CSC321 Lecture 22: Adversarial Learning

Roger Grosse

Overview

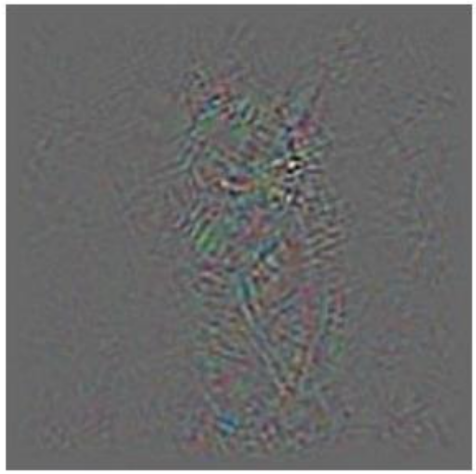
- Two topics for today:
  - **Adversarial examples:** examples carefully crafted to cause an undesirable behavior (e.g. misclassification)
  - **Generative Adversarial Network (GAN):** a kind of generative model which learns to generate images which are hard (for a conv net) to distinguish from real ones

Adversarial Examples

- We've touched upon two ways an algorithm can fail to generalize:
  - overfitting the training data
  - dataset bias (overfit the idiosyncrasies of a dataset)
- But algorithms can also be vulnerable to adversarial examples, which are examples that are crafted to cause a particular misclassification.

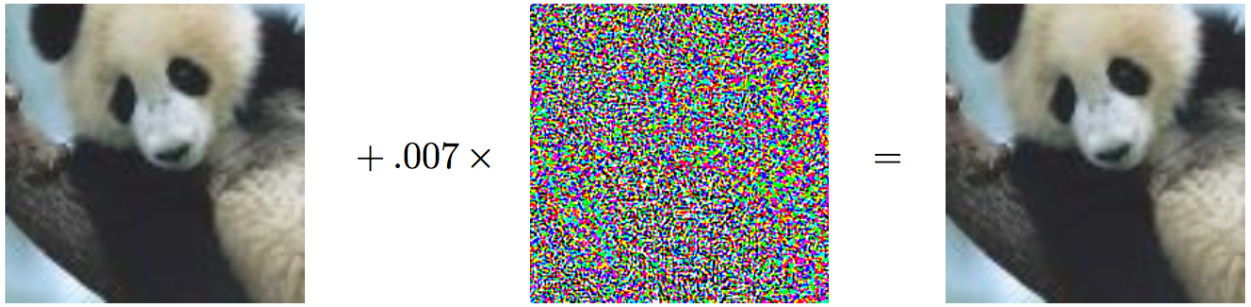
Adversarial Examples

- In our discussion of conv nets, we used backprop to perform gradient descent over the input image:
  - visualize what a given unit is responding to
  - visualize the optimal stimulus for a unit
  - inceptionism
  - style transfer
- Remember that the image gradient for maximizing an output neuron is hard to interpret:



Adversarial Examples

- Now let's say we do gradient ascent on the cross-entropy, i.e. update the image in the direction that *minimizes* the probability assigned to the correct category
  - It turns out you can make an imperceptibly small perturbation which causes a misclassification.
  - Alternatively, do gradient ascent on the probability assigned to a particular *incorrect* category.
- Slight variant: update the image based on the *sign* of the gradient, so that the perturbations of all pixels are small.



