Conditional GAN (C-GAN)

Originally proposed by Mirza et al. is their work titled Conditional Generative Adversarial Nets. This network uses a basic implementation where generator and discriminator models are MLPs with additional inputs for conditioning with class labels. This notebook trains both networks using ADAM optimizer to play the minimax game. We showcase the effectiveness using MNIST digit generation

Conditional GAN (C-GAN)

Originally proposed by Mirza et al. is their work titled Conditional Generative Adversarial Nets. This network uses a basic implementation where generator and discriminator models are MLPs with additional inputs for conditioning with class labels. This notebook trains both networks using ADAM optimizer to play the minimax game. We showcase the effectiveness using MNIST digit generation

In [1]:
```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision.datasets as datasets
import torchvision.transforms as transforms
from torch.utils.data import DataLoader
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
# Define the generator
class Generator(nn.Module):
    def __init__(self, z_dim, num_classes):
        super(Generator, self).__init__()
        self.label_emb = nn.Embedding(num_classes, z_dim)
        self.model = nn.Sequential(
            nn.Linear(z_dim * 2, 128),
            nn.ReLU(),
            nn.Linear(128, 256),
            nn.ReLU(),
            nn.Linear(256, 512),
            nn.ReLU(),
            nn.Linear(512, 28 * 28),
            nn.Tanh()
        )

    def forward(self, z, labels):
        z = torch.cat([z, self.label_emb(labels)], dim=1)
        return self.model(z).view(-1, 1, 28, 28)
```

In [3]:
```
# Define the discriminator
class Discriminator(nn.Module):
    def __init__(self, num_classes):
        super(Discriminator, self).__init__()
        self.label_emb = nn.Embedding(num_classes, 28 * 28)
```

```
        self.model = nn.Sequential(
            nn.Linear(28 * 28 * 2, 512),
            nn.LeakyReLU(0.2),
            nn.Linear(512, 256),
            nn.LeakyReLU(0.2),
            nn.Linear(256, 1),
            nn.Sigmoid()
        )

    def forward(self, img, labels):
        img = img.view(img.size(0), -1)
        d_input = torch.cat([img, self.label_emb(labels)], dim=1)
        return self.model(d_input)
```

In [4]:
```
# Set hyperparameters
z_dim = 100
num_classes = 10
batch_size = 128
epochs = 1000
```

In [5]:
```
# Load dataset
transform = transforms.Compose([transforms.ToTensor(), transforms.Normalize([0.5
dataset = datasets.MNIST(root="./data", train=True, transform=transform, downloa
dataloader = DataLoader(dataset, batch_size=batch_size, shuffle=True)

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

```
100%|
| 9.91M/9.91M [00:11<00:00, 884kB/s]
100%|
| 28.9k/28.9k [00:00<00:00, 113kB/s]
100%|
| 1.65M/1.65M [00:02<00:00, 563kB/s]
100%|
| 4.54k/4.54k [00:00<00:00, 4.55MB/s]
```

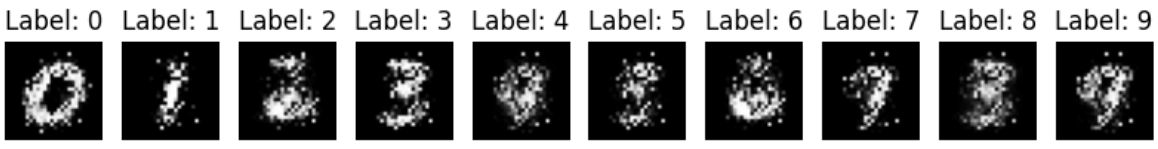
In [6]:
```
# Initialize models
generator = Generator(z_dim, num_classes).to(device)
discriminator = Discriminator(num_classes).to(device)
```

In [7]:
```
# Loss and optimizers
criterion = nn.BCELoss()
optimizer_G = optim.Adam(generator.parameters(), lr=0.0002, betas=(0.5, 0.999))
optimizer_D = optim.Adam(discriminator.parameters(), lr=0.0002, betas=(0.5, 0.99
```

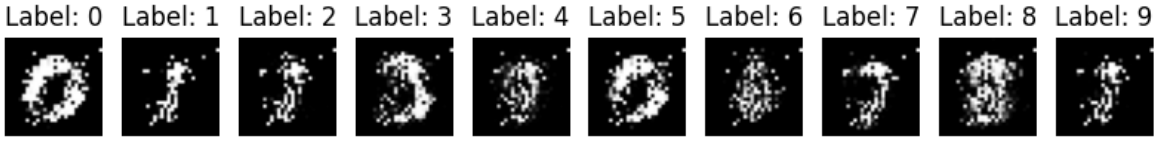
In [8]:
```python
def generate_and_show_images(epoch):
    generator.eval()
    z = torch.randn(10, z_dim, device=device)
    labels = torch.arange(10, device=device)
    with torch.no_grad():
        fake_imgs = generator(z, labels).cpu().numpy()
    generator.train()

