GENERAL

- Last class.
- No office hours tomorrow.
- Project Reports Due Dec 22nd at the latest (no extensions!)

STYLE TRANSFER

- Make one image have the "style" or texture quality of another.
- Gatys et al., CVPR 2016:
  - Don’t train a network for this
  - Instead take an existing network and look at the properties of its activations
  - Values of higher layers represent "content": Try to preserve them
  - Covariances of other layers represent style: Try to match them with other image

Set this up as an optimization problem, and minimize with SGD+Backprop from a random init.
STYLE TRANSFER

- One option: choose a parametric form for $p(y|x) = f(y; \theta)$.
- Have a network $g$ that predicts $\theta = g(x)$ for a specific $x$.
- The other option is to train a sampler. A network that given an input $x$, produces “samples” from $p(y|x)$.
- How do you produce samples, or how do you produce multiple outputs for the same input?
- You give your network access to a random generator: a noise source.

GENERAL GENERATIVE ADVERSARIAL NETWORKS

- So far, we have looked at networks that given an input $x$, produce an output $y$.
- All $(x, y)$ pairs come from some joint distribution $p(x, y)$.
- A function that maps $x \rightarrow y$ is then reasoning with the distribution $p(y|x)$.
- And producing a single guess $\hat{y}$ which minimizes $E_{p(\epsilon|y)} L(y, \hat{y})$.
- But if $p(y|x)$ is not deterministic, this expected loss won’t go to zero: Bayes Error.
- What if I didn’t want my network to produce a “best” guess, but tell me about this distribution?

GENERAL GENERATIVE ADVERSARIAL NETWORKS

- Let’s ignore conditional distributions. Consider the task of generating samples from $p_s(x)$.
- You don’t know $p(x)$, but you have training examples that are samples from $p_s(x)$.
- You want to learn a “generator” network $G(z; \theta)$ which
  - Takes in random inputs $z$ from a known distribution $p_z(z)$
  - And produces outputs $x$ from $p(x)$
  - Has learnable parameters $\theta$
- You want to select $\theta$ such that the distribution of $\{G(z; \theta) : z \sim p_z(z)\}$ to match $p_s(x)$.
- But you don’t have the data distribution, only samples from it.
GENERATIVE ADVERSARIAL NETWORKS

- Set this up as a min-max objective with a second network, a “discriminator” $D(x, \phi)$
- The discriminator is a binary classifier. Tries to determine if
  - The input $x$ is “real”, i.e., it came from the training set.
  - Or “fake”, i.e., it was the output of $G$
- Train both networks simultaneously, against the following loss:
  $$L(\theta, \phi) = -\mathbb{E}_{z \sim p_z} \log(1 - D(G(z; \theta); \phi))$$
- This is the cross-entropy loss on the discriminator saying outputs of $G$ should be labeled 0.
- What about examples that should be labeled 1?

\[
G = \arg \max_D \min_{\theta} -\mathbb{E}_{z \sim p_z} \log(1 - D(G(z))) - \mathbb{E}_{x \sim p_x} \log D(x)
\]

Theoretical Analysis

- Let’s say your discriminator and generator had infinite capacity and you had infinite training data.
- For a given input $x$, what should the optimal output of your discriminator $D(x)$ be?
  - Say you know $p_s(x)$.
  - You also know $p_g(x)$ and $G$, and therefore $p_g(x)$: probability of $x$ being an output from the generator.
  $$q = D(x) = \arg \min_q -p_g(x) \log(1 - q) - p_s(x) \log q$$
- What $q$ minimizes this, for $q \in [0, 1]$?
  $$q = \frac{p_g(x)}{p_g(x) + p_s(x)}$$
- Let’s replace $D$ with this optimal value in the loss function, and figure out what $G$ should do.
Remember that \( \mathbb{E} \) also depends on \( x \). In fact, you can replace this as an optimization on \( p_z \).

