CSCI 5922, Spring 2020

Adversarial Examples

NN Limitations

There are some fundamental challenges with neural networks

- No symbolic reasoning: hard to interpret why the NN makes a decision.
- Learned representations are unstable and networks do not behave as humans would.

We'll go over some lines of research started by Szegedy et al., 2014, "Intriguing properties of neural networks", which laid out some problems that are still open.

Interpretability

- Do NNs learn high-level concepts?
- A popular technique is to inspect images that excite specific neurons in the network.
- Szegedy et al., 2014 shows that random linear combinations of neurons are just as interpretable as individual neurons

(a) Unit sensitive to white flowers.

(b) Unit sensitive to postures.

(c) Unit senstive to round, spiky flowers.

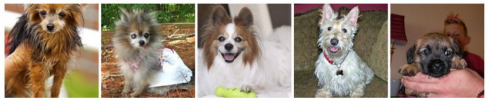
(d) Unit senstive to round green or yellow objects.

Maximizing images for individual units

(a) Direction sensitive to white, spread flowers.

(b) Direction sensitive to white dogs.

(c) Direction sensitive to spread shapes.



(d) Direction sensitive to dogs with brown heads.

Maximizing images for random directions

Interpretability

- Suggests that the space spanned by the units is what is important, not the units themselves.
- Eliciting “concepts” from neural networks is more complex than just looking at neuron activations.

Instability of NNs

- Recall that architecture of CNNs provides:
  - Stability to local geometric deformation
  - Features that are covariant with image translation
- It does not enforce Lipschitz smoothness
  - $\| \mathbf{f}(\mathbf{x} + \mathbf{r}) - \mathbf{f}(\mathbf{x}) \|$ can be large even for small perturbations $\mathbf{r}$
- However, we often assume that networks are stable in regions around the training data.

Instability of NNs

- In NNs, there is no stability enforced in the loss functions
  - Kernel methods use RKHS norm penalties
  - NN regularizers (dropout, early stopping, weight decay) do not necessarily impose smoothness
- Empirically, do neural networks learn a smooth transformation?

Instability of NNs

- Szegedy et al., 2014, “Intriguing properties of neural networks”
- Describe a method to find perturbations that reveal NN instability
- Let $\boldsymbol{l}$ be a class label, distinct from the correct ground-truth class for $\boldsymbol{x}$

$$\text{Minimize } \|r\|_2 \text{ subject to:}$$
$$1.\ \ f(x+r) = l$$
$$2.\ \ x + r \in [0,1]^m$$

- i.e. find the smallest perturbation that causes the neural network to mess up and assign label $\boldsymbol{l}$

Instability of NNs

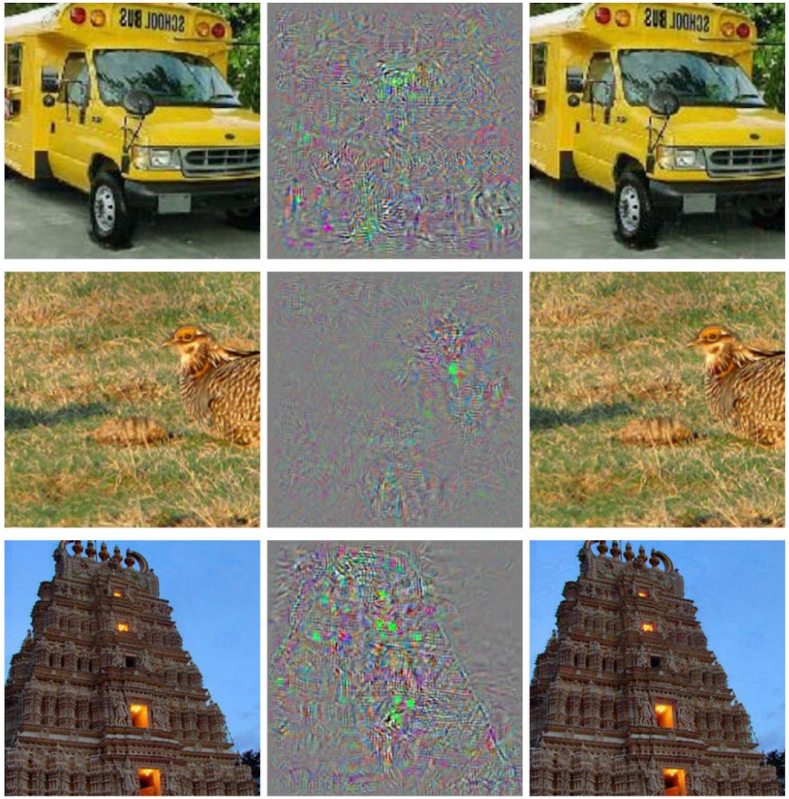
- They solve a relaxation of the problem

