Joint work with Xin Li

Characterizing adversarial examples in deep networks with convolutional filter statistics

Fuxin Li

School of EECS

Oregon State University

Oregon State University

web.engr.oregonstate.edu/~lif

Fooling a deep network(Szegedy et al. 2013)

- Optimizing a delta from the image to maximize a class prediction $f_c(x)$

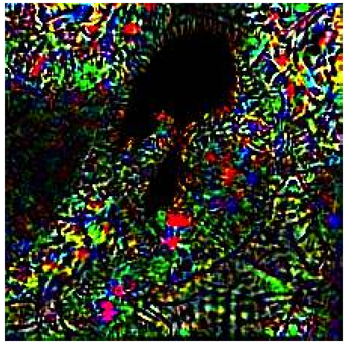
$$\max_{\Delta I} f_c(I+\Delta I)-\lambda||\Delta I||^2$$

Giant Panda (99.32% confidence)

I

$+0.03$

ΔI

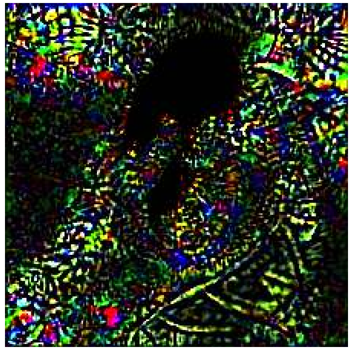


$=$

Shark (93.89% confidence)

$+0.03$

ΔI



$=$

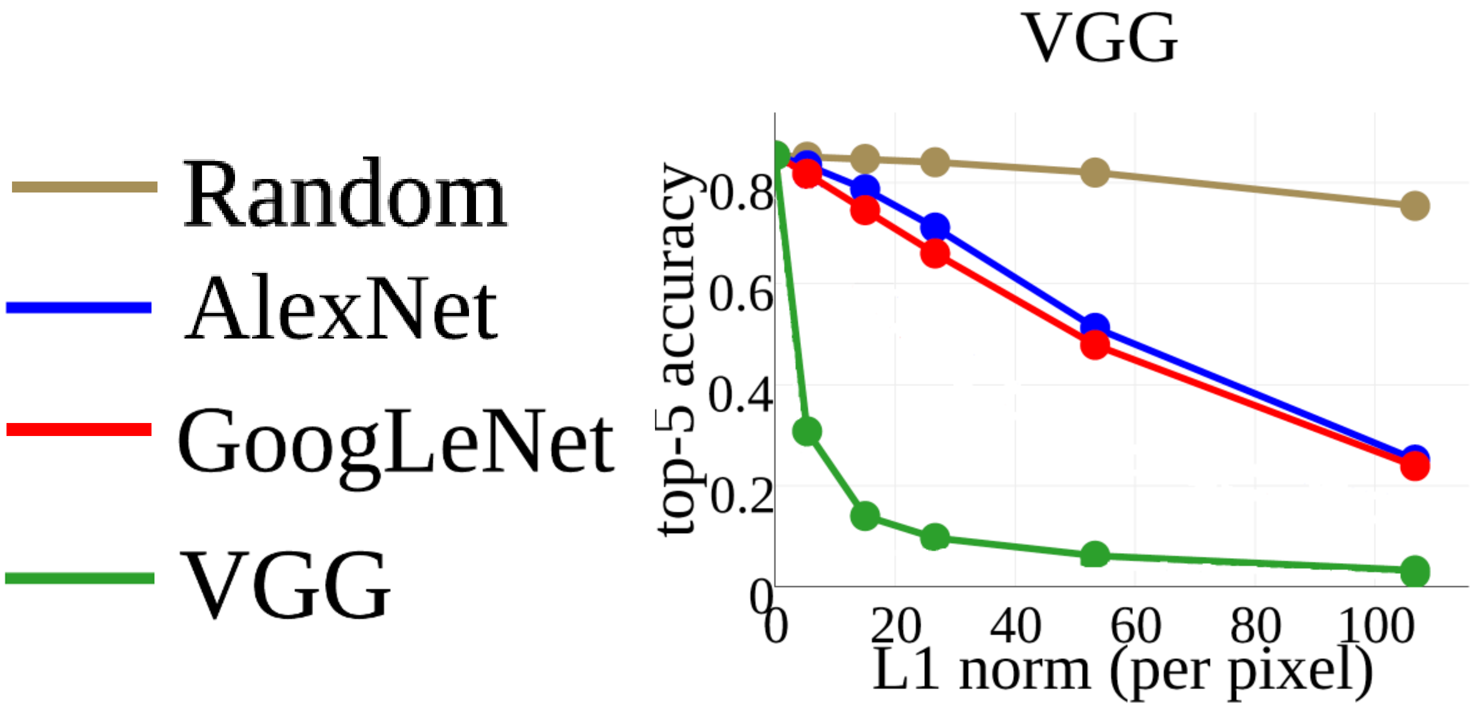
Goldfish (95.15% confidence)

(Szegedy et al. 2013, Goodfellow et al. 2014, Nguyen et al. 2015)

Generalization of fooling

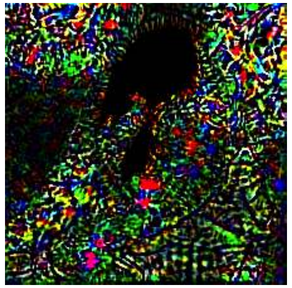
- Adversarial examples are not random
- They generalize across networks!
- Use one algorithm to generate perturbations and test on others (Luo et al. 2016)

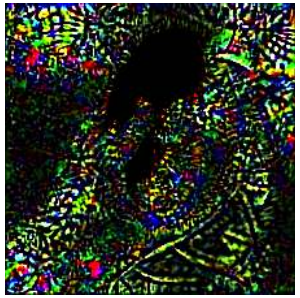
Perturbations generated from:

