start Doing Research!** -- Want to get involved? We're trying to research openly!

Notebooks

Start visualizing neural networks ***with no setup***. The following notebooks run right from your browser, thanks to Colaboratory. It's a Jupyter notebook environment that requires no setup to use and runs entirely in the cloud.

You can run the notebooks on your local machine, too. Clone the repository and find them in the `notebooks` subfolder. You will need to run a local instance of the Jupyter notebook environment to execute them.

Tutorial Notebooks



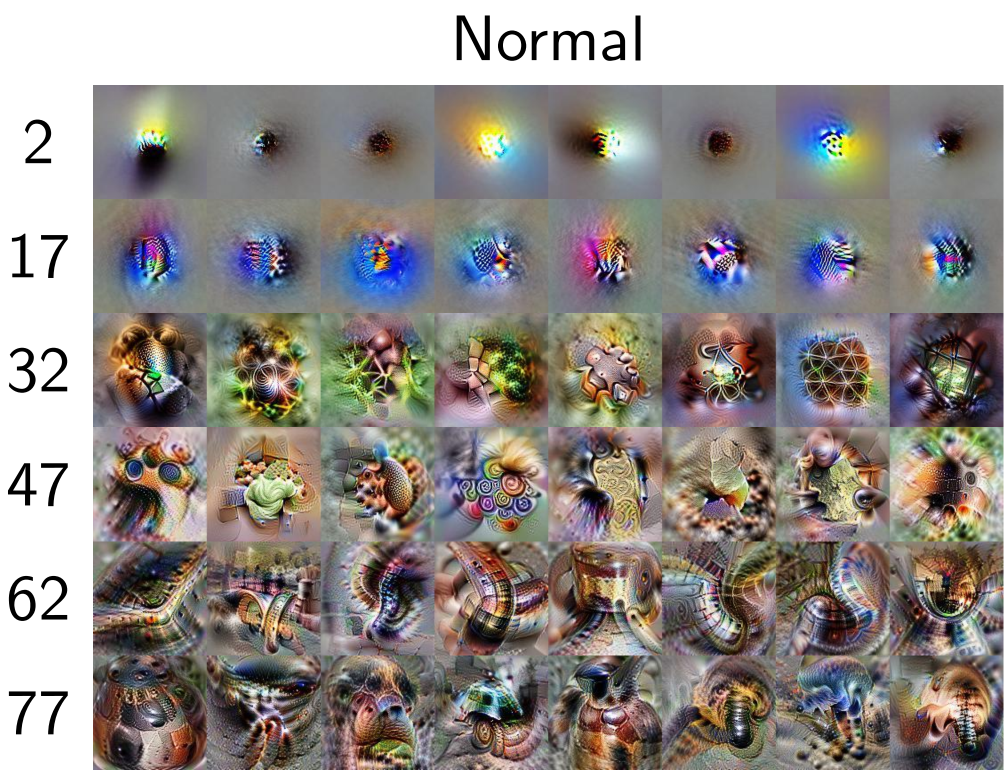
Lucid Tutorial [colab]

Quickly get started using Lucid. Become familiar with changing **objectives**, **transformations**, and **paramaterization**.

Modelzoo Introduction [colab]