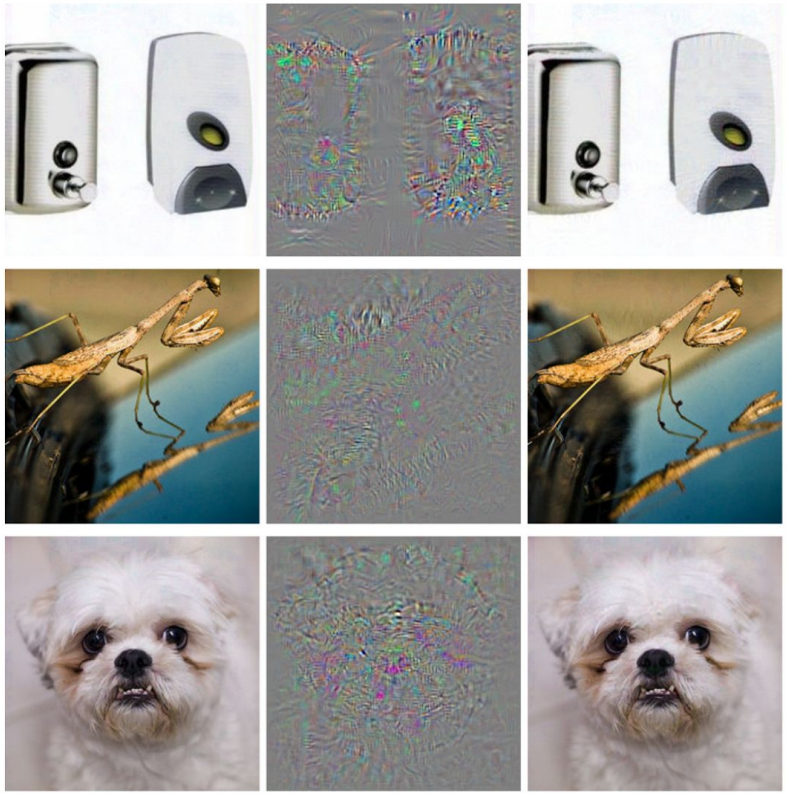
$$\text{Minimize } c|r| + \text{loss}_f(x + r, l) \text{ subject to } x + r \in [0, 1]^m$$

- Use L-BFGS, and backpropagate into the input pixels

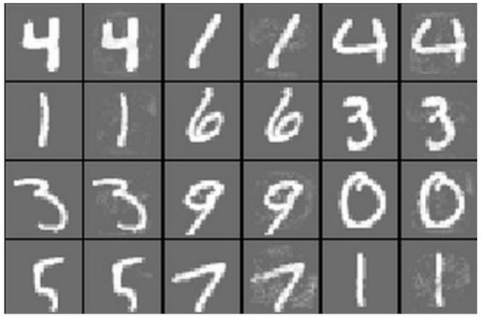
Adversarial examples

- Applied to AlexNet and several MLPs trained on MNIST.
- The optimization is remarkably successful.
- Tiny, imperceptible perturbations can cause all the neural networks to switch labels.

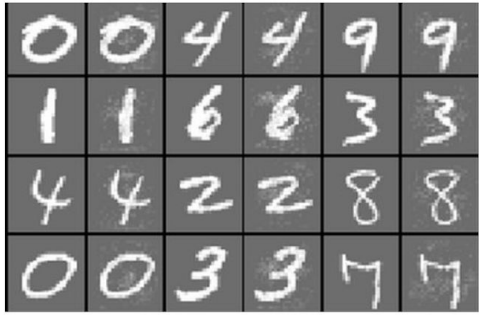




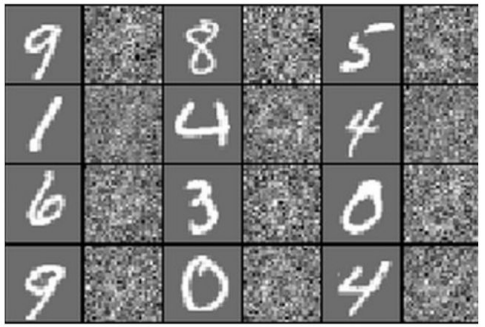
Can you spot the ostriches?



(a) Even columns: adversarial examples for a linear (FC) classifier (stddev=0.06)



(b) Even columns: adversarial examples for a 200-200-10 sigmoid network (stddev=0.063)



(c) Randomly distorted samples by Gaussian noise with stddev=1. Accuracy: 51%.

Cross model generalization

- Ok, but maybe this optimization really targets a specific model’s particular weakness?
- Or maybe it’s particular to the model architecture?

