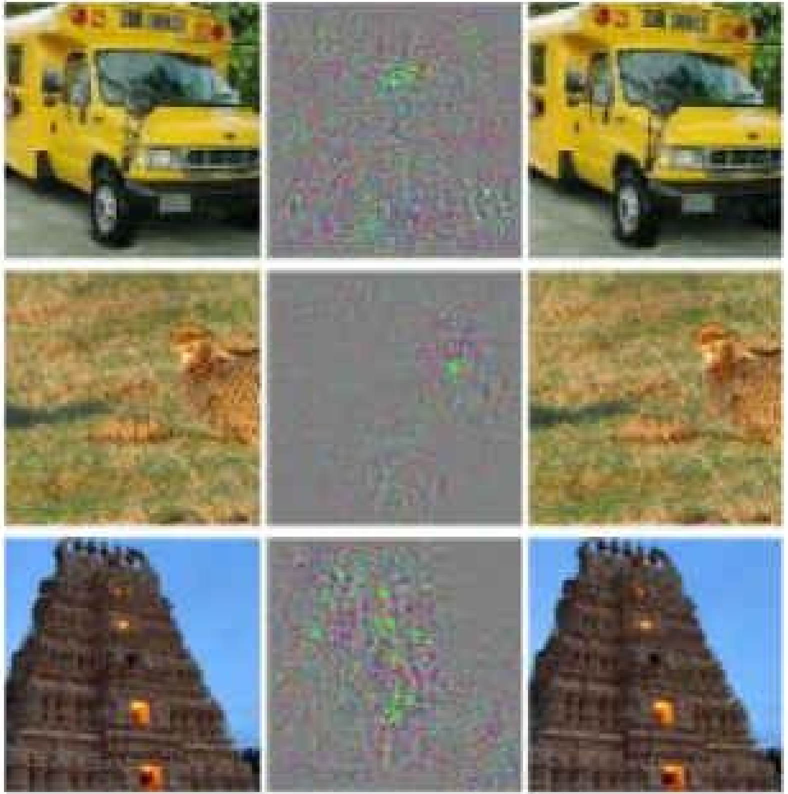
Generative Adversarial Networks

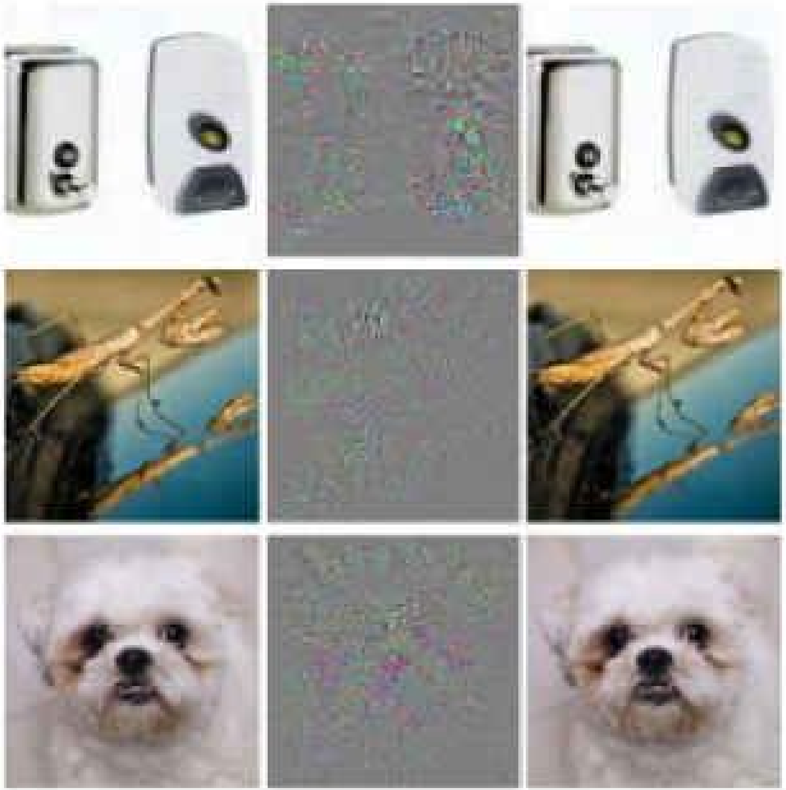
The success of deep learning thus far

- So far, we've been talking about deep learning in the context of classification
- Today we will see that
  - They aren't as good at classification as it might seem
  - They can't generate images at all

