CS4501: Introduction to Computer Vision
Adversarial Examples, and Generative Adversarial Networks (GANs)
Last Class

• Object Detection
  • The YOLO Object Detector (2016)
  • The SSD Object Detector (2016)
  • Mask-RCNN (2017) – to read on your own
  • Semantic Segmentation – Fully Convolutional Models
  • Semantic Segmentation – Transposed Convolutions
  • Semantic Segmentation – Dilated Convolutions
Today’s Class

• Adversarial Examples
• Generative Adversarial Networks (GANs)
Idea: Optimize weights to predict bus.

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

\[ w = w - \lambda \frac{\partial L}{\partial w} \]
New Idea: Optimize input to predict ostrich.

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

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

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.
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.]
Generative Network (closer look)

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

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

http://torch.ch/blog/2015/11/13/gan.html
NVidia’s progressive GANs ICLR 2018
Questions?