	FC10(10^{-4})	FC10(10^{-2})	FC10(1)	FC100-100-10	FC200-200-10	AE400-10	Av. distortion
FC10(10^{-4})	100%	11.7%	22.7%	2%	3.9%	2.7%	0.062
FC10(10^{-2})	87.1%	100%	35.2%	35.9%	27.3%	9.8%	0.1
FC10(1)	71.9%	76.2%	100%	48.1%	47%	34.4%	0.14
FC100-100-10	28.9%	13.7%	21.1%	100%	6.6%	2%	0.058
FC200-200-10	38.2%	14%	23.8%	20.3%	100%	2.7%	0.065
AE400-10	23.4%	16%	24.8%	9.4%	6.6%	100%	0.086
Gaussian noise, stddev=0.1	5.0%	10.1%	18.3%	0%	0%	0.8%	0.1
Gaussian noise, stddev=0.3	15.6%	11.3%	22.7%	5%	4.3%	3.1%	0.3

Cross training-set generalization

- Maybe the adversaries are particular to the training set?
- Experiment
  - Split MNIST training set into 2 halves (30k examples each)
  - Train models on each half
  - See if the adversaries to a model screw up the other model

	FC100-100-10	FC123-456-10	FC100-100-10'
Distorted for FC100-100-10 (av. stddev=0.062)	100%	26.2%	5.9%
Distorted for FC123-456-10 (av. stddev=0.059)	6.25%	100%	5.1%
Distorted for FC100-100-10' (av. stddev=0.058)	8.2%	8.2%	100%
Gaussian noise with stddev=0.06	2.2%	2.6%	2.4%
Distorted for FC100-100-10 amplified to stddev=0.1	100%	98%	43%
Distorted for FC123-456-10 amplified to stddev=0.1	96%	100%	22%
Distorted for FC100-100-10' amplified to stddev=0.1	27%	50%	100%
Gaussian noise with stddev=0.1	2.6%	2.8%	2.7%

The effectiveness decreases across training sets, but is still significant, especially when magnifying the perturbation.

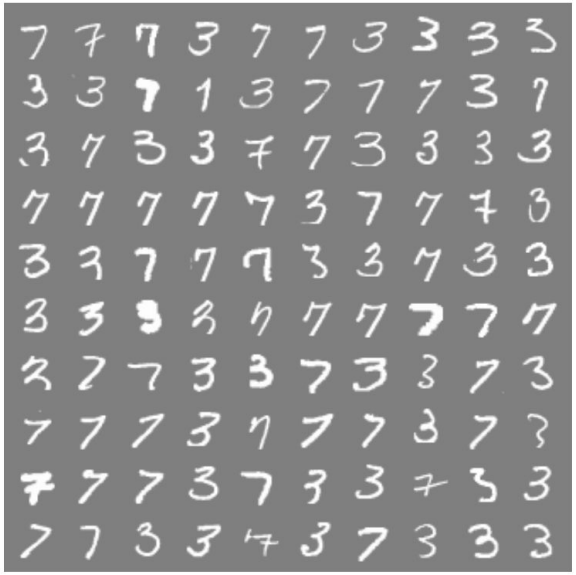
Fast gradient sign method

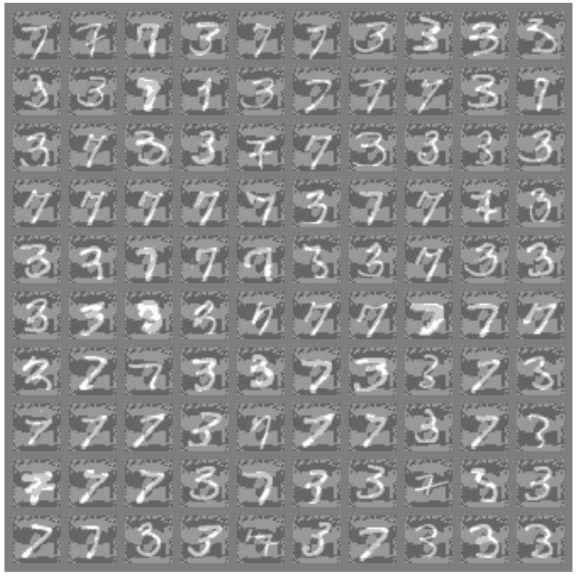
- The phenomenon was studied further in Goodfellow et al., 2015 “Explaining and harnessing adversarial examples”

