AIL 862

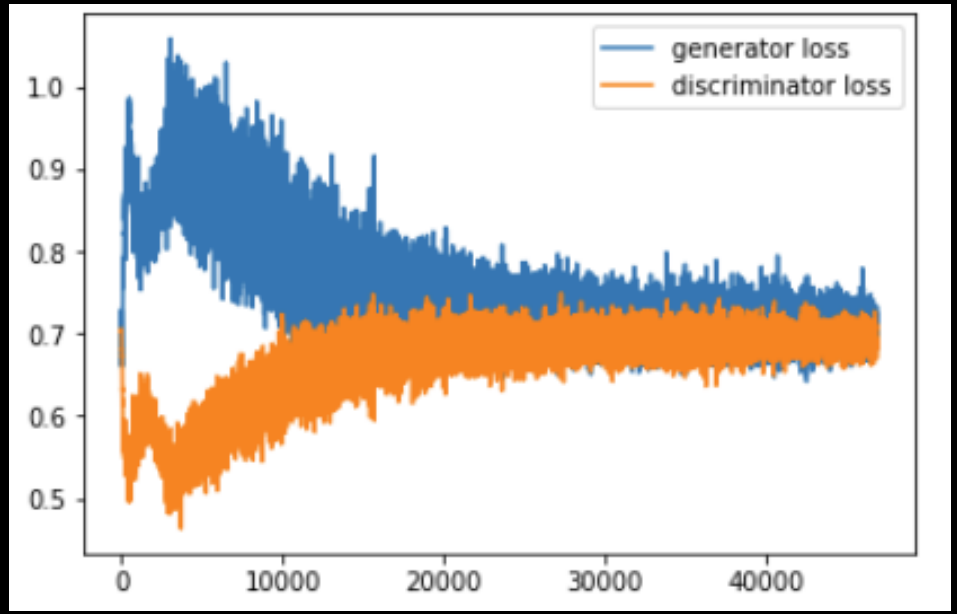
Lecture 11

GAN Issues

- Too powerful discriminator: generator cannot engage in “game”

GAN Issues

- Oscillation in discriminator and generator losses.
- Finding the right balance between the two networks is crucial for successful training.

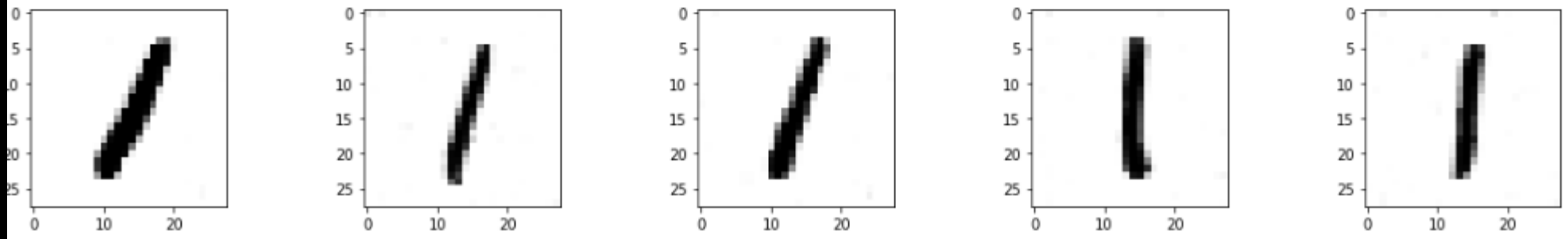


https://github.com/rasbt/stat479-deep-learning-ss19/blob/master/L17_gans/code/gan-cnn-mnist.ipynb

