Deep generative neural networks Fundamentals & problem solving

Class 4

JAMAL TOUTOUH
jamal@uma.es
jamal.es
@jamtou

Conditional GANs

Review basic ideas about GANs

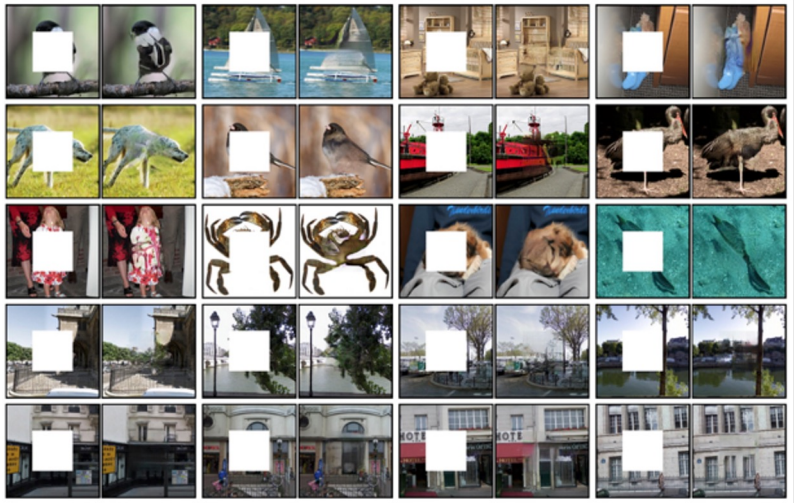
- GAN is trained in a completely unsupervised and unconditional fashion, meaning no labels involved in the training process.
- We have zero control over the type of samples generated.
- What if we want our GAN model to generate samples of given specific type (e.g., generate just the digit 9 in MNIST).
  - We could try to control the random vectors sampled from the latent space

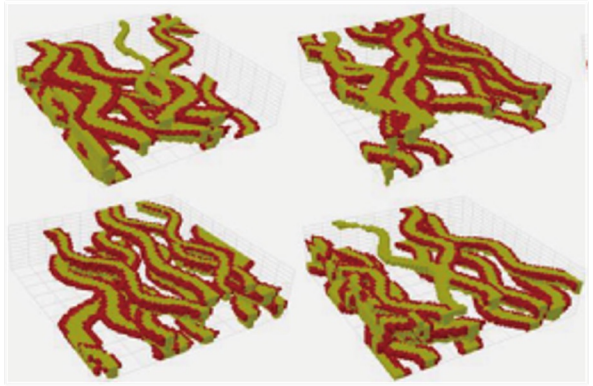
GAN applications

"a cat wearing sunglasses"

"a black and white photograph of a cat wearing sunglasses by annie lebovitz, highly-detailed"

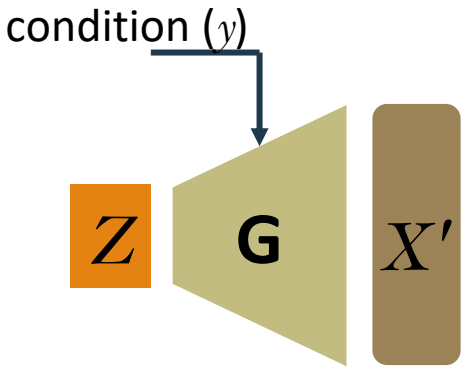
How to control the output

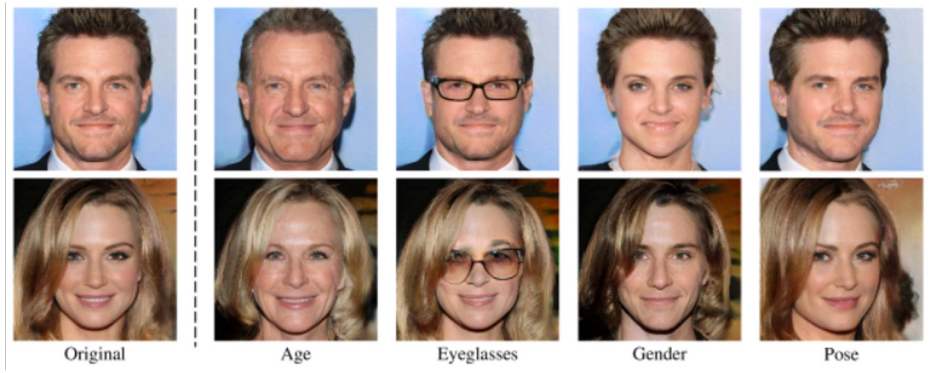




Conditional GANs

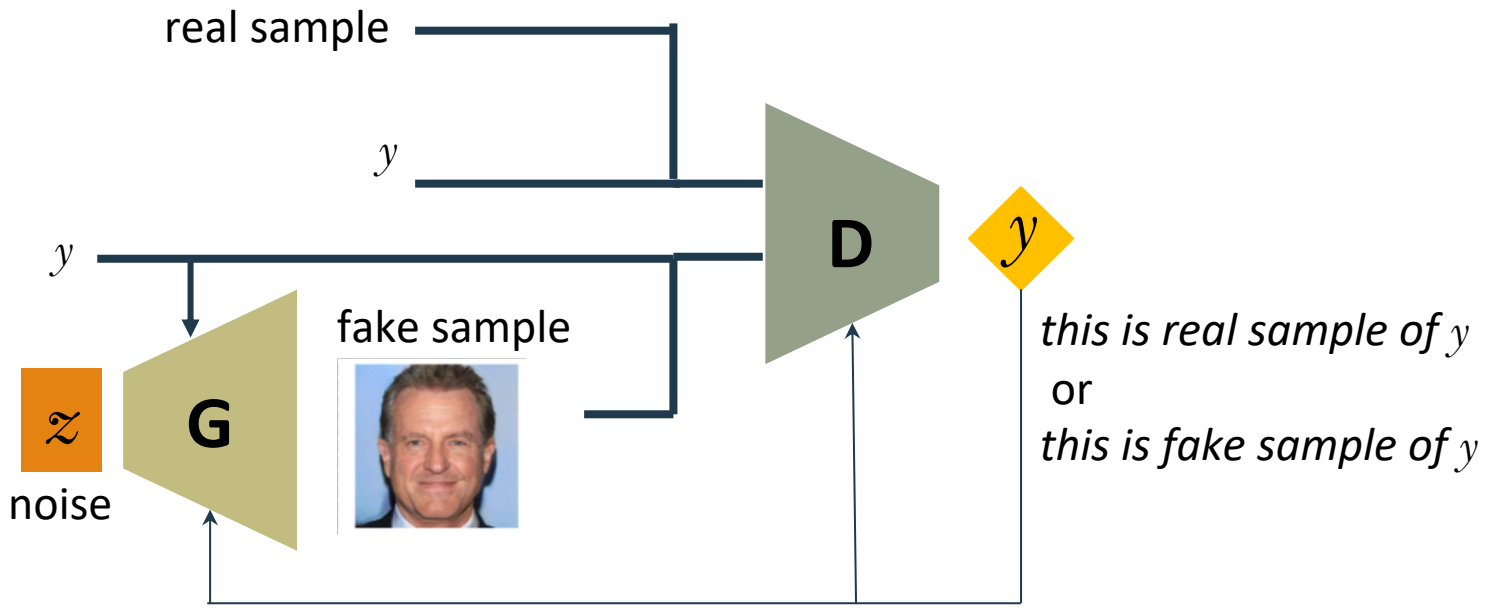
- **Goal:** Better control of the generation
- **Idea:** Add information about the generated samples (e.g., labels) to train the generator





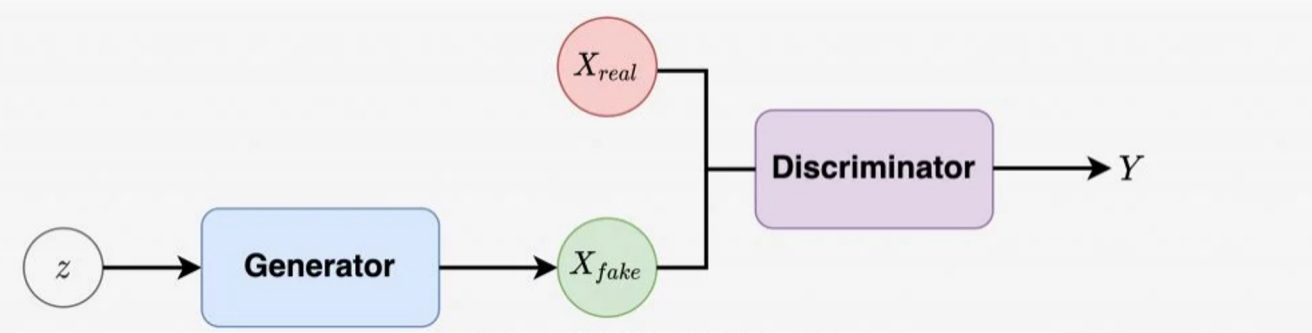
Training cGANs

- **Main difference:** Discriminator gets data sample and condition (e.g., label)

