AIL 862

Lecture 9

Fake Currency Example

Presented in the original GAN paper

GAN

In basic setup: one generator and one discriminator

Generator tries to mimic a data distribution

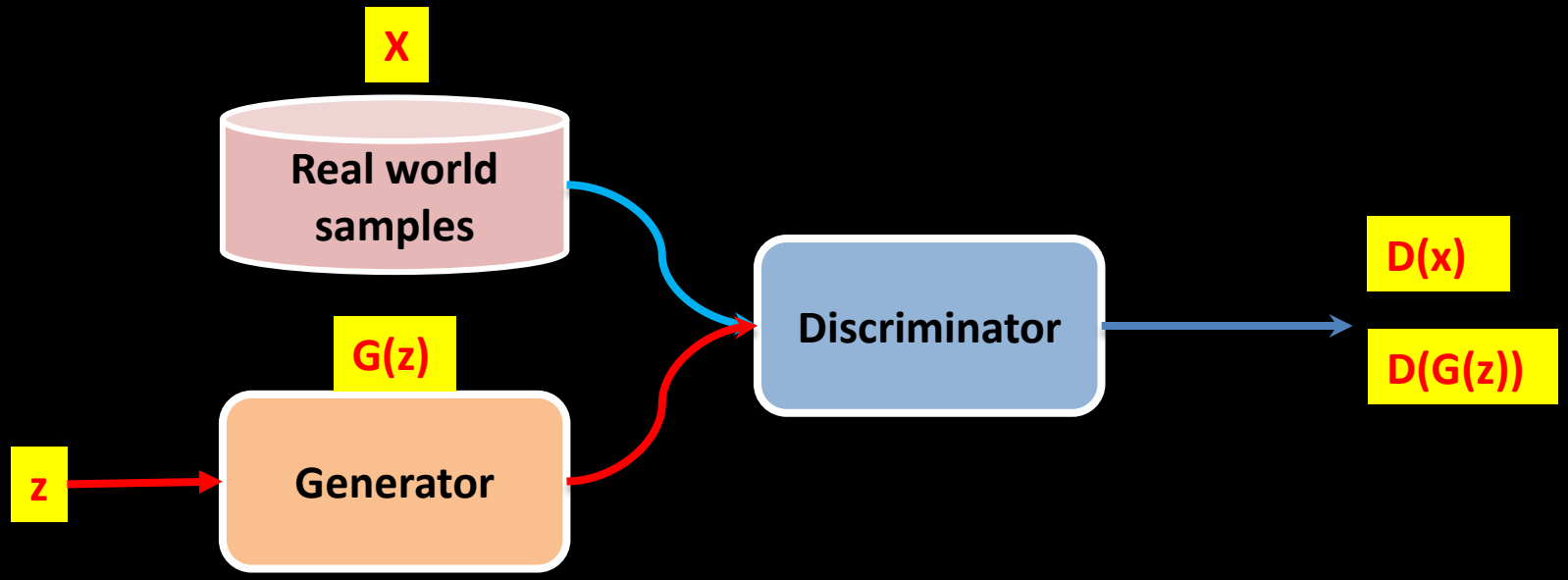
Discriminator tries to distinguish between real data and fake data (i.e., generated by the generator)

Inspiration from game theory

Objective is to reach Nash equilibrium

Nash equilibrium - Nash equilibrium, in game theory, is an outcome in a non-cooperative game for two or more players in which no player’s expected outcome can be improved by changing one’s own strategy.

GAN



GAN

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

Ideally

If G and D have enough capacity (and training algorithm is followed), p_G converges to p_data, , i.e., the generative model perfectly replicates the data generating process.

Ideally, after sufficient training

After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p\_G = p\_\{data\}$. The discriminator is unable to differentiate between the two distributions, i.e., $D(x) = 1/2$

Potential Design choices

Weaken the discriminator

Reduced generator 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.