Deep Neural Networks are Easily Fooled

Left: correctly classified; Right: Incorrectly classified





Generative Adversarial Networks

- We are going to address both of these in one algorithm:
  - Learn to generate images
  - Stop being susceptible to adversarial examples

Generative Adversarial Networks

- We will define two networks
  - $G(z; \theta_g)$ is a generator
    - Takes input noise z and has parameters θ_g
  - $D(x; \theta_d)$ is a pre-trained discriminator, 0-1 probability
    - Takes input x and has parameters θ_d
- We train D to *maximize* probability of correctly classifying training examples vs. data generated by G
- We train G to *minimize log(1 - D(G(z)))*

Generative Adversarial Networks

- We train D to *maximize* probability of correctly classifying training examples vs. data generated by G; i.e. *D(G(z))*
- We train G to *minimize log(1 - D(G(z)))*
- So *D(G(z))* is trying to reach 1
- But *log(1 - D(G(z)))* is trying to reach -∞
  - In other words, the two networks play a minimax game with value function *V(D, G).* This is just an version of continuous cross-entropy loss with two mini-batches rather than 1.

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**
  for k steps **do**
    • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
    • Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
    • Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for
  • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  • Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

4.1 Global Optimality of $p_g = p_{\text{data}}$

We first consider the optimal discriminator D for any given generator G.

Proposition 1. *For G fixed, the optimal discriminator D is*

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \tag{2}$$

Proof. The training criterion for the discriminator D, given any generator G, is to maximize the quantity $V(G, D)$

$$\begin{aligned} V(G, D) &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x}))dx + \int_{z} p_{\boldsymbol{z}}(\boldsymbol{z}) \log(1 - D(g(\boldsymbol{z})))dz \\ &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x})) + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x}))dx \end{aligned} \tag{3}$$

For any $(a, b) \in \mathbb{R}^2 \setminus \{0, 0\}$, the function $y \to a \log(y) + b \log(1 - y)$ achieves its maximum in $[0, 1]$ at $\frac{a}{a+b}$. The discriminator does not need to be defined outside of $Supp(p_{\text{data}}) \cup Supp(p_g)$, concluding the proof. $\square$

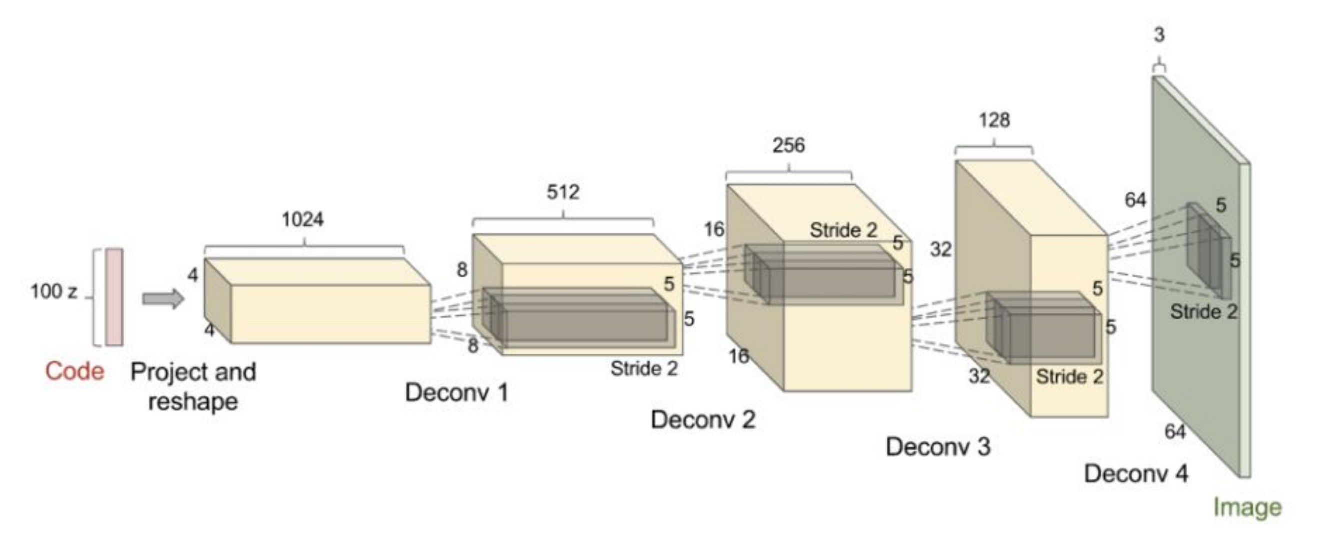
Note that the training objective for D can be interpreted as maximizing the log-likelihood for estimating the conditional probability $P(Y = y|\boldsymbol{x})$, where Y indicates whether $\boldsymbol{x}$ comes from p_{data} (with $y = 1$) or from p_g (with $y = 0$). The minimax game in Eq. 1 can now be reformulated as:

Further Proofs

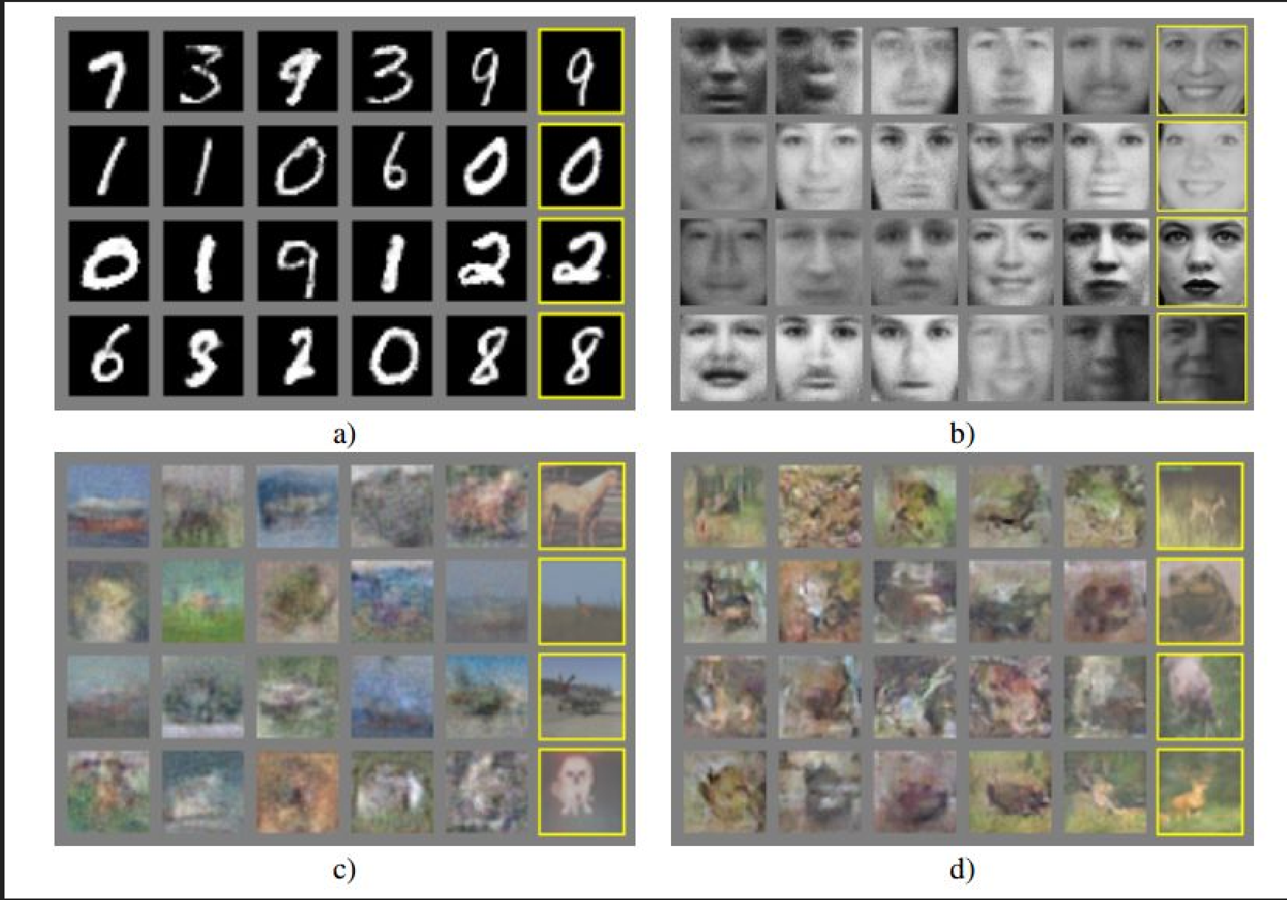
- They then go on to prove that given the algorithm, the generated data will converge to the true data samples
- Steps:
  - Show that the function V(D, G) is convex
  - Show that its global optima is at the location where $p_g = p_{data}$
    - p_g is the probability distribution of samples *G(z)*
    - $z \sim p_z$ = probability distribution of random seeds

Model Architecture

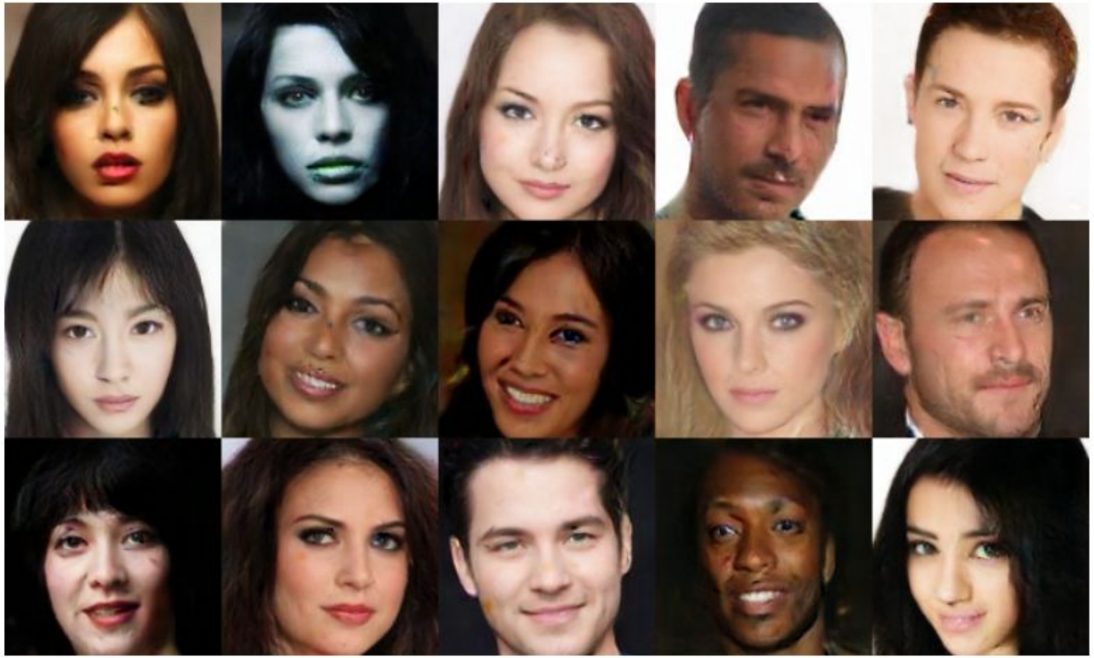
- What are potential shortcomings of this architecture?



Results



BE-GAN

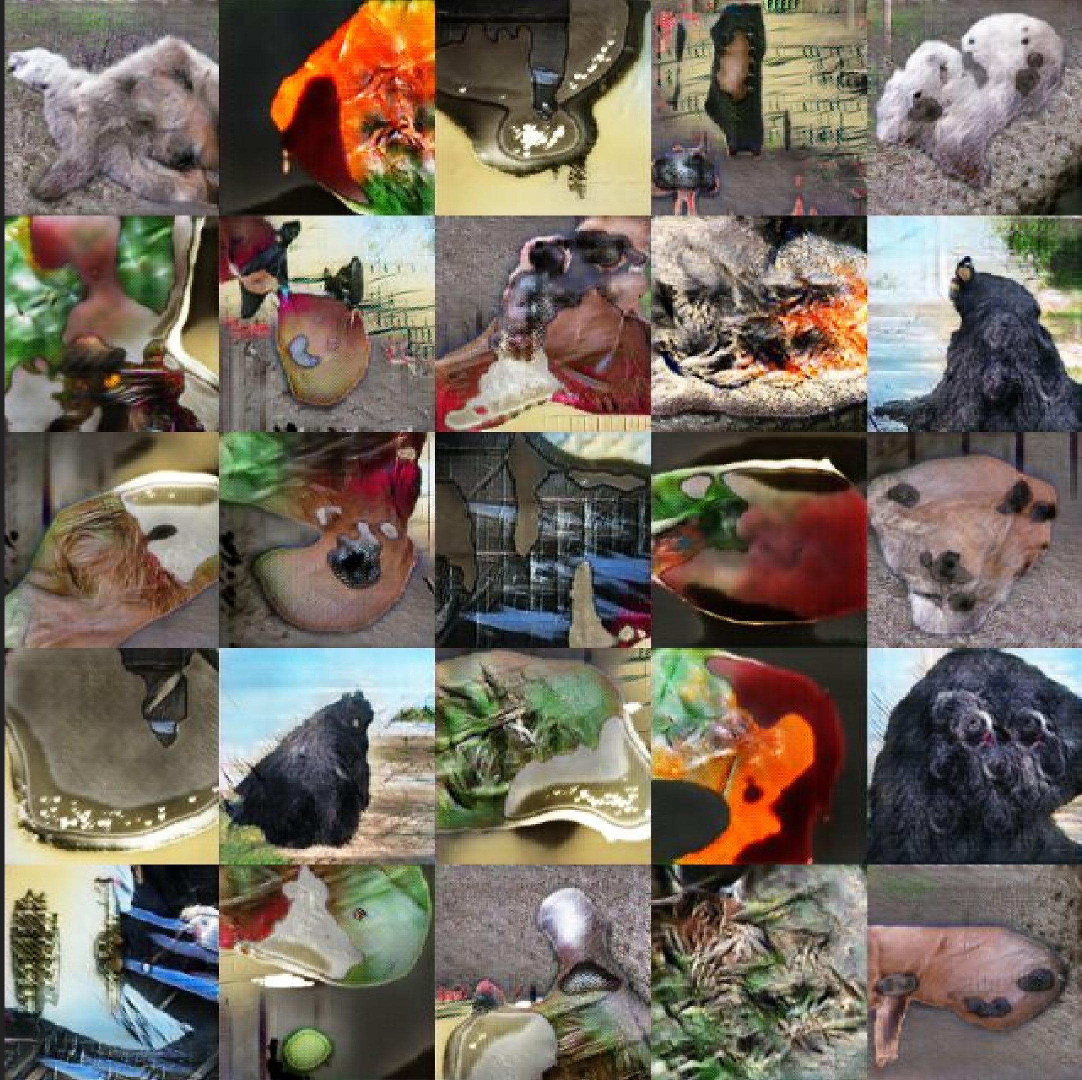


(b) Our results (128x128)

Impact

- Discriminators are less susceptible to adversarial examples
- Generators are ‘as good’ as the true data samples
  - At least in the eyes of the discriminators
- Still not great at regular every day images

- Ian Goodfellow's work at OpenAI has since been focusing on expanding GANs to work on very large datasets
- This one was trained on the ImageNet, which has 10 million samples
- Much harder task than just learning faces
- Is the generator the problem?



Further Reading

by Ian Goodfellow at OpenAI

What we've learned over the semester

- Machine Learning Basics
- Neural Network Math
  - Backpropagation and optimization
    - Batch normalization, dropout, L2 norm
- Neural Network Architectures
  - LSTM, ConvNet, Auto-Encoder
- 5 Difficult (!!!) Papers

If the material from this course makes sense...

- Then you're basically ready for a full-time job working in deep learning
- May need some practice writing the code and learning TensorFlow or Theano
- Next steps for deep learning proficiency
  - Read papers!!!!! I have a GoogleDoc of some interesting ones I can share, just email me.
  - Reddit.com/r/machinelearning
    - Just look at this thread from today for example
  - Participate in Github projects, keep up with the latest code
  - Take linear algebra, probability, math-statistics, calculus