Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Network
Alec Radford, Luke Metz and Soumith Chintala

Hammad A. Ayyubi
CSE 254

Outline

1. Goals
2. Improvement upon vanilla GANs
3. Inspecting Learned Representations
4. Applying learned features for supervised learning
Goals

1. Using GANs for unsupervised learning.
2. Demonstrating and inspecting the effectiveness of GANs for unsupervised learning.
3. Showing the applicability of learned features to general image representations by evaluating on supervised tasks.

Improvement Upon Vanilla GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Remove fully connected hidden layers for deeper architectures.
- Use batchnorm in both the generator and the discriminator.
- Use ReLU activation and LeakyReLU activation in generator and discriminator respectively.
Inspecting the Learned Representations

Experiments

- LSUN
  - 3M training examples

- Imagenet-1k
  - 1000 classes
  - > 600 img/class
Inspecting the Learned Representations

Experiments

- Faces Dataset
  - Faces scraped from web
  - Names from dbpedia
  - 3M images, 10K people

Inspecting the Learned Representations

- Walking in the latent space
  (Understanding latent space)
Inspecting the Learned Representations (Discriminator)

➢ Visualizing the discriminator features.

Inspecting the Learned Representations (Generator)

➢ Forgetting to Draw Certain Objects (Windows)

Top row: With window
Bottom row: Without window - instead wall or some other object like mirror etc.
Inspecting the Learned Representations

➢ Vector Arithmetic

Inspecting the Learned Representations

➢ Turn Vector
Applying learned features for supervised learning

- **CIFAR-10**
  - 10 classes, with 6000 images per class

<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>

Applying learned features for supervised learning

- **SVHN (StreetView House Numbers dataset)**
  - 10 classes, 1 for each digit.
  - 73257 digits for training, 26032 digits for testing

<table>
<thead>
<tr>
<th>Model</th>
<th>error rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNN</td>
<td>77.93%</td>
</tr>
<tr>
<td>T SVM</td>
<td>66.55%</td>
</tr>
<tr>
<td>M1+KNN</td>
<td>65.63%</td>
</tr>
<tr>
<td>M1+T SVM</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 archi</td>
<td>28.87% (validation)</td>
</tr>
</tbody>
</table>
Commentary on GANs

How does it compare with Autoencoders?

❖ Use Autoencoders to approximate data probability distributions, while GANs are more suitable for drawing/generating realistic samples.

❖ Autoencoders are much easier to train.

GANs achilles heel - Mode Collapse
What is mode collapse?

- The generator learns to map several different input z values to the same output point.
- In practice, partial mode collapse is more common. Partial mode collapse occurs when multiple images that contain the same color or texture themes are generated, or multiple images containing different views of the same dog are generated.

What happens during Mode Collapse?

Recall minmax game of GANs: \( G^* = \min_G \max_D V(G, D) \)

\( G^* \) here draws samples from the data distribution.

What happens often times: \( G^* = \max_D \min_G V(G, D) \)

The generator is thus asked to map every z value to the single x coordinate that the discriminator believes is most likely to be real rather than fake.

Simultaneous gradient descent doesn’t really favor one over other. We use it in the hope that it behaves like min max rather than max min.
**Cause of Mode Collapse**

It was earlier believed that mode collapse was rooted in the Jensen-Shannon Divergence that GANs reduce. It’s similar to Reverse KL, while maximum likelihood reduce KL divergence loss.

Later it was disproved by multiple works:

1. Maximum Likelihood also produces sharp images (f-GAN)
2. GANs often produce very few modes; fewer than limitations imposed by Reverse KL. Reverse KL prefers as many modes as the data allows not fewer modes in general.

It seems more like a product of training strategy. Nevertheless, it’s an open research question.

---

**Possible ways to address Mode Collapse**

- **Minibatch Features (Salimans et al.)**
  - Compare an example to a minibatch of generated samples and a minibatch of real samples.
  - Measuring distance in latent space helps the discriminator to detect if a sample is unusually similar to other generated samples. (Remember, batchnorm improved GANs performance; it’s for the same reason)

- **Unrolled GANs (Metz et al.)**
  - Ideally we would like to solve \( G^* = \arg\min_G \max_D V(G, D) \) as cost function for \( G^* \). In practice, during simultaneous update of discriminator and generator we ignore the max operation.
  - Fully maximizing discriminator in each gradient descent step of generator may become intractable. Metz et al show maximizing discriminator for \( k \) (10 or fewer) steps before minimizing discriminator works well.
Thank you!
Questions?

References

5. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization, Sebastian Nowozin, Botond Cseke, Ryota Tomioka.