    fig, axes = plt.subplots(1, 10, figsize=(10, 2))
    for i, img in enumerate(fake_imgs):
        axes[i].imshow(img[0], cmap='gray')
        axes[i].set_title(f"Label: {i}")
        axes[i].axis('off')
    plt.show()
```

In [ ]:
```python
# Training Loop
for epoch in range(epochs):
    for i, (imgs, labels) in enumerate(dataloader):
        imgs, labels = imgs.to(device), labels.to(device)
        batch_size = imgs.shape[0]

        # Prepare labels
        real_labels = torch.ones(batch_size, 1, device=device)
        fake_labels = torch.zeros(batch_size, 1, device=device)

        # Train discriminator
        optimizer_D.zero_grad()
        z = torch.randn(batch_size, z_dim, device=device)
        fake_imgs = generator(z, labels)
        real_loss = criterion(discriminator(imgs, labels), real_labels)
        fake_loss = criterion(discriminator(fake_imgs.detach(), labels), fake_la
        d_loss = real_loss + fake_loss
        d_loss.backward()
        optimizer_D.step()

        # Train generator
        optimizer_G.zero_grad()
        g_loss = criterion(discriminator(fake_imgs, labels), real_labels)
        g_loss.backward()
        optimizer_G.step()

    print(f"Epoch [{epoch+1}/{epochs}] | D Loss: {d_loss.item():.4f} | G Loss: {
    if (epoch + 1) % 1 == 0:
        generate_and_show_images(epoch)
```

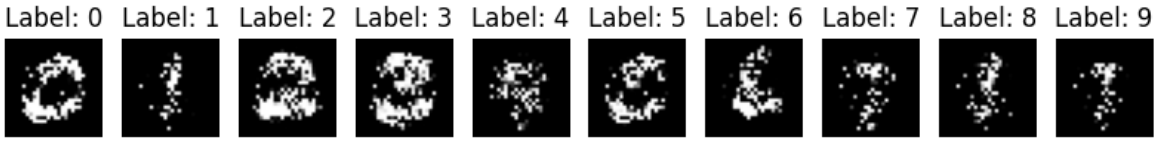
```
Epoch [1/1000] | D Loss: 1.1261 | G Loss: 2.5655
```



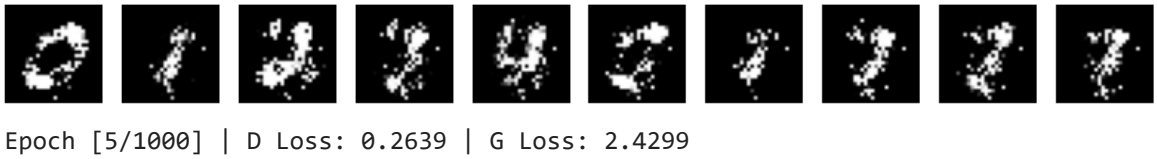

```
Epoch [2/1000] | D Loss: 0.4935 | G Loss: 3.3080
```



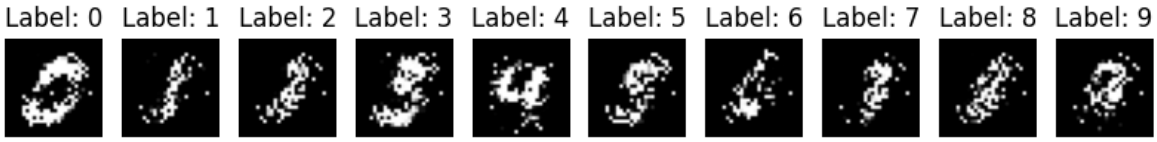