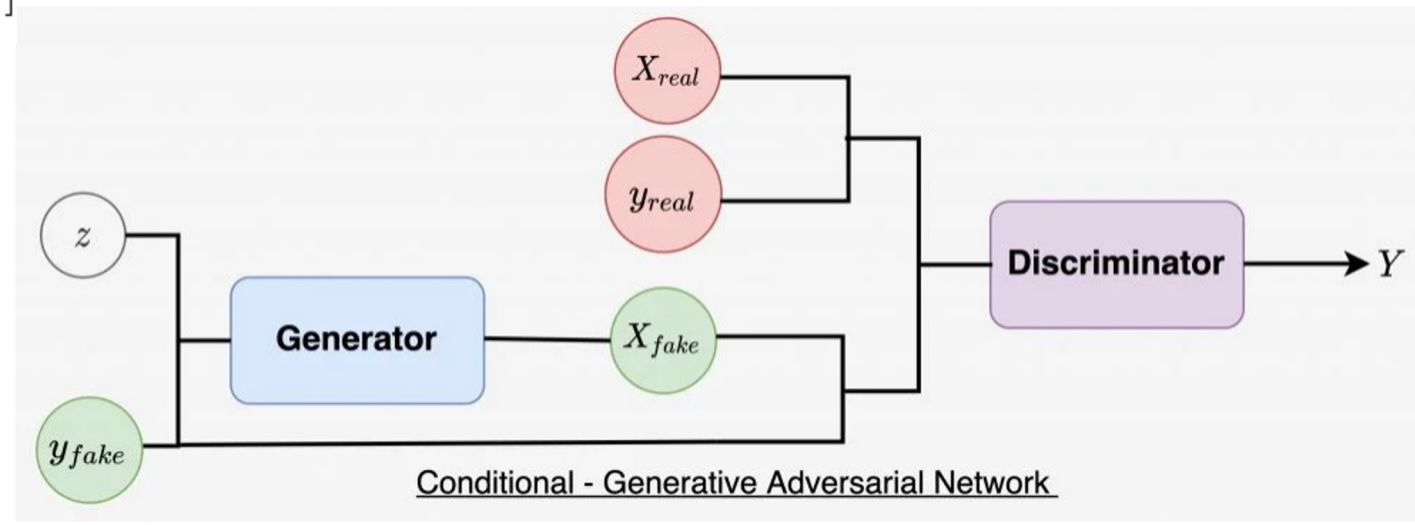


$$\min_G \max_D V(D,G) = \mathop{\mathbb{E}}_{x \sim p_{data}(x)} [\log D(x,y)] + \mathop{\mathbb{E}}_{z \sim p_z(z)} [\log(1 - D(G(z,y),y))]$$

GANs vs cGAN



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



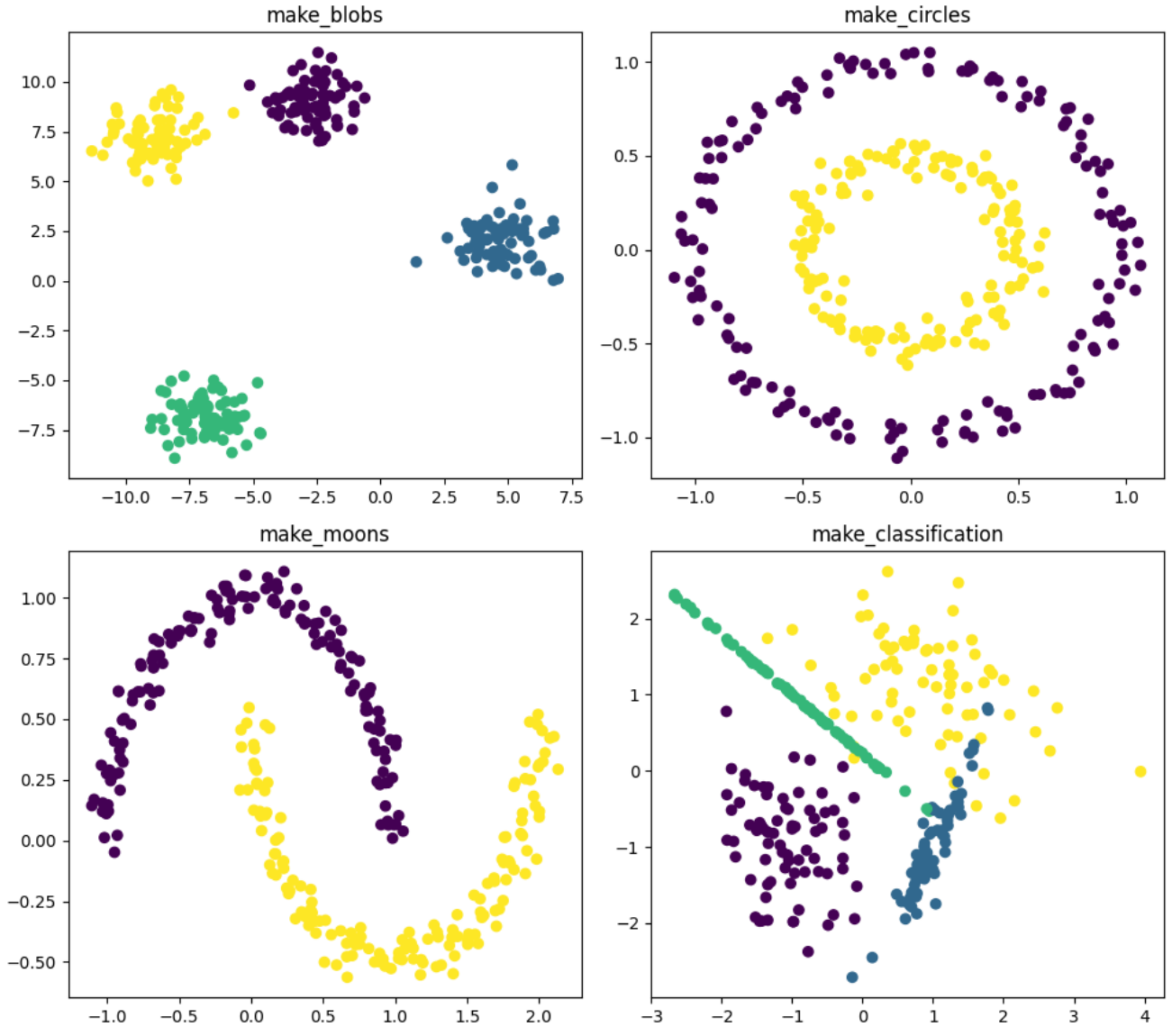
$$\mathop{\mathbb{E}}_{x\sim p_{data}(x)}[\log D(x,y)] + \mathop{\mathbb{E}}_{z\sim p_z(z)}[\log(1 - D(G(z,y),y))]$$

GANs vs cGAN. Simple Example

- **Example:** Using generative models to produce data samples of datasets provided by Sklearn.datasets.

- Vanilla GAN: https://drive.google.com/file/d/1n844Qk3T4BqHFgnnhbyBKAGg87hJdd1O/view?usp=sharing

- Conditional GAN: https://drive.google.com/file/d/1n844Qk3T4BqHFgnnhbyBKAGg87hJdd1O/view?usp=sharing



GANs vs cGAN. Simple Example

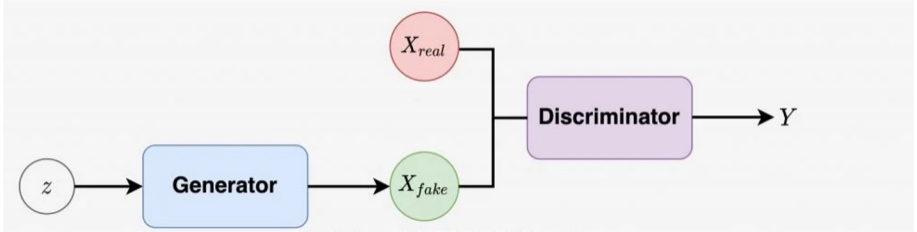
- **GANs architecture:** We have defined the same general architectures for both, generator and discriminator.

```
class Generator(nn.Module):
    def __init__(self, input_size, output_size, hidden_size=32):
        super(Generator, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size)
        self.fc2 = nn.Linear(hidden_size, hidden_size)
        self.fc3 = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = torch.tanh(self.fc3(x))
        return x
```

```
class Discriminator(nn.Module):
    def __init__(self, input_size, hidden_size=32):
        super(Discriminator, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size)
        self.fc2 = nn.Linear(hidden_size, hidden_size)
        self.fc3 = nn.Linear(hidden_size, 1)

    def forward(self, x):
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = torch.sigmoid(self.fc3(x))
        return x
```