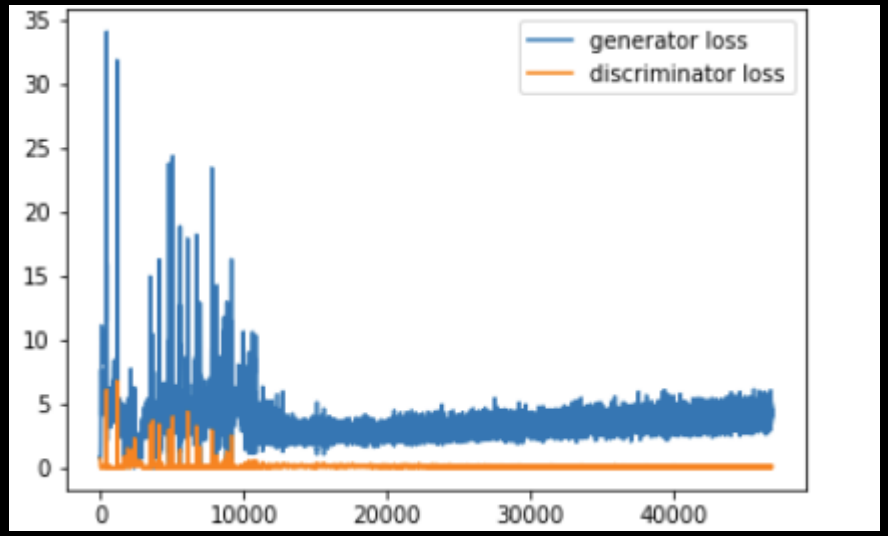
GAN Issues

- Mode collapse: The generator may collapse to producing only a limited set of samples, failing to capture the true diversity of the data distribution. This leads to a lack of variety in the generated outputs.

```
##########################
### VISUALIZATION
##########################


model.eval()
# Make new images
z = torch.zeros((5, LATENT_DIM)).uniform_(-1.0, 1.0).to(device)
generated_features = model.generator_forward(z)
imgs = generated_features.view(-1, 28, 28)

fig, axes = plt.subplots(nrows=1, ncols=5, figsize=(20, 2.5))


for i, ax in enumerate(axes):
    axes[i].imshow(imgs[i].to(torch.device('cpu')).detach(), cmap='binary')
```



https://github.com/rasbt/stat479-deep-learning-ss19/blob/master/L17_gans/code/gan-halfcnn-mnist-mode-collapse.ipynb



https://github.com/rasbt/stat479-deep-learning-ss19/blob/master/L17_gans/code/gan-halfcnn-mnist-mode-collapse.ipynb

Addressing issues: Reduced discriminator update

Update the discriminator less frequently than the generator (e.g., train the generator for multiple steps before updating the discriminator). This can give the generator more opportunities to improve.

Addressing issues: Buffer of samples

Introduce a buffer of previously generated images and mix them with current fake samples when training the discriminator. This prevents the discriminator from overfitting to the most recent generator outputs.

Learning from Simulated and Unsupervised Images through Adversarial Training

Addressing issues: Progressive training

Start training the GAN on low-resolution images and progressively increase the resolution. This allows the generator to learn simpler patterns first.

Progressive Growing of GANs for Improved Quality, Stability, and Variation

Addressing issues

- Augmenting the training data

- Simple techniques like weight clipping (or more advanced like spectral normalization)

Discriminator at different scales

- Local and global discriminator

Globally and locally consistent image completion. (Not a GAN paper)

Mixture of GANs

By building a strong generative model by combining T individual GANs, trained sequentially. After training the first GAN on the original data, each subsequent GAN focuses on the data points the previous models struggled with by reweighting the training samples. At each step, a mixture weight determines how much the new GAN contributes to the overall model. The final output is a mixture of all the GANs, with each contributing based on its mixture weight.