$\boldsymbol{x}$

"panda"
57.7% confidence

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

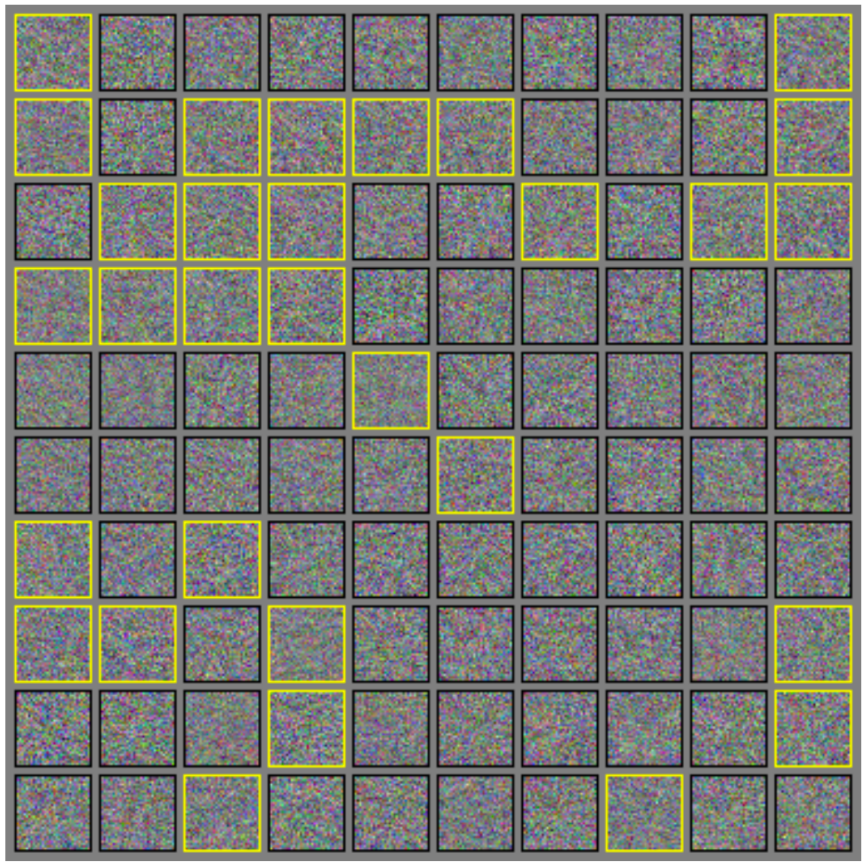
"nematode"
8.2% confidence

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

"gibbon"
99.3 % confidence

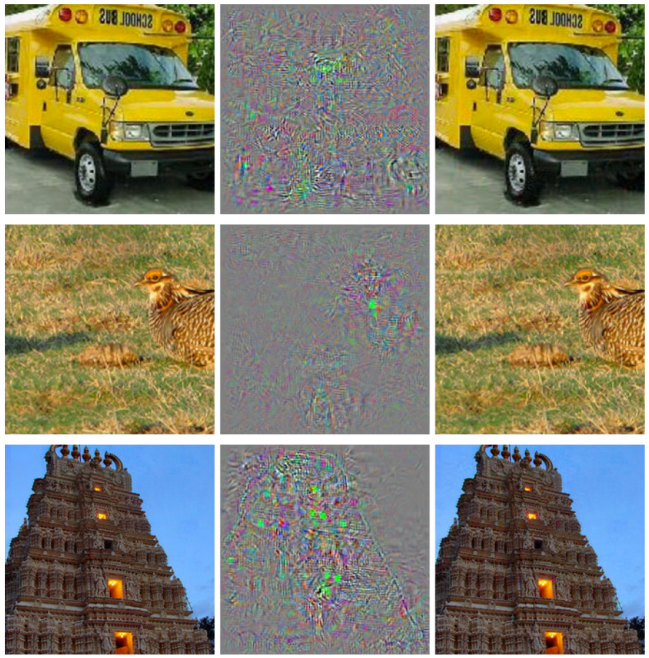
Adversrial Examples

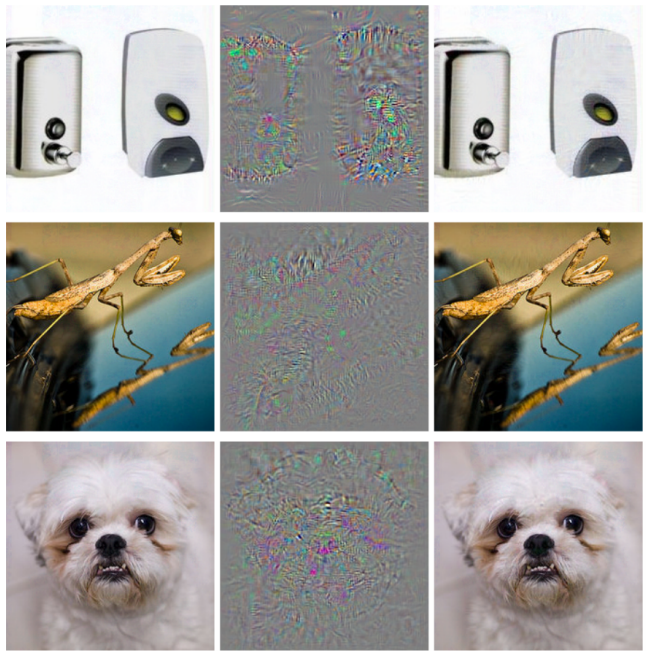
- If you start with random noise and take one gradient step, you can often produce a confident classification as some category.
- The images highlighted in yellow are classified as "airplaine" with $> 50\%$ probability.



Adversarial Examples

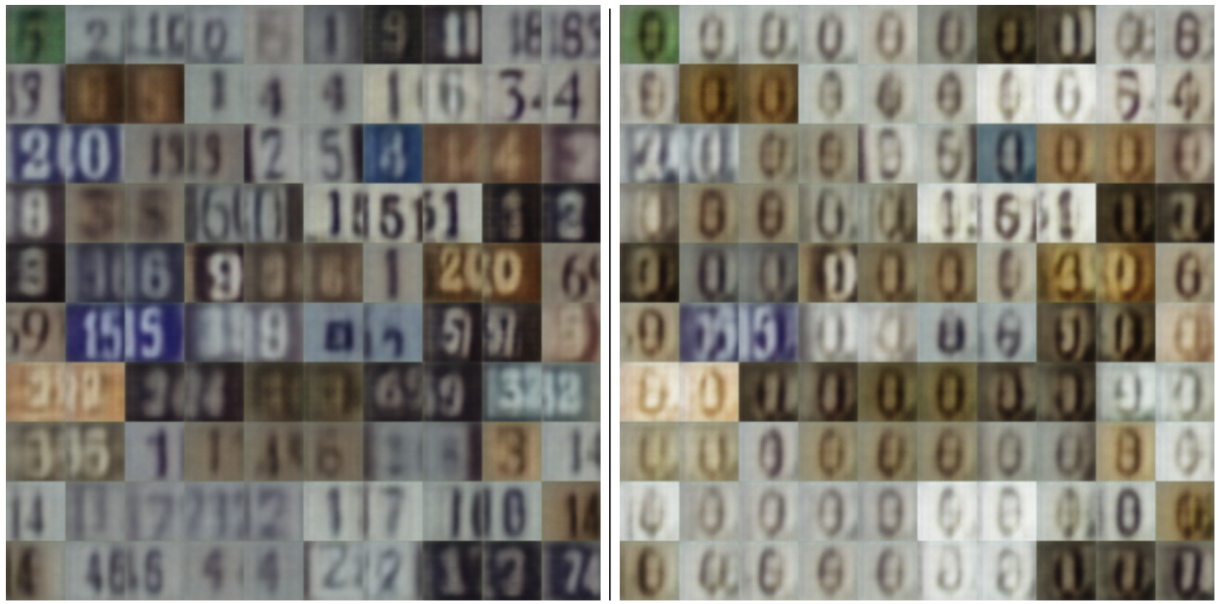
- A variant: search for the image closest to the original one which is misclassifed as a particular category (e.g. ostrich).
- This is called a targeted adversarial example, since it targets a particular category.
- The following adversarial examples are misclassified as ostriches. (Middle = perturbation $\times 10$.)