```
Epoch [3/1000] | D Loss: 0.4222 | G Loss: 2.9534
```



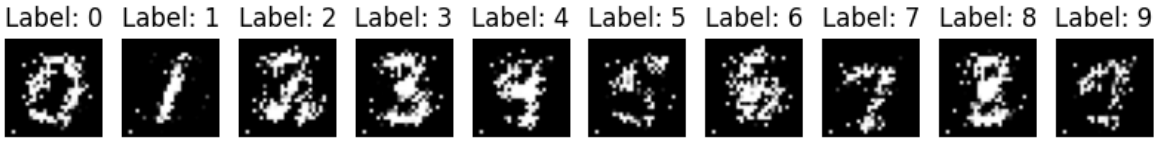
Epoch [4/1000] | D Loss: 0.3580 | G Loss: 3.3466



Epoch [5/1000] | D Loss: 0.2639 | G Loss: 2.4299

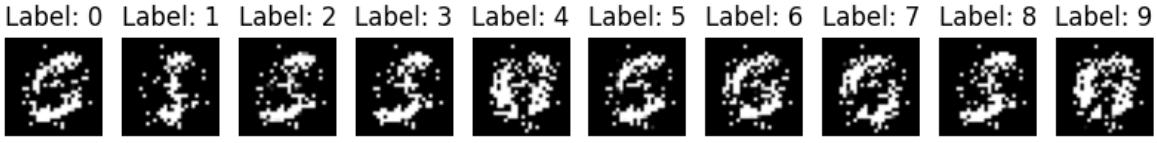


Epoch [6/1000] | D Loss: 0.3716 | G Loss: 3.4267



Epoch [7/1000] | D Loss: 0.1991 | G Loss: 3.7193

Epoch [8/1000] | D Loss: 0.0071 | G Loss: 6.4297



Epoch [9/1000] | D Loss: 0.0154 | G Loss: 5.7319

Epoch [10/1000] | D Loss: 0.0008 | G Loss: 7.7892

Epoch [11/1000] | D Loss: 0.0165 | G Loss: 6.9286

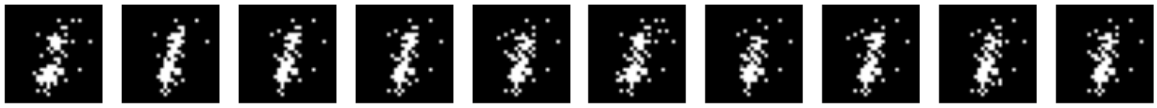
Epoch [12/1000] | D Loss: 0.0557 | G Loss: 5.7088

Epoch [13/1000] | D Loss: 0.0012 | G Loss: 7.6813

Label: 0 Label: 1 Label: 2 Label: 3 Label: 4 Label: 5 Label: 6 Label: 7 Label: 8 Label: 9

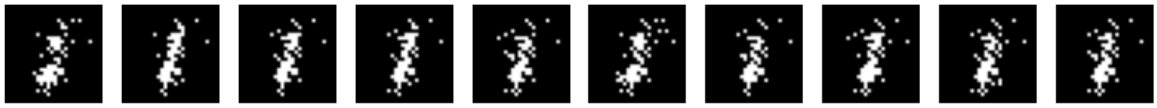
```
Epoch [14/1000] | D Loss: 0.0002 | G Loss: 8.9261
```

Label: 0 Label: 1 Label: 2 Label: 3 Label: 4 Label: 5 Label: 6 Label: 7 Label: 8 Label: 9



```
Epoch [15/1000] | D Loss: 0.0003 | G Loss: 9.1910
```

Label: 0 Label: 1 Label: 2 Label: 3 Label: 4 Label: 5 Label: 6 Label: 7 Label: 8 Label: 9