AdaGAN: Boosting Generative Models

What does latent code represent

StyleGAN – intermediate latent code

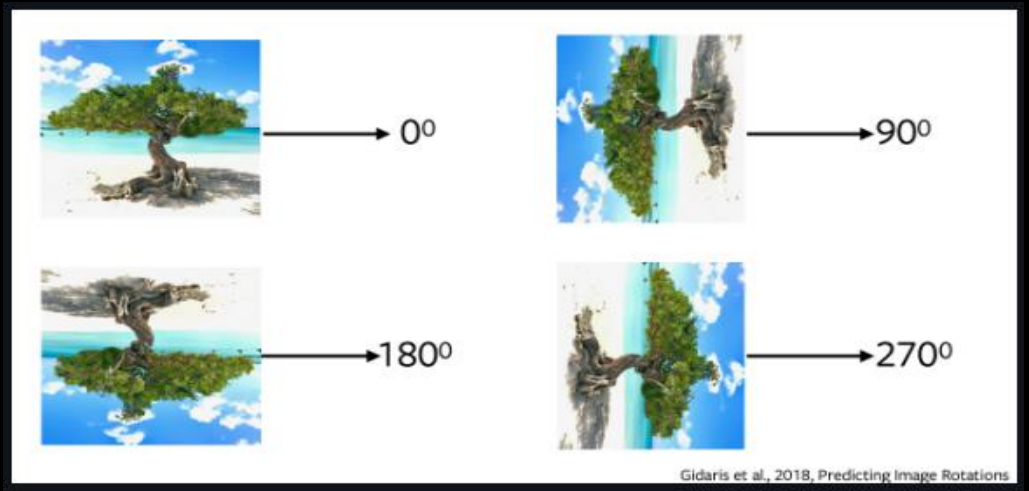
Self-Supervised Learning

- Supervised learning tasks have pre-defined (and generally human-provided) labels.
- Unsupervised learning has just the data samples without any supervision, label or correct output.
- Self-supervised learning derives its labels from a co-occurring modality for the given data sample or from a co-occurring part of the data sample itself.

Pre-Text Task

- The pretext task is the self-supervised learning task solved to learn visual representations.

- E.g., Rotation of images



Assumption

- Accuracy in pre-text tasks is closely linked to accuracy in downstream task.

- Generally, more difficult pre-text task will satisfy the assumption better.

Context Prediction, 2016

- Key Idea: By predicting the relative position of image patches, the model learns to understand object structures and scene layouts.

Context Prediction

- Patch Extraction: Extract a central patch and one of its eight neighboring patches. This results in pairs of patches with known spatial relationships.

- Prediction Task: Train a CNN to predict the position of the neighboring patch relative to the central patch. The network learns to classify the relative position into one of eight possible categories.

Context Prediction

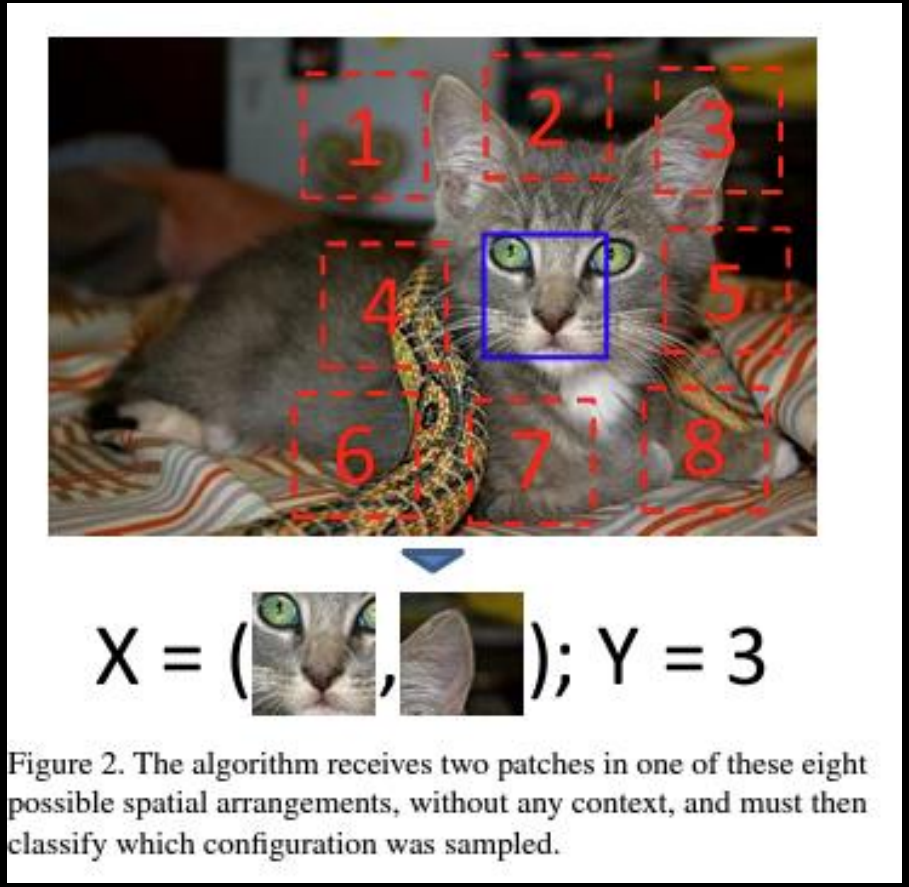
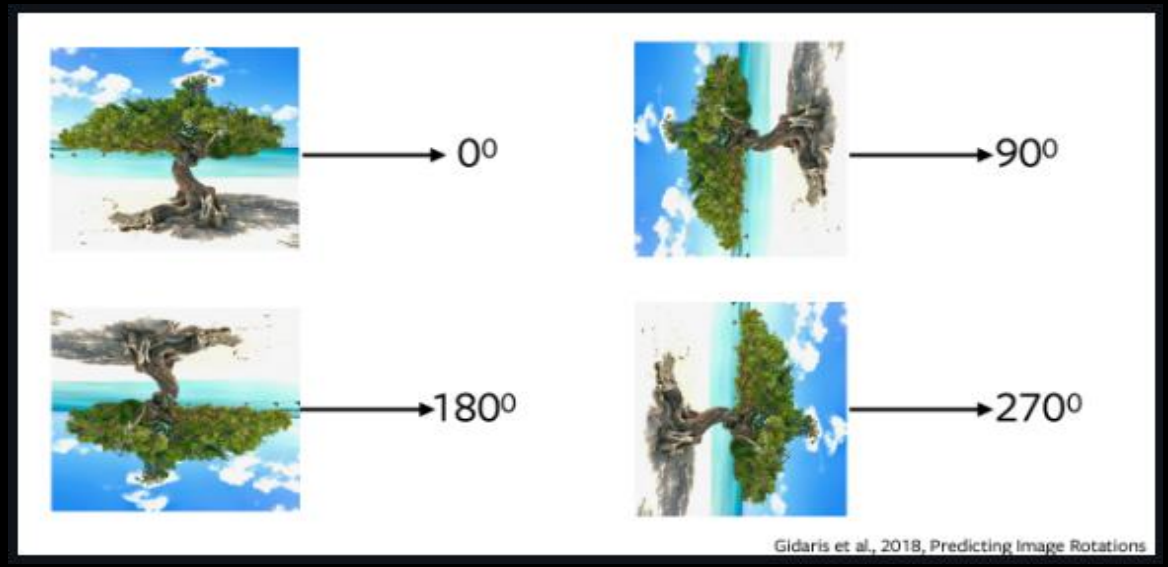


Figure 2. The algorithm receives two patches in one of these eight possible spatial arrangements, without any context, and must then classify which configuration was sampled.

Unsupervised Visual Representation Learning by Context Prediction

Rotation of Images



Simple Experiment on MNIST

Random initialization, freeze everything except FC layers

```
model = Net()

for name, param in model.named_parameters():
    if not name.startswith('fc'):  # Freezing everything except fc layers
        param.requires_grad = False
model = model.to(device)
```

Random initialization, freeze everything except FC layers

```
model = Net()

for name, param in model.named_parameters():
    if not name.startswith('fc'):  # Freezing everything except fc layers
        param.requires_grad = False
model = model.to(device)
```

Accuracy after 1 epoch of training = 95%

Pre-train based on simple rotation

```
for batch_idx, (data, target) in enumerate(train_loader):


    pretextTarget = torch.randint(0, 2, (target.shape))    ##if low and high are 0 ar


    for pretextIter in range(target.shape[0]):
         if pretextTarget[pretextIter]==1:
           thisImage = data[pretextIter,:,:,:]
           thisImageTransposed = torch.transpose(thisImage,1,2)
           data[pretextIter,:,:,:] = thisImageTransposed



    data, pretextTarget = data.to(device), pretextTarget.to(device)
    optimizer.zero_grad()
    output = model(data)
    loss = F.nll_loss(output, pretextTarget)
    loss.backward()
```

Start from this pre-trained model, freeze everything except FC layer

```
model = torch.load('mnistPretrained.pt')
model.fc2 = nn.Linear(128, 10)

for name, param in model.named_parameters():
    if not name.startswith('fc'):  # Freezing everything e
        param.requires_grad = False
model = model.to(device)
```

Start from this pre-trained model, freeze everything except FC layer

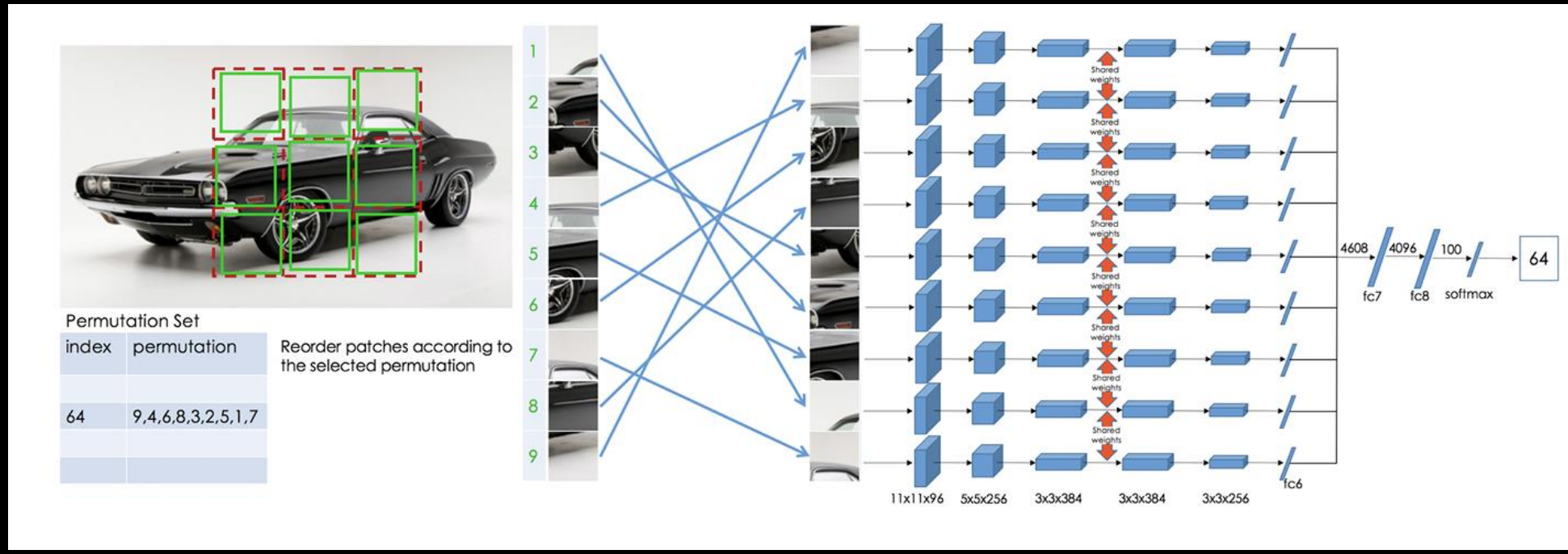
```
model = torch.load('mnistPretrained.pt')
model.fc2 = nn.Linear(128, 10)

for name, param in model.named_parameters():
    if not name.startswith('fc'):  # Freezing everything e
        param.requires_grad = False
model = model.to(device)
```

Accuracy after 1 epoch of training = 97.5%

Image Jigsaw Puzzle

- Patch Extraction: Divide the input image into a grid of patches (e.g.,).
- Shuffling: Randomly permute the patches to create a shuffled version of the image.
- Puzzle Solving Task: The model predicts the original order of the patches from the shuffled input.

Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles



Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles

Figure 1. **Learning image representations by completing damaged jigsaw puzzles.** We sample 3-by-3 patches from an image and create damaged jigsaw puzzles. (a) is the puzzles after shuffling the patches, removing one patch, and decolorizing. We push a network to recover the original arrangement, the missing patch, and the color of the puzzles. (b) shows the outputs; while the pixel-level predictions are in *ab* channels, we visualize with their original L channels for the benefit of the reader.

Learning Image Representations by Completing Damaged Jigsaw Puzzles

Jigsaw Alternative

Pre-Text Task: Rotation / Jigsaw / …

- Rarely used in Earth observation
- Such spatial correlation is less dominant in EO images

Pre-Text Task: Geolocation Classification

- Geolocation metadata is often available
- Cluster the dataset according to lat/long
- Train a model to predict these clusters

Geography-Aware Self-Supervised Learning, 2021