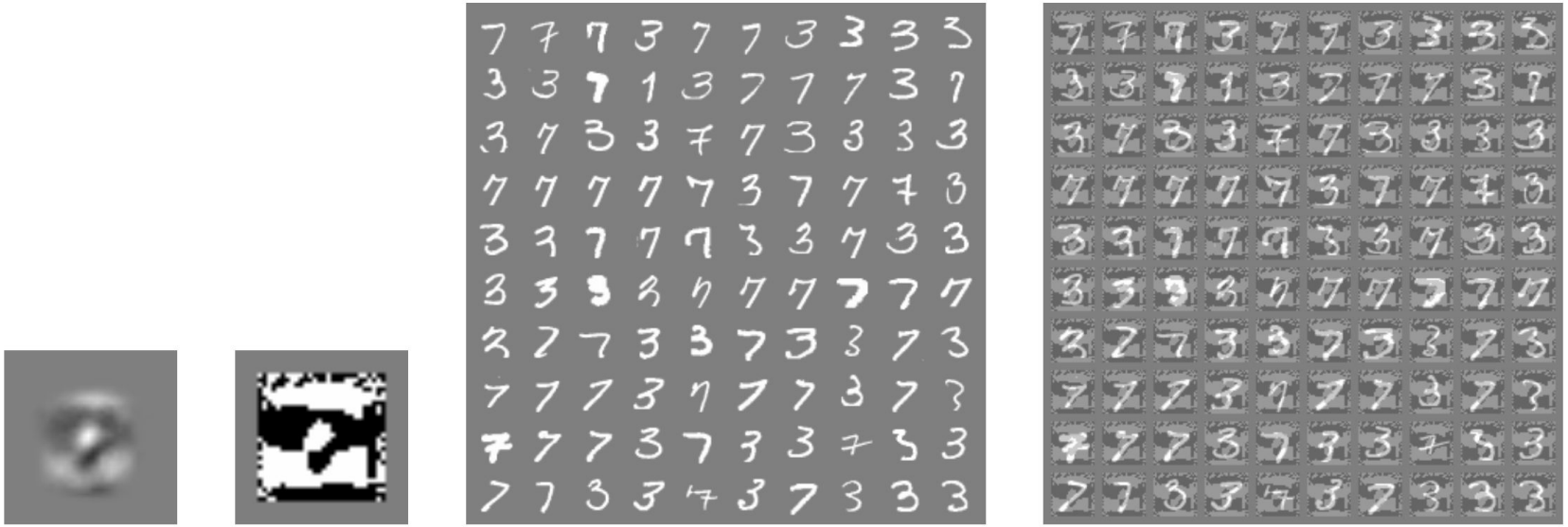
Consider a linear model $y = \boldsymbol{\omega} \cdot \mathbf{x}$, and construct an adversary $\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\eta}$, with the constraint $\|\boldsymbol{\eta}\|_\infty < \epsilon$

The difference $\boldsymbol{\omega} \cdot \tilde{\mathbf{x}} - \boldsymbol{\omega} \cdot \mathbf{x} = \boldsymbol{\omega} \cdot \boldsymbol{\eta}$ is maximized by setting $\boldsymbol{\eta} = \text{sign}\,(\boldsymbol{\omega})$

Fast gradient sign method







Fast gradient sign method applied to a linear MNIST classifier of 7s vs 3s

Figure 2 from [Goodfellow et al.]

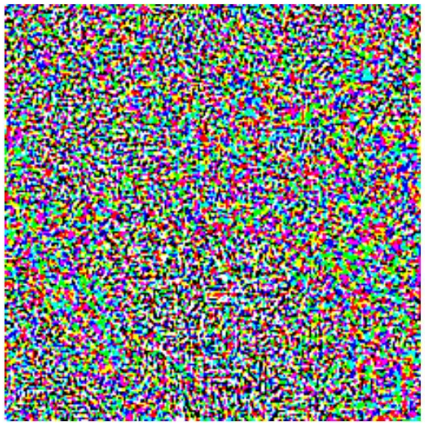
Fast gradient sign method for NNs

- NNs are well-approximated locally by a linear function of their inputs
- For a given input, set the perturbation to be equal to the sign of the gradient of the loss w.r.t. the input:

$$\boldsymbol{\eta} = \epsilon \operatorname{sign}\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y)\right)$$

- Note that this gradient can be calculated, again, by backpropagating to the input units.

$+ .007 \times$



$=$

$\boldsymbol{x}$

"panda"
57.7% confidence

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

"nematode"
8.2% confidence

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

"gibbon"
99.3 % confidence

Fast gradient sign method applied to GoogLeNet

Figure 1 from Goodfellow et al.

Harnessing adversaries

- The fast gradient sign method opens up the possibility of adversarial training
- Training with adversaries is a kind of regularization, and also aims to produce more stable models

$$\tilde{J}(\boldsymbol{\theta}, \boldsymbol{x}, y) = \alpha J(\boldsymbol{\theta}, \boldsymbol{x}, y) + (1 - \alpha) J(\boldsymbol{\theta}, \boldsymbol{x} + \epsilon \text{sign}\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y)\right))$$

- The resulting models are “somewhat resistant”, but still vulnerable to adversaries
  - Original model: 89.4% error on FGSM adversaries
  - Model trained with robust loss: 17.9% error on FGSM adversaries (still with high confidence)

NN attacks

- Maybe not just intriguing, but also a security concern?
- ML-powered systems could be tricked into giving arbitrary results by making changes to inputs that are imperceptible to humans.
- With FGSM, they are easy to compute