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

Progressive training

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

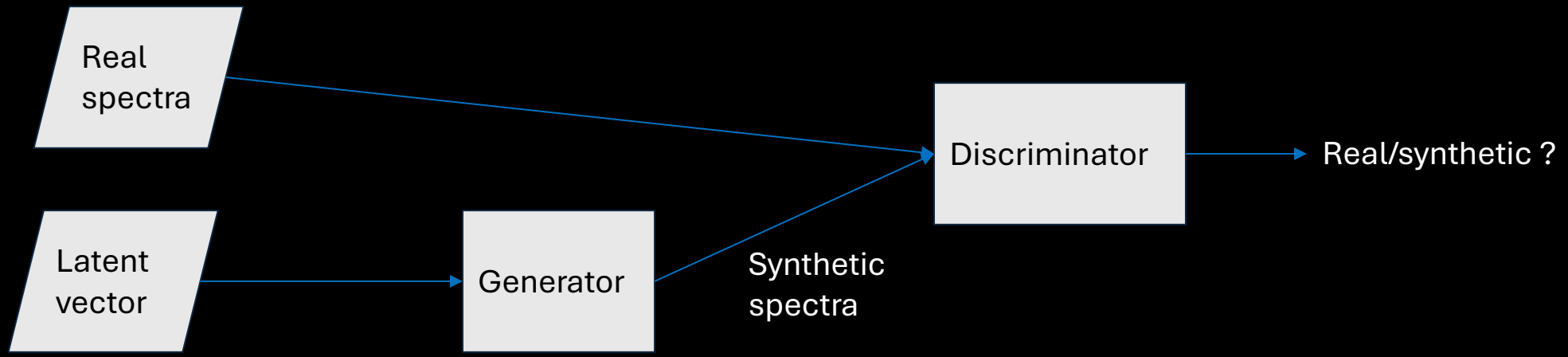
Progressive Growing of GANs for Improved Quality, Stability, and Variation

Feature matching

Train the generator to match the expected value of the features on an intermediate layer of the discriminator.

Improved Techniques for Training GANs

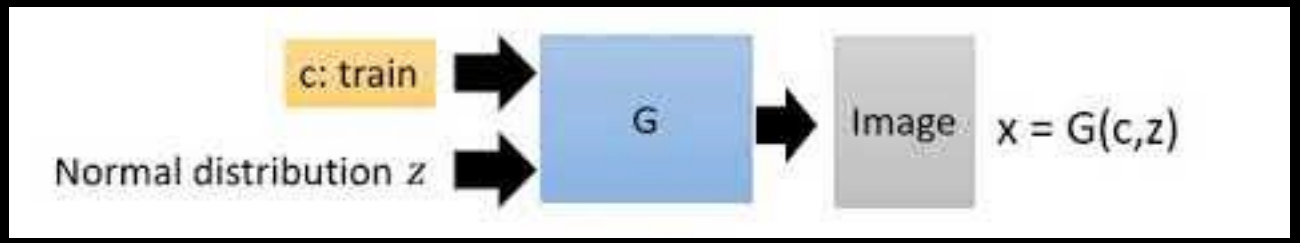
GAN-Based Synthesis of Hyperspectral Data



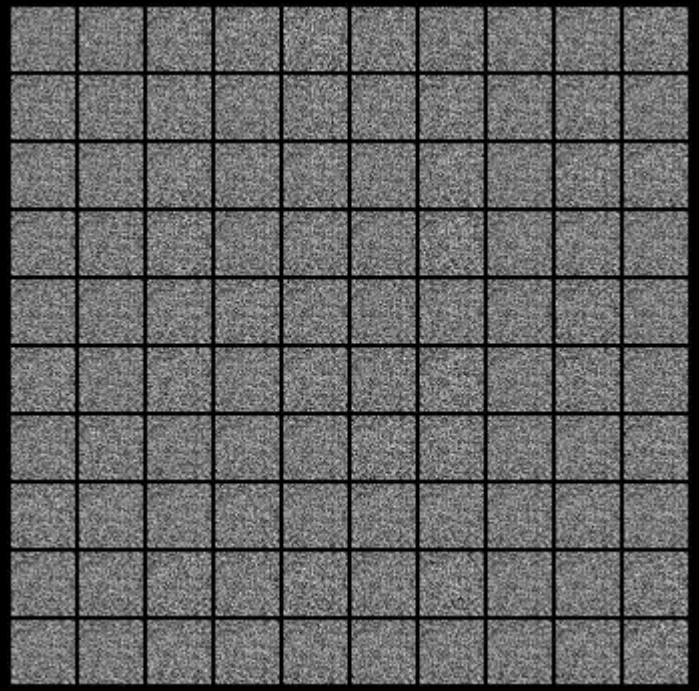
Generative Adversarial Network Synthesis of Hyperspectral Vegetation Data

Conditional GAN

Generator



Conditional GAN Example on MNIST



Just at the beginning.
Generator has not learned anything yet.

Conditional GAN Example on MNIST

After some training.

Conditional GAN Example on MNIST

Half way through the training process.

