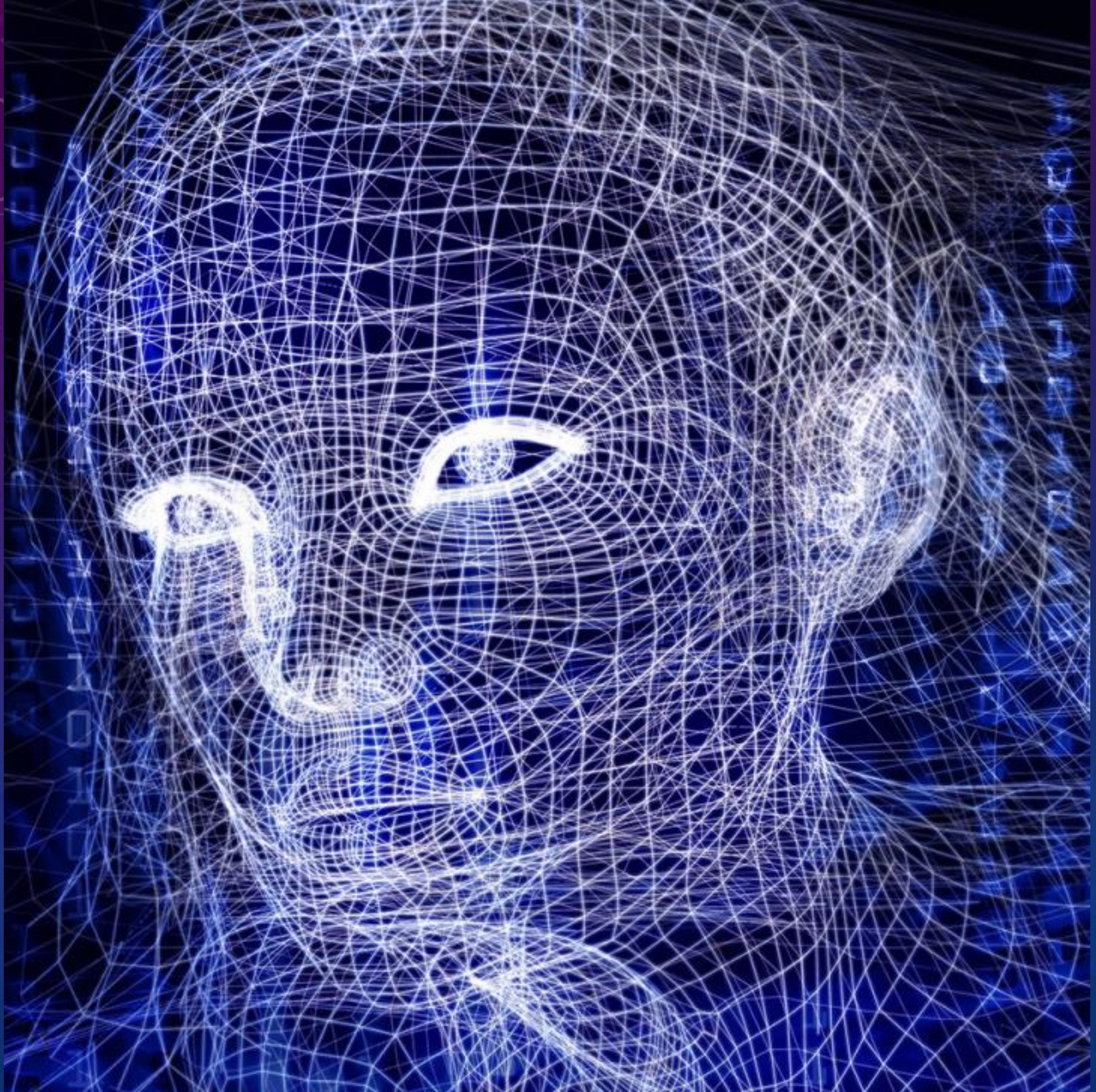
COMPUTER VISION AND ITS APPLICATIONS

CH 3_EXERCISE

徐繼聖

Gee-Sern Jison Hsu

National Taiwan University of Science and Technology



Example 3-1: Convolution and Deconvolution

- Please download the “3-1_Convolution and Deconvolution.zip” from the Moodle.
- Upload the “sample.jpg” to the Google Colab.
- Run the codes and get the output images through the convolution and deconvolution.

