Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
Alec Radford, Luke Metz, and Somit Chantal

Presented by Brandon Theodorou
Presentation Outline

- Related Work and Existing Limitations
- Problem Statement and Motivation
- Model Architecture and Novelties
- Experimental Details
- Results
Related Work

Unsupervised Representation Learning

- Well-studied
- Still relatively unrefined compared to modern results
  - K-Means
  - Autoencoders
  - Ladder Networks
  - Deep Belief Networks
Related Work

Generating Images

• Similarly well explored
• Poor generative output
  • VAEs
  • GANs
  • RNNs with Deconvolutions
• Blurry/Wobbly Images
Related Work

Existing GAN Models

• Exciting architecture and idea

• Generative output quality not yet crisp
  • “Noisy and incomprehensible”

• Unsuccessful using CNNs to model images
  • LAPGAN utilized a different, iterative upscaling approach
  • Problem specifically scaling up CNN architectures from supervised learning literature
Problem Statement

Develop an architecture to utilize CNNs within GANs in order to stabilize training and unsupervisedly learn a strong image representation.
Model Architecture

Eliminate Pooling Layers

• Uses strided convolutions in place of pooling layers

• Allows the network to learn its own upscaling and downscaling algorithms
Remove Fully Connected Layers

• Eliminate fully connected head on top of convolutional layers
  • Extreme is global average pooling
    • Helped stability but hurt convergence rate
  • Final convolution layer instead fed into a single sigmoid layer
  • Only other fully connected layer is initial generation layer matrix multiplication to reshape noise
Using Batch Normalization

- Normalizes the input to each layer to have zero mean and unit variance
- Stabilizes training and improves gradient flow for deep networks
- Helps with mode collapse
- Applied to all but last generator and first discriminator layers

Model Architecture

\[
\begin{align*}
\text{Input: } & \text{ Values of } x \text{ over a mini-batch: } B = \{x_1,...,x_m\}; \\
& \text{Parameters to be learned: } \gamma, \beta \\
\text{Output: } & \{y_i = \text{BN}_{\gamma,\beta}(x_i)\}
\end{align*}
\]

- \[\mu_B \leftarrow \frac{1}{m} \sum_{i=1}^{m} x_i\] // mini-batch mean
- \[\sigma_B^2 \leftarrow \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_B)^2\] // mini-batch variance
- \[\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}\] // normalize
- \[y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i)\] // scale and shift
Model Architecture

Activation Functions

- Tanh for generator output
- ReLU in generator otherwise
- LeakyReLU in discriminator
- Original GAN used Maxout

\[ g(z) = \max(w_1 x + b_1, w_2 x + b_2, \ldots, w_i x + b_i) \]
Model Architecture

Overview

- All CNNs
- No pooling, no fully connected layers
- Utilize batch normalization

Generator

- ReLU (+Tanh) for the generator, LeakyReLU for the discriminator
- Simple changes but the result of lots of experimentation

Discriminator
Experimental Details

Datasets

• 3 Main Datasets
  • LSUN
  • ImageNet-1k
  • Faces
• Only preprocessing is scaling to range \([-1,1]\)
• No Data Augmentation
Experimental Details

Training Specifications

• Mini-Batch SGD
• LeakyReLU slope of 0.2
• Adam Optimizer
• Learning Rate of 0.0002
• Momentum $\beta_1 = 0.5$
Results
Analysis of Possible Memorization

• Analysis of limited training
  • 1 Epoch, Small LR
• Hashing model
• Deduplication and analysis
Results

Classifying CIFAR-10 Using DCGAN Features

• Use DCGAN as feature extractor with linear model on top for supervised learning task

• Discriminator feature maps max pooled to same 4x4 size, concatenated, and flattened to 28672 dimensional vector

• Beat strong K-means benchmark

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>Accuracy (400 per class)</th>
<th>max # of features units</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Layer K-means</td>
<td>80.6%</td>
<td>63.7% (±0.7%)</td>
<td>4800</td>
</tr>
<tr>
<td>3 Layer K-means Learned RF</td>
<td>82.0%</td>
<td>70.7% (±0.7%)</td>
<td>3200</td>
</tr>
<tr>
<td>View Invariant K-means</td>
<td>81.9%</td>
<td>72.6% (±0.7%)</td>
<td>6400</td>
</tr>
<tr>
<td>Exemplar CNN</td>
<td>84.3%</td>
<td>77.4% (±0.2%)</td>
<td>1024</td>
</tr>
<tr>
<td>DCGAN (ours) + L2-SVM</td>
<td>82.8%</td>
<td>73.8% (±0.4%)</td>
<td>512</td>
</tr>
</tbody>
</table>
Results
Classifying SVHN Numbers Using DCGAN Features

• Same setup as CIFAR-10
• Here achieves state of the art
• Makes sure architecture is not the key by supervisedly training CNN with the same architecture

<table>
<thead>
<tr>
<th>Model</th>
<th>error rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNN</td>
<td>77.93%</td>
</tr>
<tr>
<td>TSVM</td>
<td>66.55%</td>
</tr>
<tr>
<td>M1+KNN</td>
<td>65.63%</td>
</tr>
<tr>
<td>M1+TSVM</td>
<td>54.33%</td>
</tr>
<tr>
<td>M1+M2</td>
<td>36.02%</td>
</tr>
<tr>
<td>SWWAE without dropout</td>
<td>27.83%</td>
</tr>
<tr>
<td>SWWAE with dropout</td>
<td>23.56%</td>
</tr>
<tr>
<td>DCGAN (ours) + L2-SVM</td>
<td>22.48%</td>
</tr>
<tr>
<td>Supervised CNN with the same architecture</td>
<td>28.87% (validation)</td>
</tr>
</tbody>
</table>
Results
Exploring the Latent Space

• Pick two random points in the latent space, generate outputs along the connecting line

• Checks for memorization

• Evaluates quality of representation
Results
Visualizing Discriminator Features

• Use guided back propagation to find exemplar activations of learned features

• Can see bedroom features corresponding to LSUN dataset
Results

Removing Features in Generations

• Using manual analysis and logistic regression, identify window features

• During forward pass, dropped all positive values for these features and replaced with noise

• Images do not have windows but remain semantically sound
Results

Performing Vector Arithmetic

- Vector manipulation similar to Word2Vec
- Use average of multiple images rather than single image for stability
Results
Conditional DCGANs

• Trained a conditional version of the model
• Evaluated using a nearest neighbor classifier on the test dataset

<table>
<thead>
<tr>
<th>Model</th>
<th>Test Error @50K samples</th>
<th>Test Error @10M samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlignMNIST</td>
<td>-</td>
<td>1.4%</td>
</tr>
<tr>
<td>InfiMNIST</td>
<td>-</td>
<td>2.6%</td>
</tr>
<tr>
<td>Real Data</td>
<td>3.1%</td>
<td>-</td>
</tr>
<tr>
<td>GAN</td>
<td>6.28%</td>
<td>5.65%</td>
</tr>
<tr>
<td>DCGAN (ours)</td>
<td>2.98%</td>
<td>1.48%</td>
</tr>
</tbody>
</table>
Conclusion

• Propose a CNN-only architecture for GANs which offers more stable training

• Learns strong representations and produces strong image generations

• Remaining work to improve generative capacity, handle mode collapse, and apply to other domains