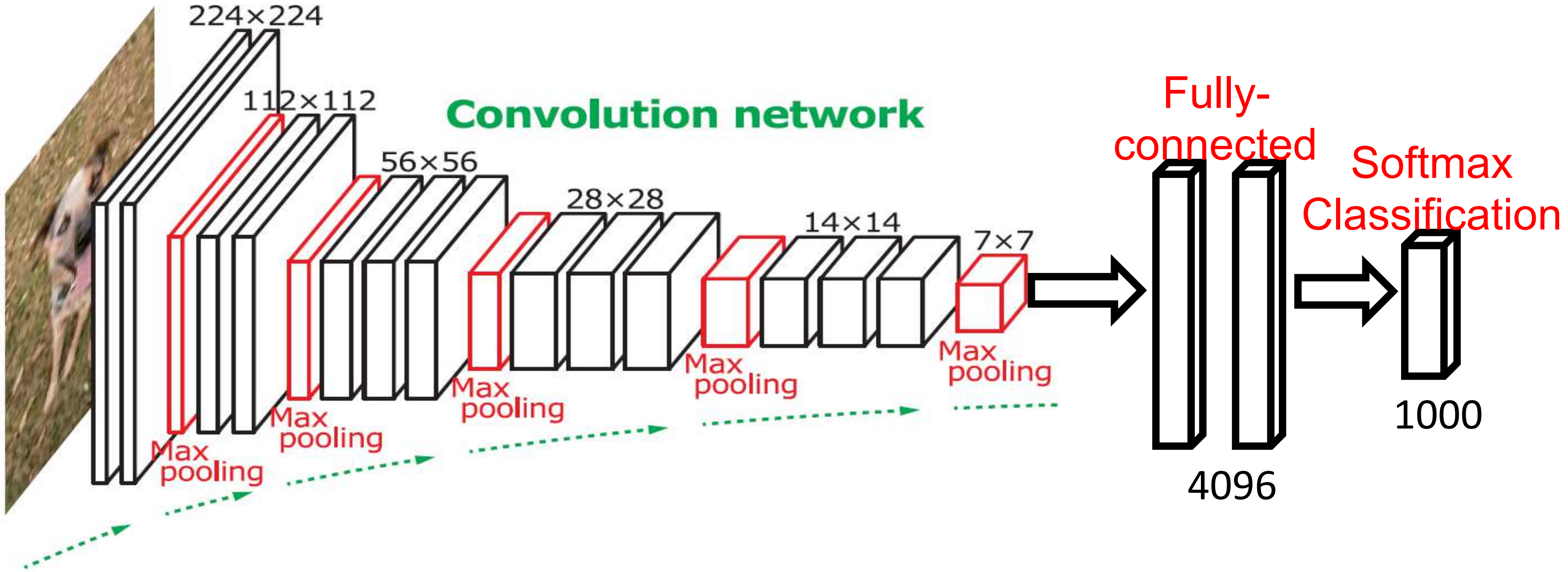


Closer Examination of Perturbations

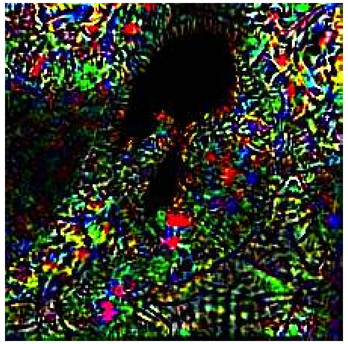
- Two networks used in analysis:
  - AlexNet (2012 state-of-the-art, 16% error on ImageNet challenge)
  - VGG Network (2014-2015 state-of-the-art, 7% error on ImageNet challenge)
  - First part on VGG, second part on both

