GANs for Exploiting Unlabeled Data

Improved Techniques for Training GANs
Learning from Simulated and Unsupervised Images through Adversarial Training

Presented by:
Uriya Pesso
Nimrod Gilboa Markevich
“[...] you could not see an article in the press [about AI] without the picture being Terminator. It was always Terminator, 100 percent. And you see less of that now, and that’s a good thing. “

Yann LeCun, Director, Facebook AI
“Generative Adversarial Networks is the most interesting idea in the last ten years in machine learning.”

Yann LeCun, Director, Facebook AI
Presentation Overview

- Motivation
- Intro to GANs
- Paper 1: Semi Supervised Learning
- Paper 2: Simulated and Unsupervised Learning
- Conclusion
Motivation – Compensating for Missing Data

- Labeled data is expensive (time, money, effort)
- Unlabeled data is cheaper
- Unlabeled data still contains information
- How can we utilize unlabeled data?
- GANs!
  - We will present two methods from two papers
Generative Adversarial Networks

- What does it do?
  - Generate **synthetic** data that is indistinguishable from **real** data

- Uses
  - Super-Resolution
  - Text-to-Speech
  - Art
  - Exploiting unlabeled data
    (for training other networks)
Generative Adversarial Networks

- Adversarial Networks – Opponent Networks
  - Generator – Create realistic samples
  - Discriminator – Distinguish generated from real
- Compete with each other
- The only labels are Real/Fake
- Gradient Descent, Train in turns
**Loss Functions**

\[
L^{(D)}(\theta^{(D)}, \theta^{(G)}) = -\mathbb{E}_{x \sim p_{data}} \log(D(x)) - \mathbb{E}_{z \sim \text{noise}} \log(1 - D(G(z)))
\]

\[
L^{(G)} = -L^{(D)}
\]

- Cross-entropy loss function
- \(D(x)\) – probability of \(x\) being real
- \(G(z)\) – generated sample from noise \(z\)
- \(\theta^{(D)}, \theta^{(G)}\) – network parameters
- \(L^{(D)}\) – discriminator loss function
- \(L^{(G)}\) – Generator loss function
Improved Techniques for Training GANs

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen
Objective: We want to train a classifier

We have:
- labeled data => supervised learning
- unlabeled data => unsupervised learning

Combining labeled and unlabeled data => semi-supervised
Semi-Supervised Learning - Idea

- Combine GAN and classifier networks.
- Add a label for the synthetic data - K+1
Semi-Supervised Loss Function

- Cross entropy loss over two distributions

\[ L = -\mathbb{E}_{x, y \sim p_{data}(x, y)} \log p_{model}(y|x) - \mathbb{E}_{x \sim G} \log p_{model}(y = K + 1|x) = L_{supervised} + L_{unsupervised} \]

- \( L_{supervised} = -\mathbb{E}_{x, y \sim p_{data}(x, y)} \log p_{model}(y|x, y < K + 1) \)
- \( L_{unsupervised} = -\mathbb{E}_{x \sim p_{data}(x)} \log (1 - p_{model}(y = K + 1|x)) - \mathbb{E}_{x \sim G} \log p_{model}(y = K + 1|x) \)

- \( p_{data}(x, y) \) - is the probability distribution of the input data.
- \( \log p_{model}(y|x) \) - is the model probability distribution to predict label y from K labels given data X.
Supervised Loss

Where:

- \( p_{\text{data}}(x) \) is the probability distribution of the input data.
- \( \log p_{\text{model}}(y|x) \) is the model probability distribution to predict label \( y \) from \( K \) labels given data \( X \).
Supervised Loss

\[ L_{\text{supervised}} = -\mathbb{E}_{x,y \sim p_{\text{data}}(x,y)} \log p_{\text{model}}(y | x) \]