```
Epoch [16/1000] | D Loss: 0.0003 | G Loss: 8.6102
```

Label: 0 Label: 1 Label: 2 Label: 3 Label: 4 Label: 5 Label: 6 Label: 7 Label: 8 Label: 9

```
Epoch [17/1000] | D Loss: 0.0211 | G Loss: 5.8722
```

Label: 0 Label: 1 Label: 2 Label: 3 Label: 4 Label: 5 Label: 6 Label: 7 Label: 8 Label: 9

```
Epoch [18/1000] | D Loss: 0.0001 | G Loss: 10.3863
```

Label: 0 Label: 1 Label: 2 Label: 3 Label: 4 Label: 5 Label: 6 Label: 7 Label: 8 Label: 9

```
Epoch [19/1000] | D Loss: 0.0042 | G Loss: 7.3401
```

Label: 0 Label: 1 Label: 2 Label: 3 Label: 4 Label: 5 Label: 6 Label: 7 Label: 8 Label: 9

```
Epoch [20/1000] | D Loss: 0.0001 | G Loss: 10.0092
```

Label: 0 Label: 1 Label: 2 Label: 3 Label: 4 Label: 5 Label: 6 Label: 7 Label: 8 Label: 9

```
Epoch [21/1000] | D Loss: 0.0000 | G Loss: 10.9068
```

Label: 0 Label: 1 Label: 2 Label: 3 Label: 4 Label: 5 Label: 6 Label: 7 Label: 8 Label: 9

```
Epoch [22/1000] | D Loss: 0.0000 | G Loss: 11.8375
```

Label: 0 Label: 1 Label: 2 Label: 3 Label: 4 Label: 5 Label: 6 Label: 7 Label: 8 Label: 9