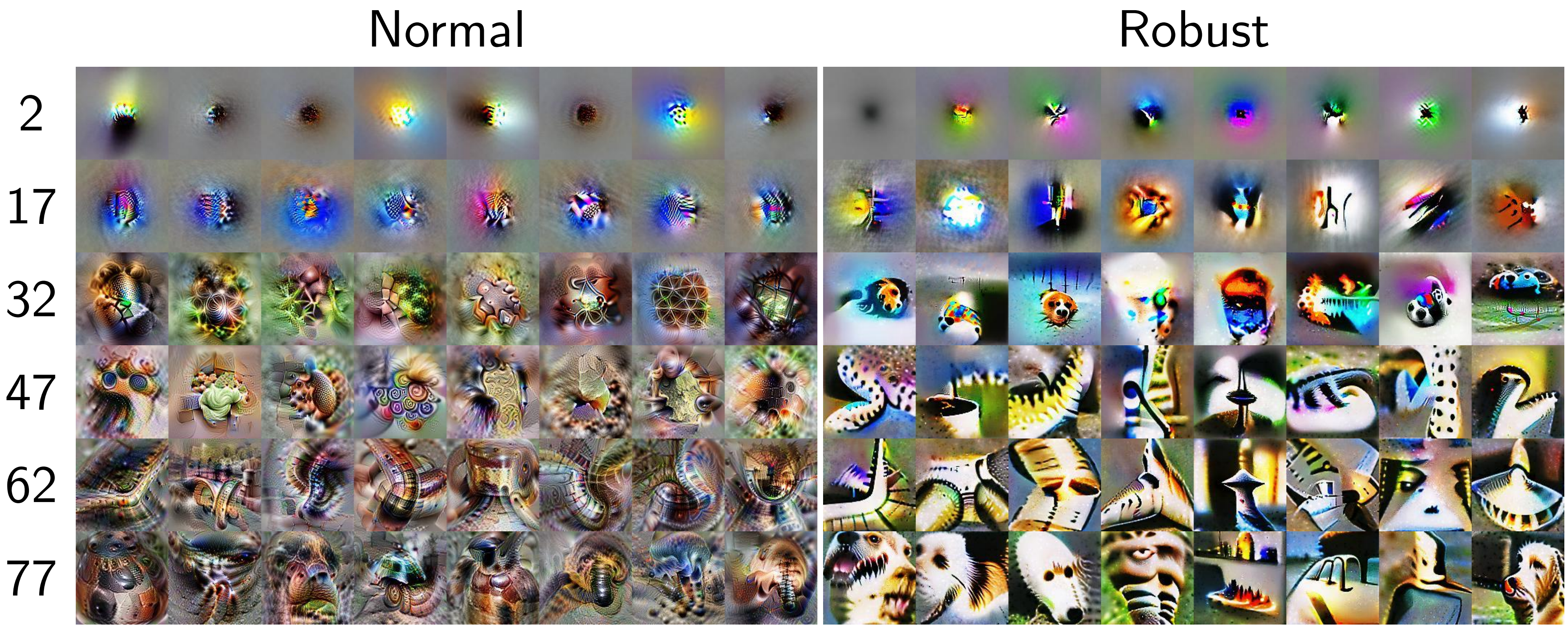
Visualizing Robust Networks

Visualization: find images that maximally excite different neurons.