- \( p_{\text{data}}(x) \) - is the probability distribution of the input data.
- \( \log p_{\text{model}}(y | x) \) - is the model probability distribution to predict label \( y \) from \( K \) labels given data \( X \).
Unsupervised Loss

\[ L^{(D)}(\theta^{(D)}, \theta^{(G)}) = -\mathbb{E}_{x \sim p_{data}} \log(D(x)) - \mathbb{E}_{z \sim \text{noise}} \log(1 - D(G(z))) \]

\[ D(x) = 1 - p_{\text{model}}(y = K + 1 \mid x) \]

\[ L_{\text{unsupervised}} = -\mathbb{E}_{x \sim p_{data}(x)} \log(1 - p_{\text{model}}(y = K + 1 \mid x)) - \mathbb{E}_{x \sim G(z)} \log(p_{\text{model}}(y = K + 1 \mid x)) \]

- \( p_{\text{model}}(y = K + 1 \mid x) \) – the probability distribution to predict that the data x is unreal.
Punchline

\[ L = L_{\text{supervised}} + L_{\text{unsupervised}} \]

- **Classifier**
  \[ L_{\text{supervised}} = -\mathbb{E}_{x,y \sim p_{\text{data}}(x,y)} \log p_{\text{model}}(y | x, y < K + 1) \]

- **GAN**
  \[ L_{\text{unsupervised}} = -\mathbb{E}_{x \sim p_{\text{data}}(x)} \log (1 - p_{\text{model}}(y = K + 1 | x)) - \mathbb{E}_{x \sim G} \log p_{\text{model}}(y = K + 1 | x) \]
Semi-Supervised Learning – Intuition

- How does a classifier benefit from unlabeled data?
- From the unlabeled data, the classifier learns to **focus on the right features**, thus reducing generalization error.
Results
Results - MNIST

Generated

Real
Results - MNIST

<table>
<thead>
<tr>
<th>Model</th>
<th>Number of incorrectly predicted test examples for a given number of labeled samples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20(*)</td>
</tr>
<tr>
<td>DGN [22]</td>
<td>333 ± 14</td>
</tr>
<tr>
<td>Virtual Adversarial [23]</td>
<td></td>
</tr>
<tr>
<td>CatGAN [14]</td>
<td>191 ± 10</td>
</tr>
<tr>
<td>Ladder network [25]</td>
<td>106 ± 37</td>
</tr>
<tr>
<td>Our model</td>
<td>1677 ± 452</td>
</tr>
<tr>
<td>Ensemble of 10 of our models</td>
<td>1134 ± 445</td>
</tr>
</tbody>
</table>

(*) number of labeled samples per class the rest are unlabeled
train size: 60,000
test size: 10,000
Results – CIFAR10
Results – Imagenet

DCGAN
(Not “Ours”)
Results – Imagenet

“Our” Method

Mission Accomplished!
Learning from Simulated and Unsupervised Images through Adversarial Training

CVPR 2017 Best Paper Award

Avish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, Russ Webb
Apple Inc
Motivation

- Labeled data is expensive (time, money, effort)
- Simulated data is cheap 😊
- Alternatively, we could learn using simulated data
Motivation

- Drawback: Simulated data may have artifacts 😊
- We want more realistic synthetic data
- We have unlabeled data
- Let’s build a refiner
Refiner Network / S+U Learning
SimGAN Architecture
SimGAN Architecture
Loss Function – Discriminator

- **Adversarial Loss**

\[
L^{(D)}(\theta^{(D)}, \theta^{(R)}) = -\mathbb{E}_{x \sim p_{\text{real}}(x)} \log(D(x)) - \mathbb{E}_{x \sim p_{\text{simulated}}(x)} \log(1 - D(R(x))]
\]

- \(D(x)\) – probability of \(x\) being real
- \(R(x)\) – refined simulated image
- \(\theta^{(D)}, \theta^{(R)}\) – network parameters
- \(L^{(D)}\) – discriminator loss function

- \(p_{\text{real}}(x)\) – pdf of real images
- \(p_{\text{simulated}}(x)\) – pdf of simulated images
Loss Function – Refiner

- Refiner has two goals
  - Generate realistic samples – Adversarial Loss
  - Preserve the label – Self-regularization

\[
L^{(R)} \left( \theta^{(D)}, \theta^{(R)} \right) = - \mathbb{E}_{x \sim p_{\text{simulated}}(x)} \left[ \ell_{\text{real}} + \lambda \ell_{\text{reg}} \right]
\]

\[
\ell_{\text{real}} = \log \left( D \left( R(x) \right) \right) \hspace{2cm} \ell_{\text{reg}} = \| R(x) - x \|_1
\]

- \( R(x) \) – refined simulated image
- \( \theta^{(D)}, \theta^{(R)} \) – network parameters
- \( L^{(R)} \) – refiner loss function
- \( p_{\text{simulated}}(x) \) – pdf of simulated images
- \( \lambda \) – weight
Minor Improvements

- Are we done?