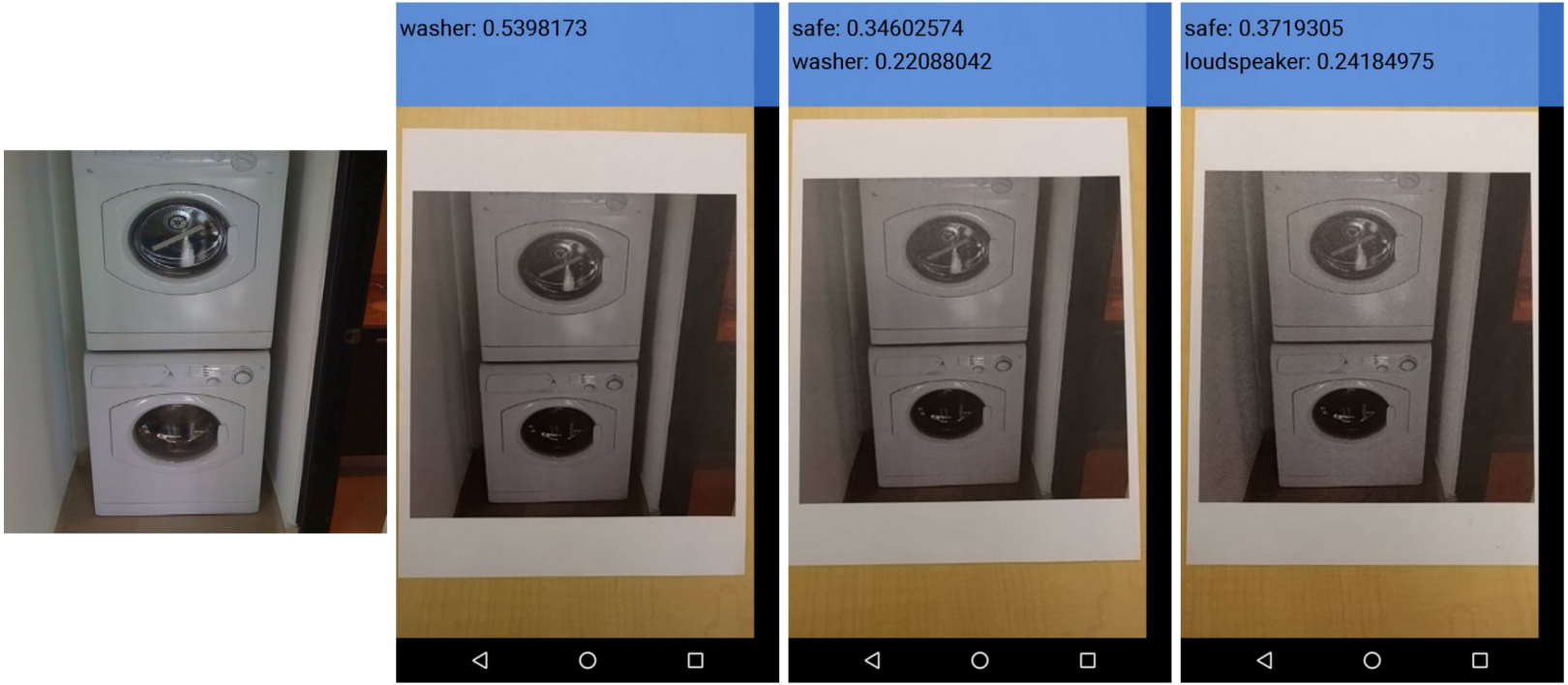
Black-box attacks

- The optimization required access to the network weights
- In real-world scenarios, the network would be hidden from the attacker
- However, Papernot et al. 2017 showed that black-box attacks are possible
- They construct adversaries while only having the ability to get model predictions.

Spot the yield sign

Physical attacks

- In some real-world systems, such attacks requires the ability to directly feed pixels to the model.
- What about real-world adversaries?
- Kurakin, et al. 2017 printed out adversaries and classified them through a cell-phone camera



(a) Image from dataset (b) Clean image (c) Adv. image, $\epsilon = 4$ (d) Adv. image, $\epsilon = 8$

