Generative Adversarial Networks
“Going From GAN to LAPGAN”

Presentation by Elad Richardson

Some Context – Discriminative Models
- We’ve talked a lot about discriminative models
  - Given some input \( \rightarrow \) Return most likely label
  - From a probabilistic point of view: \( f(x) = \arg\max_y p(y|\mathbf{x}) \)
  - Usually we just model the direct mapping
    “One should solve the problem directly and never solve a more general problem as an intermediate step” (V.N.Vapnik)
  - SVMs, KNNS and Decision Trees are just a few examples for discriminative models

Some Context – Generative Models
- The generative learning algorithms offer a different approach
- Model the joint distribution of our data and labels: \( p(\mathbf{x}, y) \)
- Can still be used to solve classification problems
  \( f(x) = \arg\max_y p(y|x) = \arg\max_y p(x|y)p(y) = \arg\max_y p(y|x, y) \)
- GMMs and Bayesian networks are generative algorithms
- A more challenging task
  - But allows to sample from our data
  - Generating new text, images, etc.

Our Mission
- Neural networks show state-of-the-art result for discriminative tasks
- So let’s use them for generative tasks!
- Given some dataset, try and model \( p(\mathbf{x}) \)
- The Problem: Modeling \( p(\mathbf{x}) \) directly is challenging
- The Trick: We will train our network to sample images directly
  - Let \( z \sim p(z) \) be some input noise
  - Our Generative network would transform \( z \) to \( \mathbf{x} = \mathbf{G}(z) \)
  - To sample an image, simply choose \( z \sim p(z) \) and run \( \mathbf{G} \) over it
  - There is no explicit modeling of \( p(\mathbf{x}) \)

The historical process
- GAN: Goodfellow et al. Jun 2014
- CGAN: Mirza et al. Nov 2014
- LAPGAN: Denton et al. Jun 2015

Generative Adversarial Networks
- The task:
  - Given some dataset with unknown \( p_{\text{data}}(x) \)
  - From \( G \) to output images from \( x \sim p_{\text{data}}(x) \)
  - But how can we train \( G \)?
  - Using an adversarial:
    - A standard discriminator
    - Gets either real images or output of \( G \)
    - Determines whether image is real or generated
  - We will train \( G \) to “cheat” \( D \), while training \( D \) to catch fakes

Some Context – Generative Models
- The generative learning algorithms offer a different approach
- Model the joint distribution of our data and labels: \( p(\mathbf{x}, y) \)
- Can still be used to solve classification problems
  \( f(x) = \arg\max_y p(y|x) = \arg\max_y p(x|y)p(y) = \arg\max_y p(y|x, y) \)
- GMMs and Bayesian networks are generative algorithms
- A more challenging task
  - But allows to sample from our data
  - Generating new text, images, etc.
GAN – Problem Formulation

- Optimize the following problem

\[
\min G \max D \mathbb{V} D, G = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [1 - \log D(G(z))]
\]

- Networks are optimized iteratively
  - Gather \( m \) noise examples \( z(1), \ldots, z(m) \) and \( m \) real examples \( x(1), \ldots, x(m) \)
  - Update \( D \) using standard gradient descent
    \[
    \nabla_D \sum_{i=1}^{m} \log D(x(i)) + \log(1 - D(G(z(i)))
    \]
  - In practice \( G \) was updated only every \( k \) steps

GAN Results

- MNIST Dataset
- Toronto Face Dataset
- CIFAR with Fully Connected
- CIFAR with Convolutional

Where are we

- GAN shows an interesting concept
- Can be trained similarly to regular neural networks
- Shows good results for small datasets
- Fails on more complex ones

GAN Convergence

- Note that a balance must be maintained
  - If \( D \) always wins, \( G \) won’t get a good signal for training
  - If \( D \) never wins, \( G \) won’t generate real looking images

GAN Results

Conditional GANs

- In the basic GAN we have no control over the sampling process

\[
\min_{model}\mathcal{F}(D,G) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [1 - \log D(G(z))]
\]

- CGAN suggests adding some prior \( y \) to the formulation

\[
\min_{model}\mathcal{F}(D,G,y) = \mathbb{E}_{x,y \sim p_{data}(x,y)} [\log D(x,y)] + \mathbb{E}_{z \sim p_z(z)} [1 - \log D(G(z),y)]
\]

- That way we can generate images from a specific class

Where are we
CGANs Results

What's next?