```
Epoch [23/1000] | D Loss: 0.0000 | G Loss: 11.9878
```

Label: 0 Label: 1 Label: 2 Label: 3 Label: 4 Label: 5 Label: 6 Label: 7 Label: 8 Label: 9

Epoch [24/1000] | D Loss: 0.0003 | G Loss: 8.7661

Label: 0 Label: 1 Label: 2 Label: 3 Label: 4 Label: 5 Label: 6 Label: 7 Label: 8 Label: 9

Epoch [25/1000] | D Loss: 0.0000 | G Loss: 11.2474

Label: 0 Label: 1 Label: 2 Label: 3 Label: 4 Label: 5 Label: 6 Label: 7 Label: 8 Label: 9

Epoch [26/1000] | D Loss: 0.0000 | G Loss: 11.7373

Label: 0 Label: 1 Label: 2 Label: 3 Label: 4 Label: 5 Label: 6 Label: 7 Label: 8 Label: 9

Epoch [27/1000] | D Loss: 0.0000 | G Loss: 12.4325

Label: 0 Label: 1 Label: 2 Label: 3 Label: 4 Label: 5 Label: 6 Label: 7 Label: 8 Label: 9

Epoch [28/1000] | D Loss: 0.0000 | G Loss: 12.0832

Label: 0 Label: 1 Label: 2 Label: 3 Label: 4 Label: 5 Label: 6 Label: 7 Label: 8 Label: 9

Epoch [29/1000] | D Loss: 0.0000 | G Loss: 12.5280

Label: 0 Label: 1 Label: 2 Label: 3 Label: 4 Label: 5 Label: 6 Label: 7 Label: 8 Label: 9

Epoch [30/1000] | D Loss: 0.0000 | G Loss: 12.6174

Label: 0 Label: 1 Label: 2 Label: 3 Label: 4 Label: 5 Label: 6 Label: 7 Label: 8 Label: 9

Epoch [31/1000] | D Loss: 0.0000 | G Loss: 12.7858

Label: 0 Label: 1 Label: 2 Label: 3 Label: 4 Label: 5 Label: 6 Label: 7 Label: 8 Label: 9

Epoch [32/1000] | D Loss: 0.0000 | G Loss: 13.0825

Label: 0 Label: 1 Label: 2 Label: 3 Label: 4 Label: 5 Label: 6 Label: 7 Label: 8 Label: 9

Epoch [33/1000] | D Loss: 0.0001 | G Loss: 9.8179

Epoch [34/1000] | D Loss: 0.0001 | G Loss: 10.3254

Epoch [35/1000] | D Loss: 0.0052 | G Loss: 8.6352

Epoch [36/1000] | D Loss: 0.0001 | G Loss: 11.0021

Epoch [37/1000] | D Loss: 0.0000 | G Loss: 11.6431

Epoch [38/1000] | D Loss: 0.0002 | G Loss: 12.1269

Epoch [39/1000] | D Loss: 0.0000 | G Loss: 12.9875

Epoch [40/1000] | D Loss: 0.0000 | G Loss: 12.8319

Epoch [41/1000] | D Loss: 0.0000 | G Loss: 13.2082

Epoch [42/1000] | D Loss: 0.0000 | G Loss: 13.4357

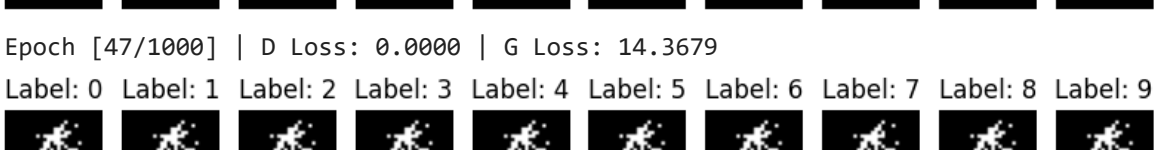