Adversarial Examples

- Here are adversarial examples constructed for a (variational) autoencoder
- Right = reconstructions of the images on the left



- This is a security threat if a web service uses an autoencoder to compress images: you share an image with your friend, and it decompresses to something entirely different

Adversarial Examples

- The paper which introduced adversarial examples (in 2013) was titled "Intriguing Properties of Neural Networks."
- Now they're regarded as a serious security threat.
  - Nobody has found a reliable method yet to defend against them.
  - Adversarial examples transfer to different networks trained on a disjoint subset of the training set!
  - You don't need access to the original network; you can train up a new network to match its predictions, and then construct adversarial examples for that.
    - Attack carried out against proprietary classification networks accessed using prediction APIs (MetaMind, Amazon, Google)

Adversarial Examples

- You can print out an adversarial image and take a picture of it, and it still works!

(a) Printout (b) Photo of printout (c) Cropped image

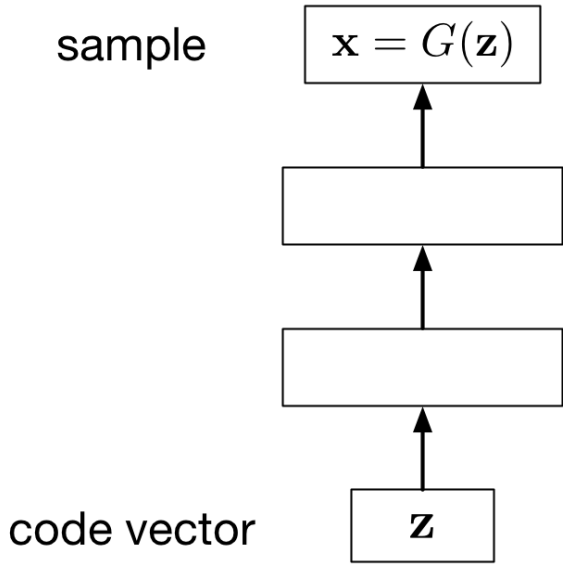
- Can someone paint over a stop sign to fool a self-driving car?

Generative Adversarial Networks

- Now for the optimistic half of the lecture: using adversarial training to learn a better generative model
- Generative models so far
  - simple distributions (Bernoulli, Gaussian, etc.)
  - mixture models
  - Boltzmann machines
  - variational autoencoders (barely mentioned these)
- Some of the things we did with generative models
  1. sample from the distribution
  2. fit the distribution to data
  3. compute the probability of a data point (e.g. to compute the likelihood)
  4. infer the latent variables
- Let's give up on items 3 and 4, and just try to learn something that gives nice samples.