See https://youtu.be/zQ_uMenoBCk and https://youtu.be/piYnd_wYlT8

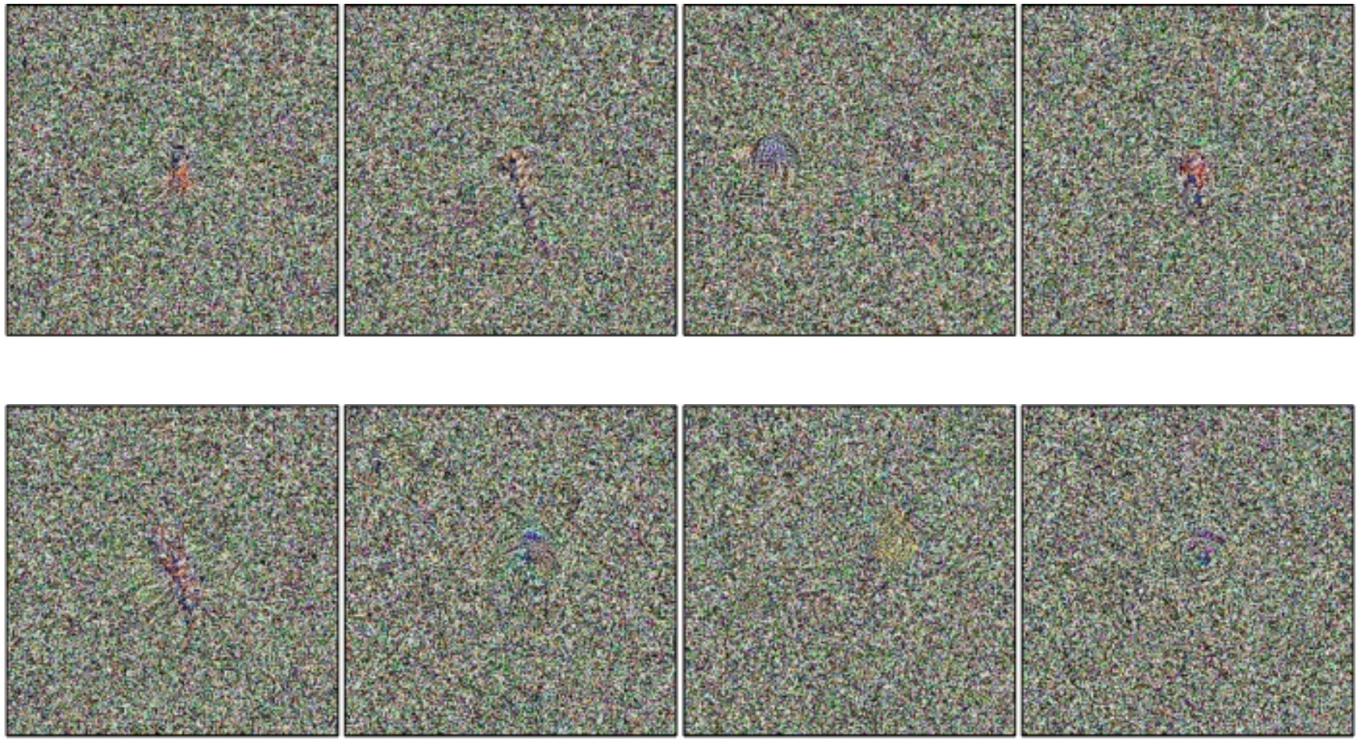
Machine learning is the end of computer security?

“Why Do Keynote Speakers Keep Suggesting That Improving Security Is Possible? Because Keynote Speakers Make Bad Life Decisions and Are Poor Role Models” USENIX Security '18-Q