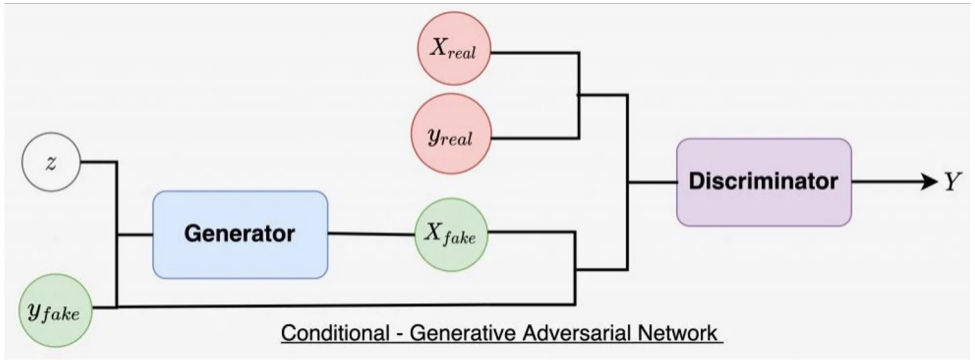
GANs vs cGAN. Simple Example

- **GANs architecture:** We have defined the same general architectures for both, generator and discriminator.




```
# Initialize generator and discriminator
generator = Generator(noise_dim, output_size, hidden_size).to(DEVICE)
discriminator = Discriminator(input_size, hidden_size).to(DEVICE)
```

y is added as an extra input for both models




```
# Initialize generator and discriminator
generator = Generator(noise_dim+1, output_size, hidden_size).to(DEVICE)
discriminator = Discriminator(input_size+1, hidden_size).to(DEVICE)
```

GANs vs cGAN. Simple Example

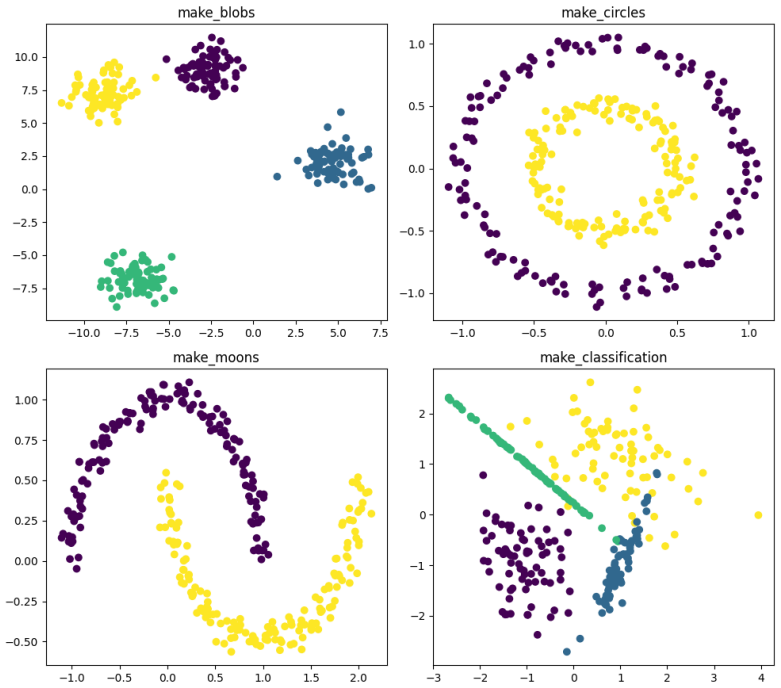
```
for epoch in range(num_epochs):
    for real_data, _ in dataloader:
        real_data = Variable(real_data)
```

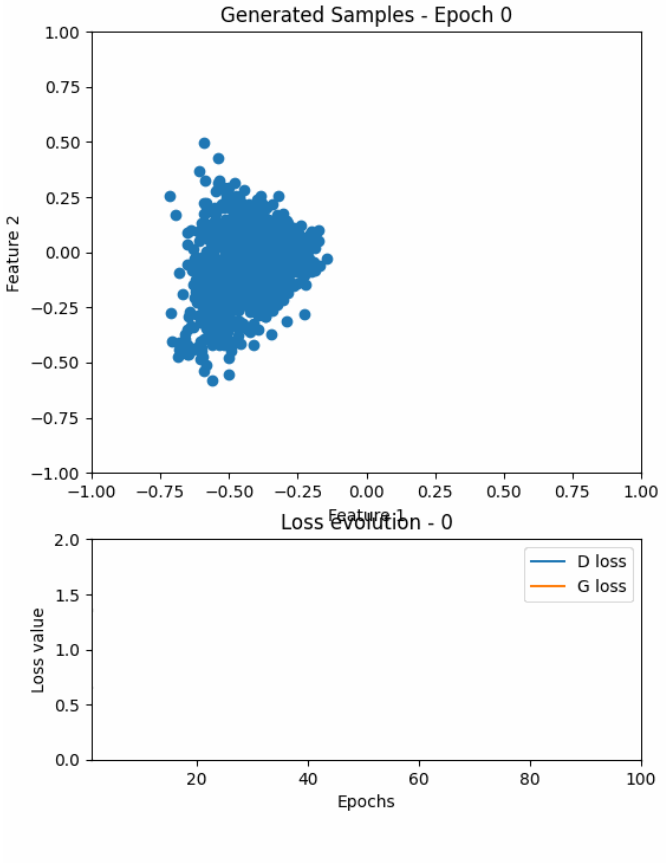
```
for epoch in range(num_epochs):
    for real_data, labels in dataloader:
        real_data = Variable(real_data)
        labels = Variable(labels.float())
```

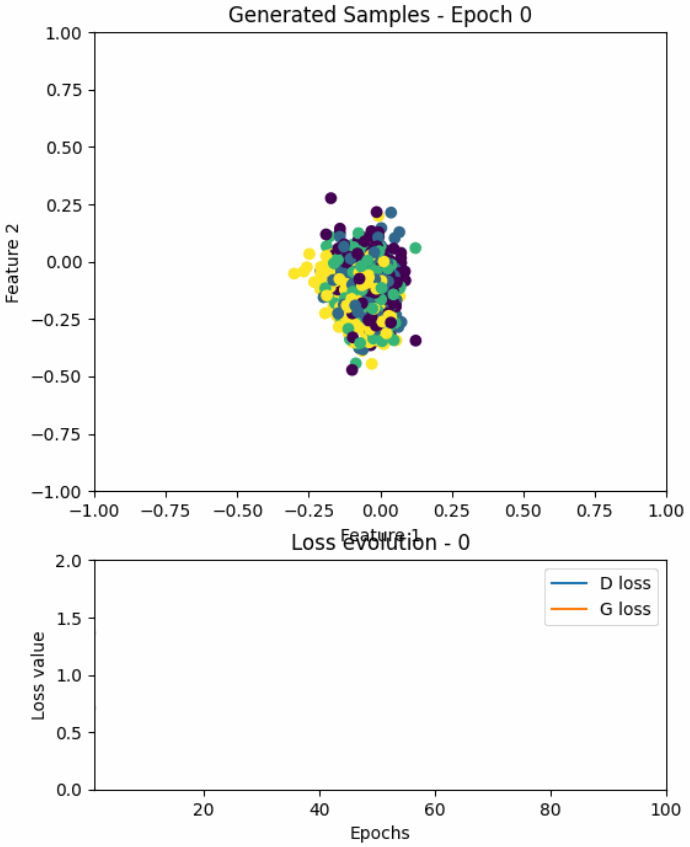
```
# Train discriminator
d_optimizer.zero_grad()
d_real_decision = discriminator(real_
d_real_error = criterion(d_real_decisi
d_real_error.backward()

fake_data = generator(read_latent_spac
d_fake_decision = discriminator(fake_
d_fake_error = criterion(d_fake_decisi
d_fake_error.backward()
d_optimizer.step()

# Train generator
g_optimizer.zero_grad()
fake_data = generator(read_latent_spac
dg_fake_decision = discriminator(fake_
g_error = criterion(dg_fake_decision,
g_error.backward()
g_optimizer.step()
```

```
# Train discriminator
d_optimizer.zero_grad()
d_real_decision = discriminator(concat_labels(real_data, labels))
d_real_error = criterion(d_real_decision, real_data_target(batch_size))
d_real_error.backward()

fake_data = generator(concat_labels(read_latent_space(batch_size, noise_dim), labels)).deta
d_fake_decision = discriminator(concat_labels(fake_data, labels))
d_fake_error = criterion(d_fake_decision, fake_data_target(batch_size))
d_fake_error.backward()
d_optimizer.step()

# Train generator
g_optimizer.zero_grad()
fake_data = generator(concat_labels(read_latent_space(batch_size, noise_dim), labels))
dg_fake_decision = discriminator(concat_labels(fake_data, labels))
g_error = criterion(dg_fake_decision, real_data_target(batch_size))
g_error.backward()
g_optimizer.step()
```