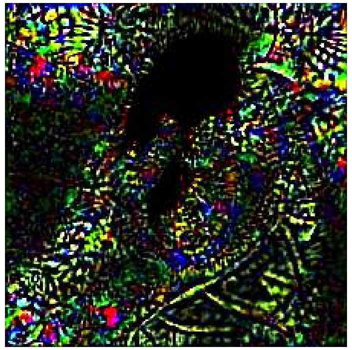




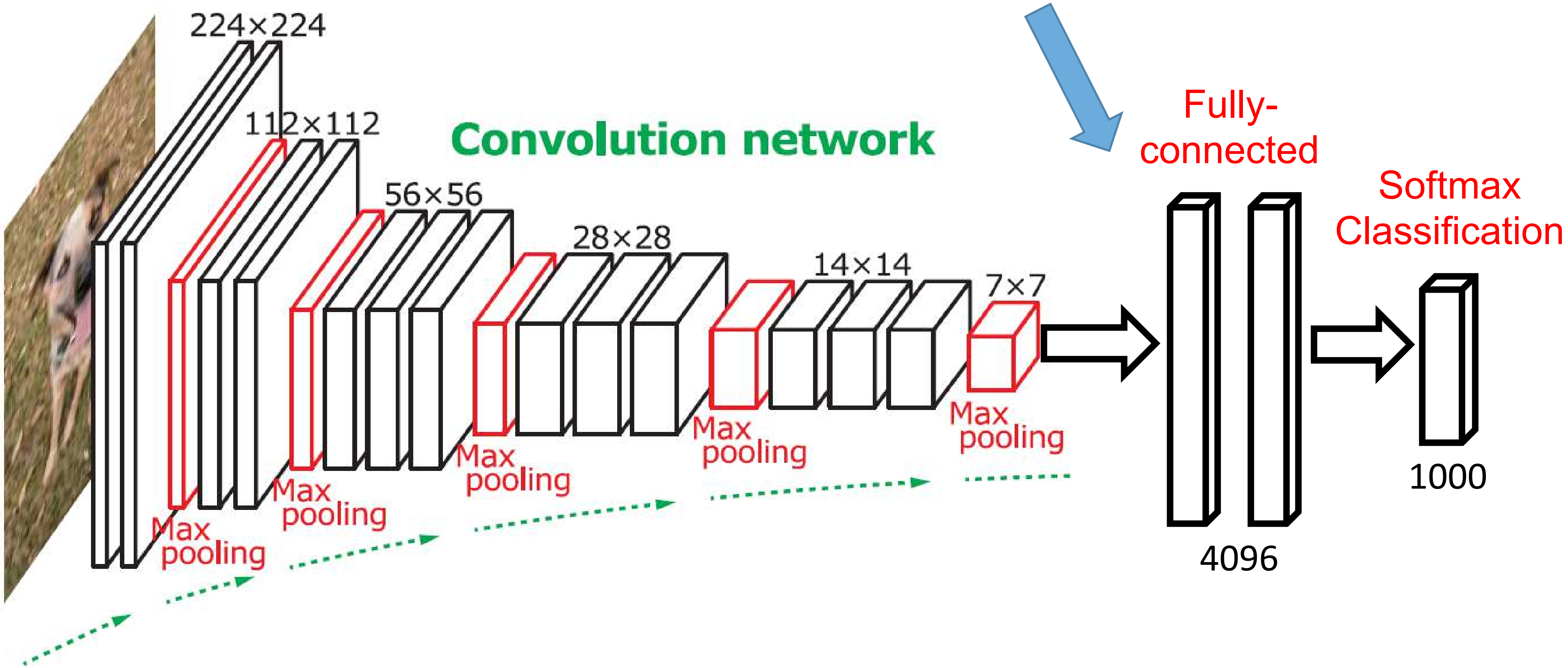


Generate Insights: Explore at the End



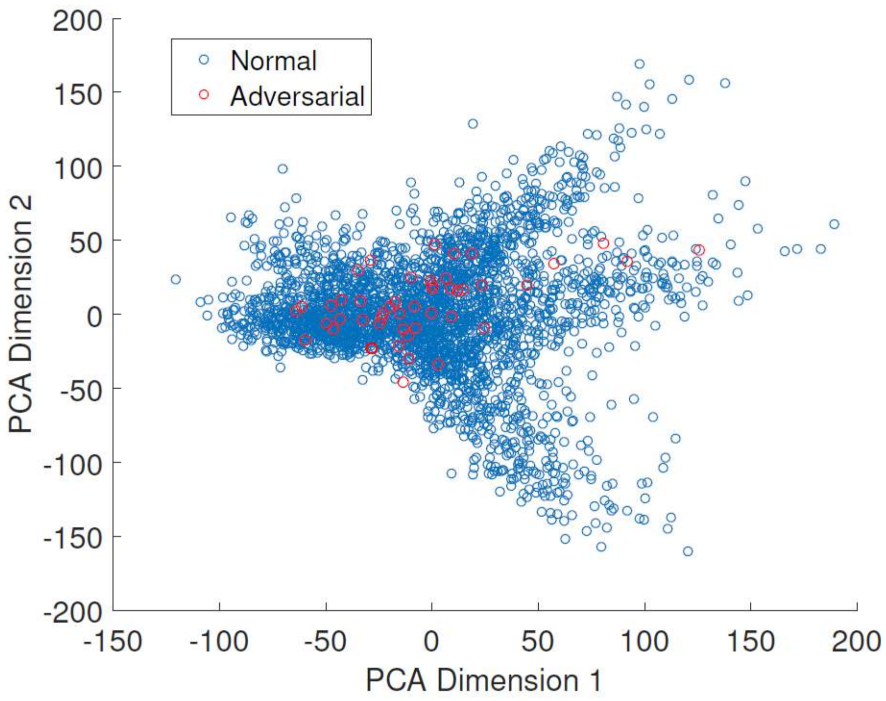


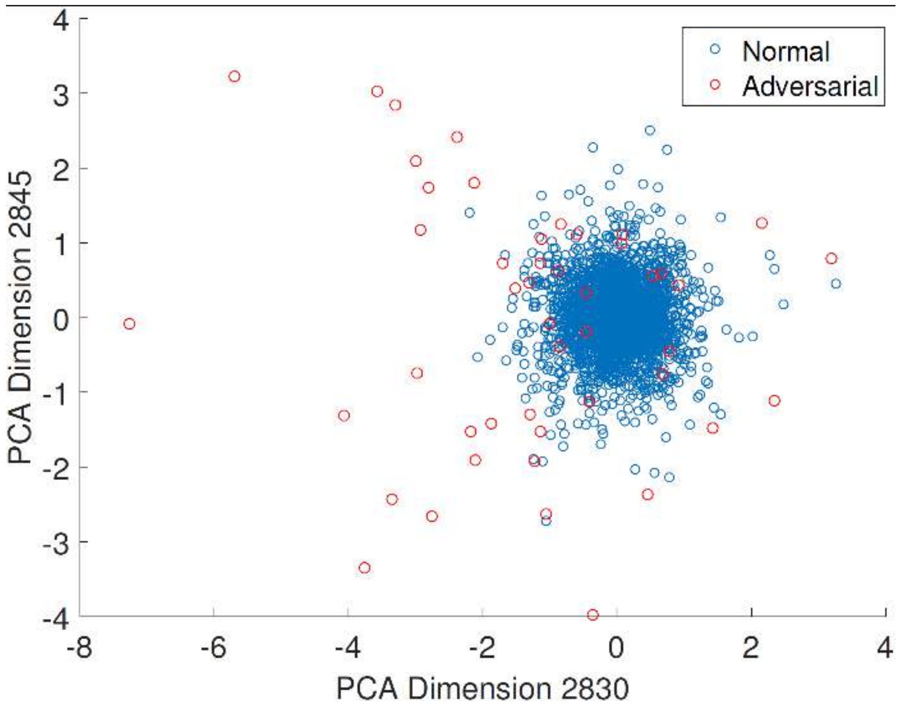
No convolution anymore
Close to final output
Use PCA (NN = linear + transformation)