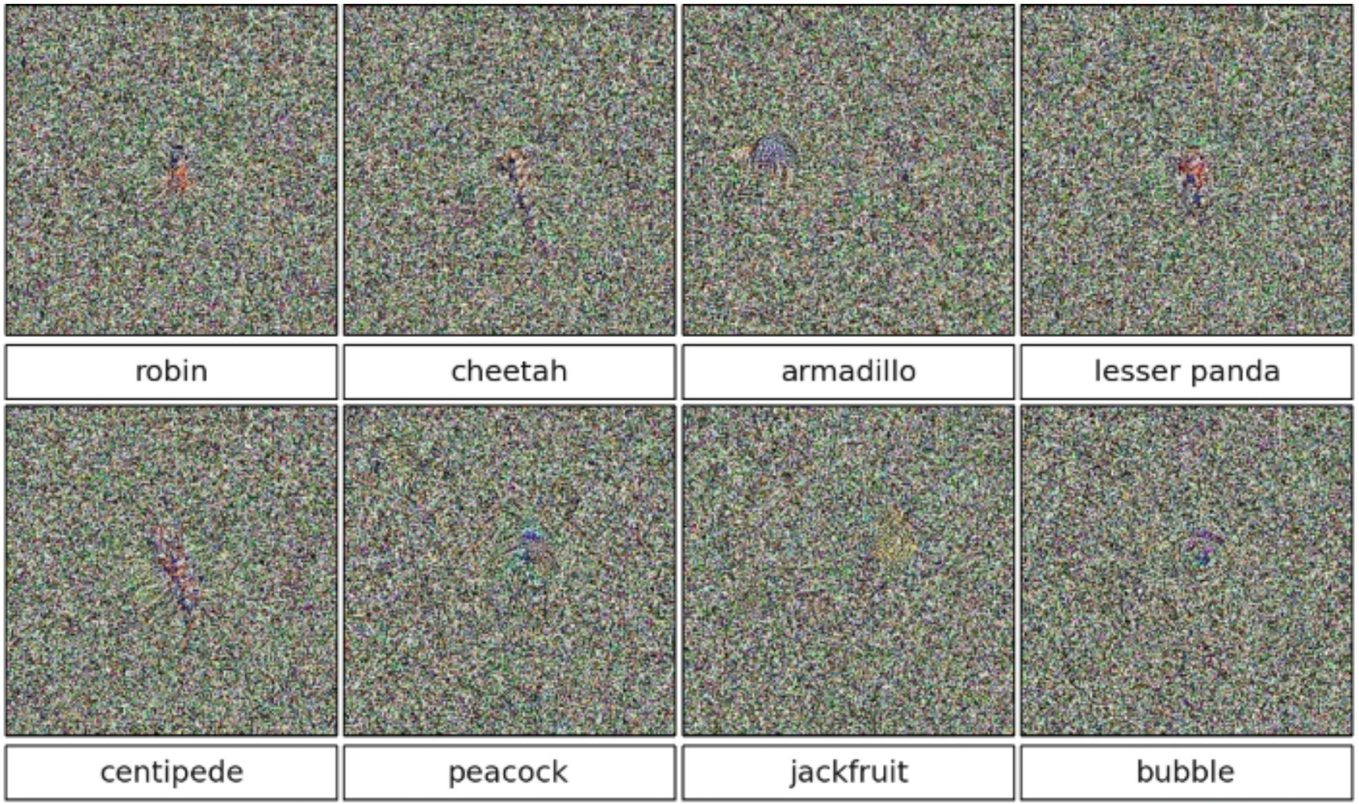
Directly fooling deep networks

- Nguyen et al. 2015 “Deep Neural Networks Are Easily Fooled: High Confidence Predictions for Unrecognizable Images”
- Use evolutionary algorithms to directly generate images that a classifier will assign a high-confidence prediction too

Optimizing in the pixel domain

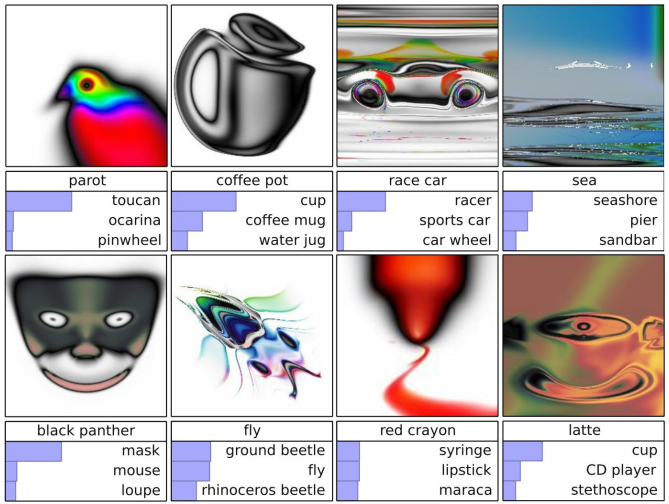


Optimizing in the pixel domain



Optimizing higher-level features

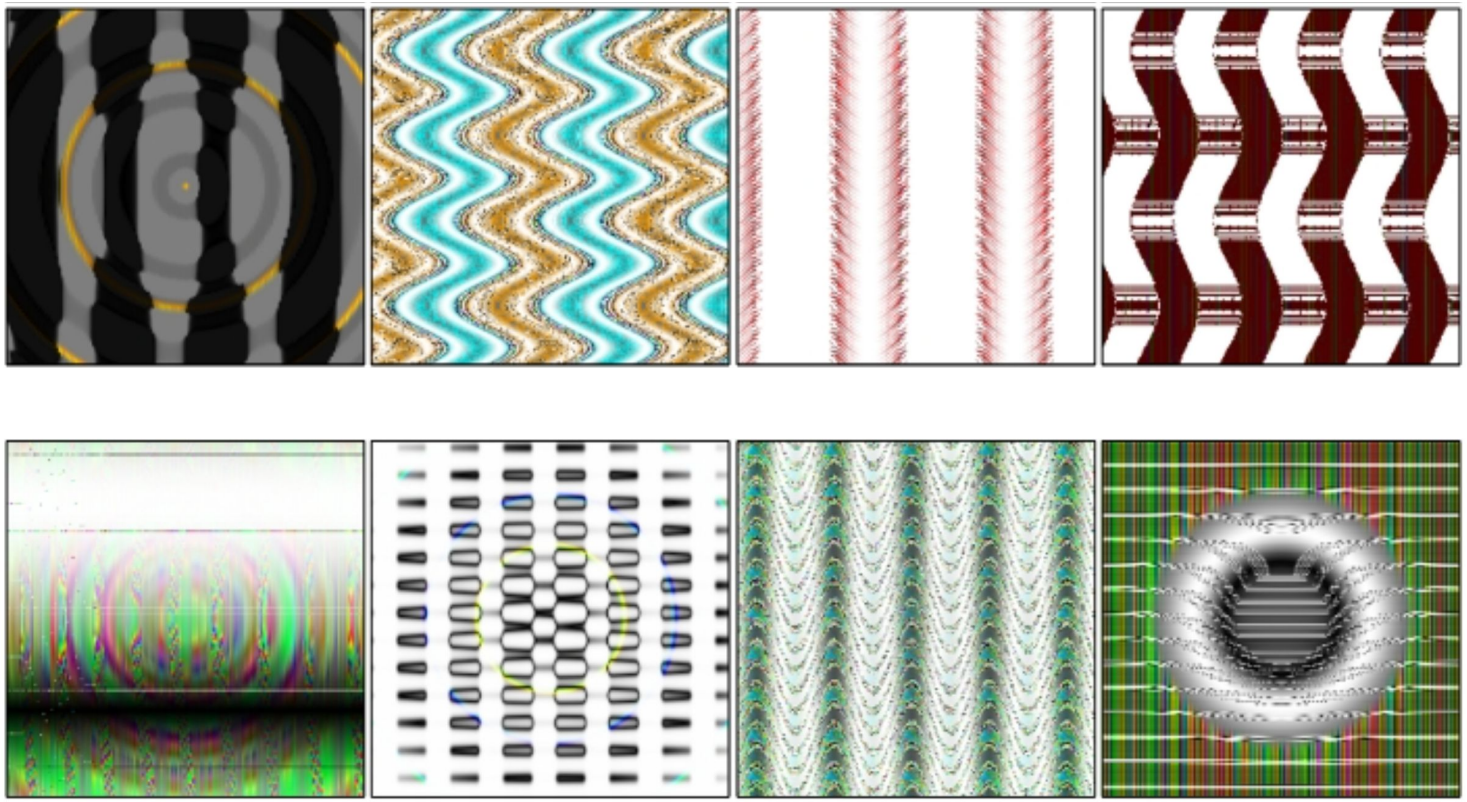
- Authors evolve a compositional pattern-producing network to generate images.

