GAN Frontiers/Related Methods
Improving GAN Training

Improved Techniques for Training GANs (Salimans, et. al 2016)

CSC 2541 (07/10/2016)  Robin Swanson (robin@cs.toronto.edu)
Training GANs is Difficult

● General Case is hard to solve
  ○ Cost functions are non-convex
  ○ Parameters are continuous
  ○ Extreme Dimensionality
● Gradient descent can’t solve everything
  ○ Reducing cost of generator could increase cost of discriminator
  ○ And vice-versa
Simple Example

- Player 1 minimizes $f(x) = xy$
- Player 2 minimizes $f(y) = -xy$
- Gradient descent enters a stable orbit
- Never reaches $x = y = 0$

Working on Converging

- Feature Mapping
- Minibatch Discrimination
- Historical Averaging
- Label Smoothing
- Virtual Normalization
Feature Matching

- Generate data that matches the statistics of real data
- Train generator to match expected value of intermediate discriminator layer:
  \[
  \|\mathbb{E}_{x \sim p_{data}} f(x) - \mathbb{E}_{z \sim p_z(z)} f(G(z))\|_2^2
  \]
  (Where \(f(x)\) is some activations of an intermediate layer)
- Still no guarantee of reaching \(G^*\)
- Works well in empirical tests
Minibatch Discrimination

- Discriminator looks at generated examples independently
- Can’t discern generator collapse
- Solution: Use other examples as side information
- KL divergence does not change
- JS favours high entropy

(Ferenc Huszár - http://www.inference.vc/understanding-minibatch-discrimination-in-gans/)
And More...

- **Historical Averaging:**\[ \| \theta - \frac{1}{t} \sum_{i=1}^{t} \theta[i] \|^2 \]
- **Label Smoothing:**
  - e.g., 0.1 or 0.9 instead of 0 or 1
  - Negative values set to zero
- **Virtual Batch Normalization:**
  - Each batch normalized w.r.t a fixed reference
  - Expensive, used only in generator
Assessing Results
Ask Somebody

- Solution: Amazon Mechanical Turk
- Problem:
  - “TASK IS HARD.”
  - Humans are slow, and unreliable, and ...
- Annotators learn from mistakes

(http://infinite-chamber-35121.herokuapp.com/cifar-minibatch/)
Inception Score

- Run output through Inception Model
- Images with meaningful objects should have a label distribution \(p(y|x)\) with low entropy
- Set of output images should be varied
- Proposed score:
  \[\exp\left(\mathbb{E}_x \text{KL}(p(y|x) \parallel p(y))\right)\]
- Requires large data sets (>50,000 images)
Semi-Supervision

- We can incorporate generator output into any classifier
- Include generated samples into data set
- New “generated” label class
  - \([\text{Label}_1, \text{Label}_2, \ldots, \text{Label}_n, \text{Generated}]\)
- Classifier can now act as our discriminator

Experimental Results
Generating from MNIST

Semi-Supervised generation without (left) and with (right) minibatch discrimination
Generating from ILSVRC2012

Using DCGAN to generate without (left) and with (right) improvements
Where to go from here
Further Work

  - Generating realistic images of galaxies for telescope calibration

- **MBD for energy based systems:**
Adversarial Autoencoders (AAEs)

Adversarial Autoencoders (Makhzani, et. al 2015)
Variational Autoencoders (VAEs)

- Maximize the variational lower bound (ELBO) of \( \log p(x) \):

\[
\mathcal{L}(\phi, \theta, x) = -D_{KL}(q_{\phi}(z|x) \| p_{\theta}(z)) + \mathbb{E}_{q_{\phi}(z|x)} \left( \log p_{\theta}(x|z) \right)
\]

- Divergence of \( q \) from prior (regularization)
- Reconstruction quality
Motivation: an issue with VAEs

- After training a VAE, we can feed samples from the latent prior \( p(z) \) to the decoder \( p(x|z) \) to generate data points.
- Unfortunately, in practice, VAEs often leave “holes” in the prior’s space which don’t map to realistic data samples.
From VAEs to Adversarial Autoencoders (AAEs)

- Both turn autoencoders into generative models
- Both try to minimize reconstruction error
- A prior distribution $p(z)$ is imposed on the encoder ($q(z)$) in both cases, but in different ways:
  - VAEs: Minimizes $KL(q(z)||p(z))$
  - AAEs: Uses adversarial training (GAN framework)
Adversarial Autoencoders (AAEs)

- Combine an autocoder with a GAN
  - Encoder is the generator, $G(x)$
  - Discriminator, $D(z)$, trained to differentiate between samples from prior $p(z)$ and encoder output ($q(z)$)
- Autoencoder portion attempts to minimize reconstruction error
- Adversarial network guides $q(z)$ to match prior $p(z)$
Draw samples from $p(z)$

Adversarial cost for distinguishing positive samples $p(z)$ from negative samples $q(z)$
Autoencoder

Draw samples from $p(z)$

Adversarial cost for distinguishing positive samples $p(z)$ from negative samples $q(z)$
Adversarial Net

Draw samples from $p(z)$

$\text{Adversarial cost for distinguishing positive samples } p(z) \text{ from negative samples } q(z)$
Training

- Train jointly with SGD in two phases
- "Reconstruction" phase (autoencoder):
  - Run data through encoder and decoder, update both based on reconstruction loss
- "Regularization" phase (adversarial net):
  - Run data through encoder to "generate" codes in the latent space
    - Update $D(z)$ based on its ability to distinguish between samples from prior and encoder output
    - Then update $G(x)$ based on its ability to fool $D(z)$ into thinking codes came from the prior, $p(z)$
Resulting latent spaces of AAEs vs VAEs

AAE vs VAE on MNIST (held out images in latent space)
- First row: Spherical 2-D Gaussian prior
- Second row: MoG prior (10 components)
Possible Modifications
Incorporating Label Info

Draw samples from $p(z)$
Incorporating Label Info

0
1
2
3
4
5
6
7
8
9
Possible Applications
Example Samples

(a) MNIST samples (8-D Gaussian)
(b) TFD samples (15-D Gaussian)
Figure 9: Unsupervised clustering of MNIST using the AAE with 16 clusters. Each row corresponds to one cluster with the first image being the cluster head. (see text)
Disentangling Style/Content

http://www.comm.utoronto.ca/~makhzani/adv_ae/svhn.gif
More Applications...

- Dimensionality reduction
- Data visualization
- ...
  (see paper for more)

Further reading

Nice blog post on AAEs: http://hjweide.github.io/adversarial-autoencoders
Thanks!