Corruption Traces before the Fully-Connected Layer

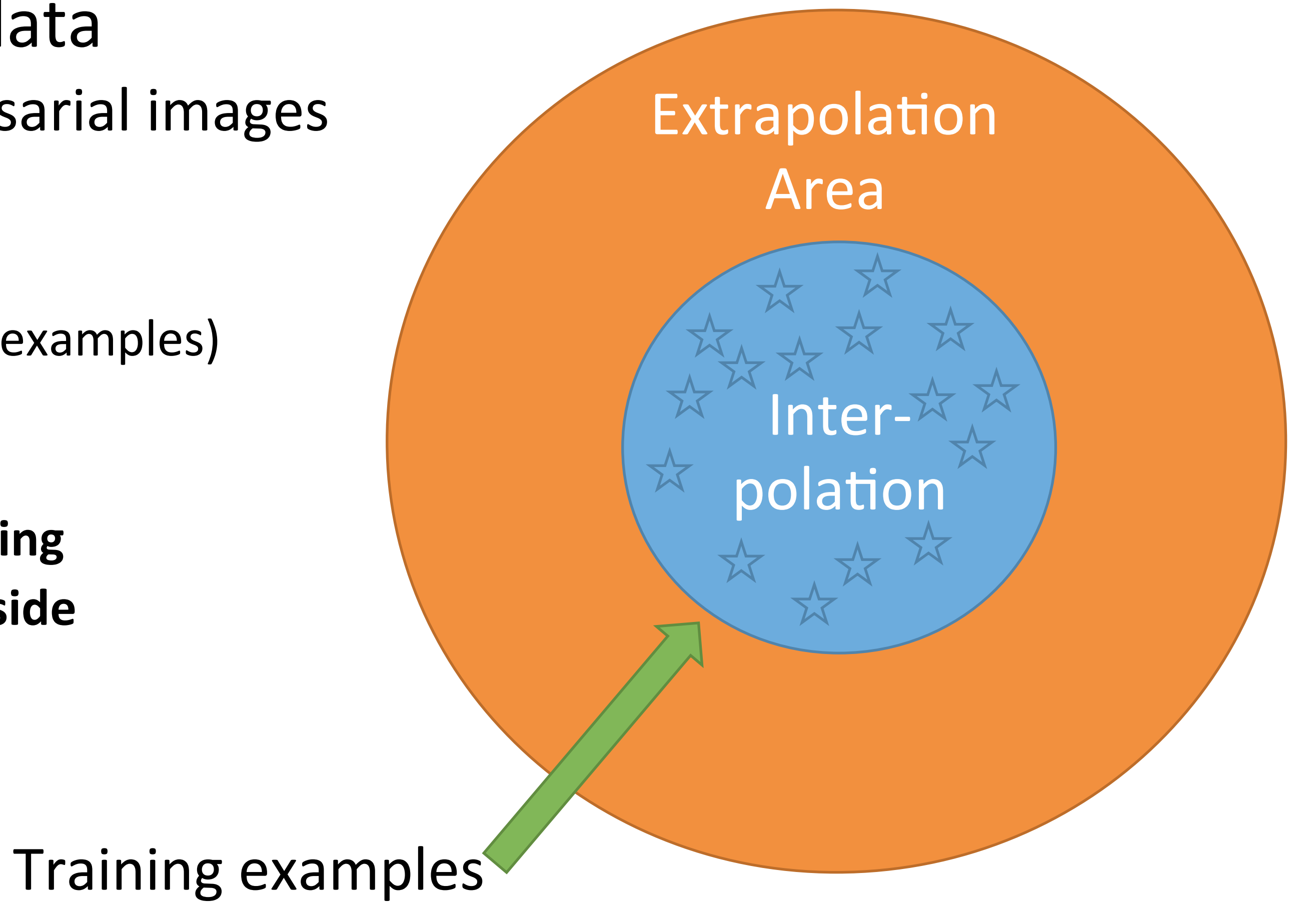
- Do a PCA on layer-14 features (after the last convolutional layer)





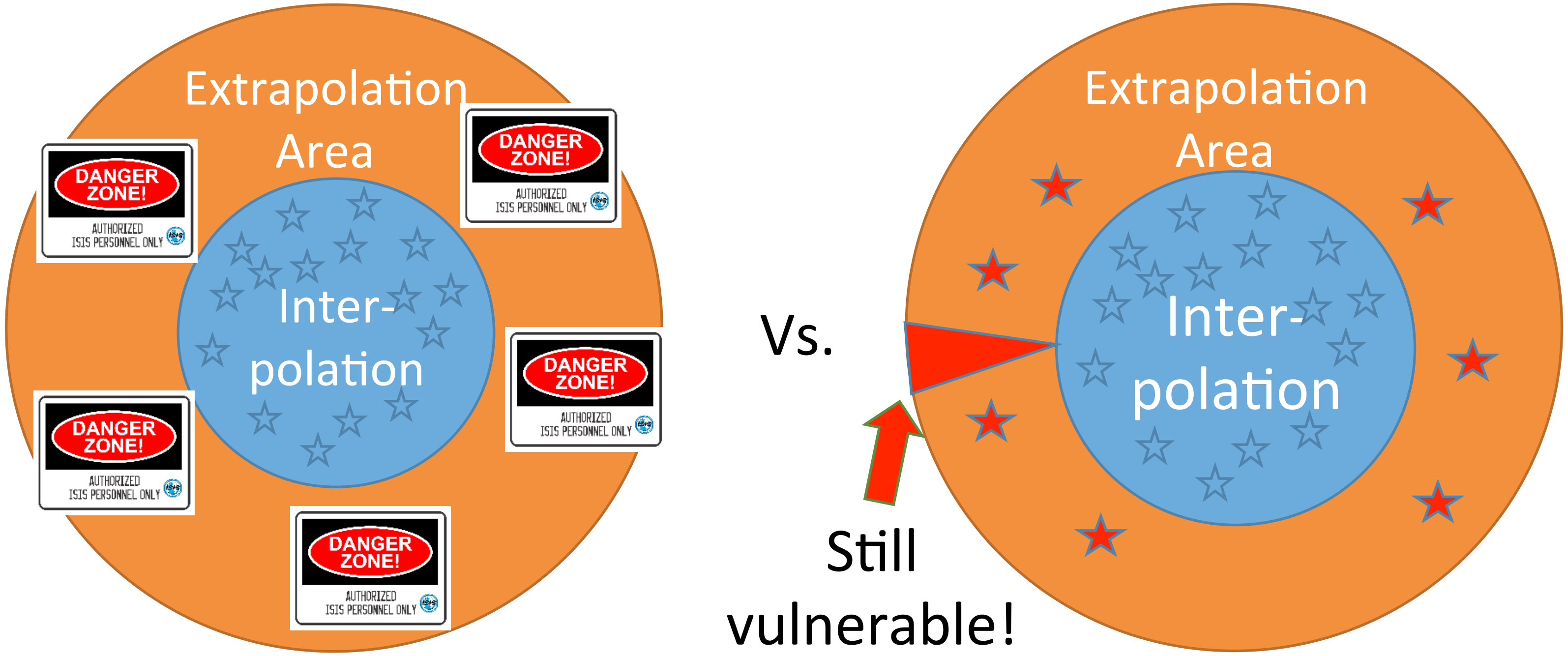
Fundamental Aspect of ML

- Machine learning works only on the test data if it's sampled from the same distribution with training data
    - No good result expected on adversarial images since never trained on it
    - Solution?
        - Enlarge training set (add adversarial examples) (Goodfellow et al. 2014)
        - Led to many GAN-type approaches
        - **Or just detect the boundary of training distribution and refuse to work outside**