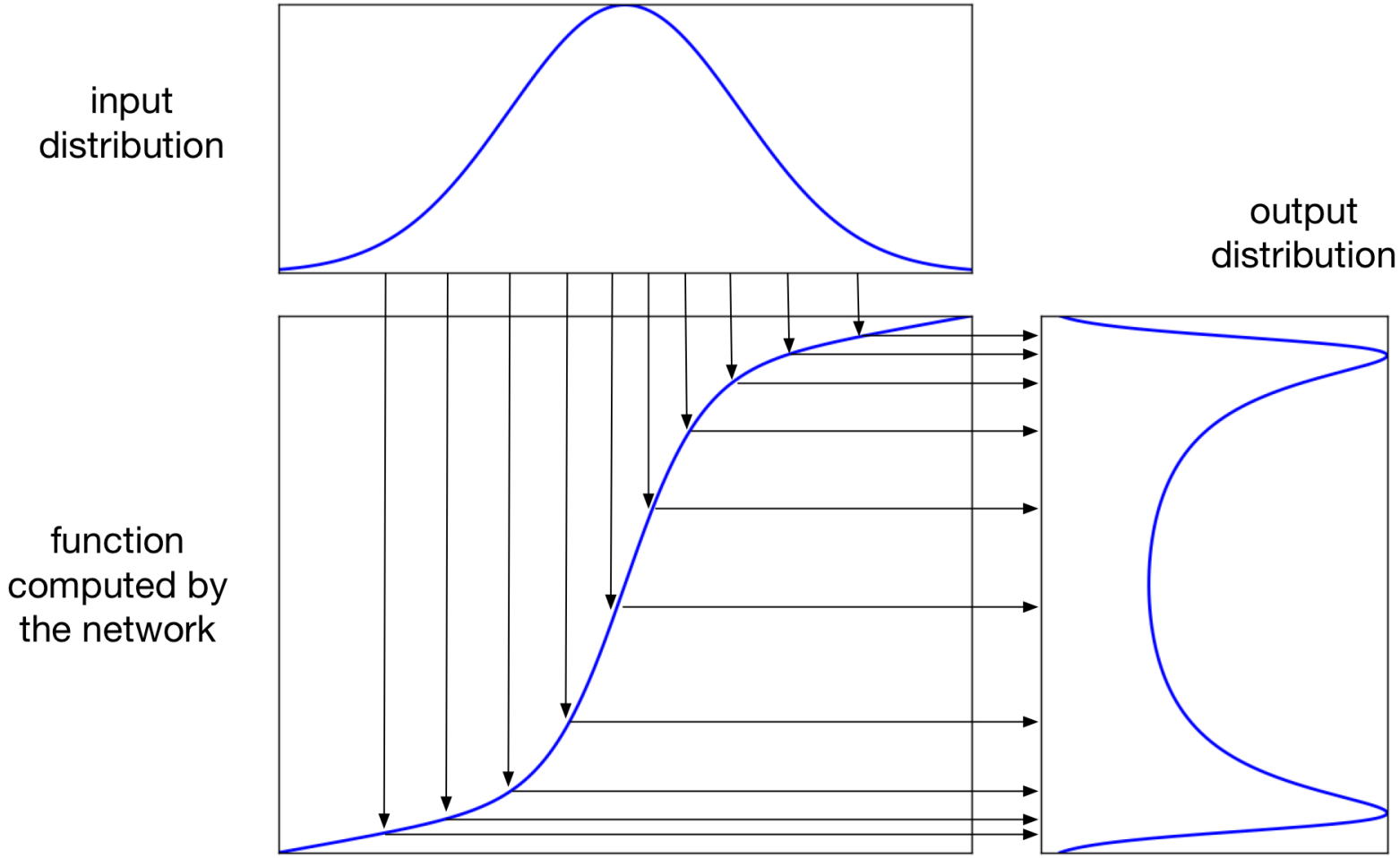
Generative Adversarial Networks

- Density networks implicitly define a probability distribution
- Start by sampling the code vector $\mathbf{z}$ from a fixed, simple distribution (e.g. spherical Gaussian)
- The network computes a differentiable function G mapping $\mathbf{z}$ to an $\mathbf{x}$ in data space



Generative Adversarial Networks

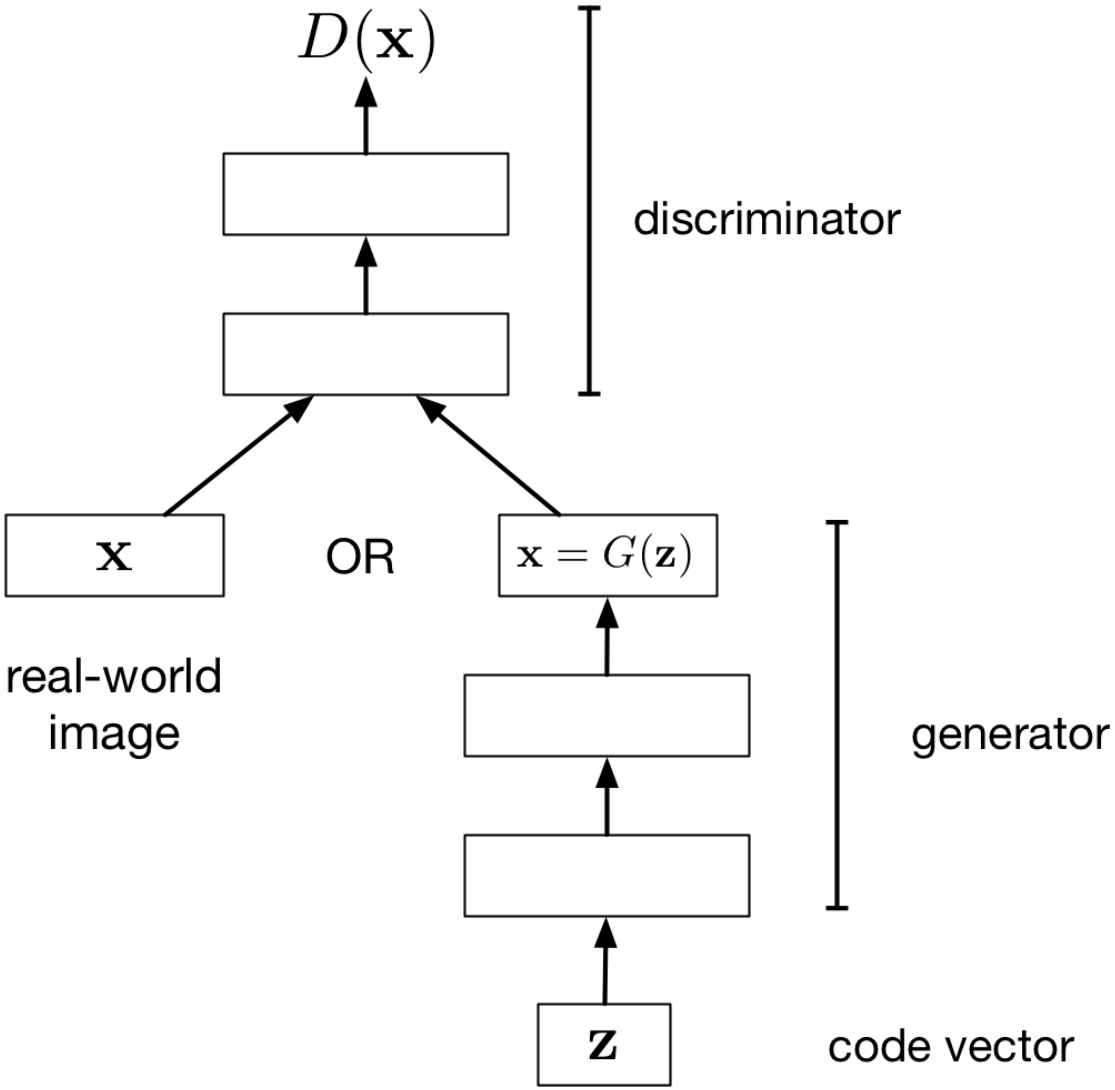
A 1-dimensional example:



Generative Adversarial Networks

- The advantage of density networks: if you have some criterion for evaluating the quality of samples, then you can compute its gradient with respect to the network parameters, and update the network's parameters to make the sample a little better
- The idea behind Generative Adversarial Networks (GANs): train two different networks
  - The generator network is a density network whose job it is to produce realistic-looking samples
  - The discriminator network tries to figure out whether an image came from the training set or the generator network
- The generator network tries to fool the discriminator network

Generative Adversarial Networks



Generative Adversarial Networks

- Let D denote the discriminator's predicted probability of being data
- Discriminator's cost function: cross-entropy loss for task of classifying real vs. fake images

$$\mathcal{J}_D = \mathbb{E}_{\mathbf{x} \sim \mathcal{D}}[-\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}}[-\log(1 - D(G(\mathbf{z})))]$$

- One possible cost function for the generator: the opposite of the discriminator's

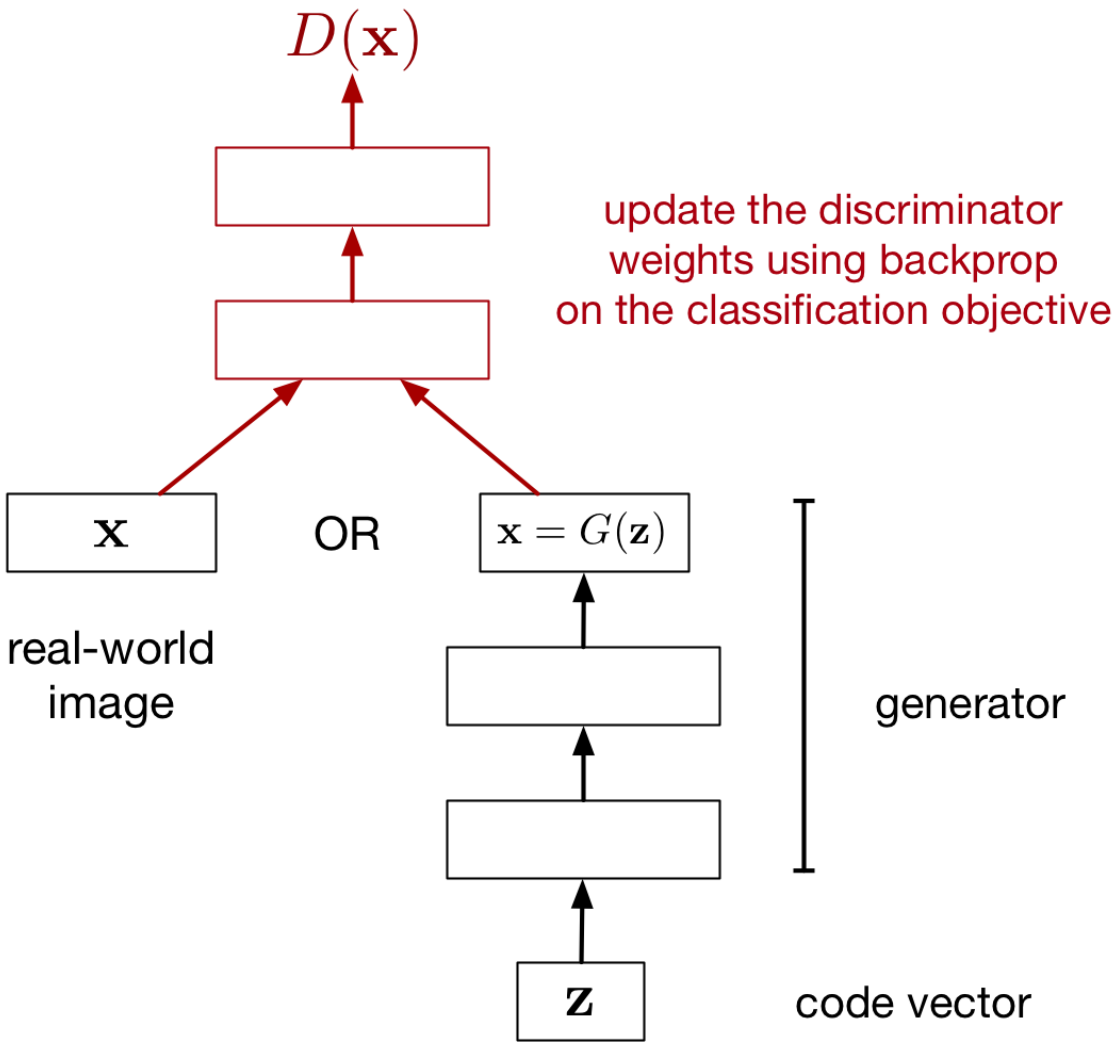
$$\begin{aligned} \mathcal{J}_G &= -\mathcal{J}_D \\ &= \text{const} + \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))] \end{aligned}$$