Example 3-1: Convolution and Deconvolution

```
# First read the input image
img = cv2.imread('./sample.jpg')
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img = cv2.resize(img,(112,112))
plt.imshow(img)
img = torch.from_numpy(img.transpose((2, 0, 1)))
img_tensor = img.float().div(255).unsqueeze(0)
print('*****Dimensions of input image******  ',img_tensor.shape)
```

Image path

Resize the image to 112 by 112

We need to transpose the image in order to match the format of pytorch
The format of image:
Numpy: (Height, Width, Channel)
Torch.Tensor(Channel, Height, Width)

Example 3-1: Convolution and Deconvolution

Create the Network with 1 forward convolution

In [4]:

```
#Define the neural Network
class Convolution(nn.Module):
    def __init__(self):
        super(Convolution, self).__init__()
        kernel = [[1/9, 1/9, 1/9],
                  [1/9, 1/9, 1/9],
                  [1/9, 1/9, 1/9]]
        kernel = torch.FloatTensor(kernel).unsqueeze(0).unsqueeze(0)
        self.weight = nn.Parameter(data=kernel, requires_grad=False)

    def forward(self, x):
        x1 = x[:, 0]
        x1 = F.conv2d(x1.unsqueeze(1), self.weight, stride=2, padding=0)

        return x1
```

Filter parameters

Single channel and create the forward path.

Example 3-1: Convolution and Deconvolution

Feed the input image to the convolution

```
5]: #create an instance of our FilterClass
    Convolution1 = Convolution()

    print('shape',img_tensor.shape)
    out = Convolution1(img_tensor)
    #out1=out.copy()
    print(out.size())
    #print(out)

    img_out = out.mul(255).byte()
    img_out = img_out.cpu().numpy().squeeze(0).transpose((1, 2, 0))
    print('*****Dimensions of output image******:',img_out.shape)

    plt.imshow(img_out[:,:,0])
```

Show the image.

Example 3-1: Convolution and Deconvolution

Create the Deconvolution Network with 1 deconvolution layer

```
#Define the class Deconvolution
class Deconvolution(nn.Module):
  def __init__(self):
    super(Deconvolution, self).__init__()

    kernel = [[1/9, 1/9, 1/9],
              [1/9, 1/9, 1/9],
              [1/9, 1/9, 1/9]]
    #kernel3d =[kernel,kernel,kernel]

    kernel = torch.FloatTensor(kernel).unsqueeze(0).unsqueeze(0)
    self.weight = nn.Parameter(data=kernel, requires_grad=False)

  def forward(self, x):
    x1 = x[:, 0]
    x1 = F.conv_transpose2d(x1.unsqueeze(1), self.weight, stride=2, padding=0)
    return x1
```

Example 3-1: Convolution and Deconvolution

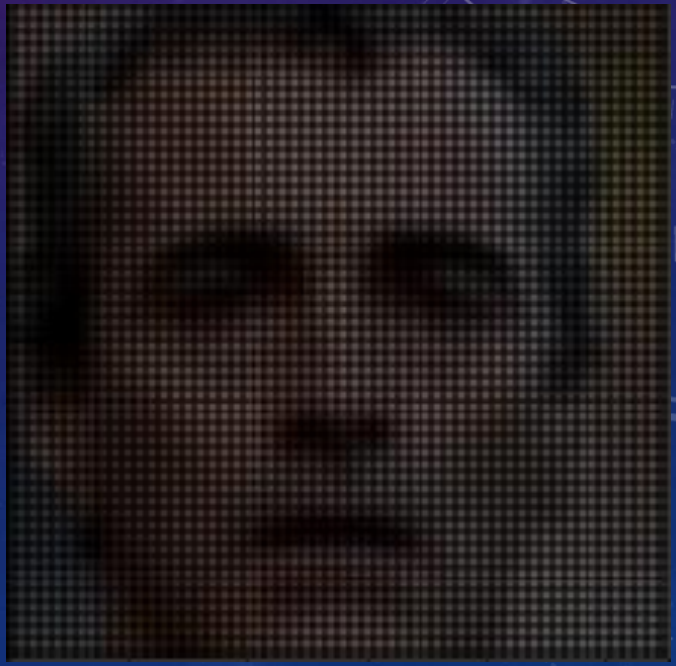
Perform the deconvolution

```
#create an instance of our Deconvolution Class


Deconvolution1 = Deconvolution()
out = Deconvolution1(out)
print(out.size())
#print(out)

img_out = out.mul(255).byte()
img_out = img_out.cpu().numpy().squeeze(0).transpose((1, 2, 0))
print('*****Dimensions of output image******:',img_out.shape)

print(plt.imshow(img_out[:,:,0]))
```