CGAN: Goodfellow et al. Jun 2014
CGAN: Miao et al. Nov 2014
LAPGAN: Denton et al. Jun 2015

- We saw that GAN fails on complex images
- CGAN can be used to add more knowledge to the generation process
- The idea: Run GAN in a coarse-to-fine manner
- Start by generating a small image
- Iteratively generate larger images, conditioned on the smaller ones

Laplacian Pyramid – Quick Recap

- The laplacian pyramid is a scale-based representation of an image
- Assume a downsampling operator \( d(\cdot) \) – blurs and decimates
- Generate a Gaussian pyramid \( \mathcal{G}(I) = [I_0, \ldots, I_K] \) by applying \( d(\cdot) \)
- \( I_j = d(I_{j-1}) \)
- Assume a upsampling operator \( u(\cdot) \) – smooths and expands
- Generate the Laplacian pyramid \( \mathcal{L}(I) = [h_0, \ldots, h_K] \) by the differences
  \( h_j = u(I_{j+1}) - I_j \)
- Note that the image can be reconstructed from \( \mathcal{L}(I) \)
  \[ I_j = h_j - u(I_{j+1}) \]

Laplacian Pyramid

The Training Process

- Each network is trained independently
- The generator gets:
  - The previous scale result
  - Noise vector of matching size
- Each discriminator gets:
  - The previous scale result
  - The generated laplacian or the real one

The Generation Process

- The last network will generate \( I_K \) directly
- Each other network would generate the matching \( h_j \)
- Based on some noise vector and the previous level
Evaluating the Log-Likelihood

- Visually, results look nice
- Traditionally, one would like to evaluate the log-likelihood of the test data
- Problematic since we don't explicitly model $p(x)$
- Can be estimated using a Gaussian Parzen Window
- A standard tool for density estimation
- Distribution is estimated from generated samples
- Which is then used for measuring likelihood of real images

<table>
<thead>
<tr>
<th>Model</th>
<th>CIFAR-10</th>
<th>STL-10</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAN</td>
<td>$-3817 \pm 53$</td>
<td>$-2961 \pm 347$</td>
</tr>
<tr>
<td>LAPGAN</td>
<td>$-1799 \pm 826$</td>
<td>$-2956 \pm 728$</td>
</tr>
</tbody>
</table>

Some More Notes

- Results were evaluated by humans
- Showing that the model is better than GAN, but still far from fooling humans
- How do we know the network creates new images?
- Results were compared to closest neighbors from the train set
- Showing significant changes
LAPGAN was able to improve GAN by breaking down the problem.
The new pipeline is complex, consuming to train.
Results are still far from looking real.
So what’s next?

GAN
Goodfellow et al. Jun 2014

CGAN
Mazuera et al. Nov 2014

LAPGAN
Denton et al. Jun 2015

DCGAN
Radford et al. Jan 2016

Deep Convolutional GANs
- Writers go back to the basic GAN formulation.
- Did some extensive exploration of different models.
- Came up with a few important changes to the architecture:
  - Replace pooling with strided convolution
  - Use batchnorm layers
  - Remove fully connected layers
  - Use ReLU in generator, with TanH in the last layer
  - Use LeakyReLU in discriminator
  - Also, when one network is too strong it stops training

GAN
Goodfellow et al. Jun 2014

CGAN
Mazuera et al. Nov 2014

LAPGAN
Denton et al. Jun 2015

DCGAN
Radford et al. Jan 2016

DCGAN Results

DCGAN Results

DCGAN Results

DCGAN Results

DCGAN Results
Traversing the Noise Space

Is it really useful?

- So generating images is cool, but who cares?
- Can be used to generate infinite data
- The adversarial idea can be incorporated into new problems
  - A recent paper suggests incorporating it into an Autoencoder

Results

Conclusion

- Neural Networks can be used to efficiently solve generation problems
- Still a lot of work to do
  - Though recent papers are getting there!
- Can be used in practical problems
- Sometimes less is more