Epoch [43/1000] | D Loss: 0.0000 | G Loss: 13.7428

Epoch [44/1000] | D Loss: 0.0000 | G Loss: 13.3872

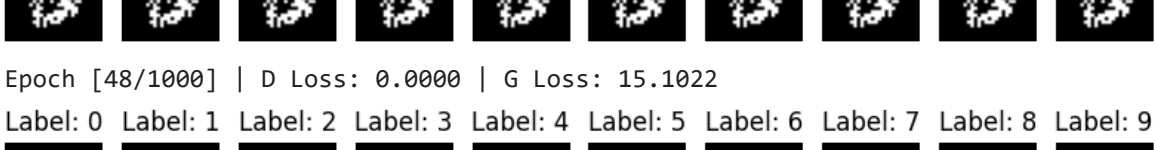
Epoch [45/1000] | D Loss: 0.0000 | G Loss: 13.8519

Epoch [46/1000] | D Loss: 0.0000 | G Loss: 13.9370

Epoch [47/1000] | D Loss: 0.0000 | G Loss: 14.3679



Epoch [48/1000] | D Loss: 0.0000 | G Loss: 15.1022



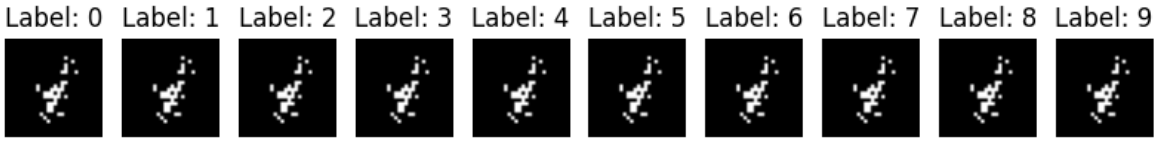
Epoch [49/1000] | D Loss: 0.0000 | G Loss: 15.0676

Epoch [50/1000] | D Loss: 0.0000 | G Loss: 15.3978

Epoch [51/1000] | D Loss: 0.0000 | G Loss: 15.5206

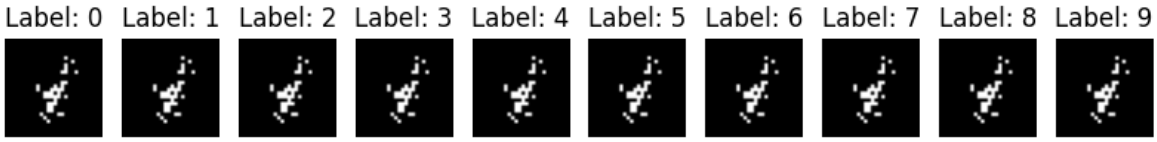
Epoch [52/1000] | D Loss: 0.0000 | G Loss: 15.7414

Epoch [53/1000] | D Loss: 0.0026 | G Loss: 7.7218



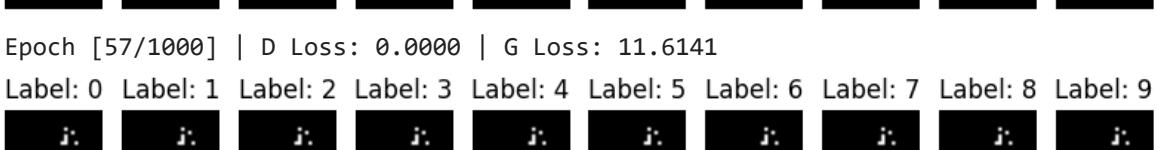
Epoch [54/1000] | D Loss: 0.0001 | G Loss: 10.0251

Epoch [55/1000] | D Loss: 0.0001 | G Loss: 10.4928

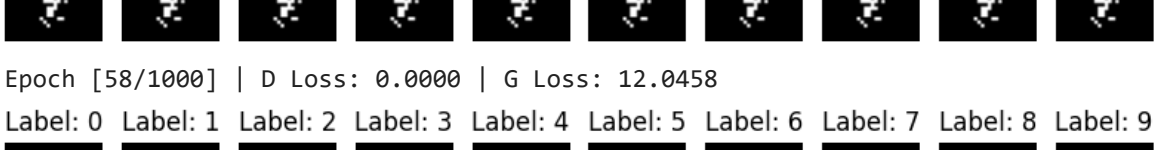


Epoch [56/1000] | D Loss: 0.0000 | G Loss: 11.2056

Epoch [57/1000] | D Loss: 0.0000 | G Loss: 11.6141



Epoch [58/1000] | D Loss: 0.0000 | G Loss: 12.0458



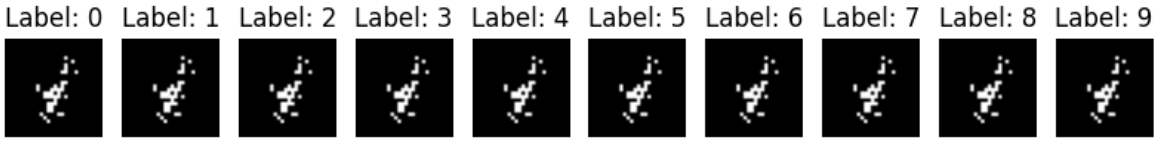
Epoch [59/1000] | D Loss: 0.0000 | G Loss: 12.4808

Epoch [60/1000] | D Loss: 0.0000 | G Loss: 13.0851

Epoch [61/1000] | D Loss: 0.0000 | G Loss: 13.3643

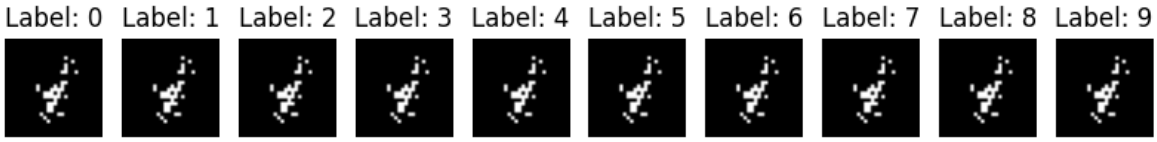
Epoch [62/1000] | D Loss: 0.0000 | G Loss: 13.7085

Epoch [63/1000] | D Loss: 0.0000 | G Loss: 13.9850



Epoch [64/1000] | D Loss: 0.0000 | G Loss: 14.1755

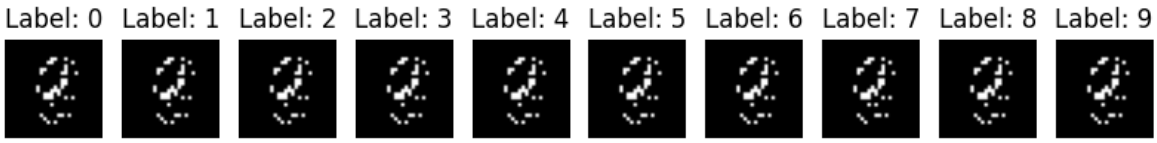
Epoch [65/1000] | D Loss: 0.0000 | G Loss: 14.3996



Epoch [66/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [67/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [68/1000] | D Loss: 100.0000 | G Loss: 0.0000



Epoch [69/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [70/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [71/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [72/1000] | D Loss: 100.0000 | G Loss: 0.0000

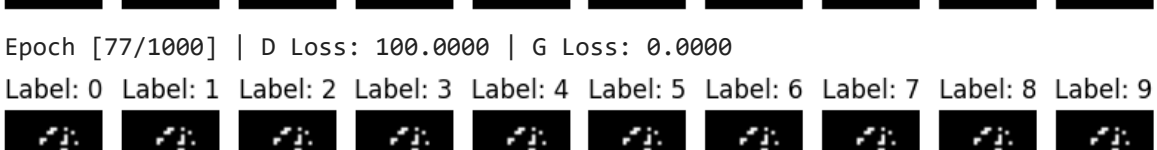