\[
G = \arg \max \min_G \left( -\mathbb{E}_{z \sim p_z} \log D(G(z)) - \mathbb{E}_{x \sim p_x} \log D(x) \right)
\]

\[
G = \arg \min_G \mathbb{E}_{z \sim p_z} \log \frac{p_z(G(z))}{p_z(G(z)) + p_x(G(z))} + \mathbb{E}_{x \sim p_x} \log \frac{p_x(x)}{p_x(x) + p_z(x)}
\]

- You can replace this as an optimization on \( p_z \).

\[
p_z = \arg \min \int_x \left[ p_z(x) \log \frac{p_z(x)}{p_z(x) + p_x(x)} + p_x(x) \log \frac{p_x(x)}{p_x(x) + p_z(x)} \right] dx
\]

- You can relate this to KL-divergences:

\[
KL(p_z \parallel \frac{p_z + p_x}{2}) + KL(p_x \parallel \frac{p_z + p_x}{2})
\]

- Called the “Jensen-Shannon” Divergence
- Minimized when \( p_x \) matches \( p_z \).

Practical Concerns

- So the procedure is, set up your generator and discriminator networks. Define the loss.
- At each iteration, pick a batch of \( z \) values from a known distribution (typically a vector of uniformly or Gaussian distributed values)
- And a batch of training samples
- Compute gradients for the discriminator and update.
- Compute gradients for the generator, by back-propagating through the discriminator, and update.
Practical Concerns

A common issue is that the discriminator has a much "easier" task
- In the initial iterations, your generator will be producing junk.
- Very easy for the discriminator to identify fake samples with high confidence.
- At that point, \( \log 1 - D(G(z)) \) will saturate.
- No gradients to generator.
- One common approach: minimize \( G \) with respect to a different loss
  - Instead of \( \max - \log(1 - D(G(z))) \)
  - Do \( \min - \log D(G(z)) \)
- View as minimizing cross-entropy wrt wrong label, rather than maximizing wrt true label.

Other approaches:
- Reduce capacity of discriminator
- Make fewer updates to discriminator, or have lower learning rate
- Provide additional losses to generator to help it train: e.g., separate network that predicts intermediate features of the discriminator.
- Other losses: See Wasserstein GANs.
- Also need to be careful how you use Batch Normalization. (Don't let the discriminator use batch statistics to tell real and fake apart!)

\[ L(\theta, \phi) = -\mathbb{E}_{z \sim p_z} \log(1 - D(G(z; \theta); \phi)) - \mathbb{E}_{x \sim p_x} \log D(x; \phi) \]
Generative Adversarial Networks

- Conditional GANs: now want to sample from $p(x|s)$ for a given $s$
- Same adversarial setting but $s$ is given as an input to both generator and discriminator
  - $G(z, s)$ and $D(x, s)$
- Sometimes a noise source is simply replaced by dropout
- Or no noise at all: called an “Adversarial Loss”. The generator is producing a deterministic output, but being trained with a distribution matching loss rather than $L_1$ or $L_2$.
  - Can be useful when the true $p(x|s)$ is multi-modal.
  - Regular networks would average the modes, adversarial loss promotes picking one of the modes.

Adversarial Loss

- Train with
  $$L(G) = \|G(x) - y\|^2 - \lambda \log D(G(x))$$
  $$L(D) = -\log(1 - D(G(x))) - \log D(y)$$
- The GAN loss is unconditional, but there is also a reconstruction loss.
- So the loss says, be close to the true answer, but make your output resemble natural images.

Generative Adversarial Networks

DCGAN: Radford et al.