- Almost. There are two additional mechanisms:
  - Local Adversarial Loss
  - Using a History of Refined Images
Local Adversarial Loss

- Discriminator outputs \( w \times h \) probability map
- The adversarial loss is the sum of the loss over the local patches
- Localization restrains artifacts – the refined image should look real in every patch
Local Adversarial Loss
Using a History of Refined Images

- **Problem**
  - Discriminator only focuses on the latest refined images
  - Refiner might reintroduce artifacts that the discriminator has forgotten

- **Solution**
  - Buffer refined images
Using a History of Refined Images
Using a History of Refined Images
Results
Stages for SimGAN Performance Evaluation

1) Train SimGAN
2) Generate Synthetic Refined Dataset => Qualitative Results
3) Train NN (Estimator) with the Dataset
4) Test NN (Estimator) => Quantitative Results
Results - Performance Evaluation using Gaze Estimator

- Simulated images – UnityEyes, 1.2M
- Real Labeled Dataset – MPIIGaze Dataset, 214K
Process – 1. Train SimGAN

MPIIGaze (without labels)
Process – 2. Trained Refiner Generates DB

Data Set of Labeled Synthetic Images

SimGAN Refiner

Data Set of Labeled Refined Synthetic Images
Qualitative Results
Process – 3. Train Gaze Estimator

Data Set of Labeled Synthetic Refined Images

Gaze Estimator
Process – 4. Test Gaze Estimator

Data Set of Real Images → Gaze Estimator CNN → Output
## Quantitative Results

<table>
<thead>
<tr>
<th>Method</th>
<th>R/S</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support Vector Regression (SVR) [33]</td>
<td>R</td>
<td>16.5</td>
</tr>
<tr>
<td>Adaptive Linear Regression ALR) [23]</td>
<td>R</td>
<td>16.4</td>
</tr>
<tr>
<td>Random Forest (RF) [36]</td>
<td>R</td>
<td>15.4</td>
</tr>
<tr>
<td>kNN with UT Multiview [47]</td>
<td>R</td>
<td>16.2</td>
</tr>
<tr>
<td>CNN with UT Multiview [47]</td>
<td>R</td>
<td>13.9</td>
</tr>
<tr>
<td>k-NN with UnityEyes [43]</td>
<td>S</td>
<td>9.9</td>
</tr>
<tr>
<td>CNN with UnityEyes Synthetic Images</td>
<td>S</td>
<td>11.2</td>
</tr>
<tr>
<td>CNN with UnityEyes Refined Images</td>
<td>S</td>
<td>7.8</td>
</tr>
</tbody>
</table>
Results – Performance Evaluation using Hand Pose Estimator

- NYU hand pose dataset, 73,000 training, 8,000 testing

Diagram:
- Depth Image
- Hand Pose Estimator
- Selected Points Coordinates
Qualitative Result

Unlabeled Real Images  |  Synthetic  |  Refined  |  Simulated images
Quantitative Results

<table>
<thead>
<tr>
<th>Training data</th>
<th>% of images within $d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthetic Data</td>
<td>69.7</td>
</tr>
<tr>
<td>Refined Synthetic Data</td>
<td>72.4</td>
</tr>
<tr>
<td>Real Data</td>
<td>74.5</td>
</tr>
<tr>
<td>Synthetic Data 3x</td>
<td>77.7</td>
</tr>
<tr>
<td>Refined Synthetic Data 3x</td>
<td>83.3</td>
</tr>
</tbody>
</table>
Quantitative Results
Conclusion

- Generative Adversarial Networks are awesome
  - Generator vs. Discriminator

- Unlabeled data can be used for supervised learning
  - Semi-Supervised Learning
    - Classifier combined with Discriminator
    - Train GAN with labeled and unlabeled data
  - Simulated and Unsupervised Learning
    - Train Refiner
    - Generate large synthetic refined dataset