Exercise 3-1: Convolution and Deconvolution

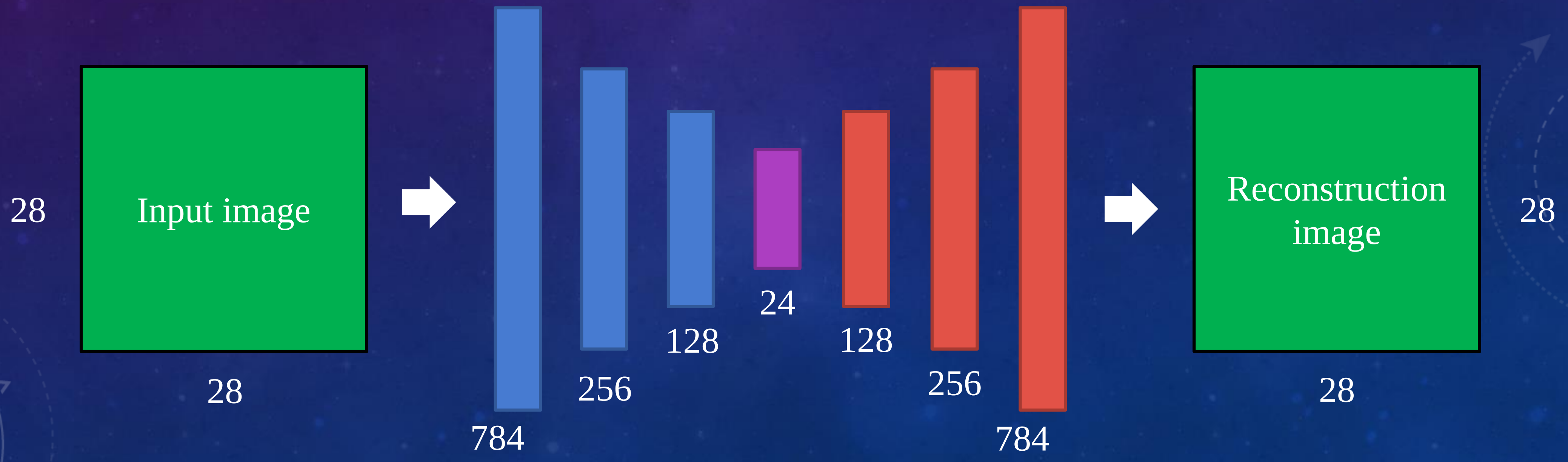
- Please download the “3-1_Convolution and Deconvolution.zip” from the Moodle.
- Upload the “sample.jpg” to the Google Colab.
- Follow the example code and design a convolution kernal and a deconvolution kernal.
- Run the codes and get the output images through the designed convolution and deconvolution kernals.

Please write down your results and codes in MS Word, then upload to the Moodle.

Exercise 3.2 : Autoencoder

- Please download the "exercise4.1_Autoencoder.ipynb" on Moodle.
    - Train the autoencoder and compare the images that reconstruct from the different epoch.
    - Change the encoder and decoder to the below architecture and compare the difference.

Please copy your results and code and paste to a MS Word, then upload to Moodle.



Example 3.2 : Autoencoder

Define the hyper-parameter and load the training data

```
num_epochs = 10
batch_size = 32
learning_rate = 1e-3

img_transform = transforms.Compose([transforms.ToTensor()])

dataset = torchvision.datasets.MNIST(root='./data', train=True, download=True, transform=img_transform)
dataloader = torch.utils.data.DataLoader(dataset, batch_size=batch_size, shuffle=False)
```

Define the hyperparameter

Download the Mnist dataset to the folder './data'