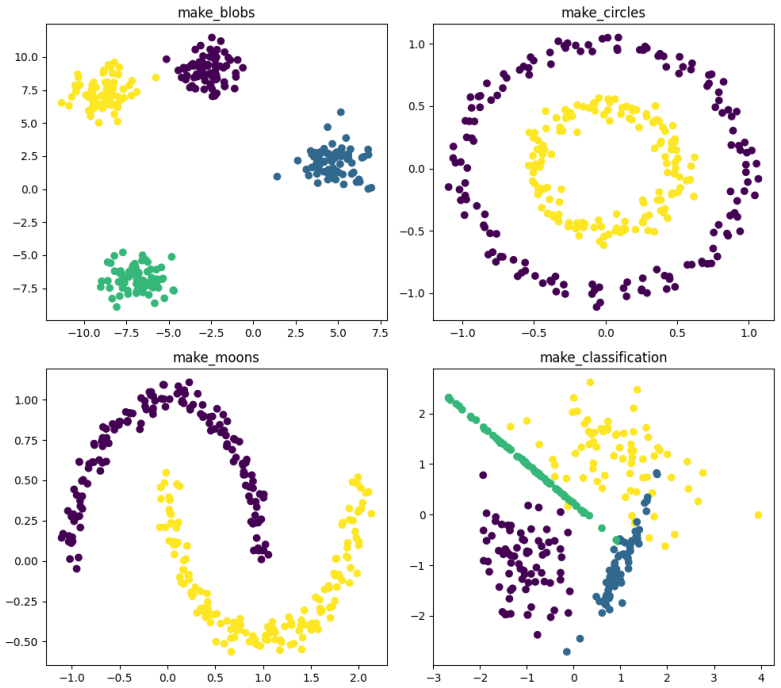
GANs vs cGAN. Simple Example

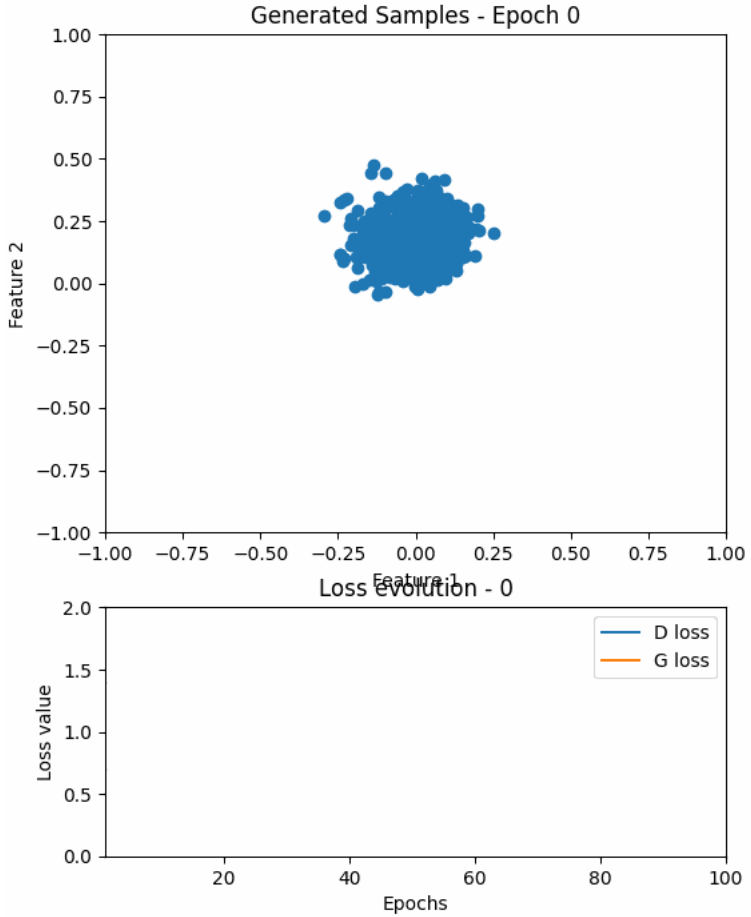


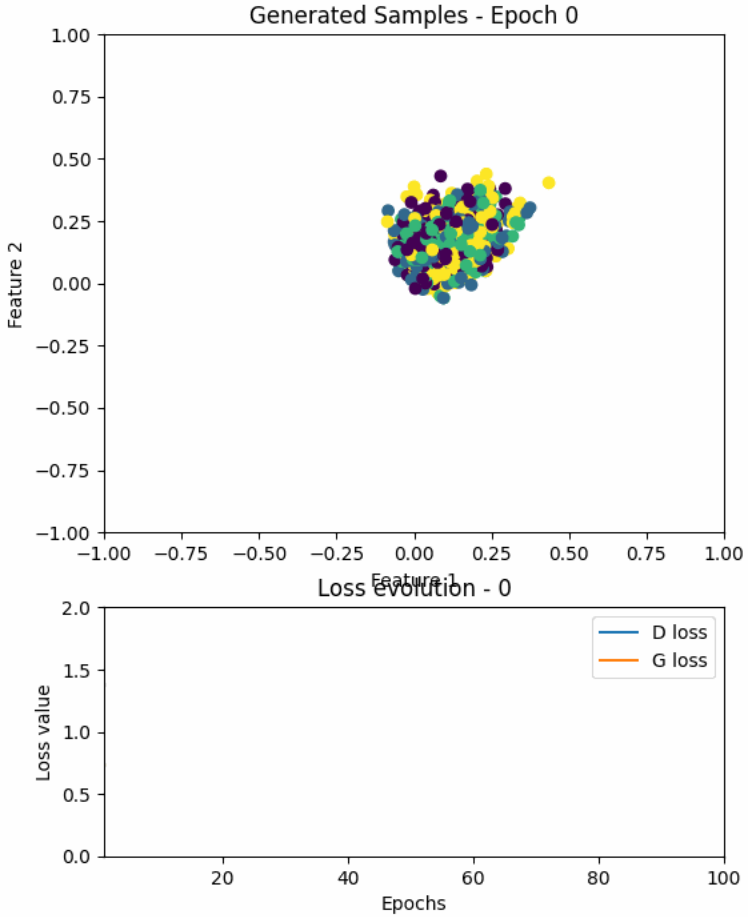




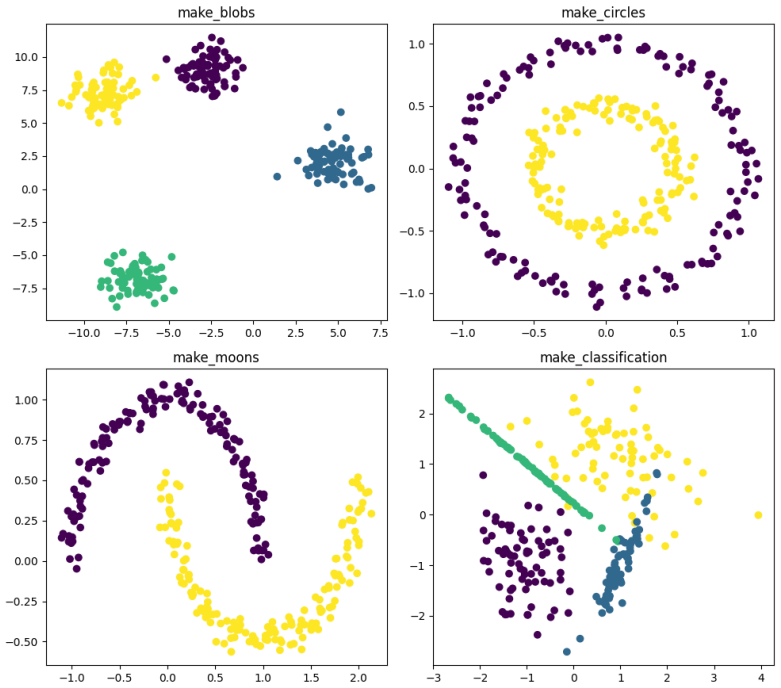
GANs vs cGAN. Simple Example

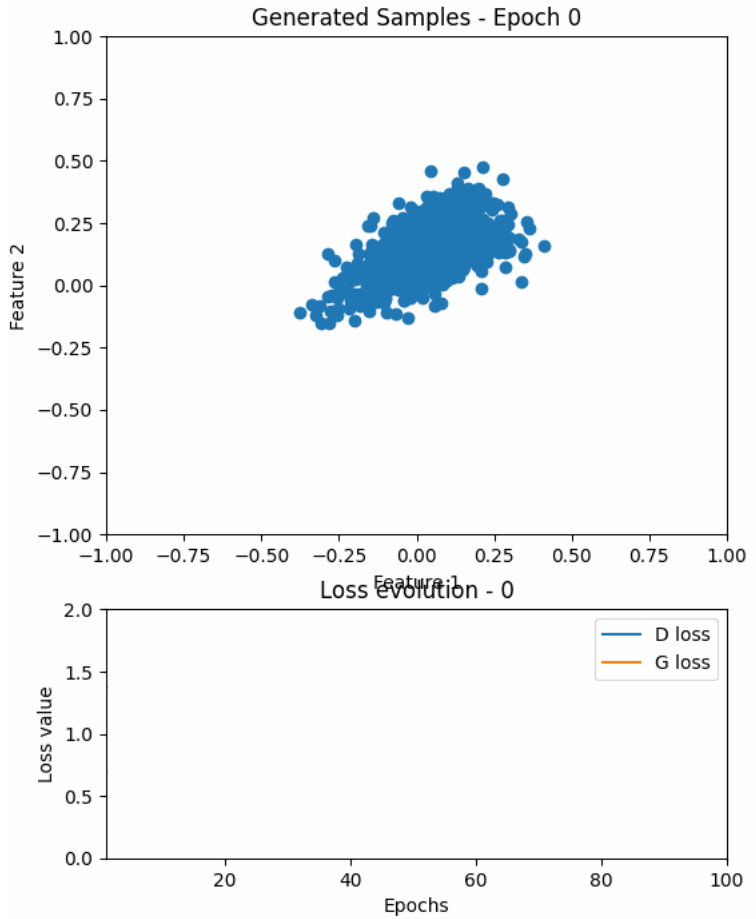


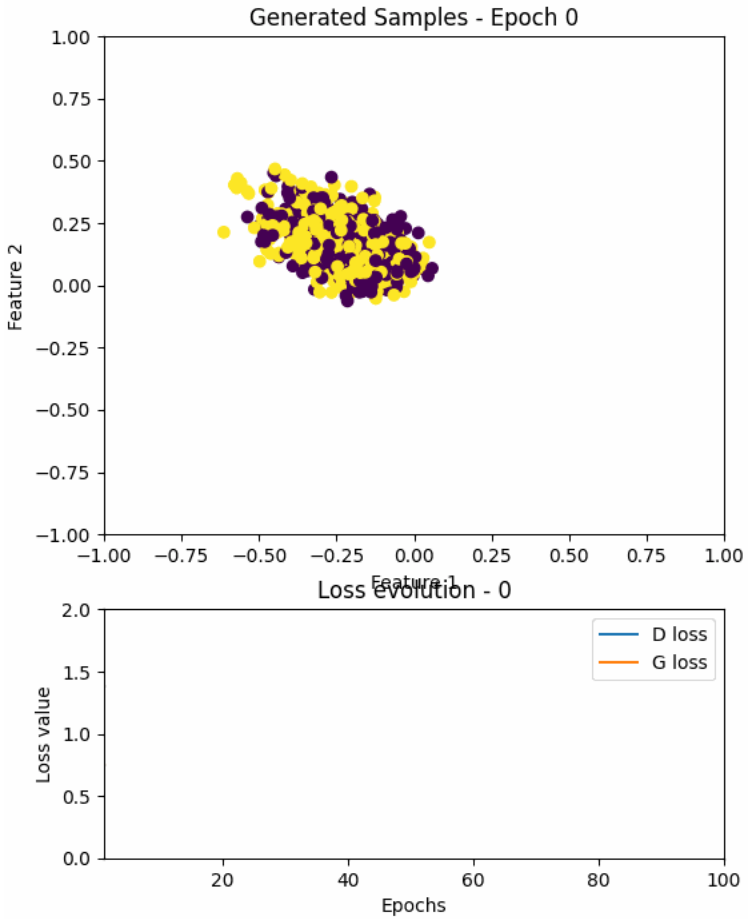




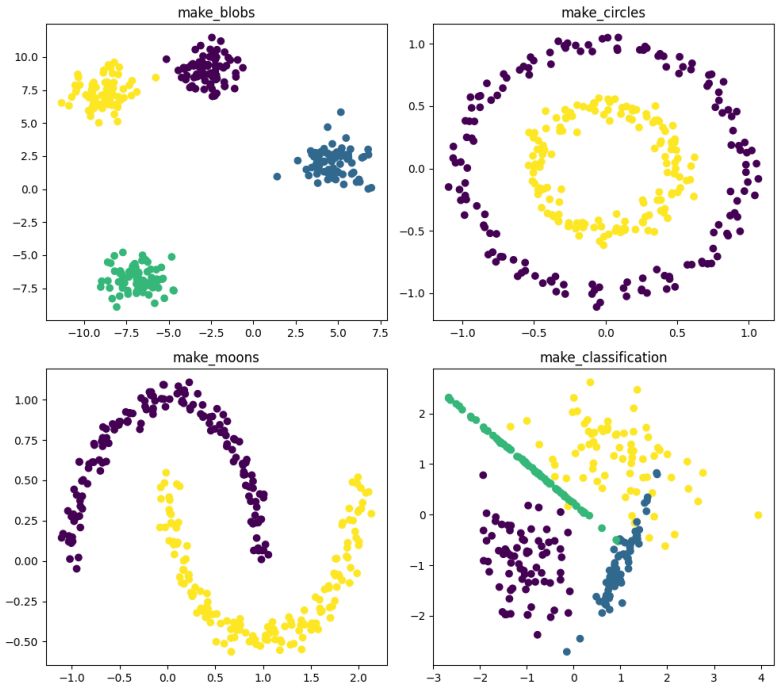
GANs vs cGAN. Simple Example

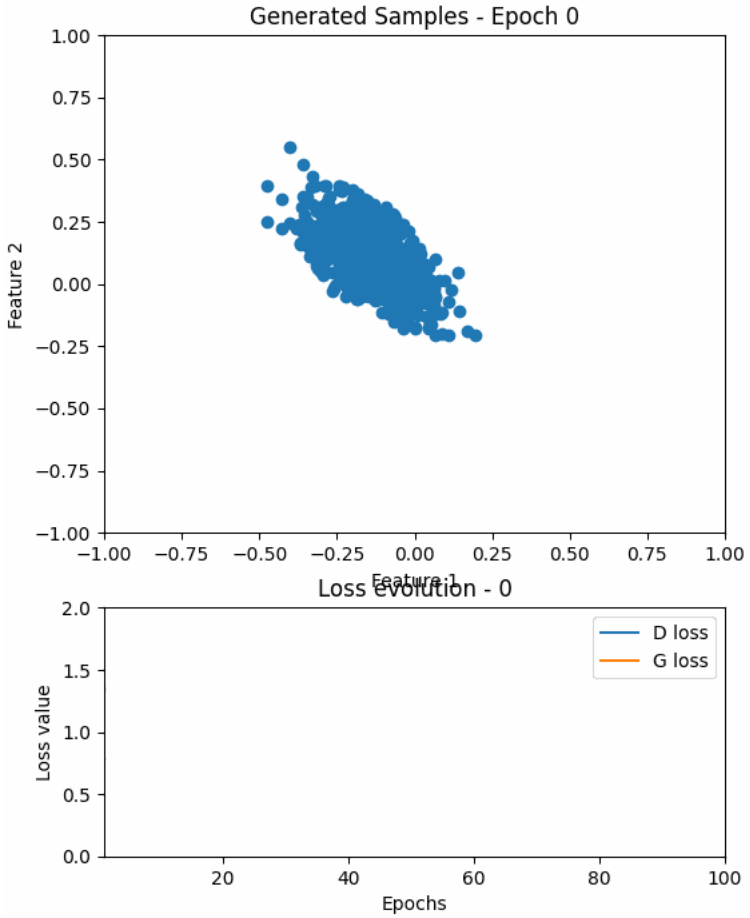


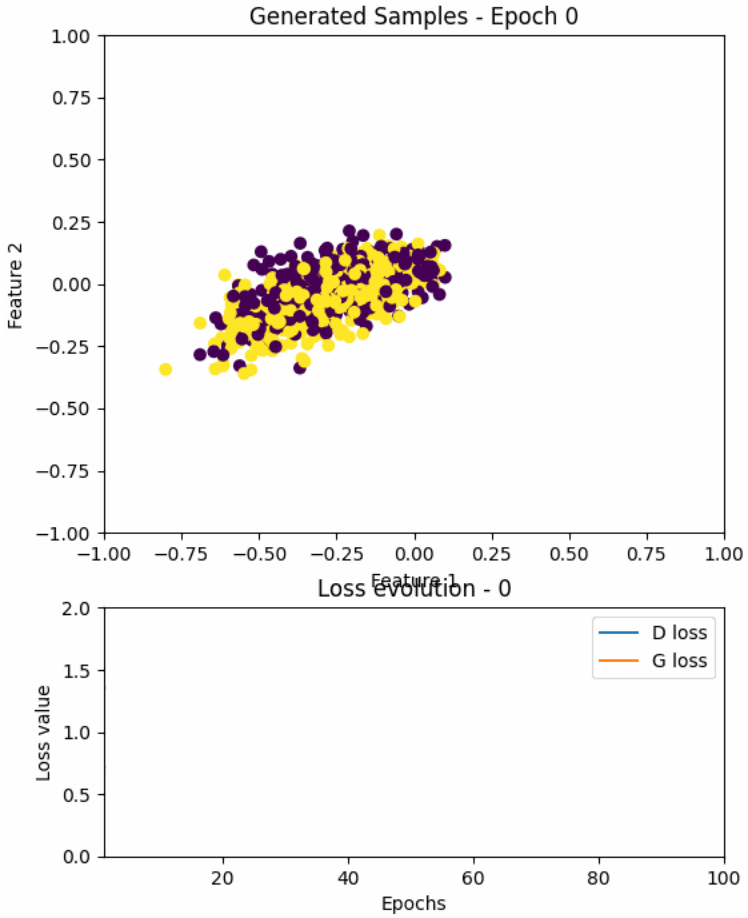




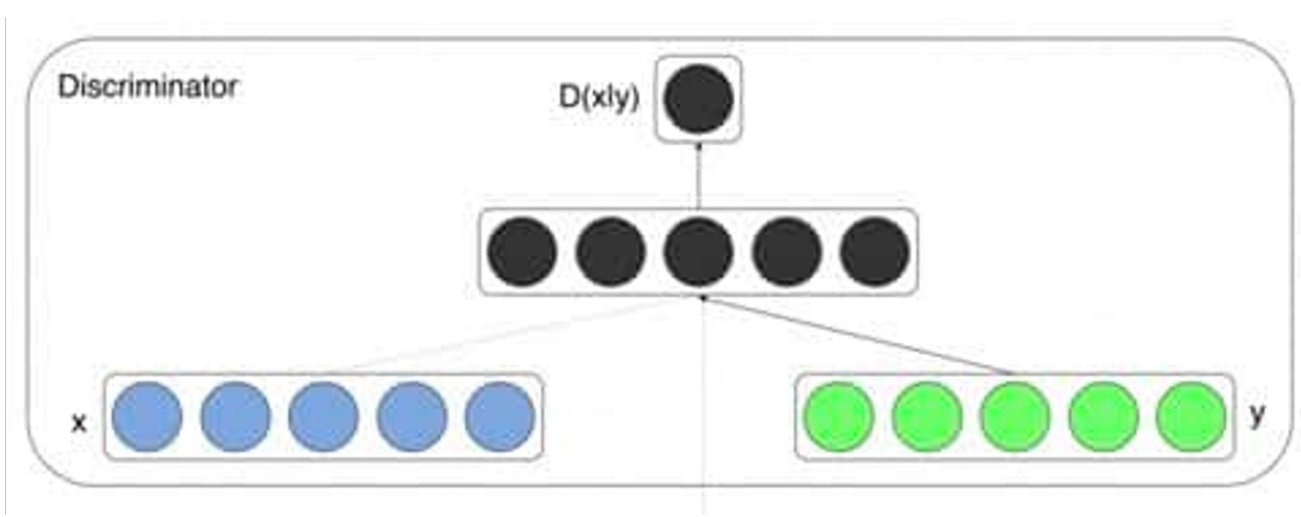
GANs vs cGAN. Simple Example