- This is called the minimax formulation, since the generator and discriminator are playing a zero-sum game against each other:

$$\max_G \min_D \mathcal{J}_D$$

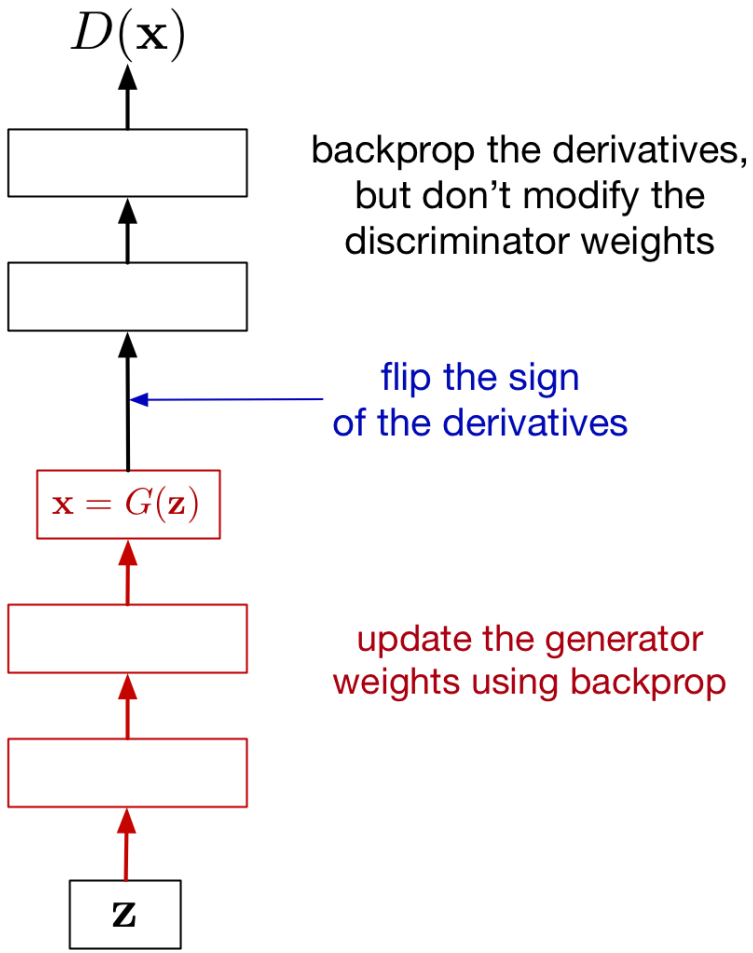
Generative Adversarial Networks

Updating the discriminator:



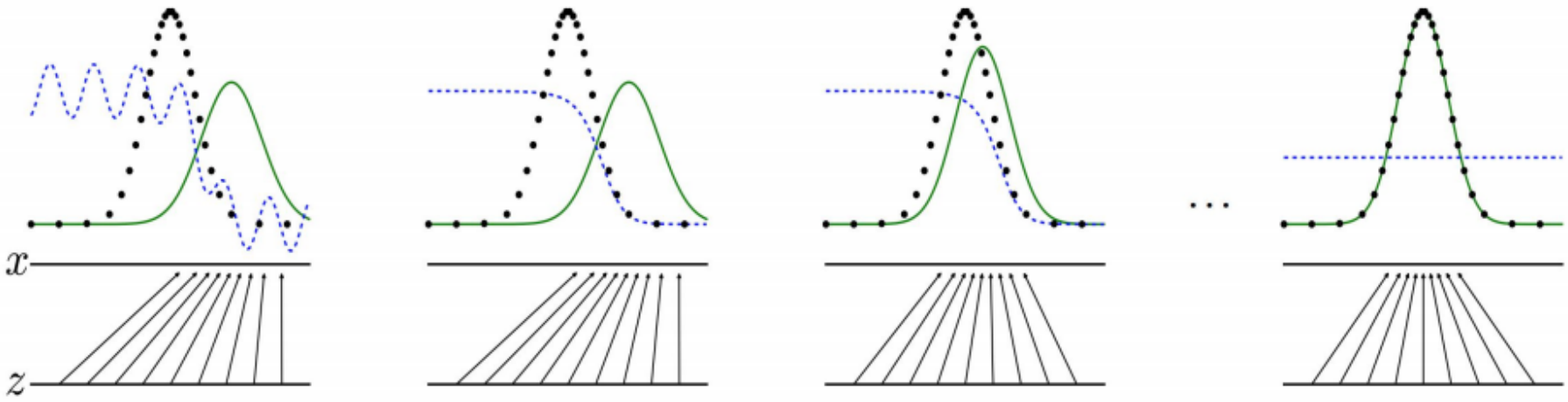
Generative Adversarial Networks

Updating the generator:



Generative Adversarial Networks

Alternating training of the generator and discriminator:



Generative Adversarial Networks

- We introduced the minimax cost function for the generator:

$$\mathcal{J}_G = \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]$$

- One problem with this is saturation.
- Recall from our lecture on classification: when the prediction is really wrong,
  - "Logistic + squared error" gets a weak gradient signal
  - "Logistic + cross-entropy" gets a strong gradient signal
- Here, if the generated sample is really bad, the discriminator's prediction is close to 0, and the generator's cost is flat.