Example 3.2 : Autoencoder

Define the model architecture

```
class autoencoder(nn.Module):
    def __init__(self):
        super(autoencoder, self).__init__()
        self.encoder = nn.Sequential(
            nn.Linear(28 * 28, 128),
            nn.ReLU(True),
            nn.Linear(128, 64),
            nn.ReLU(True),
            nn.Linear(64, 12),
            nn.ReLU(True),
            nn.Linear(12, 3))
        self.decoder = nn.Sequential(
            nn.Linear(3, 12),
            nn.ReLU(True),
            nn.Linear(12, 64),
            nn.ReLU(True),
            nn.Linear(64, 128),
            nn.ReLU(True), nn.Linear(128, 28 * 28), nn.Tanh())
```

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

```
autoencoder(
  (encoder): Sequential(
    (0): Linear(in_features=784, out_features=128, bias=True)
    (1): ReLU(inplace=True)
    (2): Linear(in_features=128, out_features=64, bias=True)
    (3): ReLU(inplace=True)
    (4): Linear(in_features=64, out_features=12, bias=True)
    (5): ReLU(inplace=True)
    (6): Linear(in_features=12, out_features=3, bias=True)
  )
  (decoder): Sequential(
    (0): Linear(in_features=3, out_features=12, bias=True)
    (1): ReLU(inplace=True)
    (2): Linear(in_features=12, out_features=64, bias=True)
    (3): ReLU(inplace=True)
    (4): Linear(in_features=64, out_features=128, bias=True)
    (5): ReLU(inplace=True)
    (6): Linear(in_features=128, out_features=784, bias=True)
    (7): Tanh()
  )
)
```

Example 3.2 : Autoencoder

Define the model architecture

```
class autoencoder(nn.Module):
    def __init__(self):
        super(autoencoder, self).__init__()
        self.encoder = nn.Sequential(
            nn.Linear(28 * 28, 128),
            nn.ReLU(True),
            nn.Linear(128, 64),
            nn.ReLU(True),
            nn.Linear(64, 12),
            nn.ReLU(True),
            nn.Linear(12, 3))
```

Define the encoder

Define the decoder

```
self.decoder = nn.Sequential(
            nn.Linear(3, 12),
            nn.ReLU(True),
            nn.Linear(12, 64),
            nn.ReLU(True),
            nn.Linear(64, 128),
            nn.ReLU(True),
            nn.Linear(128, 28 * 28),
nn.Tanh())

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

Example 3.2 : Autoencoder

Define the model architecture

```
autoencoder(
  (encoder): Sequential(
    (0): Linear(in_features=784, out_features=128, bias=True)
    (1): ReLU(inplace=True)
    (2): Linear(in_features=128, out_features=64, bias=True)
    (3): ReLU(inplace=True)
    (4): Linear(in_features=64, out_features=12, bias=True)
    (5): ReLU(inplace=True)
    (6): Linear(in_features=12, out_features=3, bias=True)
  )
  (decoder): Sequential(
    (0): Linear(in_features=3, out_features=12, bias=True)
    (1): ReLU(inplace=True)
    (2): Linear(in_features=12, out_features=64, bias=True)
    (3): ReLU(inplace=True)
    (4): Linear(in_features=64, out_features=128, bias=True)
    (5): ReLU(inplace=True)
    (6): Linear(in_features=128, out_features=784, bias=True)
    (7): Tanh()
  )
)
```

Example 3.2 : Autoencoder

Use the Mean Square Error as the loss function

```
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate, weight_decay=1e-5)
```

```
for epoch in range(num_epochs):
    for data in dataloader:
        img, _ = data
        img = img.view(img.size(0), -1)
        img = Variable(img).cuda()
        output = model(img)
        loss = criterion(output, img)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
    print('epoch [{}/{}], loss:{:.4f}'.format(epoch + 1, num_epochs, loss.item()))
```

Flatten the input images

Compute the loss

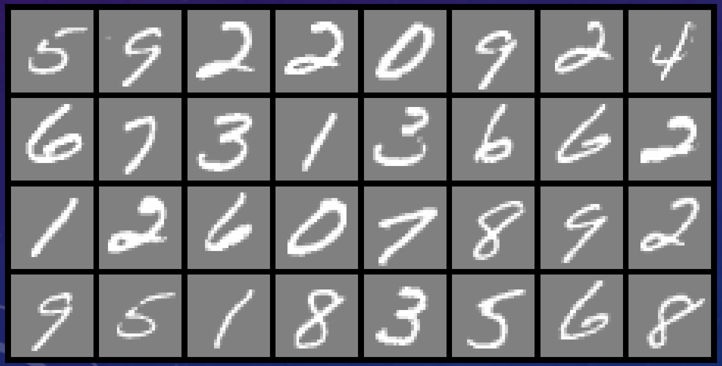
Example 3.2 : Autoencoder

