EECS 442 Discussion 6 FA 2023

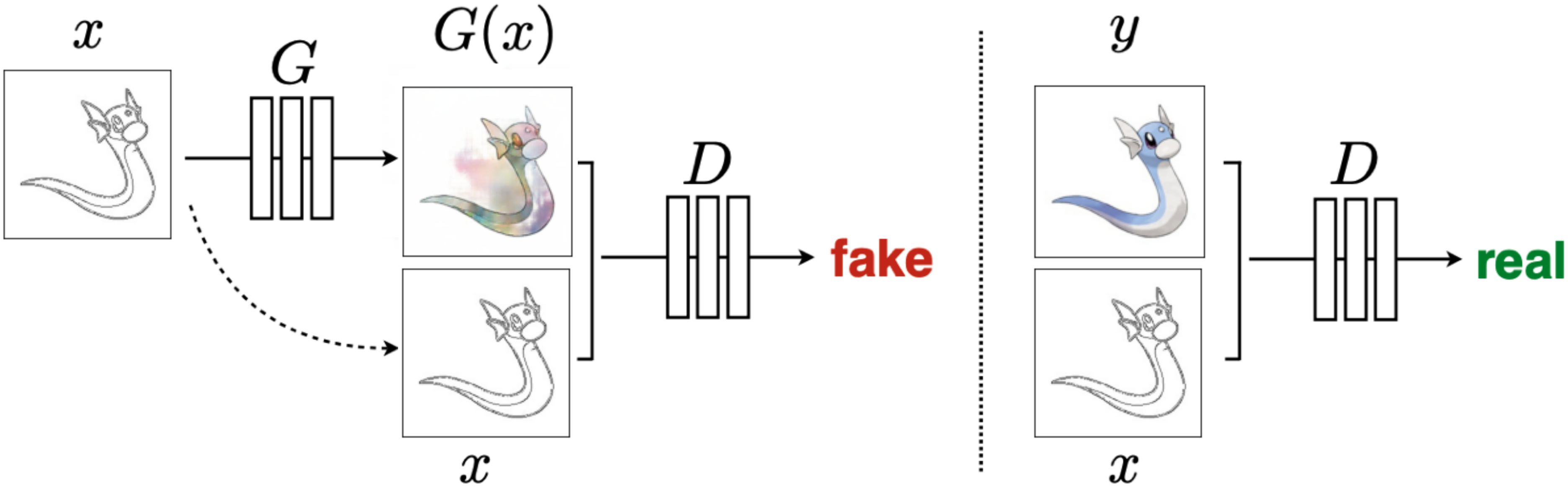
GANs

Datasets

- mini-edges2shoes
  - Contains 1,000 training and 100 validation images
  - Sketch and image pairs

Pytorch Data Loader

- Loads a custom set of data
- Creates a class containing the images
  1. Get the file names of images (use glob.glob)
  2. Apply the transformations to those images (transform parameter)
- Construct Pytorch DataLoader
  - Construct the custom class containing transformed images (separate train/val)
  - Giving it the custom class containing transformed images
  - Set batch_size and shuffle



G tries to synthesize fake images that fool D
D tries to identify the fakes

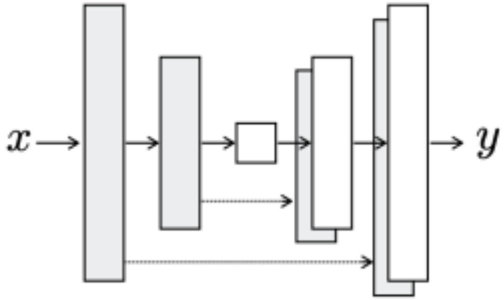
[Goodfellow et al., 2014]

U-Net

Encoder: downsample images by applying convolution

Decoder:

- Concatenate encoder and decoder
- Upsample images by applying convolution to concatenated image



Toy U-Net Example

Encoder: C64-C128-C256

Decoder: C128-C64-C3

```python
# (Not a part of your solution) Toy example of an U-net architecture
class toy_unet(nn.Module):
  # initializers
  def __init__(self):
    super(generator, self).__init__()
    # encoder
    self.conv1 = nn.Conv2d(3, 64, 4, 2, 1)
    self.conv2 = nn.Conv2d(64, 64 * 2, 4, 2, 1)
    self.conv3 = nn.Conv2d(64 * 2, 64 * 4, 4, 2, 1)
    # decoder
    self.deconv1 = nn.ConvTranspose2d(64 * 4, 64 * 2, 4, 2, 1)
    self.deconv2 = nn.ConvTranspose2d(64 * 2 * 2, 64, 4, 2, 1)
    self.deconv3 = nn.ConvTranspose2d(64 * 2, 3, 4, 2, 1)

  # forward method
  def forward(self, input):
    # pass through encoder
    e1 = self.conv1(input)
    e2 = self.conv2(F.relu(e1))
    e3 = self.conv3(F.relu(e2, 0.2))
    # pass through decoder
    d1 = self.deconv1(F.relu(e3))
    d1 = torch.cat([d1, e2], 1) # Concatenation
    d2 = self.deconv2(F.relu(d1))
    d2 = torch.cat([d2, e1], 1) # Concatenation
    d3 = self.deconv3(F.relu(d2))
    return d3
```

U-net for GAN

- Ck denotes a Convolution-BatchNorm-ReLU layer with k filters (output channels)
- All convolutions are 4 × 4 spatial filters applied with stride 2 and padding 1
- Convolutions in the encoder and the discriminator downsample the input by a factor of 2
- Convolutions in the decoder upsample the input by a factor of 2
- Batch normalization not applied to the first layer in the encoder for both the generator and the discriminator

Leaky ReLU

- Problems with ReLU
  - ReLU sets all values smaller than 0 to 0
  - Gradients of ReLU functions around 0 gradient are all 0s
- Solution: allows for small values < 0
- Uses a slope to represent negative values

Training Discriminator

- Feed the discriminator the real/fake images and labels
- Compute the real/fake BCE losses
- Training loss for the discriminator = average of real and fake BCE losses
- Backprop + optimize with Adam

$$\mathcal{L}_{cGAN}(G, D) = \frac{1}{N}\sum_{i=1}^{N} \log D(x_i, y_i) + \frac{1}{N}\sum_{i=1}^{N} \log(1 - D(x_i, G(x_i))).$$

Training Generator

Code provided for you

Same process as training the discriminator

L1 loss + BCE loss: the actual generator that we will use

L1 loss only: just for you to see the effects of L1 loss

$$\mathcal{L}_{L1}(G) = \frac{1}{N}\sum_{i=1}^{N}[\|y_i - G(x_i)\|_1]$$

Calculating receptive field size

$$r_i = r_{i-1} + (\left(k_i - 1\right) * \prod_{j=0}^{i-1} s_j)$$

r_i: Receptive field at stage i

k_i: Kernel size at stage i

s_j: Stride at stage j

$r_0 = 1,\ s_0 = 1$