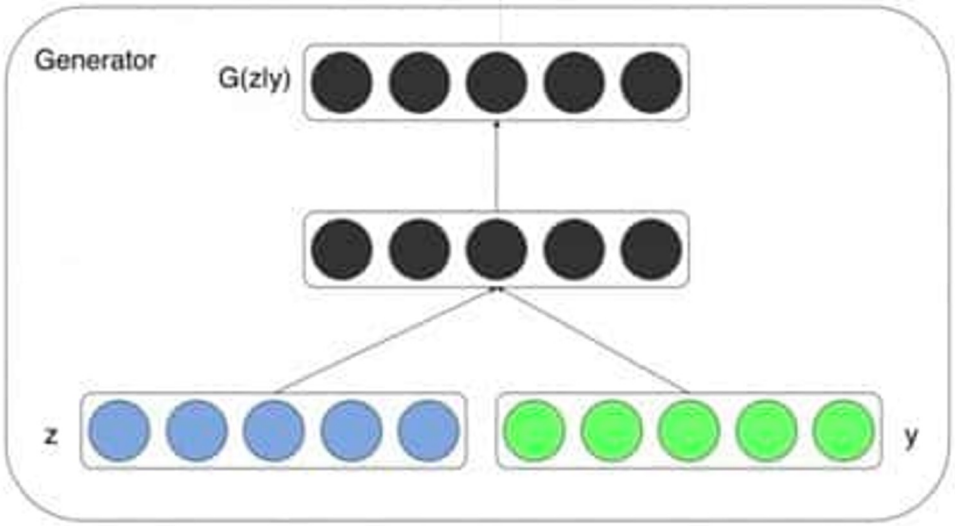






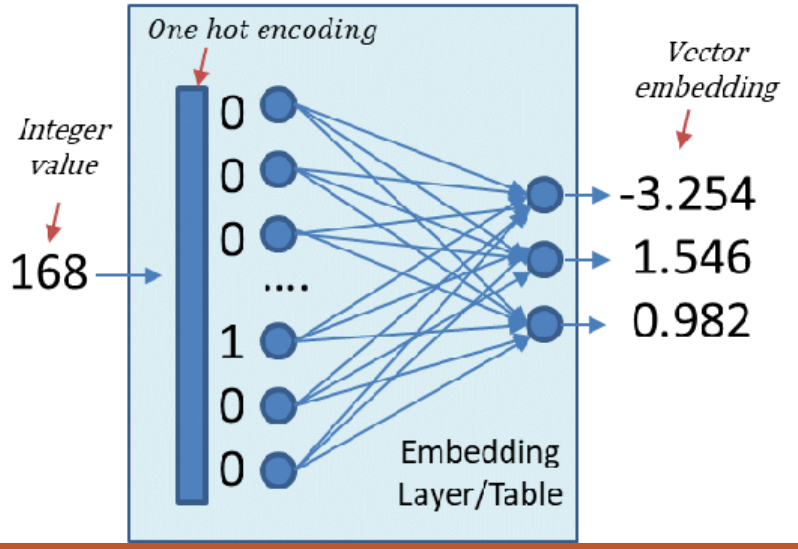
Conditional GANs for complex problems





Using embedding layer to créate the data to be concatenated to the data.

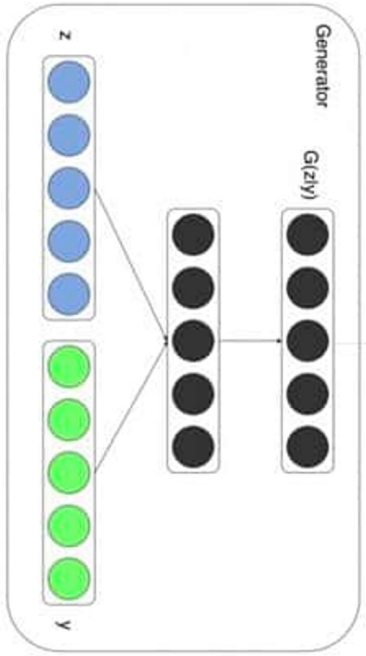
The input size of the embedding is given by the numer of classes in the real data



GAN Training. General Code – MLP GAN

0. Create ANNs

cGAN




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

        self.label_emb = nn.Embedding(10, 10)

        self.model = nn.Sequential(
            nn.Linear(110, 256),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Linear(256, 512),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Linear(512, 1024),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Linear(1024, 784),
            nn.Tanh()
        )

    def forward(self, z, labels):
        z = z.view(z.size(0), 100)
        c = self.label_emb(labels)
        x = torch.cat([z, c], 1)
        out = self.model(x)
        return out.view(x.size(0), 28, 28)
```

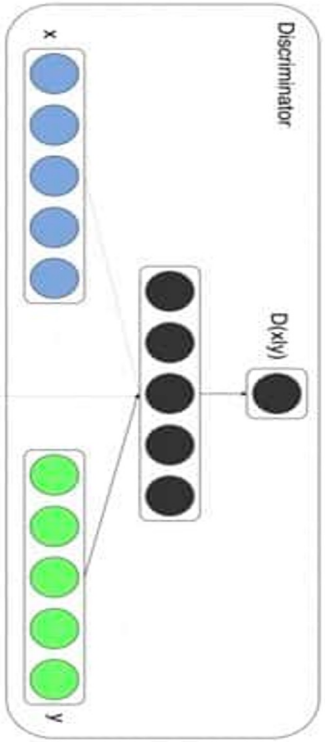
GAN

