

# GANs-PyTorch

November 28, 2024

```
[1]: #COMMENT IF NOT USING COLAB VM

# This mounts your Google Drive to the Colab VM.
from google.colab import drive
drive.mount('/content/drive')

# TODO: Enter the foldername in your Drive where you have saved the unzipped
# assignment folder, e.g. 'DeepLearning/assignments/assignment5/'
FOLDERNAME = 'U of U/Semester 7 (Fall 24)/CS Deep Learning/assignment5'
assert FOLDERNAME is not None, "[!] Enter the foldername."

# Now that we've mounted your Drive, this ensures that
# the Python interpreter of the Colab VM can load
# python files from within it.
import sys
sys.path.append('/content/drive/My Drive/{}'.format(FOLDERNAME))

# This downloads the CIFAR-10 dataset to your Drive
# if it doesn't already exist.
%cd /content/drive/My\ Drive/$FOLDERNAME/cs6353/datasets/
!bash get_datasets.sh
%cd /content/drive/My\ Drive/$FOLDERNAME
```

```
Mounted at /content/drive
/content/drive/My Drive/U of U/Semester 7 (Fall 24)/CS Deep
Learning/assignment5/cs6353/datasets
--2024-11-28 03:42:05--
http://supermoe.cs.umass.edu/682/asgns/coco_captioning.zip
Resolving supermoe.cs.umass.edu (supermoe.cs.umass.edu)... 128.119.244.95
Connecting to supermoe.cs.umass.edu
(supermoe.cs.umass.edu)|128.119.244.95|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1035210391 (987M) [application/zip]
Saving to: 'coco_captioning.zip.1'

coco_captioning.zip 12%[=>                ] 119.09M  8.54MB/s    eta 4m 6s
^C
/content/drive/My Drive/U of U/Semester 7 (Fall 24)/CS Deep Learning/assignment5
```

```

[ ]: # #UNCOMMENT IF USING CADE
# import os
# ##### Request a GPU #####
# ## This function locates an available gpu for usage. In addition, this
#   ↳function reserves a specified
#   ↳memory space exclusively for your account. The memory reservation prevents
#   ↳the decrement in computational
#   ↳speed when other users try to allocate memory on the same gpu in the
#   ↳shared systems, i.e., CADE machines.
# ## Note: If you use your own system which has a GPU with less than 4GB of
#   ↳memory, remember to change the
#   ↳specified minimum memory.
# def define_gpu_to_use(minimum_memory_mb = 3500):
#     thres_memory = 600 #
#     gpu_to_use = None
#     try:
#         os.environ['CUDA_VISIBLE_DEVICES']
#         print('GPU already assigned before: ' + str(os.
#   ↳environ['CUDA_VISIBLE_DEVICES']))
#         return
#     except:
#         pass

#     for i in range(16):
#         free_memory = !nvidia-smi --query-gpu=memory.free -i $i
#   ↳--format=csv,nounits,noheader
#         if free_memory[0] == 'No devices were found':
#             break
#         free_memory = int(free_memory[0])

#         if free_memory>minimum_memory_mb-thres_memory:
#             gpu_to_use = i
#             break

#     if gpu_to_use is None:
#         print('Could not find any GPU available with the required free memory
#   ↳of ' + str(minimum_memory_mb) \
#         + 'MB. Please use a different system for this assignment.')
#     else:
#         os.environ['CUDA_VISIBLE_DEVICES'] = str(gpu_to_use)
#         print('Chosen GPU: ' + str(gpu_to_use))

# ## Request a gpu and reserve the memory space
# define_gpu_to_use(4000)

```

# 1 Generative Adversarial Networks (GANs)

So far in cs6353, all the applications of neural networks that we have explored have been **discriminative models** that take an input and are trained to produce a labeled output. This has ranged from straightforward classification of image categories to sentence generation (which was still phrased as a classification problem, our labels were in vocabulary space and we'd learned a recurrence to capture multi-word labels). In this notebook, we will expand our repertoire, and build **generative models** using neural networks. Specifically, we will learn how to build models which generate novel images that resemble a set of training images.

## 1.0.1 What is a GAN?

In 2014, [Goodfellow et al.](#) presented a method for training generative models called Generative Adversarial Networks (GANs for short). In a GAN, we build two different neural networks. Our first network is a traditional classification network, called the **discriminator**. We will train the discriminator to take images, and classify them as being real (belonging to the training set) or fake (not present in the training set). Our other network, called the **generator**, will take random noise as input and transform it using a neural network to produce images. The goal of the generator is to fool the discriminator into thinking the images it produced are real.

We can think of this back and forth process of the generator ( $G$ ) trying to fool the discriminator ( $D$ ), and the discriminator trying to correctly classify real vs. fake as a minimax game:

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

where  $z \sim p(z)$  are the random noise samples,  $G(z)$  are the generated images using the neural network generator  $G$ , and  $D$  is the output of the discriminator, specifying the probability of an input being real. In [Goodfellow et al.](#), they analyze this minimax game and show how it relates to minimizing the Jensen-Shannon divergence between the training data distribution and the generated samples from  $G$ .

To optimize this minimax game, we will alternate between taking gradient *descent* steps on the objective for  $G$ , and gradient *ascent* steps on the objective for  $D$ : 1. update the **generator** ( $G$ ) to minimize the probability of the **discriminator making the correct choice**. 2. update the **discriminator** ( $D$ ) to maximize the probability of the **discriminator making the correct choice**.

While these updates are useful for analysis, they do not perform well in practice. Instead, we will use a different objective when we update the generator: maximize the probability of the **discriminator making the incorrect choice**. This small change helps to alleviate problems with the generator gradient vanishing when the discriminator is confident. This is the standard update used in most GAN papers, and was used in the original paper from [Goodfellow et al.](#)

In this assignment, we will alternate the following updates: 1. Update the generator ( $G$ ) to maximize the probability of the discriminator making the incorrect choice on generated data:

$$\underset{G}{\text{maximize}} \mathbb{E}_{z \sim p(z)} [\log D(G(z))]$$

2. Update the discriminator ( $D$ ), to maximize the probability of the discriminator making the correct choice on real and generated data:

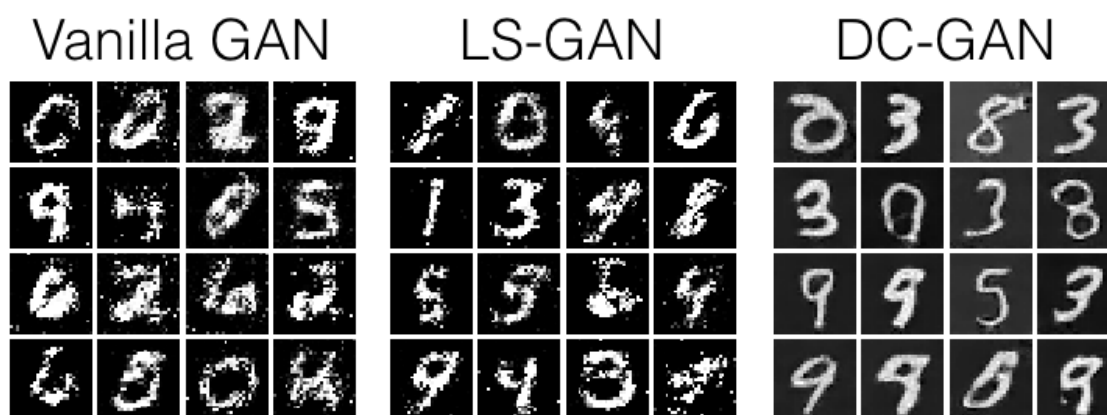
$$\underset{D}{\text{maximize}} \mathbb{E}_{x \sim p_{\text{data}}} [\log D(x)] + \mathbb{E}_{z \sim p(z)} [\log (1 - D(G(z)))]$$

### 1.0.2 What else is there?

Since 2014, GANs have exploded into a huge research area, with massive [workshops](#), and [hundreds of new papers](#). Compared to other approaches for generative models, they often produce the highest quality samples but are some of the most difficult and finicky models to train (see [this github repo](#) that contains a set of 17 hacks that are useful for getting models working). Improving the stability and robustness of GAN training is an open research question, with new papers coming out every day! For a more recent tutorial on GANs, see [here](#). There is also some even more recent exciting work that changes the objective function to Wasserstein distance and yields much more stable results across model architectures: [WGAN](#), [WGAN-GP](#).

GANs are not the only way to train a generative model! For other approaches to generative modeling check out the [deep generative model chapter](#) of the Deep Learning [book](#). Another popular way of training neural networks as generative models is Variational Autoencoders (co-discovered [here](#) and [here](#)). Variational autoencoders combine neural networks with variational inference to train deep generative models. These models tend to be far more stable and easier to train but currently don't produce samples that are as pretty as GANs.

Here's an example of what your outputs from the 3 different models you're going to train should look like... note that GANs are sometimes finicky, so your outputs might not look exactly like this... this is just meant to be a *rough* guideline of the kind of quality you can expect:



## 1.1 Setup

```
[2]: import torch
import torch.nn as nn
from torch.nn import init
import torchvision
import torchvision.transforms as T
import torch.optim as optim
from torch.utils.data import DataLoader
from torch.utils.data import sampler
import torchvision.datasets as dset

import numpy as np
```

```

import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec

%matplotlib inline
plt.rcParams['figure.figsize'] = (10.0, 8.0) # set default size of plots
plt.rcParams['image.interpolation'] = 'nearest'
plt.rcParams['image.cmap'] = 'gray'

def show_images(images):
    images = np.reshape(images, [images.shape[0], -1]) # images reshape to
    ↪ (batch_size, D)
    sqrtn = int(np.ceil(np.sqrt(images.shape[0])))
    sqrtimg = int(np.ceil(np.sqrt(images.shape[1])))

    fig = plt.figure(figsize=(sqrtn, sqrtn))
    gs = gridspec.GridSpec(sqrtn, sqrtn)
    gs.update(wspace=0.05, hspace=0.05)

    for i, img in enumerate(images):
        ax = plt.subplot(gs[i])
        plt.axis('off')
        ax.set_xticklabels([])
        ax.set_yticklabels([])
        ax.set_aspect('equal')
        plt.imshow(img.reshape([sqrtimg, sqrtimg]))
    return

def preprocess_img(x):
    return 2 * x - 1.0

def deprocess_img(x):
    return (x + 1.0) / 2.0

def rel_error(x, y):
    return np.max(np.abs(x - y) / (np.maximum(1e-8, np.abs(x) + np.abs(y))))

def count_params(model):
    """Count the number of parameters"""
    param_count = np.sum([np.prod(p.size()) for p in model.parameters()])
    return param_count

answers = dict(np.load('gan-checks-tf.npz'))

```

## 1.2 Dataset

GANs are notoriously finicky with hyperparameters, and also require many training epochs. In order to make this assignment approachable without a GPU, we will be working on the MNIST

dataset, which is 60,000 training and 10,000 test images. Each picture contains a centered image of white digit on black background (0 through 9). This was one of the first datasets used to train convolutional neural networks and it is fairly easy – a standard CNN model can easily exceed 99% accuracy.