Epoch [73/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [74/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [75/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [76/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [77/1000] | D Loss: 100.0000 | G Loss: 0.0000

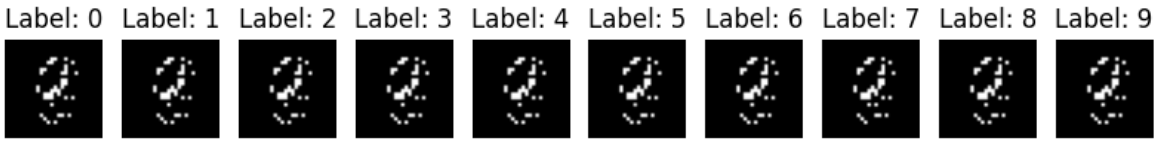


Epoch [78/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [79/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [80/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [81/1000] | D Loss: 100.0000 | G Loss: 0.0000



Epoch [82/1000] | D Loss: 100.0000 | G Loss: 0.0000

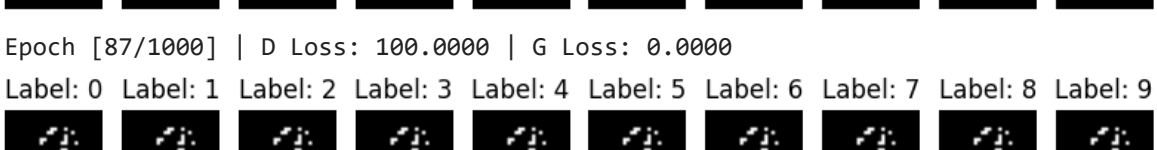
Epoch [83/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [84/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [85/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [86/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [87/1000] | D Loss: 100.0000 | G Loss: 0.0000

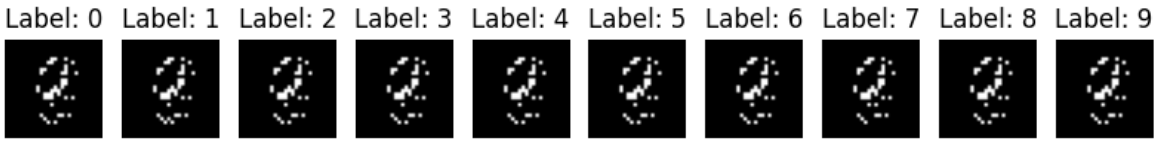


Epoch [88/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [89/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [90/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [91/1000] | D Loss: 100.0000 | G Loss: 0.0000



Epoch [92/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [93/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [94/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [95/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [96/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [97/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [98/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [99/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [100/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [101/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [102/1000] | D Loss: 100.0000 | G Loss: 0.0000

Epoch [103/1000] | D Loss: 100.0000 | G Loss: 0.0000


```
Epoch [104/1000] | D Loss: 100.0000 | G Loss: 0.0000
```


```
Epoch [105/1000] | D Loss: 100.0000 | G Loss: 0.0000
```


```
Epoch [106/1000] | D Loss: 100.0000 | G Loss: 0.0000
```

In [ ]: