CS4501: Introduction to Computer Vision
Generative Adversarial Networks (GANs)
Today’s Class

• Adversarial Examples – Input Optimization
• Generative Adversarial Networks (GANs)
What we have been doing: Optimize weights in the network to predict bus (correct class).

\[ I \quad y = f(I; w) \quad L(y, bus) \]

\[ w = w - \lambda \frac{\partial L}{\partial w} \]
New Idea: Create Adversarial Inputs by optimizing the input image to predict ostrich (wrong class).

\[ I \quad y = f(I; w) \quad L(y, \text{ostrich}) \]

\[ I = I - \lambda \frac{\partial L}{\partial I} \]

Work on Adversarial examples by Goodfellow et al., Szegedy et. al., etc.
Convnets (optimize input to predict ostrich)

Work on Adversarial examples by Goodfellow et al., Szegedy et. al., etc.
All get predicted as ostrich
Taking the idea to the extreme: Google’s DeepDream

https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

Generate your own in Pytorch: https://github.com/XavierLinNow/deepdream_pytorch
Generative Adversarial Networks (GAN) [Goodfellow et al.]

https://deeplearning4j.org/generative-adversarial-network
Generative Network (closer look)

Generative Network (closer look)

- Deconvolutional Layers
- Upconvolutional Layers
- Backwards Strided Convolutional Layers
- Fractionally Strided Convolutional Layers
- Transposed Convolutional Layers
- Spatial Full Convolutional Layers

Radford et al. 
Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. ICLR 2016
Generative Adversarial Networks (GAN) [Goodfellow et al.]

https://deeplearning4j.org/generative-adversarial-network
Generative Adversarial Networks (GAN) [Goodfellow et al.]

https://deeplearning4j.org/generative-adversarial-network
**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 $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
    • Sample minibatch of $m$ examples $\{x^{(1)}, \ldots, x^{(m)}\}$ from data generating distribution $p_{data}(x)$.
    • Update the discriminator by ascending its stochastic gradient:
      $$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D(x^{(i)}) + \log \left(1 - D\left(G\left(z^{(i)}\right)\right)\right) \right].$$
  end for
  • Sample minibatch of $m$ noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
  • Update the generator by descending its stochastic gradient:
    $$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log \left(1 - D\left(G\left(z^{(i)}\right)\right)\right).$$
end for

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

\begin{align*}
\text{for number of training iterations do} \\
\text{for } k \text{ steps do} \\
\quad \bullet \text{Sample minibatch of } m \text{ noise samples } \{ z^{(1)}, \ldots, z^{(m)} \} \text{ from noise prior } p_g(z). \\
\quad \bullet \text{Sample minibatch of } m \text{ examples } \{ x^{(1)}, \ldots, x^{(m)} \} \text{ from data generating distribution } p_{data}(x). \\
\quad \bullet \text{Update the discriminator by ascending its stochastic gradient:} \\
\quad \quad \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D(x^{(i)}) + \log \left( 1 - D \left( G \left( z^{(i)} \right) \right) \right) \right]. \\
\text{end for} \\
\quad \bullet \text{Sample minibatch of } m \text{ noise samples } \{ z^{(1)}, \ldots, z^{(m)} \} \text{ from noise prior } p_g(z). \\
\quad \bullet \text{Update the generator by descending its stochastic gradient:} \\
\quad \quad \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log \left( 1 - D \left( G \left( z^{(i)} \right) \right) \right). \\
\text{end for} \\
\end{align*}

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.
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 $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
    - Sample minibatch of $m$ examples $\{x^{(1)}, \ldots, x^{(m)}\}$ from data generating distribution $p_{data}(x)$.
    - Update the discriminator by ascending its stochastic gradient:
      \[
      \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D (x^{(i)}) + \log \left( 1 - D \left( G \left( z^{(i)} \right) \right) \right) \right].
      \]
  end for
  - Sample minibatch of $m$ noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
  - Update the generator by descending its stochastic gradient:
    \[
    \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log \left( 1 - D \left( G \left( z^{(i)} \right) \right) \right).
    \]
end for

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

\[\textbf{for number of training iterations do}\]
\[\textbf{for } k \text{ steps do}\]

\[\begin{align*}
&\text{• Sample minibatch of } m \text{ noise samples } \{z^{(1)}, \ldots, z^{(m)}\} \text{ from noise prior } p_g(z). \\
&\text{• Sample minibatch of } m \text{ examples } \{x^{(1)}, \ldots, x^{(m)}\} \text{ from data generating distribution } p_{data}(x). \\
&\text{• Update the discriminator by ascending its stochastic gradient:}\n\end{align*}\]

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

\[\textbf{end for}\]

\[\begin{align*}
&\text{• Sample minibatch of } m \text{ noise samples } \{z^{(1)}, \ldots, z^{(m)}\} \text{ from noise prior } p_g(z). \\
&\text{• Update the generator by descending its stochastic gradient:}\n\end{align*}\]

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

\[\textbf{end for}\]

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.
Generative Adversarial Networks (GAN) [Goodfellow et al.]
Generative Adversarial Networks (GAN) [Goodfellow et al.]

- GANs are hard to train, loss for the discriminator and generator might fluctuate.

- There are many choices for loss, and other auxiliary signals.

- Training of these models is even less well understood than for other deep models.
Basic GAN Results (Example implementation is provided in Pytorch’s examples)

https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html
NVidia’s progressive GANs ICLR 2018
Google’s BigGAN
Google’s BigGAN

Teddy Bear

Microphone

Conditional GANs: Input is not just Noise

Isola et al. CVPR 2017: Image-to-Image Translation with Conditional Adversarial Networks
Conditional GANs: Also Hard to Train

Result they obtained with a regular Fully Convolutional Network

Result they obtained with a U-Net network (with skip-connections)

Isola et al. CVPR 2017: Image-to-Image Translation with Conditional Adversarial Networks
Conditional GANs: Also Hard to Train
GAN Zoo

• https://github.com/hindupuravinash/the-gan-zoo
Questions?