```python
class Generator(nn.Module):
    """
    Class that defines the the Generator Neural Network
    """
    def __init__(self, input_size, hidden_size, output_
        super(Generator, self).__init__()

        self.net = nn.Sequential(
            nn.Linear(input_size, hidden_size),
            nn.SELU(),
            nn.Linear(hidden_size, hidden_size),
            nn.SELU(),
            nn.Linear(hidden_size, output_size),
            nn.SELU(),
        )

    def forward(self, x):
        x = self.net(x)
        return x
```

GAN Training. General Code – MLP GAN

0. Create ANNs

cGAN




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

        self.label_emb = nn.Embedding(10, 10)

        self.model = nn.Sequential(
            nn.Linear(794, 1024),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Dropout(0.3),
            nn.Linear(1024, 512),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Dropout(0.3),
            nn.Linear(512, 256),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Dropout(0.3),
            nn.Linear(256, 1),
            nn.Sigmoid()
        )

    def forward(self, x, labels):
        x = x.view(x.size(0), 784)
        c = self.label_emb(labels)
        x = torch.cat([x, c], 1)
        out = self.model(x)
        return out.squeeze()
```

GAN

```
class Discriminator(nn.Module):
    """
    Class that defines the the Discriminator Neura
    """
    def __init__(self, input_size, hidden_size, ou
        super(Discriminator, self).__init__()

        self.net = nn.Sequential(
            nn.Linear(input_size, hidden_size),
            nn.ELU(),
            nn.Linear(hidden_size, hidden_size),
            nn.ELU(),
            nn.Linear(hidden_size, output_size),
            nn.Sigmoid()
        )

    def forward(self, x):
        x = self.net(x)
        return x
```

GAN Training. General Code – MLP GAN

1. Train discriminator

GAN

```
# 1. Train the discriminator
discriminator.zero_grad()
# 1.1 Train discriminator on real data
input_real = get_data_samples(batch_size)
discriminator_real_out = discriminator(input_real.res
discriminator_real_loss = discriminator_loss(discrimi
discriminator_real_loss.backward()
# 1.2 Train the discriminator on data produced by the
input_fake = read_latent_space(batch_size)
generator_fake_out = generator(input_fake).detach()
discriminator_fake_out = discriminator(generator_fake
discriminator_fake_loss = discriminator_loss(discrimi
discriminator_fake_loss.backward()
# 1.3 Optimizing the discriminator weights
discriminator_optimizer.step()
```

cGAN

```
# 1 Train discriminator on real data
real_validity = discriminator(real_images, labels)
real_loss = criterion(real_validity, Variable(torch.ones(batch_size)).to(device))

# 2 Train the discriminator on data produced by the generator
z = Variable(torch.randn(batch_size, 100)).to(device)
fake_labels = Variable(torch.LongTensor(np.random.randint(0, 10, batch_size))).to(device)
fake_images = generator(z, fake_labels)
fake_validity = discriminator(fake_images, fake_labels)
fake_loss = criterion(fake_validity, Variable(torch.zeros(batch_size)).to(device))

# 3 Get loss to train the discriminator from both losses
d_loss = real_loss + fake_loss
d_loss.backward()
```

GAN Training. General Code – MLP GAN

2. Train generator

GAN

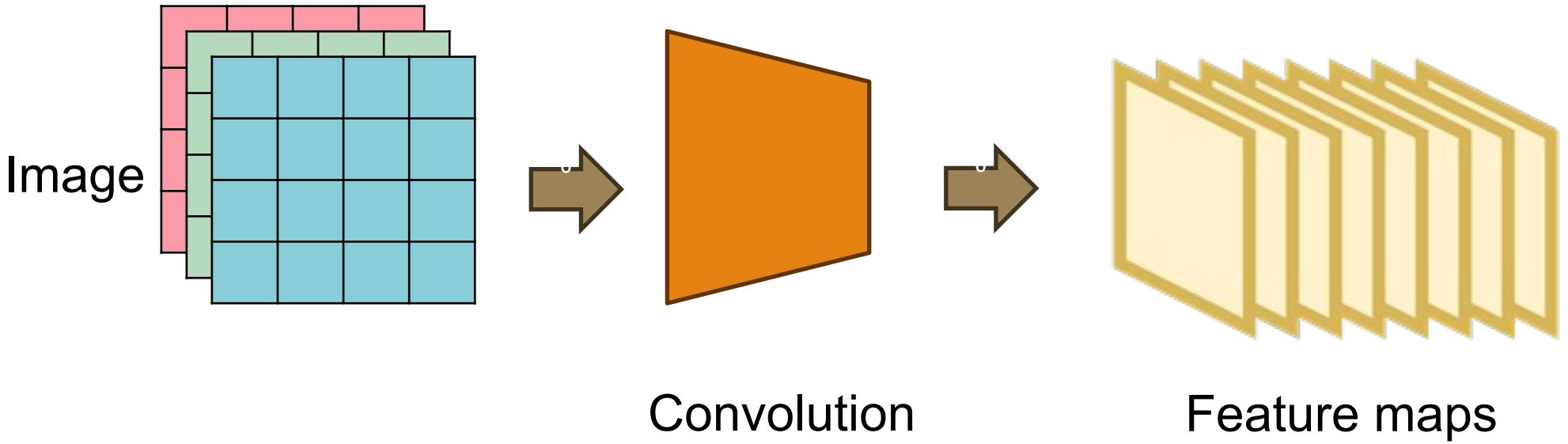
```
# 2.1 Create fake data
input_fake = read_latent_space(batch_size)
generator_fake_out = generator(input_fake)
# 2.2 Try to fool the discriminator with fak
discriminator_out_to_train_generator = discr
discriminator_loss_to_train_generator = gene

discriminator_loss_to_train_generator.backwa
# 2.3 Optimizing the generator weights
generator_optimizer.step()
```

cGAN

```
# 1 Create fake data
z = Variable(torch.randn(batch_size, 100)).to(device)
fake_labels = Variable(torch.LongTensor(np.random.randint(0, 10, batch_size))).to(devic
fake_images = generator(z, fake_labels)

# 2 Try to fool the discriminator with fake data
validity = discriminator(fake_images, fake_labels)
g_loss = criterion(validity, Variable(torch.ones(batch_size)).to(device))
g_loss.backward()

# 3 Optimize the generator weights
g_optimizer.step()
```