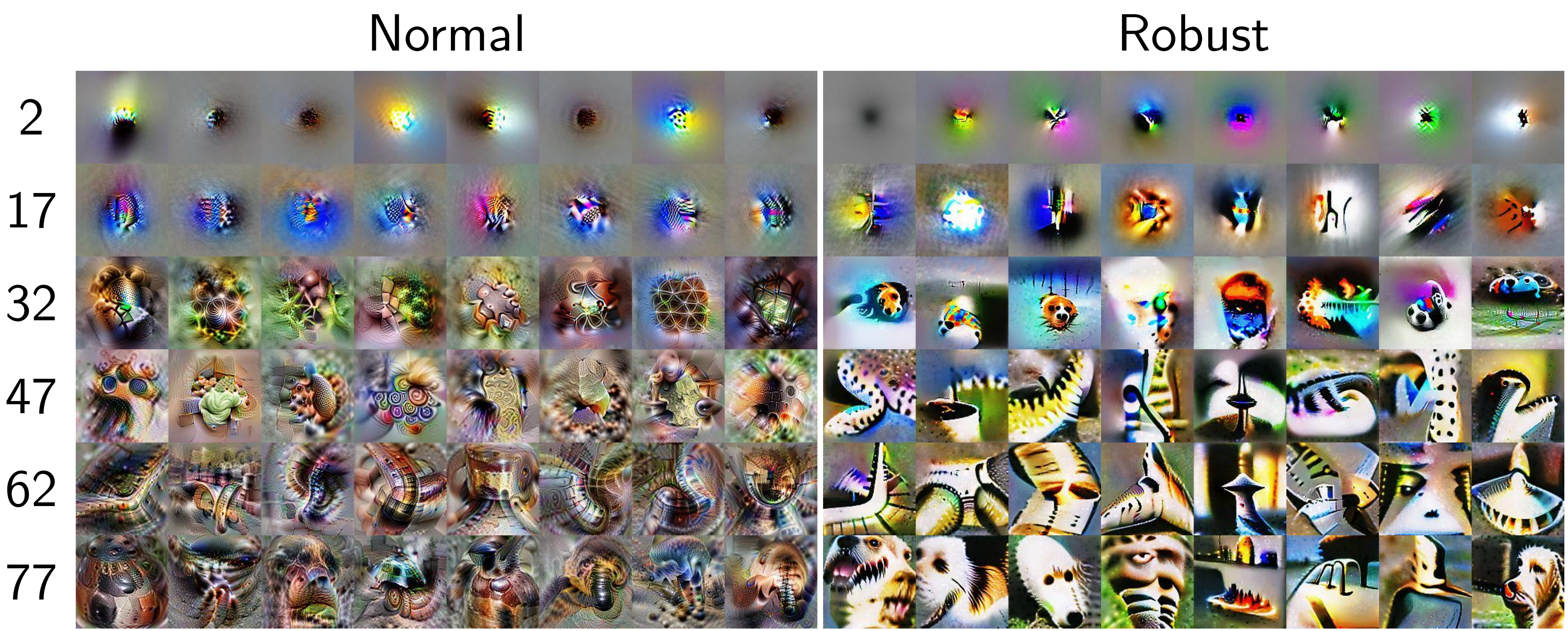
Visualizing Robust Networks

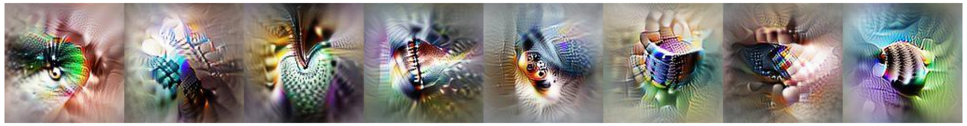
Visualization: find images that maximally excite different neurons.



Visualizing Robust Networks

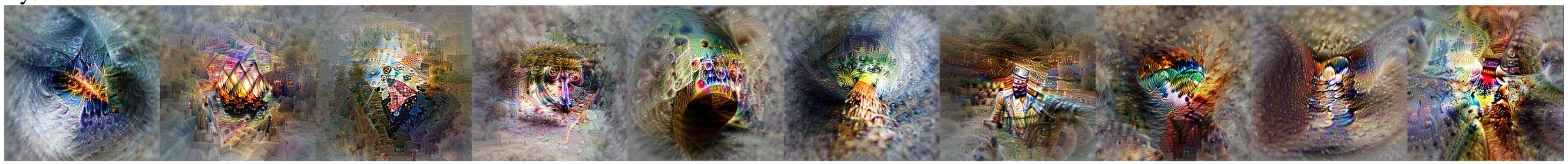
Visualization: find images that maximally excite different neurons.



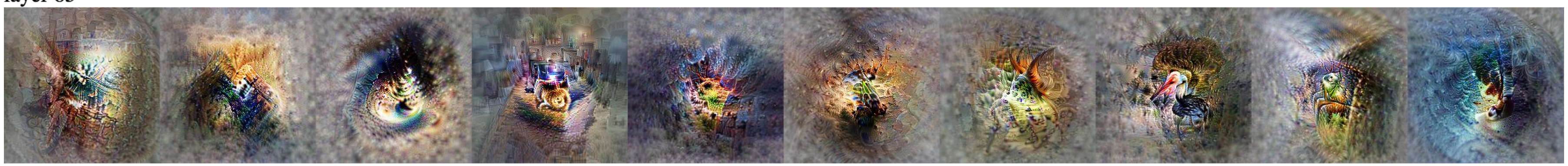
Other non-robust model: 

Regular network (zoomed in)

layer 84



layer 85



layer 86

layer 87

layer 88

layer 89

Robust network (zoomed in)