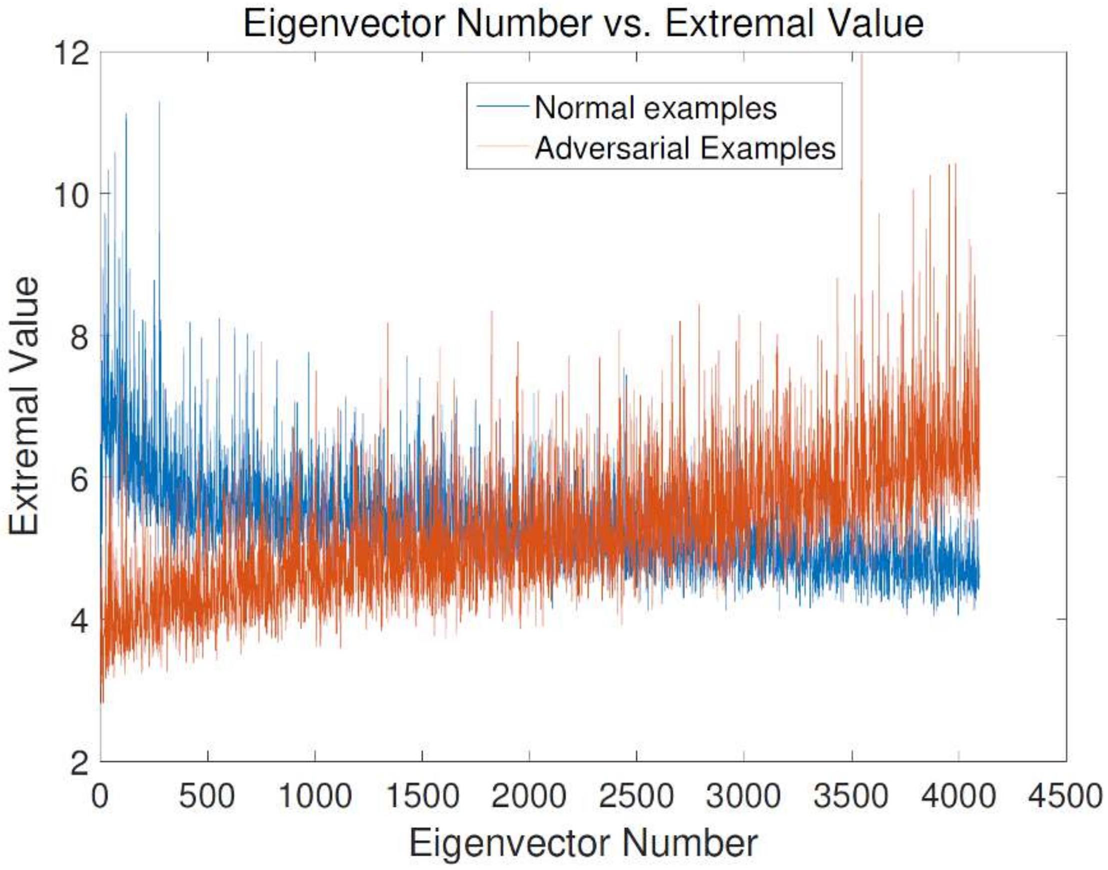


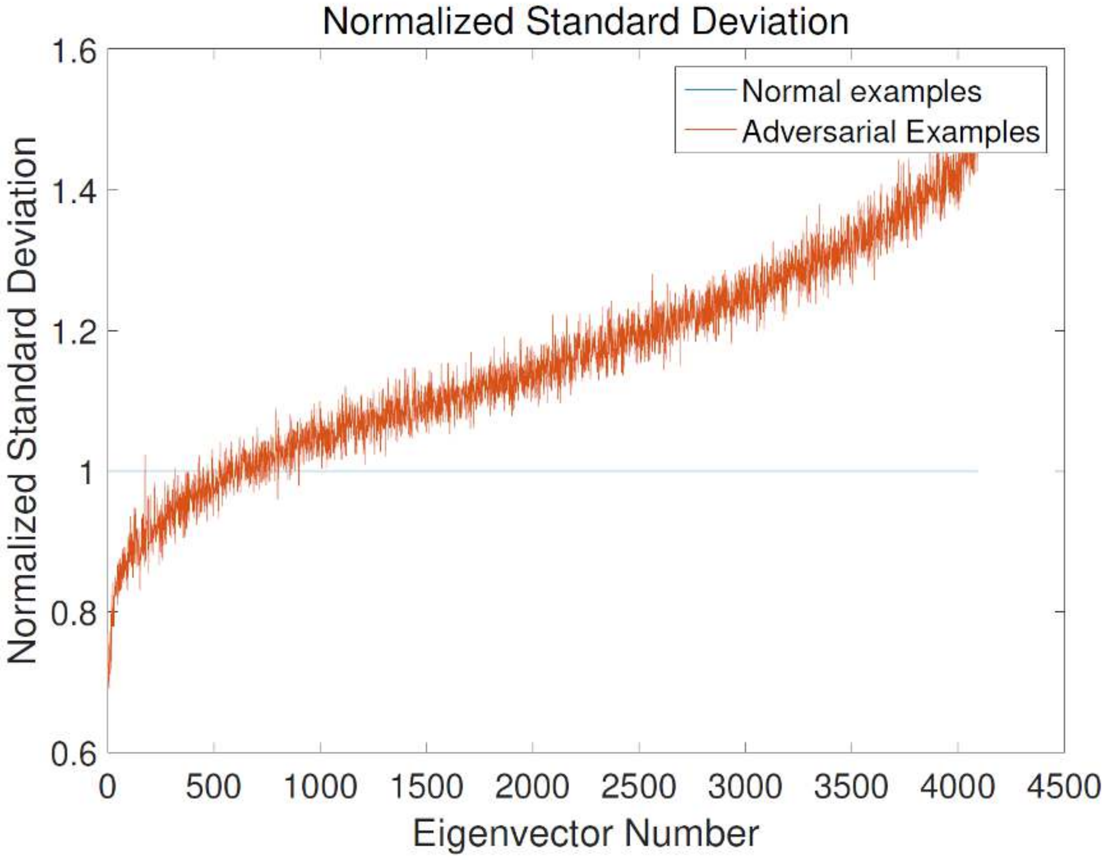
A conservative approach

- Never do extrapolation!
  - Instead, identify intruder attempts for doing so
  - Has been studied in machine learning, e.g. self-aware learning (Li et al. 2008)
- Instead of “adding adversarial examples back to training”
  - Which never ends!