```
if epoch % 1 == 0:
    img = to_img(img.cpu().data)
    save_image(img, ‘./AE_img/input_{}.png'.format(epoch))
    pic = to_img(output.cpu().data)
    save_image(pic, './ AE _img/output_{}.png'.format(epoch))
```

→ Save the input images

→ Save the reconstruction images

Input images



Epoch 1 : reconstruction images

Epoch 10 : reconstruction images

Exercise 3.3 - VAE

- Please download the “exercise4.2_VAE.ipynb” on Moodle.
- Please use different latent vector dimension
    1. 1 dimension
    2. 10 dimension
    3. 100 dimension

Please copy your results and code and paste to a MS Word, then upload to Moodle.

Example 3.3 : VAE

Define the model architecture

Define the latent vector dimension

```
class VAE(nn.Module):
    def __init__(self, image_channels=1, h_dim=1600, z_dim=20):
        super(VAE, self).__init__()

        self.conv1 = nn.Conv2d(image_channels, 32, kernel_size=4, stride=2)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=4, stride=2)

        self.fc1 = nn.Linear(h_dim, z_dim)
        self.fc2 = nn.Linear(h_dim, z_dim)
        self.fc3 = nn.Linear(z_dim, h_dim)

        self.deconv2 = nn.ConvTranspose2d(64, 32, kernel_size=5, stride=2)
        self.deconv1 = nn.ConvTranspose2d(32, image_channels, kernel_size=4, stride=2)
```

Example 3.3 : VAE

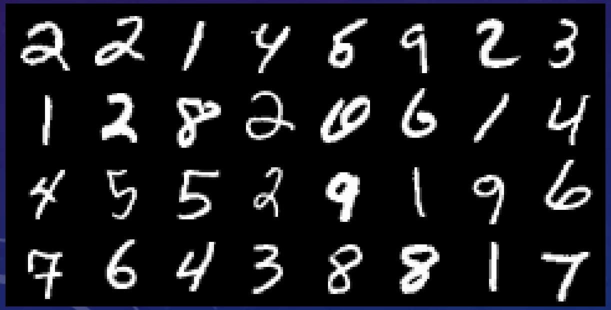
```
if epoch % 1 == 0:
    save = img.cpu().data
    save_image(img, './vae_img/input_{}.png'.format(epoch))
    save = recon_batch.cpu().data
    save_image(save, './vae_img/output_{}.png'.format(epoch))
```

Save the input images

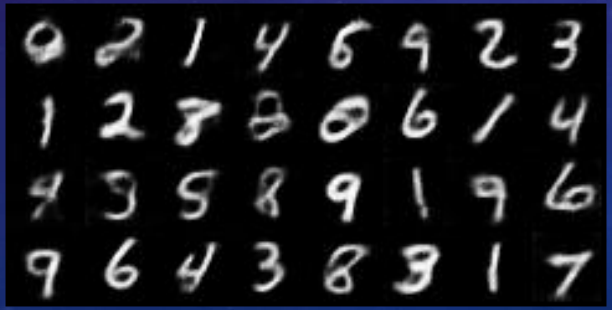
Save the reconstruction images

Epoch: 0

Input

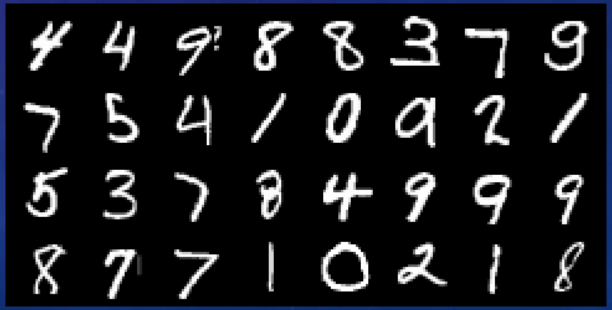


Reconstruct



Epoch: 4

Input



Reconstruct