Generative Adversarial Networks

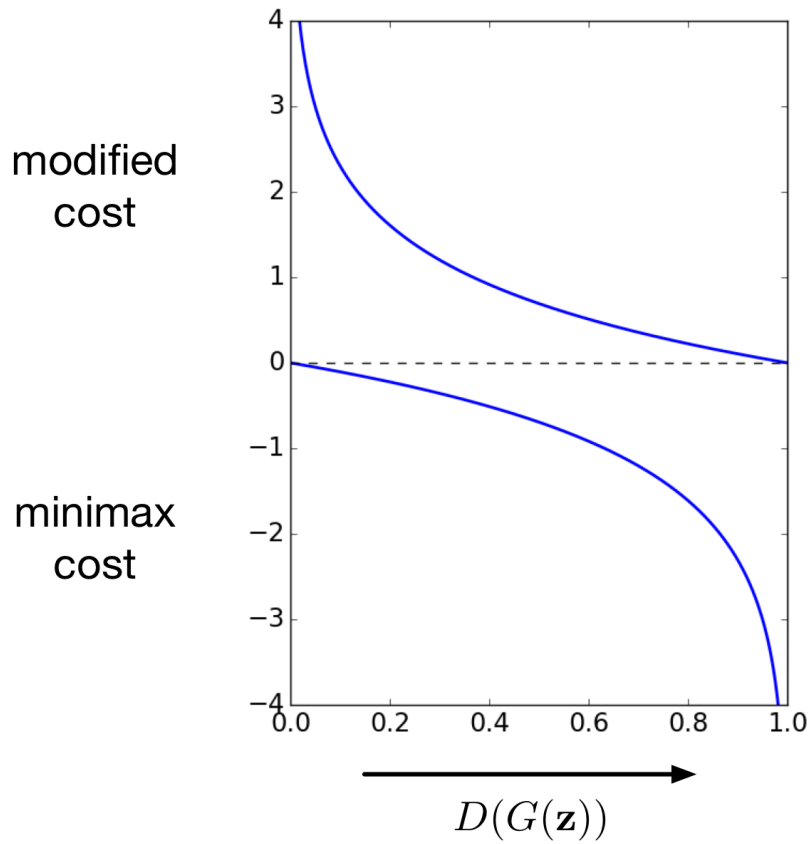
- Original minimax cost:

$$\mathcal{J}_G = \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]$$

- Modified generator cost:

$$\mathcal{J}_G = \mathbb{E}_{\mathbf{z}}[-\log D(G(\mathbf{z}))]$$

- This fixes the saturation problem.



Generative Adversarial Networks

Recall our generative models so far:

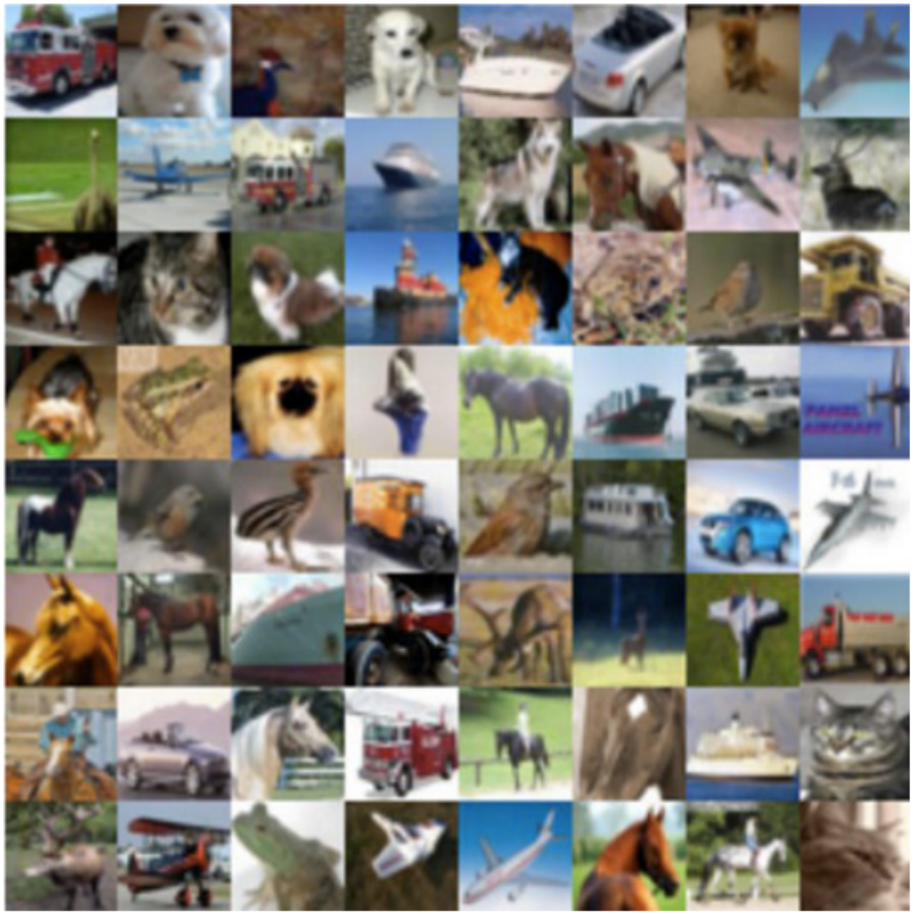
mixture of Bernoullis

RBM

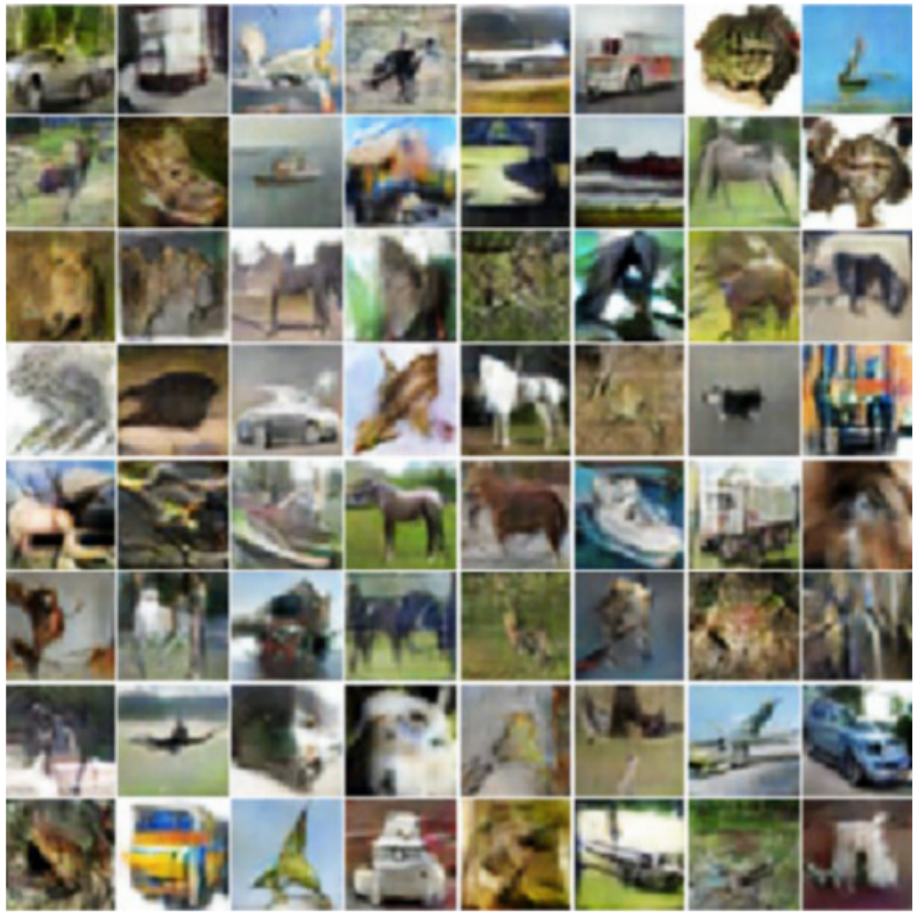
variational autoencoder

Generative Adversarial Networks

GANs produce crisp samples:



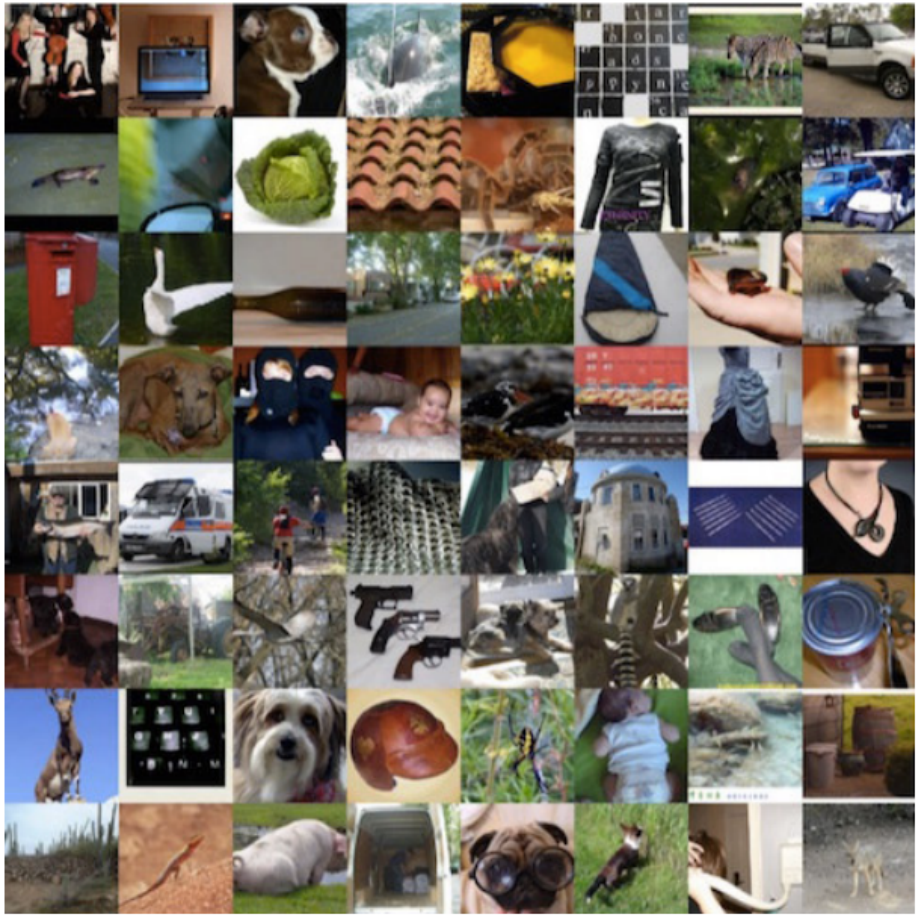
Training Data



Samples

Generative Adversarial Networks

ImageNet:



Generative Adversarial Networks

Generative Adversarial Networks

A variant of GANs was recently applied to supervsed image-to-image translation problems.