Back to Difference between Normal and Adversarial Examples



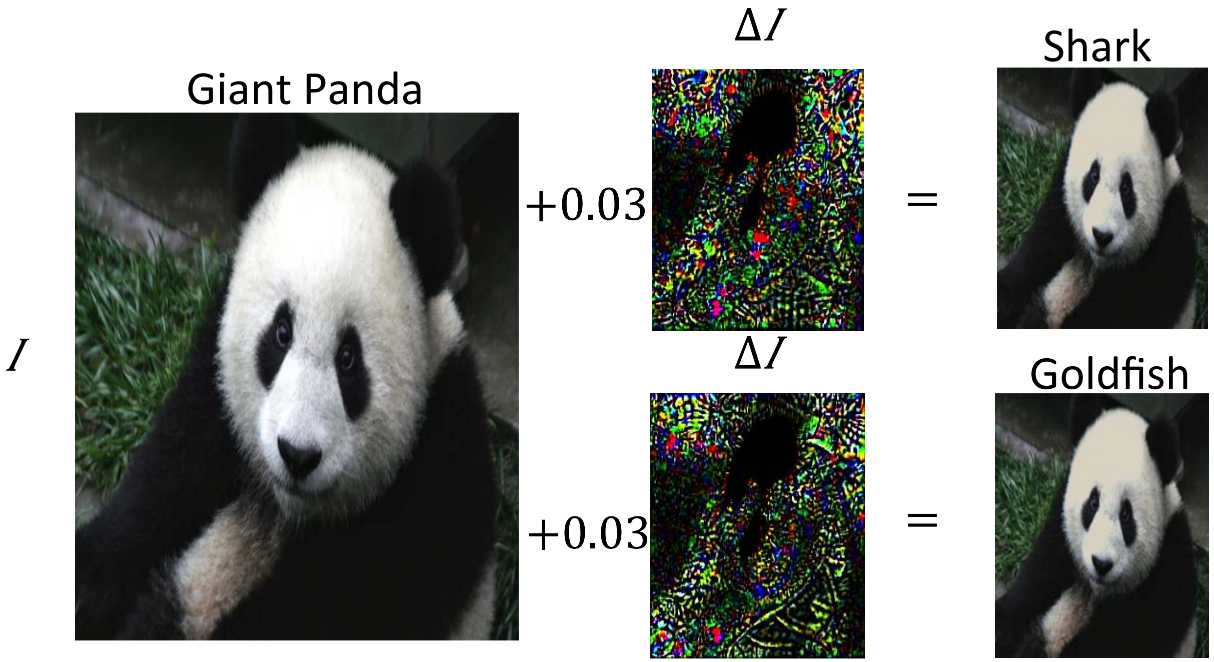


How to observe distributional statistics from a single image?

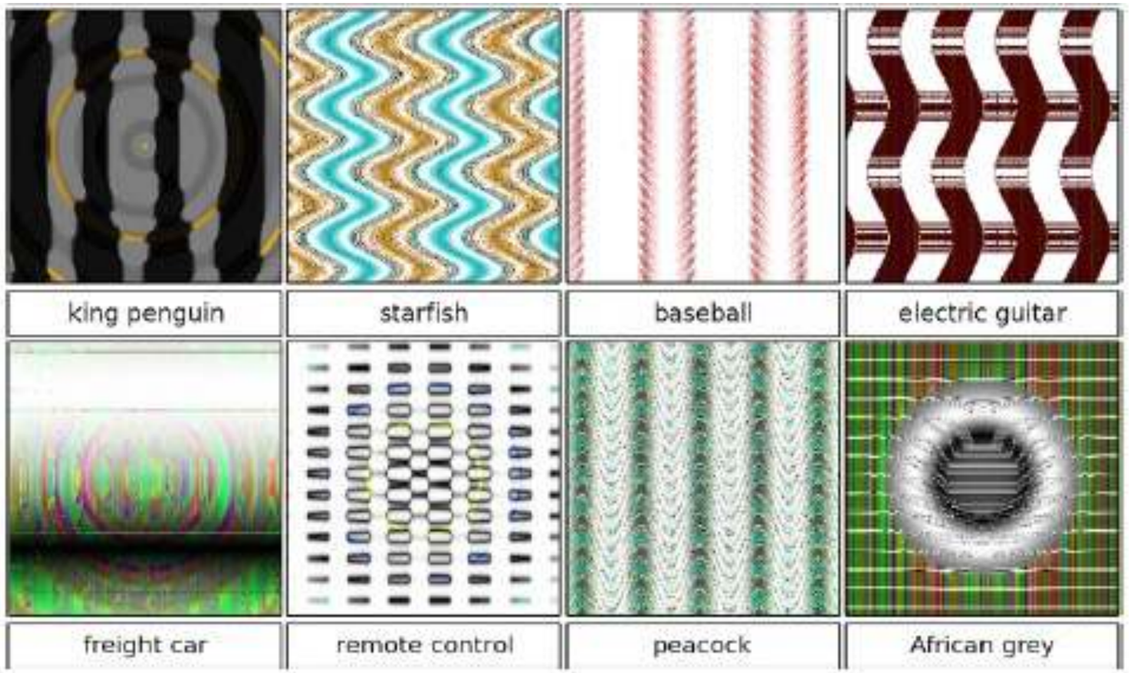
- An image is a distribution of pixels
- Each convolutional layer output is a distribution of pixels
  - K-dimensional distribution on k filter outputs
- Try not to use features to train directly (overfitting!)
- Instead, collect statistics:
  - Mean absolute value of normalized PCA coefficients
  - Minimal and maximal values
  - 25-th, 50-th and 75-th percentiles