Conditional GAN Example on MNIST

After training.

cGAN MNIST Code

```
class Generator(nn.Module):
    def __init__(self):
        super(Generator, self).__init__()

        self.label_emb = nn.Embedding(opt.n_classes, opt.n_classes)

        def block(in_feat, out_feat, normalize=True):
            layers = [nn.Linear(in_feat, out_feat)]
            if normalize:
                layers.append(nn.BatchNorm1d(out_feat, 0.8))
            layers.append(nn.LeakyReLU(0.2, inplace=True))
            return layers

        self.model = nn.Sequential(
            *block(opt.latent_dim + opt.n_classes, 128, normalize=Fals
            *block(128, 256),
            *block(256, 512),
            *block(512, 1024),
            nn.Linear(1024, int(np.prod(img_shape))),
            nn.Tanh()
        )

    def forward(self, noise, labels):
        # Concatenate label embedding and noise to produce input
        gen_input = torch.cat((self.label_emb(labels), noise), -1)
        img = self.model(gen_input)
        img = img.view(img.size(0), *img_shape)
        return img
```

```python
class Discriminator(nn.Module):
    def __init__(self):
        super(Discriminator, self).__init__()

        self.label_embedding = nn.Embedding(opt.n_classes, opt.n_classes)

        self.model = nn.Sequential(
            nn.Linear(opt.n_classes + int(np.prod(img_shape)), 512),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Linear(512, 512),
            nn.Dropout(0.4),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Linear(512, 512),
            nn.Dropout(0.4),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Linear(512, 1),
        )

    def forward(self, img, labels):
        # Concatenate label embedding and image to produce input
        d_in = torch.cat((img.view(img.size(0), -1), self.label_embedding(labels)), -1)
        validity = self.model(d_in)
        return validity
```

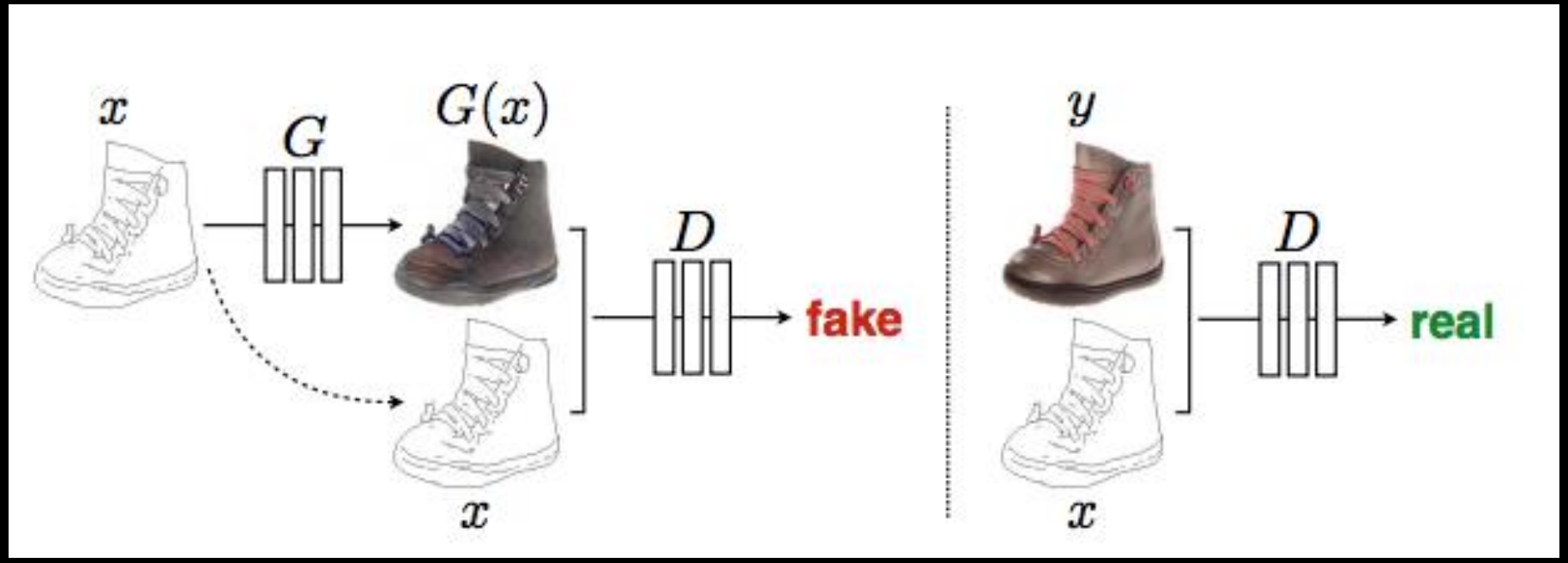
```
valid = Variable(FloatTensor(batch_size, 1).fill_(1.0), requires_grad=False)
fake = Variable(FloatTensor(batch_size, 1).fill_(0.0), requires_grad=False)
```

```
# -----------------
#  Train Generator
# -----------------

optimizer_G.zero_grad()

# Sample noise and labels as generator input
z = Variable(FloatTensor(np.random.normal(0, 1, (batch_size, opt.latent_dim))))
gen_labels = Variable(LongTensor(np.random.randint(0, opt.n_classes, batch_size)))

# Generate a batch of images
gen_imgs = generator(z, gen_labels)

# Loss measures generator's ability to fool the discriminator
validity = discriminator(gen_imgs, gen_labels)
g_loss = adversarial_loss(validity, valid)

g_loss.backward()
optimizer_G.step()
```

```
# ---------------------
#  Train Discriminator
# ---------------------

optimizer_D.zero_grad()

# Loss for real images
validity_real = discriminator(real_imgs, labels)
d_real_loss = adversarial_loss(validity_real, valid)

# Loss for fake images
validity_fake = discriminator(gen_imgs.detach(), gen_labels)
d_fake_loss = adversarial_loss(validity_fake, fake)

# Total discriminator loss
d_loss = (d_real_loss + d_fake_loss) / 2

d_loss.backward()
optimizer_D.step()
```