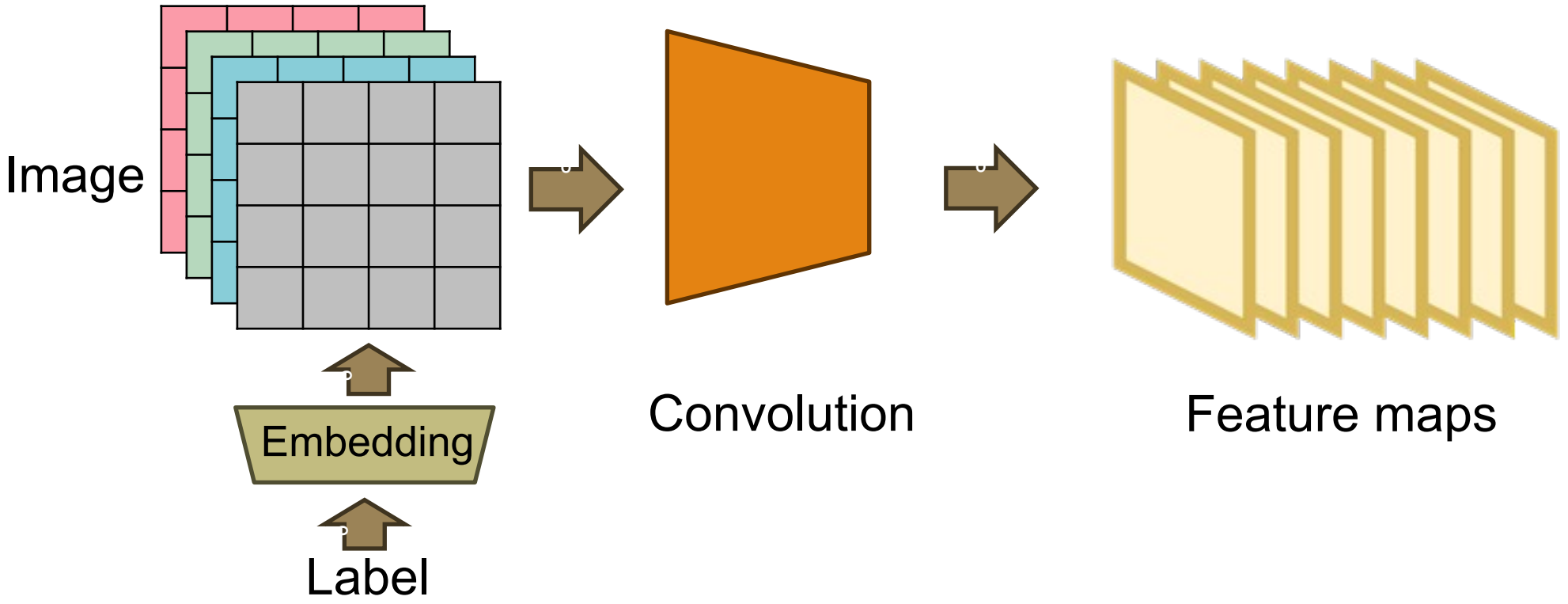
GAN Training. General Code – DCGAN

- **CNN** based classifier

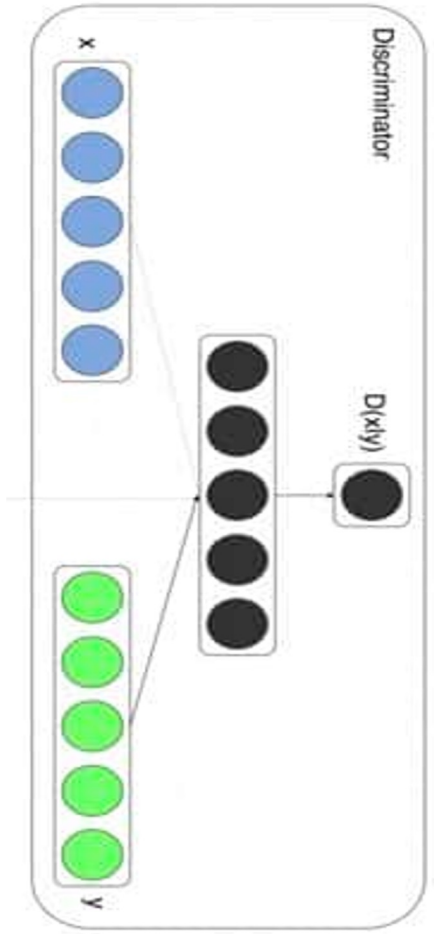


GAN Training. General Code – DCGAN

- **CNN** based discriminator



GAN Training. General Code – DC-cGAN



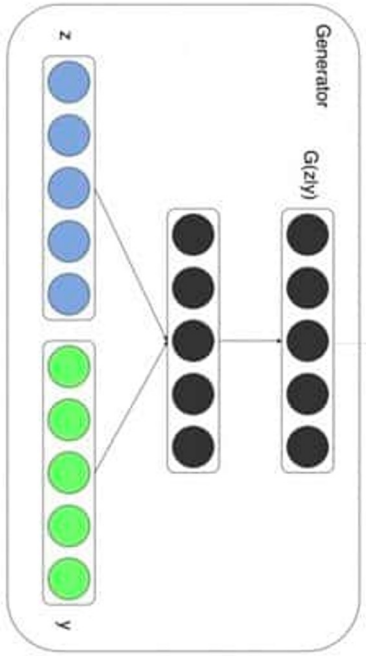
DC- cGAN Discriminator

```
class Discriminator(nn.Module):
    def __init__(self, num_classes=10, channels=3, conv_dim=64):
        super(Discriminator, self).__init__()
        self.image_size = 32
        self.label_embedding = nn.Embedding(num_classes, self.image_size*self.image_size)

        self.conv1 = nn.Conv2d(channels + 1, conv_dim, 4, 2, 1, bias=True)
        self.conv2 = nn.Conv2d(conv_dim, conv_dim * 2, 4, 2, 1, bias=False)
        self.conv2_bn=nn.BatchNorm2d(conv_dim * 2)
        self.conv3 = nn.Conv2d(conv_dim * 2, conv_dim * 4, 4, 2, 1, bias=False)
        self.conv3_bn=nn.BatchNorm2d(conv_dim * 4)
        self.conv4 = nn.Conv2d(conv_dim * 4, 1, 4, 1, 0, bias=True)
```

GAN Training. General Code – DC-cGAN

0. Create ANNs

cGAN



DC-cGAN Generator

```
class Generator(nn.Module):
    def __init__(self, z_dim=10, num_classes=10, label_embed_size=5, channels=3, conv_dim=64):
        super(Generator, self).__init__()
        self.label_embedding = nn.Embedding(num_classes, label_embed_size)
        self.tconv1 = nn.ConvTranspose2d(z_dim + label_embed_size, conv_dim * 4, 4, 2, 0, bias=False)
        self.tconv1_bn=nn.BatchNorm2d(conv_dim * 4)
        self.tconv2 = nn.ConvTranspose2d(conv_dim * 4, conv_dim * 2, 4, 2, 1, bias=False)
        self.tconv2_bn=nn.BatchNorm2d(conv_dim * 2)
        self.tconv3 = nn.ConvTranspose2d(conv_dim * 2, conv_dim, 4, 2, 1, bias=False)
        self.tconv3_bn=nn.BatchNorm2d(conv_dim)
        self.tconv4 = nn.ConvTranspose2d(conv_dim, channels, 4, 2, 1, bias=True)
```

GAN Training. Source Code Example 1

Example: Train a generator to create samples of Fashion-MNIST (grayscale clothing 28x28 images). Fashion-MNIST is a dataset of Zalando's article images—consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes.
It shares the same image size and structure of MNIST training and testing splits

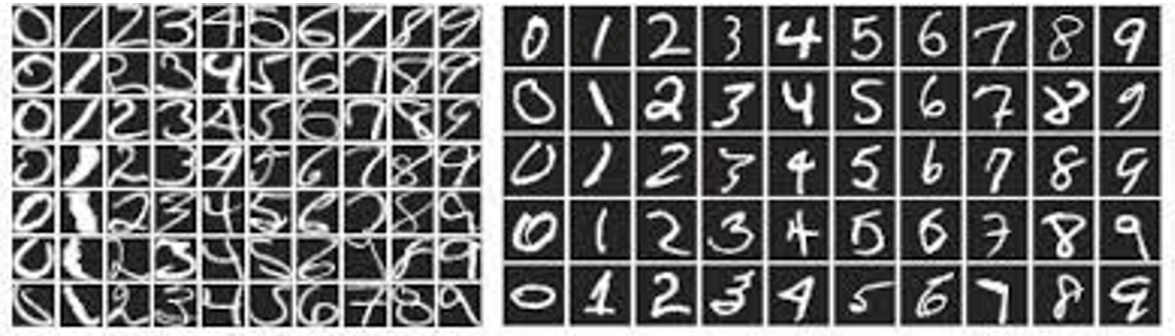
A MLP-based GAN is used
The models are created using Sequential API

Source code:
https://drive.google.com/file/d/108AqhK8awDbw6CMpVg850HhLb_aLGUTf/view?usp=sharing

GAN Training. Source Code Example 2

Example: CNN-based cGAN used to generate grayscale images. No Sequential API is used. The images are resized to be 32x32 pixels. The GAN is applied over three different datasets: MNIST, FashionMNIST, and USPS.

```
# Dataset parameters
DB = 'FashionMNIST'  # MNIST | FashionMNIST | USPS
CHANNELS = 1
```



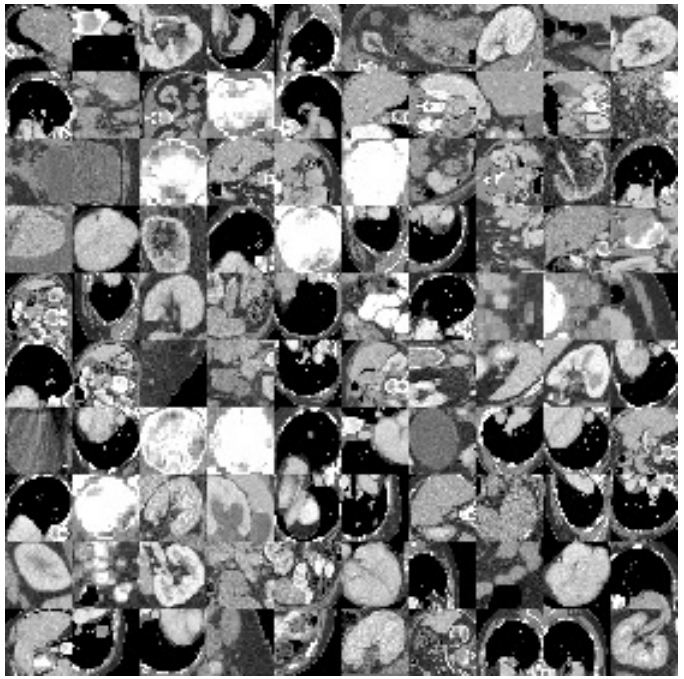
Source code:
https://drive.google.com/file/d/1ryc3k1wP1vo3fxvmS2gKfU_dXP05lymC/view?usp=sharing

GAN Training. Exercise

MedMinst offers a series of small medical image datasets for training machine learning models (https://medmnist.com/). These sets provide with different image types, sizes and colors.

Implement an MLP-based cGAN that learns to generate images from the OrganaMnist subset.

Source code:
https://drive.google.com/file/d/19FldXoOL1ETrtmXXno0zHfGdjbTN5JTd/view?usp=sharing



Thanks!

Comments?

JAMAL TOUTOUH
jamal@uma.es
jamal.es
@jamtou

Sergio Nesmachnow
sergion@fing.edu.uy