Visualization: 2 types of adversarials

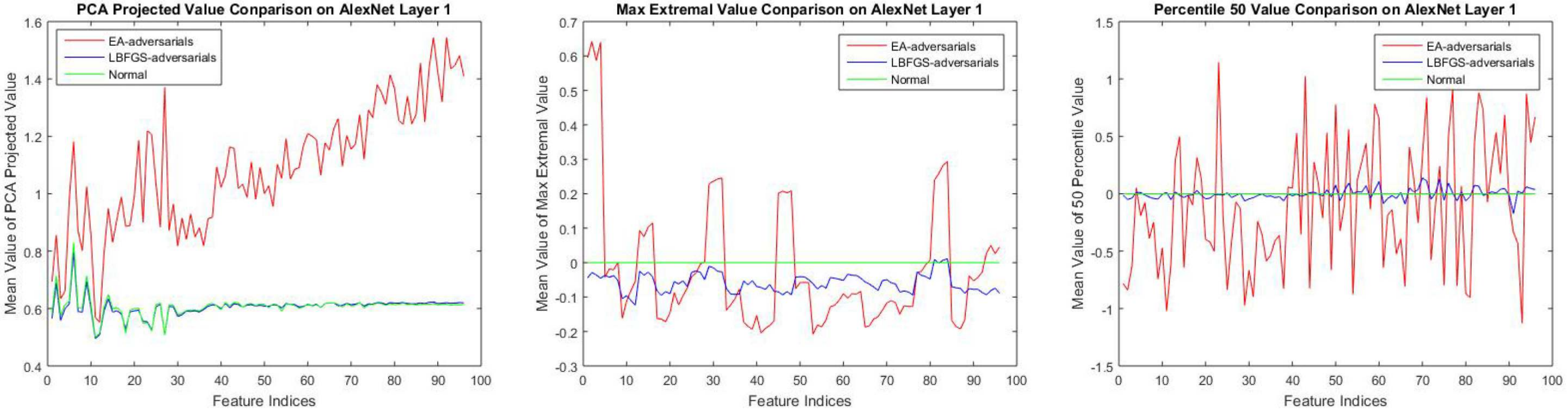
LBFGS-Adversarials (Nguyen et al. 2015)



EA-Adversarials (Nguyen et al. 2015)



Visualization:



Single-Layer Results

- Single-layer results are OK, not fantastic
  - Imaginable with oversimplified features
  - EA-Adversarials much easier to detect

Table 1. Classification Result with AlexNet for Normal vs. LBFGS-adversarials

Network Layer	2nd	3rd	4th
Accuracy	57.5 ± 0.7	67.3 ± 0.7	70.9 ± 0.6
Network Layer	5th	6th	
Accuracy	74.9 ± 0.9	78.95 ± 0.6	

Table 2. Classification Result with VGG-16 for Normal vs. LBFGS-Adersarials

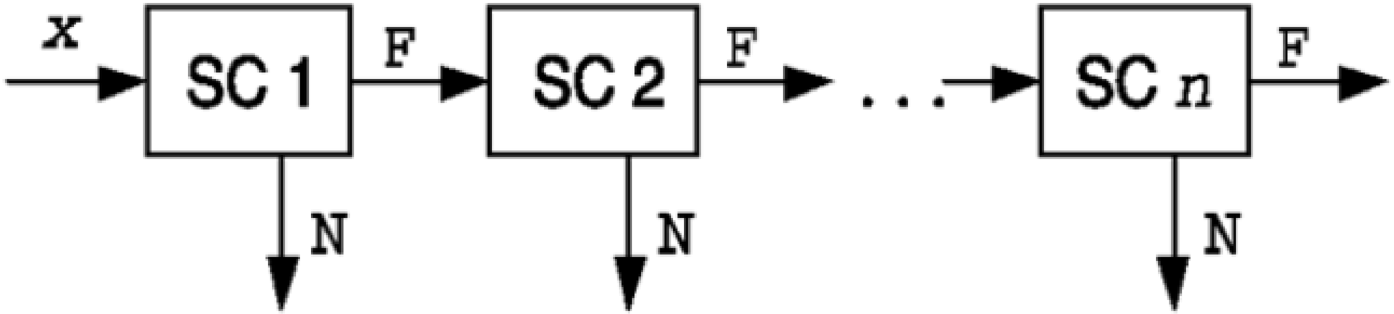
Network Layer	2nd	3rd	4th
Accuracy	72.1 ± 0.7	84.1 ± 0.7	80.3 ± 0.6
Network Layer	5th	6th	7th
Accuracy	81.4 ± 0.9	74.3 ± 0.6	73.9 ± 0.6
Network Layer	8th	9th	10th
Accuracy	74.2 ± 0.7	71.2 ± 0.7	74.3 ± 0.8

Table 3. Classification Result for Normal vs. EA-Adversarials

Layer	2nd	3rd	4th
Accuracy	93.45 ± 0.69	98.3 ± 0.73	97.9 ± 0.57

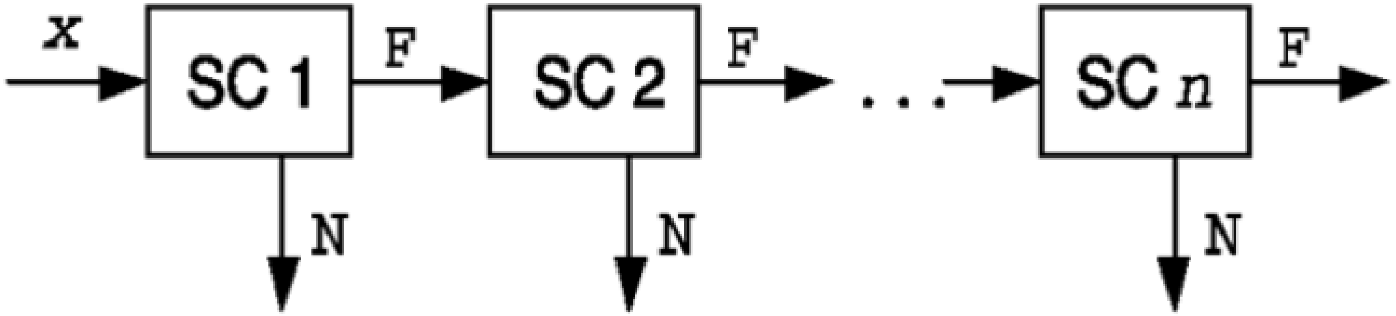
Classifier Cascade

- Proposed by Viola-Jones in 2001 for face detection
- Idea: discard large amount of examples that are simple to classify
- Leave those hard to classify to the next (more expensive) stage



Classifier Cascade

- 1 classifier for each convolutional layer
  - Classify on layer 1:
    - Normal: do not continue
    - Unsure: go to layer 2
    - Classify on layer 2…



Result

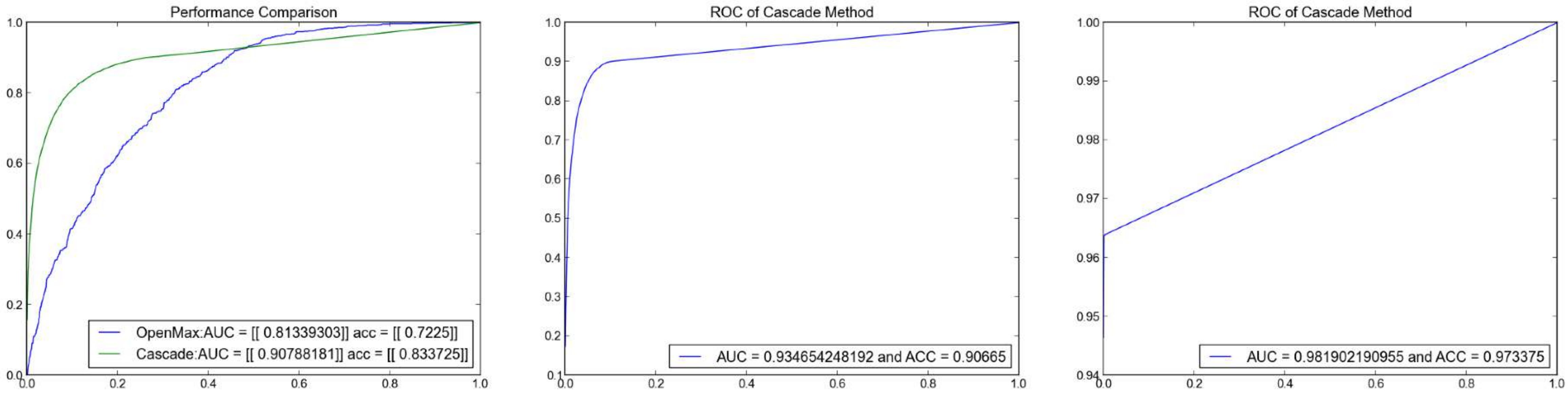
LBFGS Adversarials

EA-Adversarials

AlexNet: 83.3% Accuracy
90.7% AUC

VGG: 90.7% Accuracy
93.5% AUC

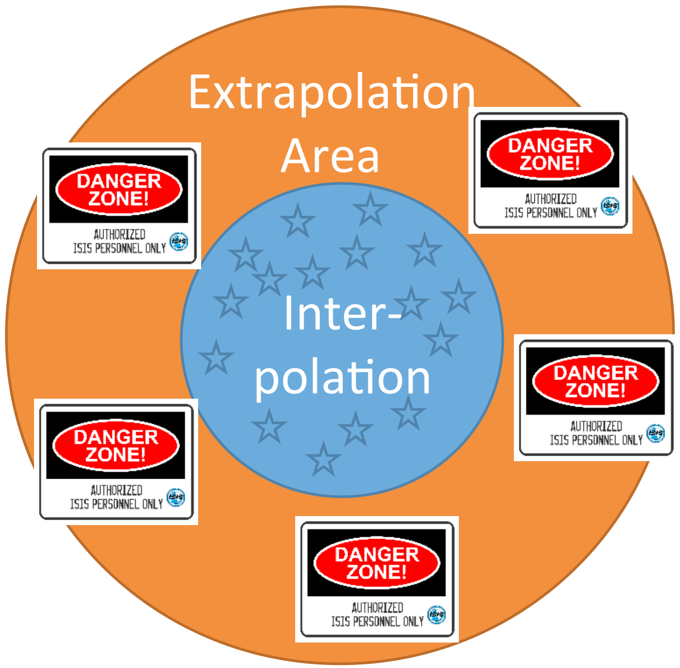
AlexNet: 97.3% Accuracy
98.2% AUC



(a) (b) (c)

Figure 7. (a) Comparison Between OpenMax detection Methods and Cascade Classifier: The blue curve represents the performace of OpenMax Method, and green curve represents the perfornace for Cascade Classifier.(b) Overall ROC Performance Curve of Cascade Classifier Trained on VGG-16 Network. (c) Overall ROC of data generated from EA-adversarials dataset on AlexNet.

Conclusion



- A different approach geared toward AI safety
  - Conservative
  - Avoids extrapolation
  - Try to perform “distribution tests” to test whether an example comes from input distribution
  - Classifier cascades on convolutional filter statistics work well
- Future work:
  - Generative Adversarial Network (GAN) -type approach to detect intrusion

Image Recovery for LBFGS-adversarials

- Insight: LBFGS adversarials attacks the extremal value of gradient output
- This is very specific to manipulating pixels to lower the magnitude of certain outputs
- One can counter even with simple average filtering

Approach	Top-5 Accuracy (Recovered Images)
Original Image (Non-corrupted)	86.5%
3×3 Average Filter	73.0%
5×5 Average Filter	68.0%
Foveation (Object Crop MP) [16]	82.6%

Another side of the story

bell pepper (946), score 0.848

Noise std = 16

bell pepper (946), score 0.841

Noise std = 32

bell pepper (946), score 0.531

Noise std = 40

bell pepper (946), score 0.294

Noise std = 48

cucumber, cuke (944), score 0.175

- It’s also not that hard to contaminate CNN!