Pix2Pix

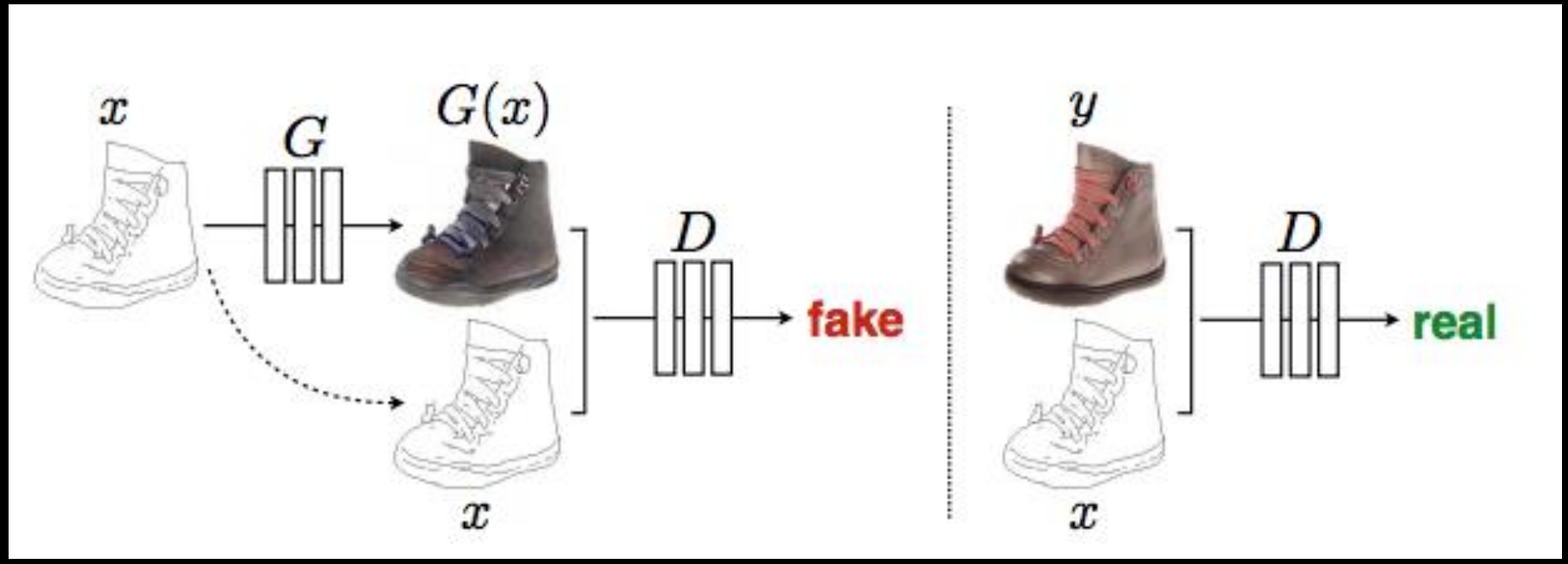
A type of conditional GAN with image as input



Pix2Pix

A type of conditional GAN with image as input

Noise in form of dropout
(both training and test time)



Paper reading allotment

Do You Remember? Dense Video Captioning with Cross-Modal Memory Retrieval

Anant

Paper reading allotment

Focal Foundation Models

Abdul

Paper reading allotment

VDT: General Purpose Video Diffusion Transformer via Mask Modeling

Taki

BEVFusion: Multi-task Multi-sensor Fusion with Unified Bird’s-Eye View Representation

Shashwat

Guiding a Diffusion Model with a Bad Version of Itself

Anamika