Generated images from a training set of bedrooms (LSUN dataset).
Neyshabur et al., Stabilizing GAN Training with Multiple Random Projections

Denton et al., Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks
GENERATIVE ADVERSARIAL NETWORKS

Karras et al., PROGRESSIVE GROWING OF GANS FOR IMPROVED QUALITY, STABILITY, AND VARIATION

Figure 1: Our training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4 x 4 pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout the process. Here $N \times N$ refers to convolutional layers operating on $N \times N$ spatial resolution. This allows stable synthesis in high resolutions and also speeds up training considerably. One the right we show six example images generated using progressive growing at 1024 x 1024.

CONDITIONAL GANS

Image-to-Image Translation with Conditional Adversarial Nets

Philip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros
University of California, Berkeley
In CVPR 2017

[Diagram of image-to-image translation process]

Image-to-Image Translation with Conditional Adversarial Nets

Philip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros
University of California, Berkeley
In CVPR 2017

[Diagram of various image-to-image translation tasks]
**UN-SUPERVISED LEARNING**

- For a lot of tasks, it is hard to collect enough training data.
- We saw for the stereo example, how you can have an indirect supervision.
- But in other cases, you have to use transfer learning.
  - Train a network on a large dataset for a related task for which you have ground truth.
  - Remove last layer, and use / finetune feature extractor for new task.
- Researchers are exploring tasks made specifically to transfer from.

---

**Pre-train by learning to add color**

Larsson, Maire, Shakhnarovich, CVPR 2017.
UN-SUPERVISED LEARNING

- Pre-train by solving jigsaw puzzles

Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles

Mehdi Noroozi and Paolo Favaro
Institute for Informatics
University of Bern
noroozi.guido.bern@inf.ethz.ch

UN-SUPERVISED LEARNING

- Pre-train by predicting sound from video

Visually Indicated Sounds

Andrew Owens  Philip Isola  Josh McDermott  Antonio Torralba  Edward H. Adelson  William T. Freeman

Examples of sounds from the Greatful Hits dataset. Click each image to play.

DOMAIN ADAPTATION

- Generate synthetic training data using renderers.
- But networks trained on synthetic data need not generalize to real data.
- (In fact, they may not transfer from high-quality Flickr data to cell-phone camera data)

Problem Setting

- Have input-output training pairs of \((x', y)\) from source domain: renderings/high-quality images/…
- Have only inputs \(x\) from target domain: where we actually want to use this.
- Train a network so that features computed from \(x'\) and \(x\) have the same distribution …
  i.e., use GANs!

DOMAIN ADAPTATION

Adversarial Discriminative Domain Adaptation

Erick S. Tong  University of California, Berkeley
tonges@berkeley.edu
Karen Simoens  Boston University
karen@bu.edu
Julie Hoffman  Stanford University
juliemail@stanford.edu
Trevor Darrell  University of California, Berkeley
trevor@eecs.berkeley.edu

Adversarial Adaptation

Pre-training

Testing

source images
class label
source images
discriminator

domain label

source images
class label
target images
discriminator

target image
class label

Pre-training

Testing

source images
class label
source images
discriminator

domain label

source images
class label
target images
discriminator

target image
class label
We've covered what forms the foundations of state-of-the-art vision algorithms
- Will help you read, understand, and implement vision papers
- But things are changing rapidly: not just new solutions, but new problems
- So keep reading!

**We hope that you have …**

- An understanding of the basic math and programming tools to approach vision problems
- Are as surprised as we are that humans and animals are able to solve this so easily

**That’s all folks!**

<table>
<thead>
<tr>
<th>Digits adaptation</th>
<th>Cross-modality adaptation (NYUD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td><img src="image1.jpg" alt="rgb_images" /> <img src="image2.jpg" alt="hh_images" /></td>
</tr>
<tr>
<td>USPS</td>
<td><img src="image3.jpg" alt="rgb_images" /> <img src="image4.jpg" alt="hh_images" /></td>
</tr>
<tr>
<td>SVHN</td>
<td><img src="image5.jpg" alt="rgb_images" /> <img src="image6.jpg" alt="hh_images" /></td>
</tr>
</tbody>
</table>