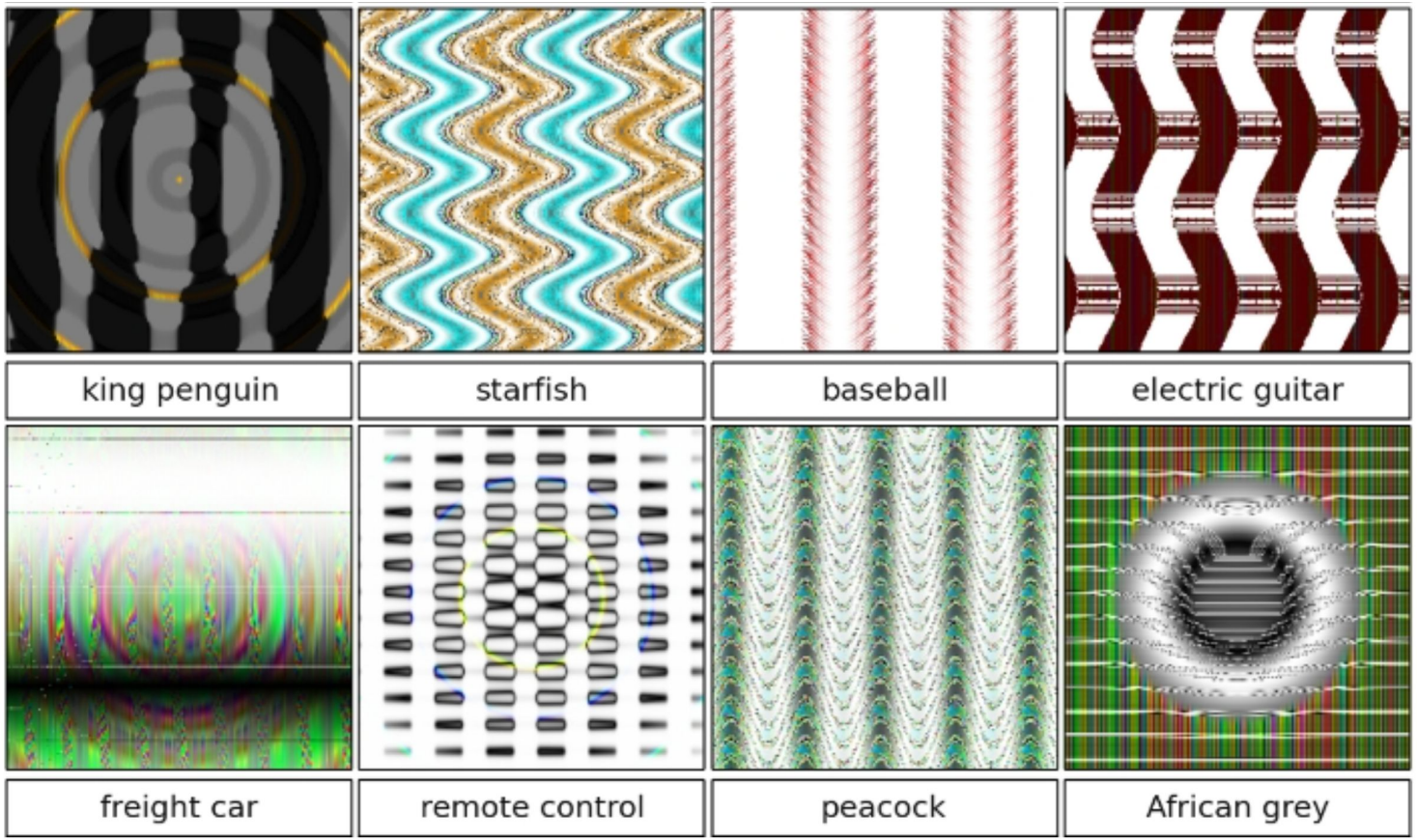


- When optimized with human raters, the CPPN generates recognizable objects.

Optimizing higher-level features

- When optimized against AlexNet, the generated images are not recognizable




king penguin
starfish
baseball
electric guitar
freight car
remote control
peacock
African grey