To simplify our code here, we will use the PyTorch MNIST wrapper, which downloads and loads the MNIST dataset. See the [documentation](#) for more information about the interface. The default parameters will take 5,000 of the training examples and place them into a validation dataset. The data will be saved into a folder called MNIST\_data.

```
[3]: class ChunkSampler(sampler.Sampler):
    """Samples elements sequentially from some offset.
    Arguments:
        num_samples: # of desired datapoints
        start: offset where we should start selecting from
    """
    def __init__(self, num_samples, start=0):
        self.num_samples = num_samples
        self.start = start

    def __iter__(self):
        return iter(range(self.start, self.start + self.num_samples))

    def __len__(self):
        return self.num_samples

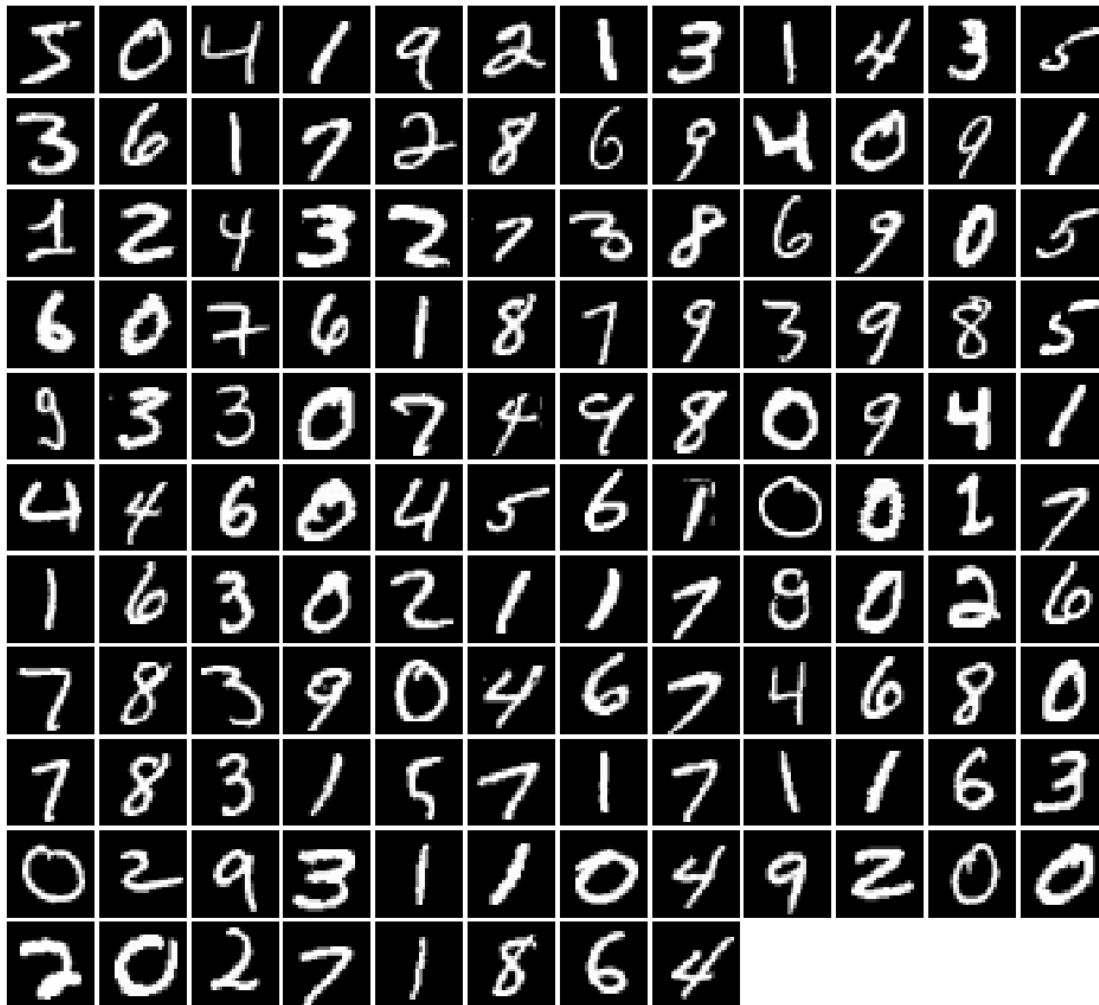
NUM_TRAIN = 50000
NUM_VAL = 5000

NOISE_DIM = 96
batch_size = 128

mnist_train = dset.MNIST('./cs6353/datasets/MNIST_data', train=True,
    ↪download=True,
                           transform=T.ToTensor())
loader_train = DataLoader(mnist_train, batch_size=batch_size,
                           sampler=ChunkSampler(NUM_TRAIN, 0))

mnist_val = dset.MNIST('./cs6353/datasets/MNIST_data', train=True,
    ↪download=True,
                           transform=T.ToTensor())
loader_val = DataLoader(mnist_val, batch_size=batch_size,
                           sampler=ChunkSampler(NUM_VAL, NUM_TRAIN))

imgs = next(loader_train.__iter__())[0].view(batch_size, 784).numpy().squeeze()
show_images(imgs)
```



### 1.3 Random Noise

Generate uniform noise from -1 to 1 with shape [batch\_size, dim].

Hint: use `torch.rand`.

```
[4]: def sample_noise(batch_size, dim):
      """
      Generate a PyTorch Tensor of uniform random noise.

      Input:
      - batch_size: Integer giving the batch size of noise to generate.
      - dim: Integer giving the dimension of noise to generate.

      Output:
      - A PyTorch Tensor of shape (batch_size, dim) containing uniform
```

```

        random noise in the range (-1, 1).
    """
    #Add your code here
    return torch.rand(batch_size, dim) * 2 - 1

```

Make sure noise is the correct shape and type:

```

[5]: def test_sample_noise():
    batch_size = 3
    dim = 4
    torch.manual_seed(231)
    z = sample_noise(batch_size, dim)
    np_z = z.cpu().numpy()
    assert np_z.shape == (batch_size, dim)
    assert torch.is_tensor(z)
    assert np.all(np_z >= -1.0) and np.all(np_z <= 1.0)
    assert np.any(np_z < 0.0) and np.any(np_z > 0.0)
    print('All tests passed!')

test_sample_noise()

```

All tests passed!

## 1.4 Flatten

Recall our Flatten operation from previous notebooks... this time we also provide an Unflatten, which you might want to use when implementing the convolutional generator. We also provide a weight initializer (and call it for you) that uses Xavier initialization instead of PyTorch's uniform default.

```

[18]: class Flatten(nn.Module):
    def forward(self, x):
        N, C, H, W = x.size() # read in N, C, H, W
        return x.view(N, -1) # "flatten" the C * H * W values into a single
        ↪ vector per image

class Unflatten(nn.Module):
    """
    An Unflatten module receives an input of shape (N, C*H*W) and reshapes it
    to produce an output of shape (N, C, H, W).
    """
    def __init__(self, N=-1, C=128, H=7, W=7):
        super(Unflatten, self).__init__()
        self.N = N
        self.C = C
        self.H = H
        self.W = W
    def forward(self, x):

```



```

        return x.view(self.N, self.C, self.H, self.W)

def initialize_weights(m):
    if isinstance(m, nn.Linear) or isinstance(m, nn.ConvTranspose2d):
        init.xavier_uniform_(m.weight.data)

```

## 1.5 CPU / GPU

By default all code will run on CPU. GPUs are not needed for this assignment, but will help you to train your models faster. If you do want to run the code on a GPU, then change the `dtype` variable in the following cell.

```

[17]: dtype = torch.FloatTensor
      #dtype = torch.cuda.FloatTensor ## UNCOMMENT THIS LINE IF YOU'RE ON A GPU!

```

## 2 Discriminator

Our first step is to build a discriminator. Fill in the architecture as part of the `nn.Sequential` constructor in the function below. All fully connected layers should include bias terms. The architecture is: \* Fully connected layer with input size 784 and output size 256 \* LeakyReLU with alpha 0.01 \* Fully connected layer with input size 256 and output size 256 \* LeakyReLU with alpha 0.01 \* Fully connected layer with input size 256 and output size 1

Recall that the Leaky ReLU nonlinearity computes  $f(x) = \max(\alpha x, x)$  for some fixed constant  $\alpha$ ; for the LeakyReLU nonlinearities in the architecture above we set  $\alpha = 0.01$ .

The output of the discriminator should have shape `[batch_size, 1]`, and contain real numbers corresponding to the scores that each of the `batch_size` inputs is a real image.

```

[30]: def discriminator():
      """
      Build and return a PyTorch model implementing the architecture above.
      """
      model = nn.Sequential(
          nn.Flatten(),
          nn.Linear(784, 256, bias=True),      # Fully connected layer input size 784
          # and output size 256
          nn.LeakyReLU(negative_slope=0.01),  # LeakyReLU with alpha 0.01
          nn.Linear(256, 256, bias=True),      # Fully connected layer input size 256
          # and output size 256
          nn.LeakyReLU(negative_slope=0.01),  # LeakyRelu again
          nn.Linear(256, 1, bias=True)
      )
      return model

```

Test to make sure the number of parameters in the discriminator is correct:

```
[31]: def test_discriminator(true_count=267009):
    model = discriminator()
    cur_count = count_params(model)
    if cur_count != true_count:
        print('Incorrect number of parameters in discriminator. Check your_
↪achitecture.')
    else:
        print('Correct number of parameters in discriminator.')

test_discriminator()
```

Correct number of parameters in discriminator.

### 3 Generator

Now to build the generator network: \* Fully connected layer from noise\_dim to 1024 \* ReLU \* Fully connected layer with size 1024 \* ReLU \* Fully connected layer with size 784 \* TanH (to clip the image to be in the range of [-1,1])

```
[24]: def generator(noise_dim=NOISE_DIM):
    """
    Build and return a PyTorch model implementing the architecture above.
    """
    model = nn.Sequential(
        nn.Linear(noise_dim, 1024), # Fully connected layer from noise_dim to_
↪1024
        nn.ReLU(inplace=True),
        nn.Linear(1024, 1024),      # Fully connected layer with size 1024
        nn.ReLU(inplace=True),
        nn.Linear(1024, 784),      # Fully connected layer with size 784
        nn.Tanh()
    )
    return model
```

Test to make sure the number of parameters in the generator is correct:

```
[25]: def test_generator(true_count=1858320):
    model = generator(4)
    cur_count = count_params(model)
    if cur_count != true_count:
        print('Incorrect number of parameters in generator. Check your_
↪achitecture.')
    else:
        print('Correct number of parameters in generator.')

test_generator()
```

Correct number of parameters in generator.

## 4 GAN Loss

Compute the generator and discriminator loss. The generator loss is:

$$\ell_G = -\mathbb{E}_{z \sim p(z)} [\log D(G(z))]$$

and the discriminator loss is:

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

Note that these are negated from the equations presented earlier as we will be *minimizing* these losses.

**HINTS:** You should use the `bce_loss` function defined below to compute the binary cross entropy loss which is needed to compute the log probability of the true label given the logits output from the discriminator. Given a score  $s \in \mathbb{R}$  and a label  $y \in \{0, 1\}$ , the binary cross entropy loss is

$$bce(s, y) = -y * \log(s) - (1 - y) * \log(1 - s)$$

A naive implementation of this formula can be numerically unstable, so we have provided a numerically stable implementation for you below.

You will also need to compute labels corresponding to real or fake and use the logit arguments to determine their size. Make sure you cast these labels to the correct data type using the global `dtype` variable, for example:

```
true_labels = torch.ones(size).type(dtype)
```

Instead of computing the expectation of  $\log D(G(z))$ ,  $\log D(x)$  and  $\log (1 - D(G(z)))$ , we will be averaging over elements of the minibatch, so make sure to combine the loss by averaging instead of summing.

```
[14]: def bce_loss(input, target):
    """
    Numerically stable version of the binary cross-entropy loss function.

    As per https://github.com/pytorch/pytorch/issues/751

    Inputs:
    - input: PyTorch Tensor of shape (N, ) giving scores.
    - target: PyTorch Tensor of shape (N,) containing 0 and 1 giving targets.

    Returns:
    - A PyTorch Tensor containing the mean BCE loss over the minibatch of input_
    ↪ data.
    """
    neg_abs = - input.abs()
    loss = input.clamp(min=0) - input * target + (1 + neg_abs.exp()).log()
```

```
return loss.mean()
```

```
[15]: def discriminator_loss(logits_real, logits_fake):  
    """  
    Computes the discriminator loss described above.  
  
    Inputs:  
    - logits_real: PyTorch Tensor of shape (N,) giving scores for the real data.  
    - logits_fake: PyTorch Tensor of shape (N,) giving scores for the fake data.  
  
    Returns:  
    - loss: PyTorch Tensor containing (scalar) the loss for the discriminator.  
    """  
    #loss = None  
  
    batch_size = logits_real.shape[0]  
    real_targets = torch.ones_like(logits_real, device=logits_real.device,   
    ↪dtype=logits_real.dtype)  
    fake_targets = torch.zeros_like(logits_fake, device=logits_fake.device,   
    ↪dtype=logits_fake.dtype)  
  
    real_loss = bce_loss(logits_real, real_targets)  
    fake_loss = bce_loss(logits_fake, fake_targets)  
  
    loss = real_loss + fake_loss  
    return loss  
  
def generator_loss(logits_fake):  
    """  
    Computes the generator loss described above.  
  
    Inputs:  
    - logits_fake: PyTorch Tensor of shape (N,) giving scores for the fake data.  
  
    Returns:  
    - loss: PyTorch Tensor containing the (scalar) loss for the generator.  
    """  
    loss = None  
  
    batch_size = logits_fake.shape[0]  
    target_labels = torch.ones_like(logits_fake, device=logits_fake.device,   
    ↪dtype=logits_fake.dtype)  
  
    loss = bce_loss(logits_fake, target_labels)  
  
    return loss
```

Test your generator and discriminator loss. You should see errors  $< 1e-7$ .

```
[19]: def test_discriminator_loss(logits_real, logits_fake, d_loss_true):
        d_loss = discriminator_loss(torch.Tensor(logits_real).type(dtype),
                                     torch.Tensor(logits_fake).type(dtype)).cpu().
        ↪numpy()
        print("Maximum error in d_loss: %g"%rel_error(d_loss_true, d_loss))

test_discriminator_loss(answers['logits_real'], answers['logits_fake'],
                        answers['d_loss_true'])
```

Maximum error in d\_loss: 2.83811e-08

```
[26]: def test_generator_loss(logits_fake, g_loss_true):
        g_loss = generator_loss(torch.Tensor(logits_fake).type(dtype)).cpu().numpy()
        print("Maximum error in g_loss: %g"%rel_error(g_loss_true, g_loss))

test_generator_loss(answers['logits_fake'], answers['g_loss_true'])
```

Maximum error in g\_loss: 4.4518e-09

## 5 Optimizing our loss

Make a function that returns an `optim.Adam` optimizer for the given model with a  $1e-3$  learning rate,  $\beta_1=0.5$ ,  $\beta_2=0.999$ . You'll use this to construct optimizers for the generators and discriminators for the rest of the notebook.

```
[27]: def get_optimizer(model):
        """
        Construct and return an Adam optimizer for the model with learning rate_
        ↪1e-3,
        beta1=0.5, and beta2=0.999.

        Input:
        - model: A PyTorch model that we want to optimize.

        Returns:
        - An Adam optimizer for the model with the desired hyperparameters.
        """
        #optimizer = None
        #Add your code here
        optimizer = torch.optim.Adam(
            model.parameters(),
            lr=1e-3,
            betas=(0.5, 0.999)
        )
        return optimizer
```

## 6 Training a GAN!

We provide you the main training loop... you won't need to change this function, but we encourage you to read through and understand it.

```
[22]: def run_a_gan(D, G, D_solver, G_solver, discriminator_loss, generator_loss,
    ↪ show_every=250,
        batch_size=128, noise_size=96, num_epochs=10):
    """
    Train a GAN!

    Inputs:
    - D, G: PyTorch models for the discriminator and generator
    - D_solver, G_solver: torch.optim Optimizers to use for training the
      discriminator and generator.
    - discriminator_loss, generator_loss: Functions to use for computing the
    ↪ generator and
      discriminator loss, respectively.
    - show_every: Show samples after every show_every iterations.
    - batch_size: Batch size to use for training.
    - noise_size: Dimension of the noise to use as input to the generator.
    - num_epochs: Number of epochs over the training dataset to use for
    ↪ training.
    """
    iter_count = 0
    for epoch in range(num_epochs):
        for x, _ in loader_train:
            if len(x) != batch_size:
                continue
            D_solver.zero_grad()
            real_data = x.type(dtype)
            logits_real = D(2* (real_data - 0.5)).type(dtype)
            g_fake_seed = sample_noise(batch_size, noise_size).type(dtype)
            fake_images = G(g_fake_seed).detach()
            logits_fake = D(fake_images.view(batch_size, 1, 28, 28))

            d_total_error = discriminator_loss(logits_real, logits_fake)
            d_total_error.backward()
            D_solver.step()

            G_solver.zero_grad()
            g_fake_seed = sample_noise(batch_size, noise_size).type(dtype)
            fake_images = G(g_fake_seed)

            gen_logits_fake = D(fake_images.view(batch_size, 1, 28, 28))
            g_error = generator_loss(gen_logits_fake)
            g_error.backward()
```

```

G_solver.step()

if (iter_count % show_every == 0):
    print('Iter: {}, D: {:.4}, G:{:.4}'.
    ↪format(iter_count,d_total_error.item(),g_error.item()))
    imgs_numpy = fake_images.data.cpu().numpy()
    show_images(imgs_numpy[0:16])
    plt.show()
    print()
    iter_count += 1

```

```

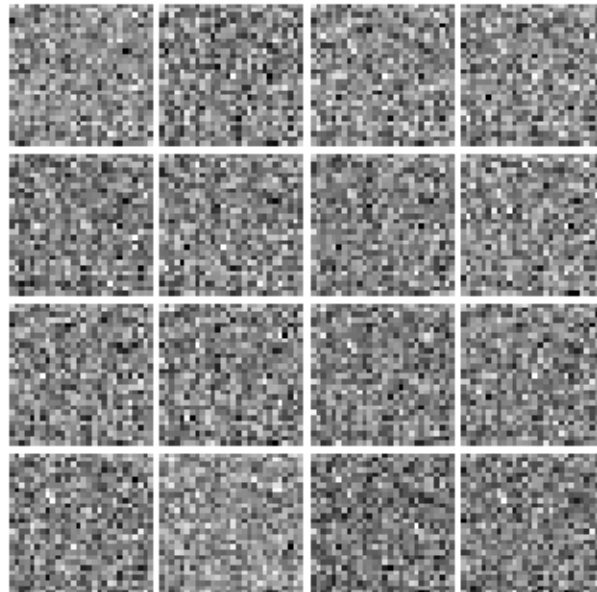
[32]: # Make the discriminator
D = discriminator().type(dtype)

# Make the generator
G = generator().type(dtype)

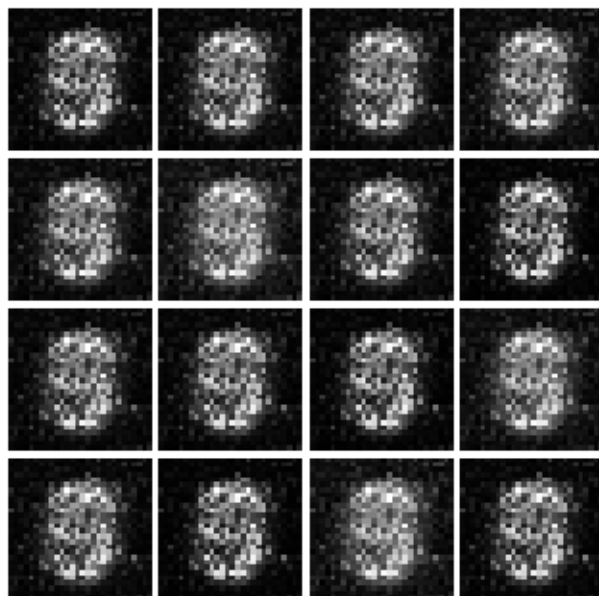
# Use the function you wrote earlier to get optimizers for the Discriminator
↪and the Generator
D_solver = get_optimizer(D)
G_solver = get_optimizer(G)
# Run it!
run_a_gan(D, G, D_solver, G_solver, discriminator_loss, generator_loss)

```

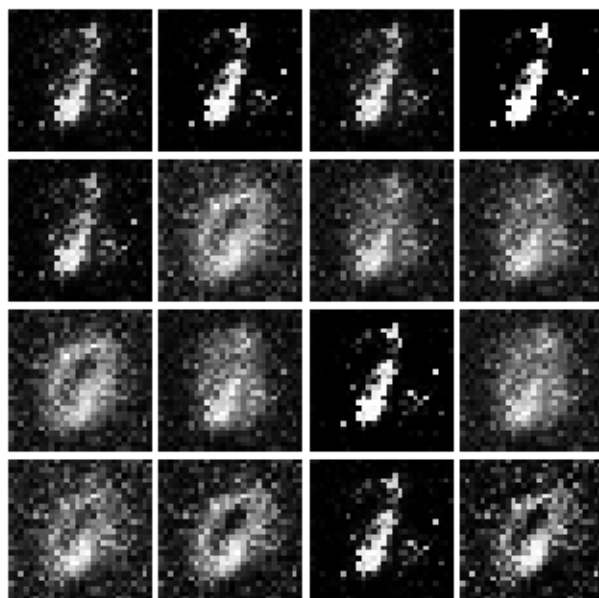
Iter: 0, D: 1.446, G:0.7261



Iter: 250, D: 1.968, G:0.6824

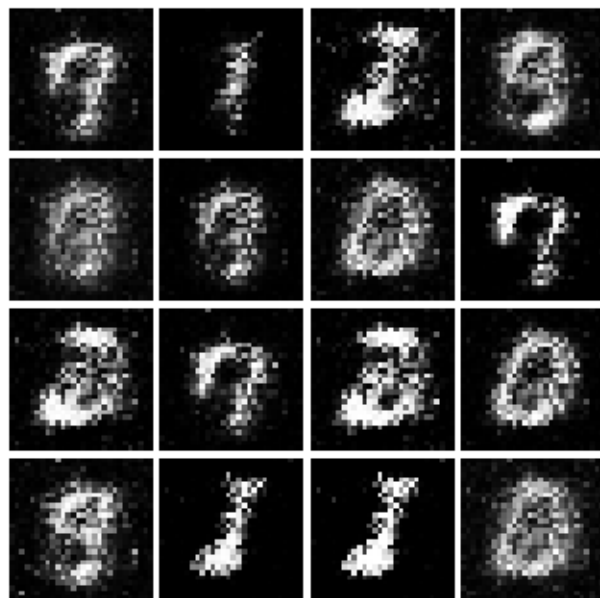


Iter: 500, D: 0.8349, G:1.963

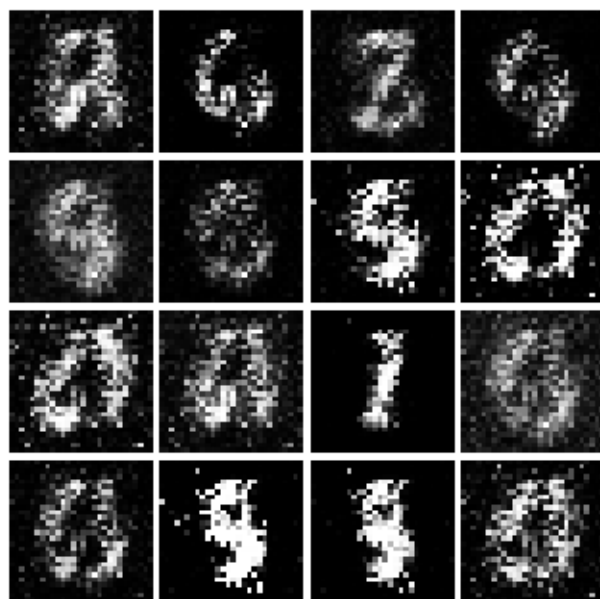


Iter: 750, D: 1.233, G:0.8651

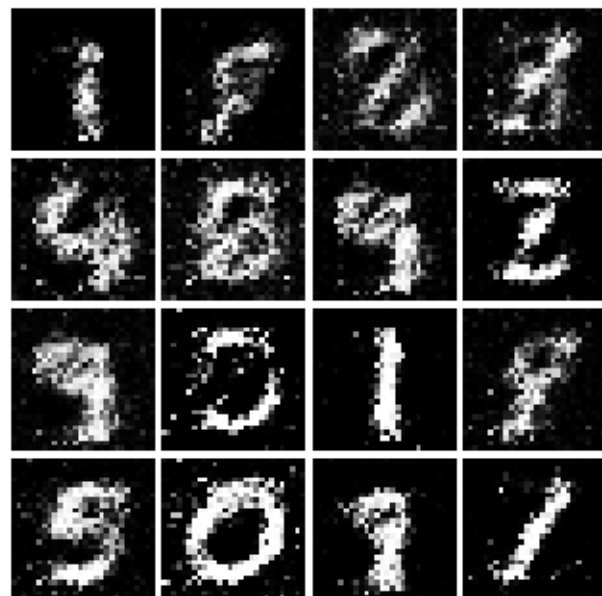




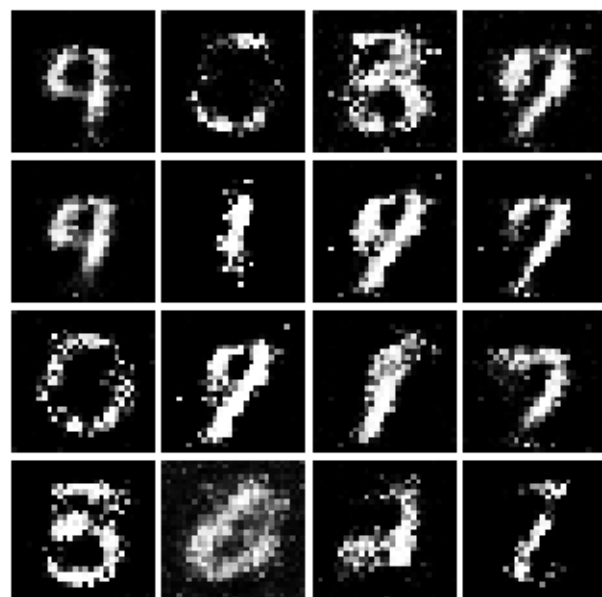
Iter: 1000, D: 1.181, G:1.043



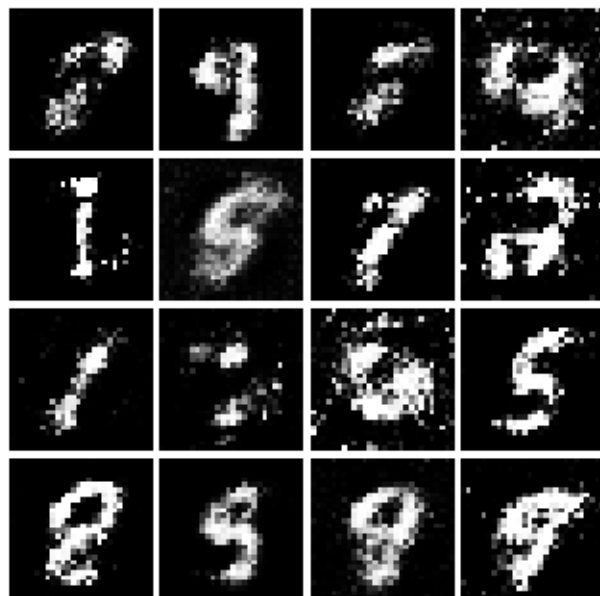
Iter: 1250, D: 1.129, G:1.125



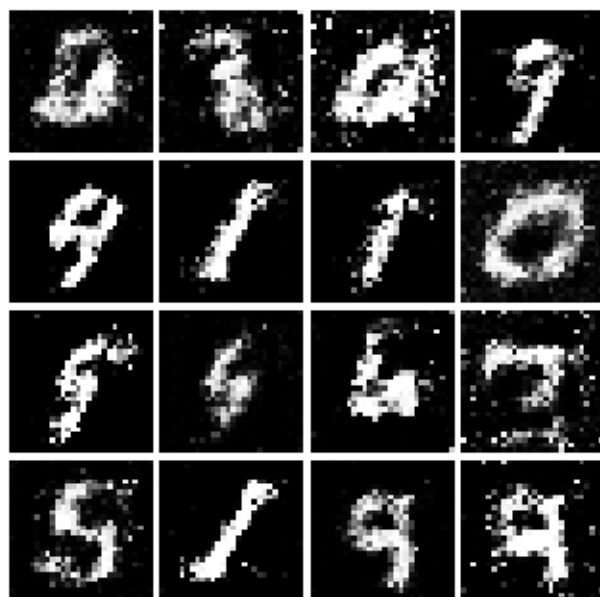
Iter: 1500, D: 1.257, G:0.9679



Iter: 1750, D: 1.268, G:0.8794



Iter: 2000, D: 1.33, G:1.065



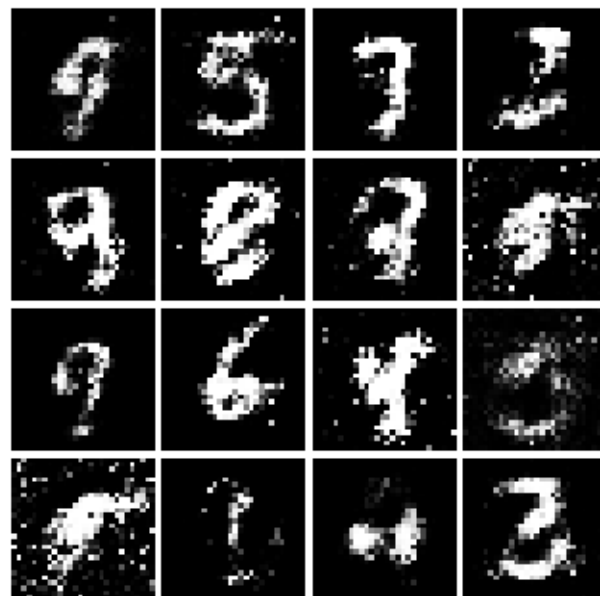
Iter: 2250, D: 1.316, G:0.8799



Iter: 2500, D: 1.391, G:0.9014



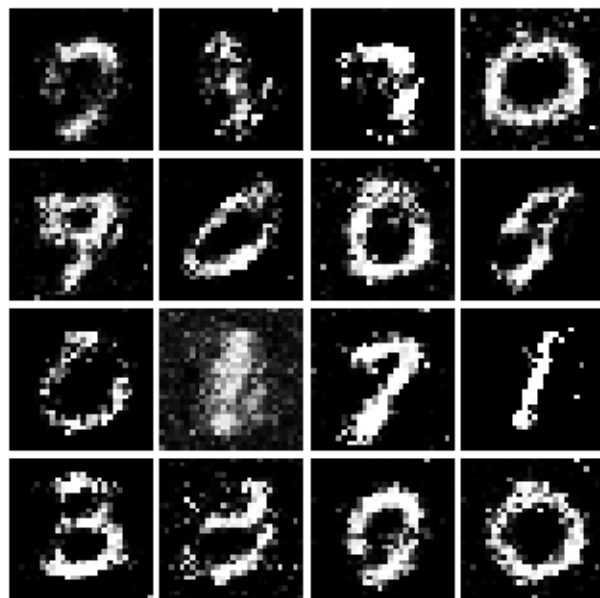
Iter: 2750, D: 1.344, G:0.7988



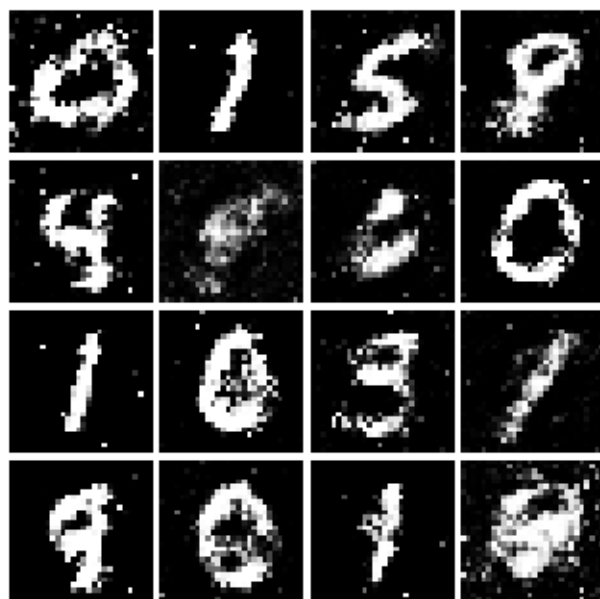
Iter: 3000, D: 1.359, G:0.7506



Iter: 3250, D: 1.353, G:0.8317



Iter: 3500, D: 1.297, G:0.838



Iter: 3750, D: 1.367, G:0.8564



Well that wasn't so hard, was it? In the iterations in the low 100s you should see black backgrounds, fuzzy shapes as you approach iteration 1000, and decent shapes, about half of which will be sharp and clearly recognizable as we pass 3000.

## 7 Least Squares GAN

We'll now look at [Least Squares GAN](#), a newer, more stable alternative to the original GAN loss function. For this part, all we have to do is change the loss function and retrain the model. We'll implement equation (9) in the paper, with the generator loss:

$$\ell_G = \frac{1}{2} \mathbb{E}_{z \sim p(z)} [(D(G(z)) - 1)^2]$$

and the discriminator loss:

$$\ell_D = \frac{1}{2} \mathbb{E}_{x \sim p_{\text{data}}} [(D(x) - 1)^2] + \frac{1}{2} \mathbb{E}_{z \sim p(z)} [(D(G(z)))^2]$$

**HINTS:** Instead of computing the expectation, we will be averaging over elements of the minibatch, so make sure to combine the loss by averaging instead of summing. When plugging in for  $D(x)$  and  $D(G(z))$  use the direct output from the discriminator (`scores_real` and `scores_fake`).

```
[33]: def ls_discriminator_loss(scores_real, scores_fake):
      """
      Compute the Least-Squares GAN loss for the discriminator.

      Inputs:
```

```

- scores_real: PyTorch Tensor of shape (N,) giving scores for the real data.
- scores_fake: PyTorch Tensor of shape (N,) giving scores for the fake data.

Outputs:
- loss: A PyTorch Tensor containing the loss.
"""

loss = None
#Add your code here
real_loss = 0.5 * torch.mean((scores_real - 1) ** 2)

fake_loss = 0.5 * torch.mean(scores_fake ** 2)

loss = real_loss + fake_loss
return loss

def ls_generator_loss(scores_fake):
    """
    Computes the Least-Squares GAN loss for the generator.

    Inputs:
    - scores_fake: PyTorch Tensor of shape (N,) giving scores for the fake data.

    Outputs:
    - loss: A PyTorch Tensor containing the loss.
    """
    loss = None
    #Add your code here
    loss = 0.5 * torch.mean((scores_fake - 1) ** 2)
    return loss

```

Before running a GAN with our new loss function, let's check it:

```

[34]: def test_lsgan_loss(score_real, score_fake, d_loss_true, g_loss_true):
    score_real = torch.Tensor(score_real).type(dtype)
    score_fake = torch.Tensor(score_fake).type(dtype)
    d_loss = ls_discriminator_loss(score_real, score_fake).cpu().numpy()
    g_loss = ls_generator_loss(score_fake).cpu().numpy()
    print("Maximum error in d_loss: %g"%rel_error(d_loss_true, d_loss))
    print("Maximum error in g_loss: %g"%rel_error(g_loss_true, g_loss))

test_lsgan_loss(answers['logits_real'], answers['logits_fake'],
                answers['d_loss_lsgan_true'], answers['g_loss_lsgan_true'])

```

Maximum error in d\_loss: 1.53171e-08

Maximum error in g\_loss: 3.36961e-08

Run the following cell to train your model!

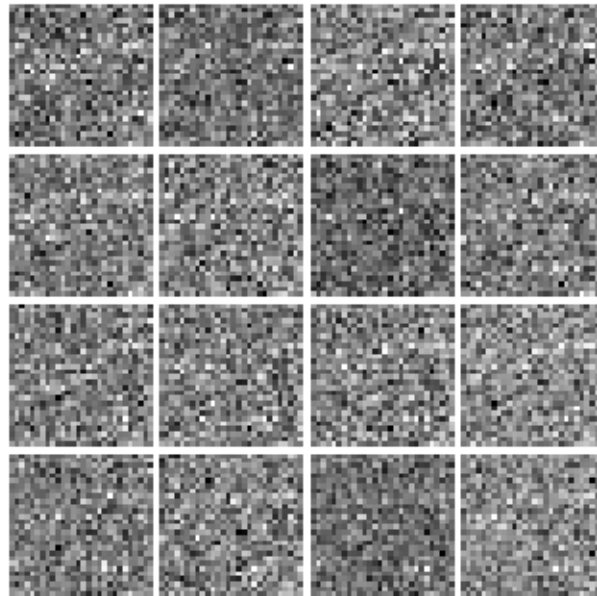


```
[35]: D_LS = discriminator().type(dtype)
      G_LS = generator().type(dtype)

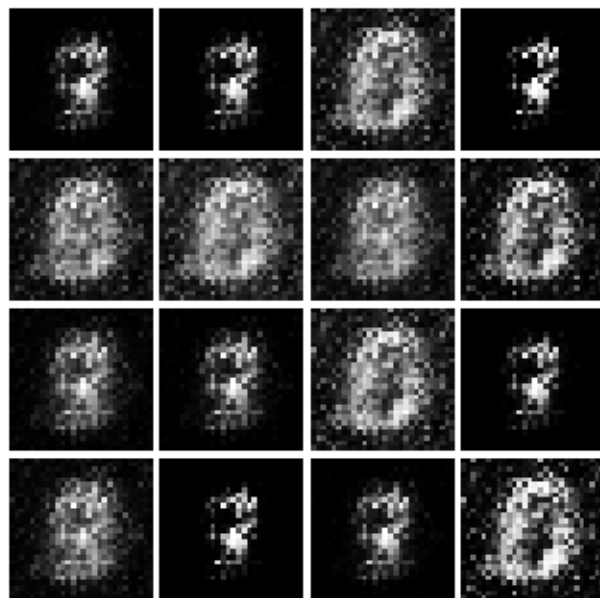
      D_LS_solver = get_optimizer(D_LS)
      G_LS_solver = get_optimizer(G_LS)

      run_a_gan(D_LS, G_LS, D_LS_solver, G_LS_solver, ls_discriminator_loss, ↵
               ↵ls_generator_loss)
```

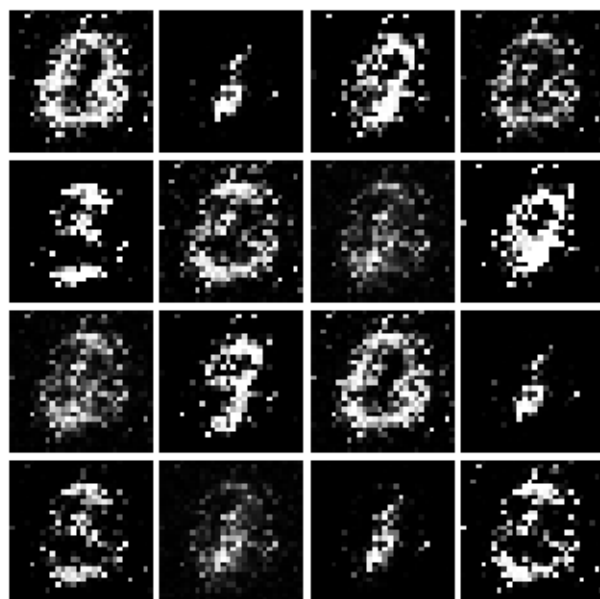
Iter: 0, D: 0.3926, G:0.453



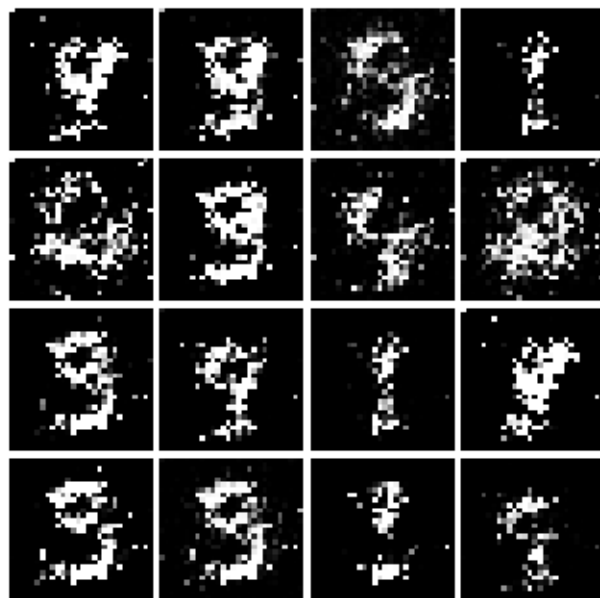
Iter: 250, D: 0.2045, G:0.1813



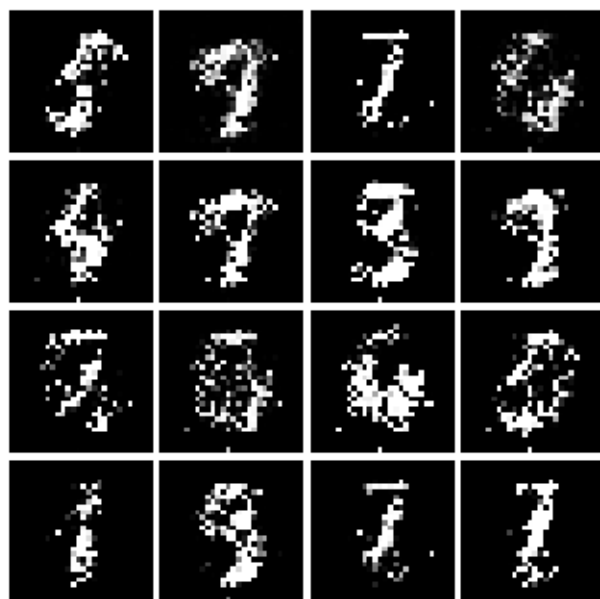
Iter: 500, D: 0.198, G:0.2658



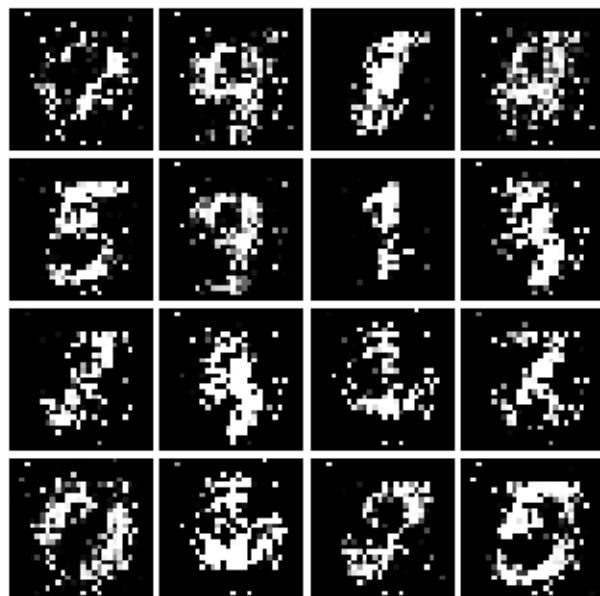
Iter: 750, D: 0.1268, G:0.3558



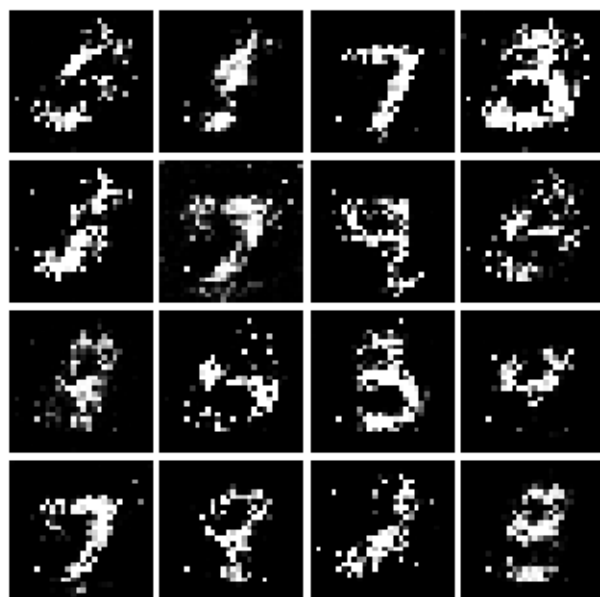
Iter: 1000, D: 0.145, G:0.2485



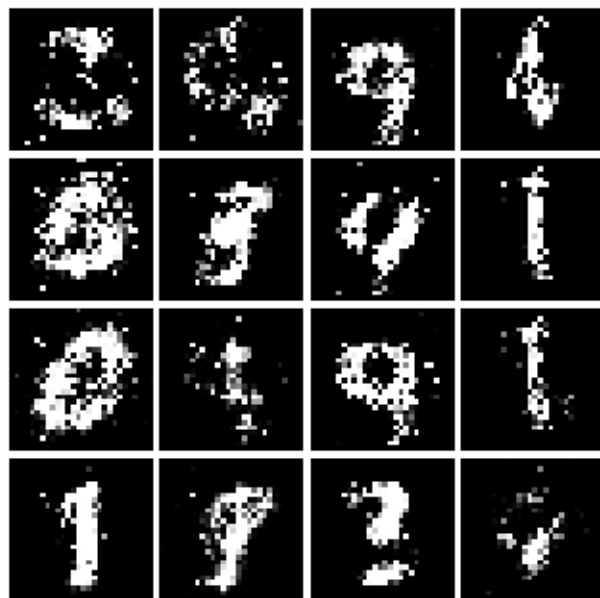
Iter: 1250, D: 0.1547, G:0.2411



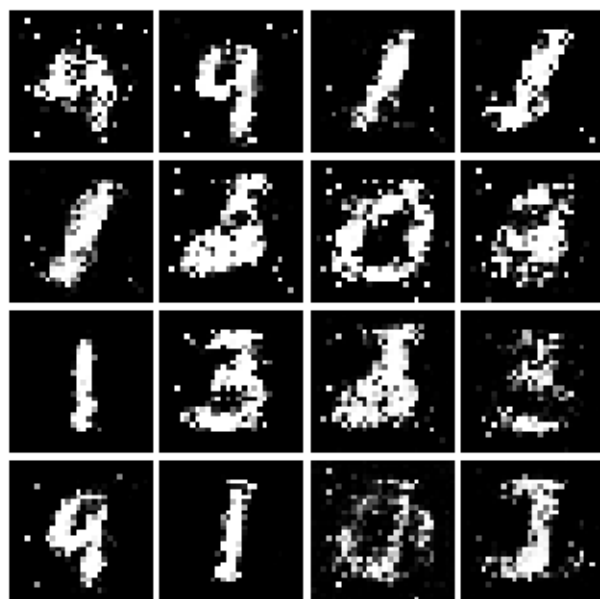
Iter: 1500, D: 0.1859, G:0.2574



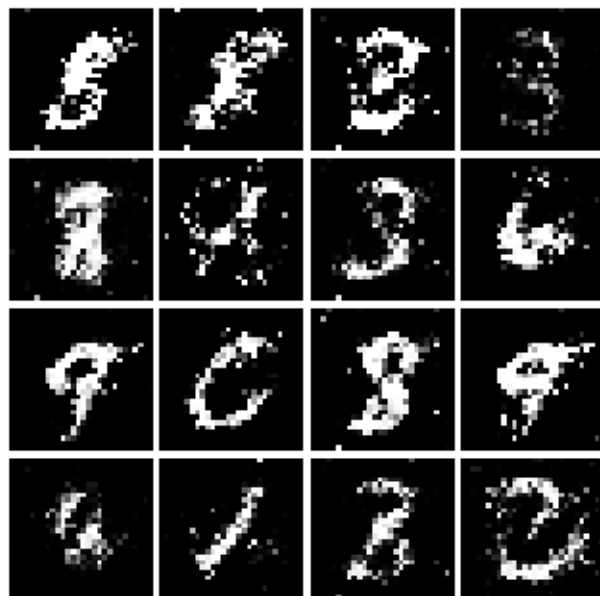
Iter: 1750, D: 0.1876, G:0.2109



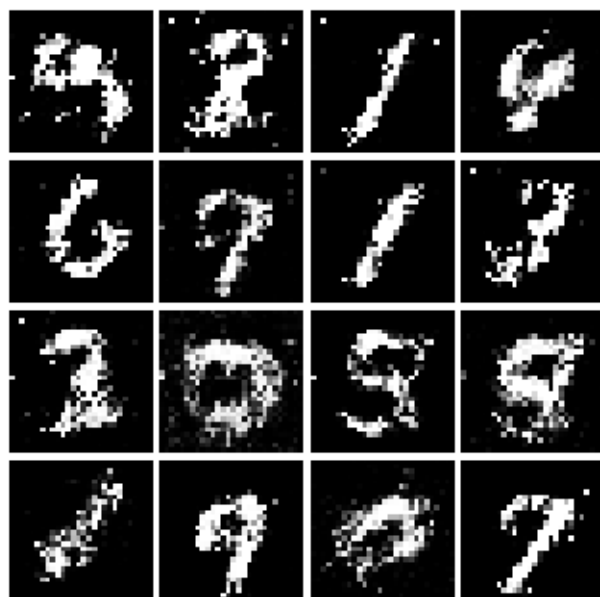
Iter: 2000, D: 0.2645, G:0.1884



Iter: 2250, D: 0.2364, G:0.1845



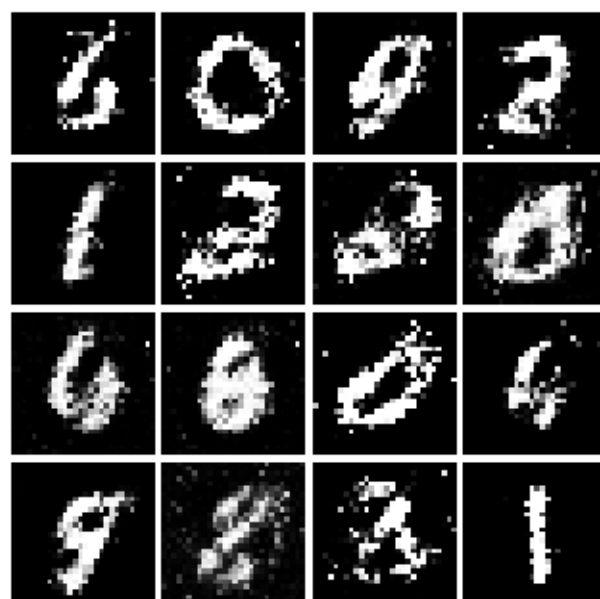
Iter: 2500, D: 0.219, G:0.1589



Iter: 2750, D: 0.2189, G:0.156



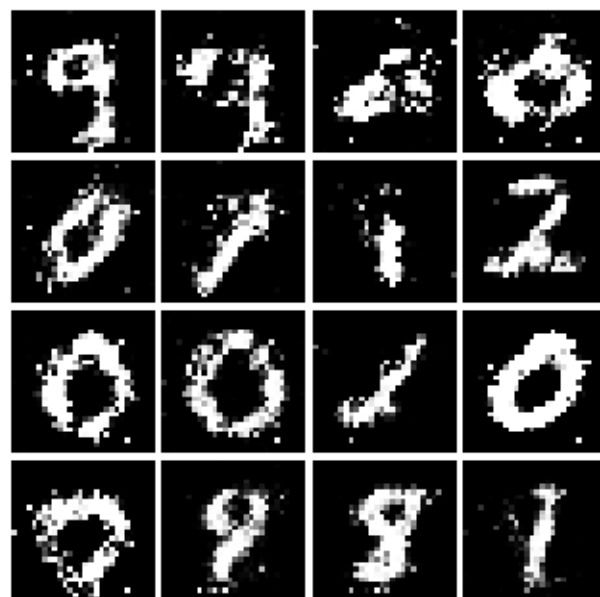
Iter: 3000, D: 0.2209, G:0.1562



Iter: 3250, D: 0.2277, G:0.189

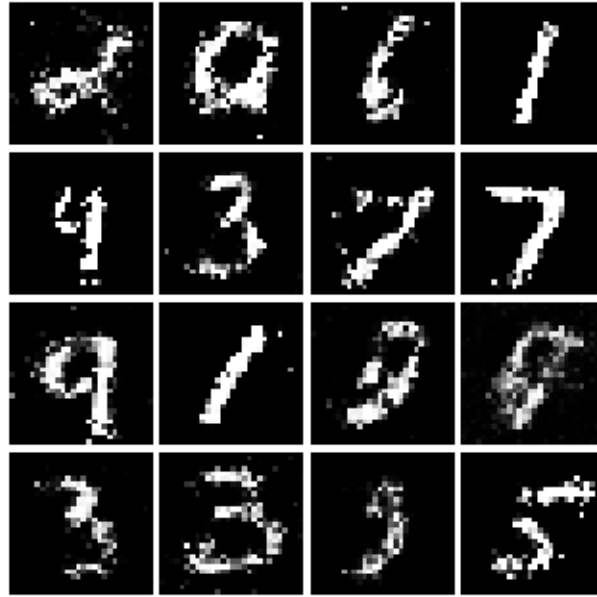


Iter: 3500, D: 0.2337, G:0.1757



Iter: 3750, D: 0.2065, G:0.1628





## 8 Deeply Convolutional GANs

In the first part of the notebook, we implemented an almost direct copy of the original GAN network from Ian Goodfellow. However, this network architecture allows no real spatial reasoning. It is unable to reason about things like “sharp edges” in general because it lacks any convolutional layers. Thus, in this section, we will implement some of the ideas from [DCGAN](#), where we use convolutional networks

### Discriminator

- Reshape into image tensor (Use Unflatten!)
- Conv2D: 32 Filters, 5x5, Stride 1
- Leaky ReLU(alpha=0.01)
- Max Pool 2x2, Stride 2
- Conv2D: 64 Filters, 5x5, Stride 1
- Leaky ReLU(alpha=0.01)
- Max Pool 2x2, Stride 2
- Flatten
- Fully Connected with output size 4 x 4 x 64
- Leaky ReLU(alpha=0.01)
- Fully Connected with output size 1

```
[36]: def build_dc_classifier():
      """
      Build and return a PyTorch model for the DCGAN discriminator implementing
```

```

the architecture above.
"""
return nn.Sequential(
    #####
    ##### TO DO #####
    #####
    Unflatten(N=-1, C=1, H=28, W=28), # Convert input to image tensor
    nn.Conv2d(1, 32, kernel_size=5, stride=1), # Conv layer
    nn.LeakyReLU(negative_slope=0.01), # Leaky ReLU
    nn.MaxPool2d(kernel_size=2, stride=2), # Max pooling
    nn.Conv2d(32, 64, kernel_size=5, stride=1), # Conv layer
    nn.LeakyReLU(negative_slope=0.01), # Leaky ReLU
    nn.MaxPool2d(kernel_size=2, stride=2), # Max pooling
    Flatten(), # Flatten the feature maps
    nn.Linear(64 * 4 * 4, 4 * 4 * 64), # Fully connected layer
    nn.LeakyReLU(negative_slope=0.01), # Leaky ReLU
    nn.Linear(4 * 4 * 64, 1) # Fully connected layer with output size 1
)

data = next(enumerate(loader_train))[-1][0].type(dtype)
b = build_dc_classifier().type(dtype)
out = b(data)
print(out.size())

```

```
torch.Size([128, 1])
```

Check the number of parameters in your classifier as a sanity check:

```

[37]: def test_dc_classifier(true_count=1102721):
    model = build_dc_classifier()
    cur_count = count_params(model)
    if cur_count != true_count:
        print('Incorrect number of parameters in generator. Check your_
architecture.')
    else:
        print('Correct number of parameters in generator.')

test_dc_classifier()

```

Correct number of parameters in generator.

**Generator** For the generator, we will copy the architecture exactly from the [InfoGAN paper](#). See Appendix C.1 MNIST. See the documentation for [tf.nn.conv2d\\_transpose](#). We are always “training” in GAN mode. \* Fully connected with output size 1024 \* ReLU \* BatchNorm \* Fully connected with output size 7 x 7 x 128 \* ReLU \* BatchNorm \* Reshape into Image Tensor of shape 7, 7, 128 \* Conv2D<sup>T</sup> (Transpose): 64 filters of 4x4, stride 2, ‘same’ padding \* ReLU \* BatchNorm \* Conv2D<sup>T</sup> (Transpose): 1 filter of 4x4, stride 2, ‘same’ padding \* TanH \* Should have a 28x28x1 image, reshape back into 784 vector

```
[38]: def build_dc_generator(noise_dim=NOISE_DIM):
    """
    Build and return a PyTorch model implementing the DCGAN generator using
    the architecture described above.
    """
    return nn.Sequential(
        #####
        ##### TO DO #####
        #####
        nn.Linear(noise_dim, 1024),
        nn.ReLU(True),
        nn.BatchNorm1d(1024),

        # Fully connected layer, output size 7 * 7 * 128
        nn.Linear(1024, 7 * 7 * 128),
        nn.ReLU(True),
        nn.BatchNorm1d(7 * 7 * 128),
        nn.Unflatten(1, (128, 7, 7)),
        nn.ConvTranspose2d(128, 64, kernel_size=4, stride=2, padding=1),
        nn.ReLU(True),
        nn.BatchNorm2d(64),
        nn.ConvTranspose2d(64, 1, kernel_size=4, stride=2, padding=1),
        nn.Tanh(),
        nn.Flatten()
    )

test_g_gan = build_dc_generator().type(dtype)
test_g_gan.apply(initialize_weights)

fake_seed = torch.randn(batch_size, NOISE_DIM).type(dtype)
fake_images = test_g_gan.forward(fake_seed)
fake_images.size()
```

```
[38]: torch.Size([128, 784])
```

Check the number of parameters in your generator as a sanity check:

```
[39]: def test_dc_generator(true_count=6580801):
    model = build_dc_generator(4)
    cur_count = count_params(model)
    if cur_count != true_count:
        print('Incorrect number of parameters in generator. Check your_
architecture.')
    else:
        print('Correct number of parameters in generator.')

test_dc_generator()
```

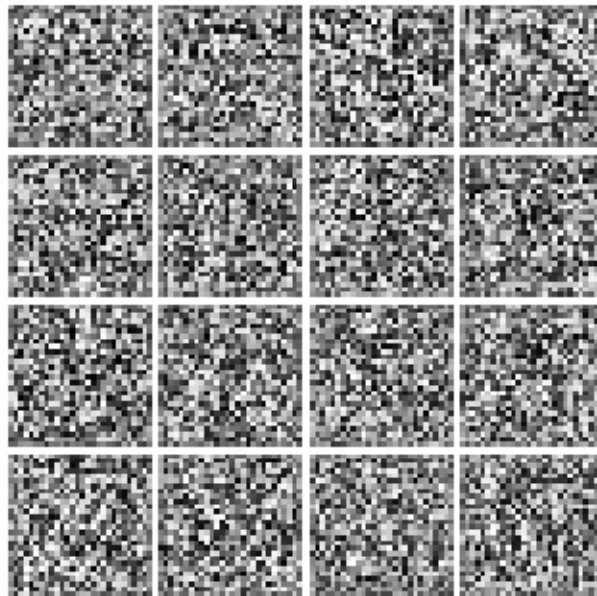
Correct number of parameters in generator.

```
[40]: D_DC = build_dc_classifier().type(dtype)
      D_DC.apply(initialize_weights)
      G_DC = build_dc_generator().type(dtype)
      G_DC.apply(initialize_weights)

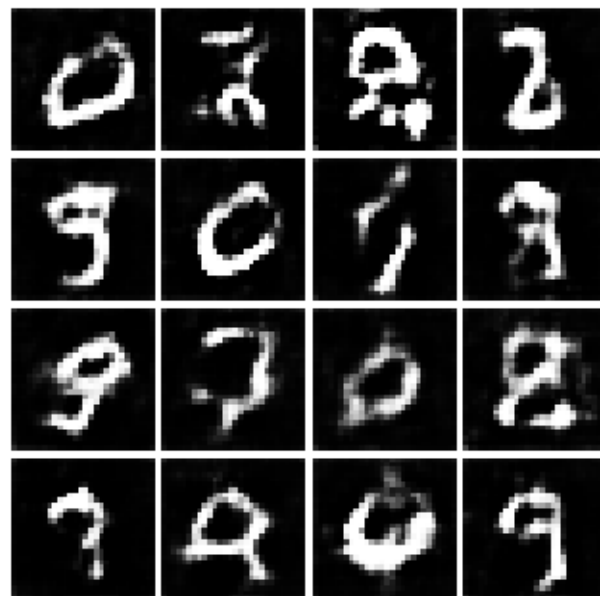
      D_DC_solver = get_optimizer(D_DC)
      G_DC_solver = get_optimizer(G_DC)

      run_a_gan(D_DC, G_DC, D_DC_solver, G_DC_solver, discriminator_loss, ↵
               ↵generator_loss, num_epochs=5)
```

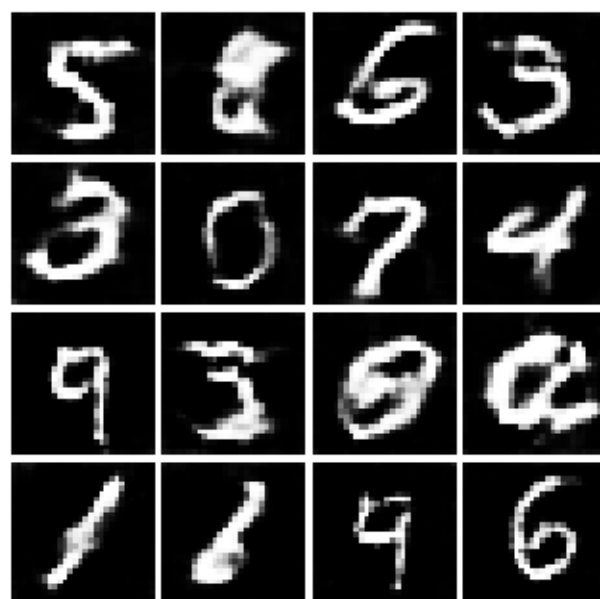
Iter: 0, D: 1.458, G:0.163



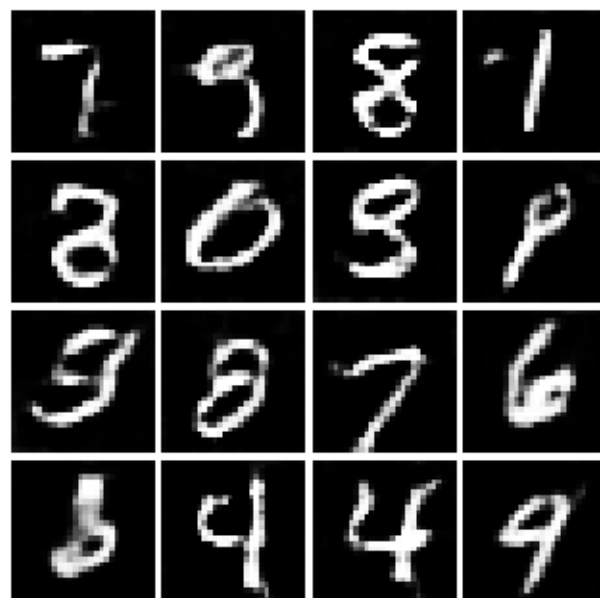
Iter: 250, D: 1.386, G:1.034



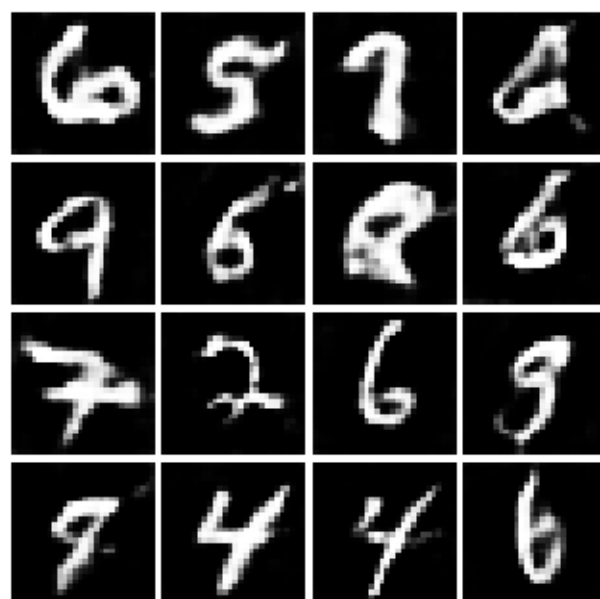
Iter: 500, D: 1.192, G:0.8889



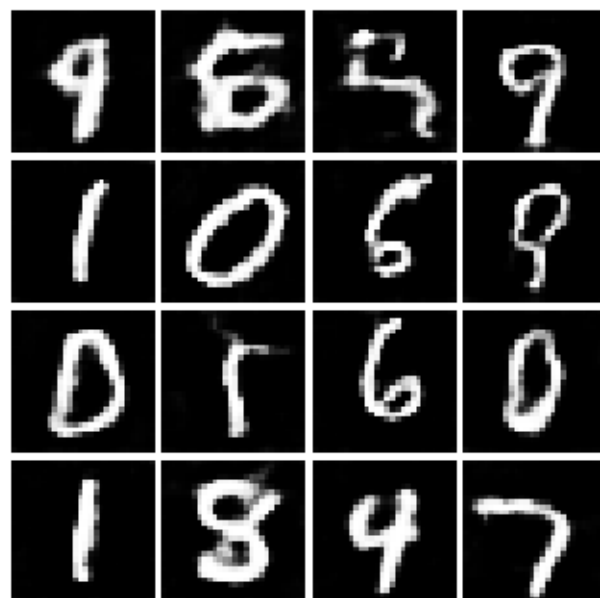
Iter: 750, D: 1.182, G:1.037



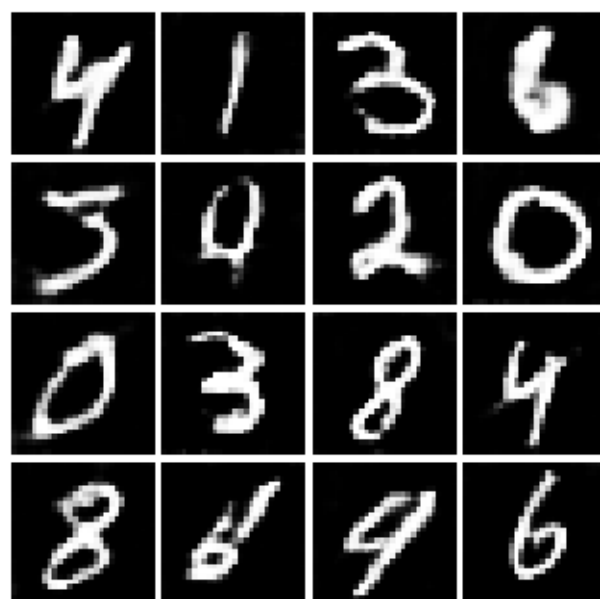
Iter: 1000, D: 1.264, G:0.8693



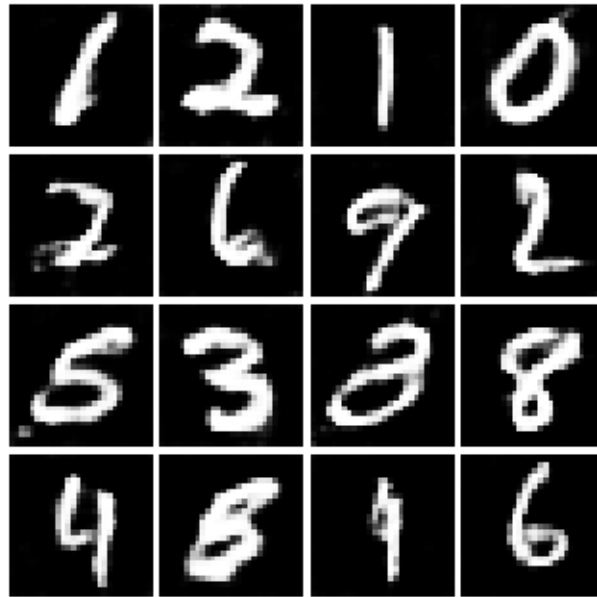
Iter: 1250, D: 1.267, G:1.002



Iter: 1500, D: 1.106, G:0.932



Iter: 1750, D: 1.221, G:0.7755



## 8.1 INLINE QUESTION 1

If the generator loss decreases during training while the discriminator loss stays at a constant high value from the start, is this a good sign? Why or why not? A qualitative answer is sufficient

### 8.1.1 Your answer:

It is not a good sign. If the discriminator loss stays constant and high, it means that it is not learning well. Both the generator and discriminator should improve together.

[42]: `!apt-get install texlive texlive-xetex texlive-latex-extra pandoc`

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  dvipng fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono fonts-
texgyre
  fonts-urw-base35 libapache-pom-java libcbmark-gfm-extensions0.29.0.gfm.3
libcbmark-gfm0.29.0.gfm.3
  libcommons-logging-java libcommons-parent-java libfontbox-java libfontenc1
libgs9 libgs9-common
  libidn12 libijs-0.35 libjbig2dec0 libkpathsea6 libpdfbox-java libptexenc1
libruby3.0 libsynchronex2
  libteckit0 libtexlua53 libtexluajit2 libwoff1 libzzip-0-13 lmodern pandoc-data
poppler-data
```



```

    preview-latex-style rake ruby ruby-net-telnet ruby-rubygems ruby-webrick ruby-
xmlrpc ruby3.0
    rubygems-integration tlutils teckit tex-common tex-gyre texlive-base texlive-
binaries
    texlive-fonts-recommended texlive-latex-base texlive-latex-recommended
texlive-pictures
    texlive-plain-generic tipa xfonts-encodings xfonts-utils
Suggested packages:
    fonts-noto fonts-freefont-otf | fonts-freefont-ttf libavalon-framework-java
    libcommons-logging-java-doc libexcalibur-logkit-java liblog4j1.2-java texlive-
luatex
    pandoc-citeproc context wkhtmltopdf librsvg2-bin groff ghc nodejs php python
libjs-mathjax
    libjs-katex citation-style-language-styles poppler-utils ghostscript fonts-
japanese-mincho
    | fonts-ipafont-mincho fonts-japanese-gothic | fonts-ipafont-gothic fonts-
arphic-ukai
    fonts-arphic-uming fonts-nanum ri ruby-dev bundler debhelper gv | postscript-
viewer perl-tk xpdf
    | pdf-viewer xzdec texlive-fonts-recommended-doc texlive-latex-base-doc
python3-pygments
    icc-profiles libfile-which-perl libspreadsheet-parseexcel-perl texlive-latex-
extra-doc
    texlive-latex-recommended-doc texlive-pstricks dot2tex prerex texlive-
pictures-doc vprerex
    default-jre-headless tipa-doc
The following NEW packages will be installed:
    dvisvgm fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono fonts-
texgyre
    fonts-urw-base35 libapache-pom-java libcmark-gfm-extensions0.29.0.gfm.3
libcmark-gfm0.29.0.gfm.3
    libcommons-logging-java libcommons-parent-java libfontbox-java libfontenc1
libgs9 libgs9-common
    libidn12 libijs-0.35 libjbig2dec0 libkpathsea6 libpdfbox-java libptexenc1
libruby3.0 libsynchronet2
    libteckit0 libtexlua53 libtexluaajit2 libwoff1 libzzip-0-13 lmodern pandoc
pandoc-data
    poppler-data preview-latex-style rake ruby ruby-net-telnet ruby-rubygems ruby-
webrick ruby-xmlrpc
    ruby3.0 rubygems-integration tlutils teckit tex-common tex-gyre texlive
texlive-base
    texlive-binaries texlive-fonts-recommended texlive-latex-base texlive-latex-
extra
    texlive-latex-recommended texlive-pictures texlive-plain-generic texlive-xetex
tipa
    xfonts-encodings xfonts-utils
0 upgraded, 59 newly installed, 0 to remove and 49 not upgraded.
Need to get 202 MB of archives.

```

After this operation, 728 MB of additional disk space will be used.

Get:1 <http://archive.ubuntu.com/ubuntu> jammy/main amd64 fonts-droid-fallback all 1:6.0.1r16-1.1build1 [1,805 kB]

Get:2 <http://archive.ubuntu.com/ubuntu> jammy/main amd64 fonts-lato all 2.0-2.1 [2,696 kB]

Get:3 <http://archive.ubuntu.com/ubuntu> jammy/main amd64 poppler-data all 0.4.11-1 [2,171 kB]

Get:4 <http://archive.ubuntu.com/ubuntu> jammy/universe amd64 tex-common all 6.17 [33.7 kB]

Get:5 <http://archive.ubuntu.com/ubuntu> jammy/main amd64 fonts-urw-base35 all 20200910-1 [6,367 kB]

Get:6 <http://archive.ubuntu.com/ubuntu> jammy-updates/main amd64 libgs9-common all 9.55.0~dfsg1-0ubuntu5.10 [752 kB]

Get:7 <http://archive.ubuntu.com/ubuntu> jammy-updates/main amd64 libidn12 amd64 1.38-4ubuntu1 [60.0 kB]

Get:8 <http://archive.ubuntu.com/ubuntu> jammy/main amd64 libijs-0.35 amd64 0.35-15build2 [16.5 kB]

Get:9 <http://archive.ubuntu.com/ubuntu> jammy/main amd64 libjbig2dec0 amd64 0.19-3build2 [64.7 kB]

Get:10 <http://archive.ubuntu.com/ubuntu> jammy-updates/main amd64 libgs9 amd64 9.55.0~dfsg1-0ubuntu5.10 [5,031 kB]

Get:11 <http://archive.ubuntu.com/ubuntu> jammy-updates/main amd64 libkpathsea6 amd64 2021.20210626.59705-1ubuntu0.2 [60.4 kB]

Get:12 <http://archive.ubuntu.com/ubuntu> jammy/main amd64 libwoff1 amd64 1.0.2-1build4 [45.2 kB]

Get:13 <http://archive.ubuntu.com/ubuntu> jammy/universe amd64 dvisvgm amd64 2.13.1-1 [1,221 kB]

Get:14 <http://archive.ubuntu.com/ubuntu> jammy/universe amd64 fonts-lmodern all 2.004.5-6.1 [4,532 kB]

Get:15 <http://archive.ubuntu.com/ubuntu> jammy/main amd64 fonts-noto-mono all 20201225-1build1 [397 kB]

Get:16 <http://archive.ubuntu.com/ubuntu> jammy/universe amd64 fonts-texgyre all 20180621-3.1 [10.2 MB]

Get:17 <http://archive.ubuntu.com/ubuntu> jammy/universe amd64 libapache-pom-java all 18-1 [4,720 B]

Get:18 <http://archive.ubuntu.com/ubuntu> jammy/universe amd64 libcmark-gfm0.29.0.gfm.3 amd64 0.29.0.gfm.3-3 [115 kB]

Get:19 <http://archive.ubuntu.com/ubuntu> jammy/universe amd64 libcmark-gfm-extensions0.29.0.gfm.3 amd64 0.29.0.gfm.3-3 [25.1 kB]

Get:20 <http://archive.ubuntu.com/ubuntu> jammy/universe amd64 libcommons-parent-java all 43-1 [10.8 kB]

Get:21 <http://archive.ubuntu.com/ubuntu> jammy/universe amd64 libcommons-logging-java all 1.2-2 [60.3 kB]

Get:22 <http://archive.ubuntu.com/ubuntu> jammy/main amd64 libfontenc1 amd64 1:1.1.4-1build3 [14.7 kB]

Get:23 <http://archive.ubuntu.com/ubuntu> jammy-updates/main amd64 libptexenc1 amd64 2021.20210626.59705-1ubuntu0.2 [39.1 kB]

Get:24 <http://archive.ubuntu.com/ubuntu> jammy/main amd64 rubygems-integration

all 1.18 [5,336 B]  
 Get:25 <http://archive.ubuntu.com/ubuntu> jammy-updates/main amd64 ruby3.0 amd64 3.0.2-7ubuntu2.8 [50.1 kB]  
 Get:26 <http://archive.ubuntu.com/ubuntu> jammy/main amd64 ruby-rubygems all 3.3.5-2 [228 kB]  
 Get:27 <http://archive.ubuntu.com/ubuntu> jammy/main amd64 ruby amd64 1:3.0~exp1 [5,100 B]  
 Get:28 <http://archive.ubuntu.com/ubuntu> jammy/main amd64 rake all 13.0.6-2 [61.7 kB]  
 Get:29 <http://archive.ubuntu.com/ubuntu> jammy/main amd64 ruby-net-telnet all 0.1.1-2 [12.6 kB]  
 Get:30 <http://archive.ubuntu.com/ubuntu> jammy-updates/main amd64 ruby-webrick all 1.7.0-3ubuntu0.1 [52.1 kB]  
 Get:31 <http://archive.ubuntu.com/ubuntu> jammy-updates/main amd64 ruby-xmlrpc all 0.3.2-1ubuntu0.1 [24.9 kB]  
 Get:32 <http://archive.ubuntu.com/ubuntu> jammy-updates/main amd64 libruby3.0 amd64 3.0.2-7ubuntu2.8 [5,113 kB]  
 Get:33 <http://archive.ubuntu.com/ubuntu> jammy-updates/main amd64 libsyntax2 amd64 2021.20210626.59705-1ubuntu0.2 [55.6 kB]  
 Get:34 <http://archive.ubuntu.com/ubuntu> jammy/universe amd64 libteckit0 amd64 2.5.11+ds1-1 [421 kB]  
 Get:35 <http://archive.ubuntu.com/ubuntu> jammy-updates/main amd64 libtexlua53 amd64 2021.20210626.59705-1ubuntu0.2 [120 kB]  
 Get:36 <http://archive.ubuntu.com/ubuntu> jammy-updates/main amd64 libtexluajit2 amd64 2021.20210626.59705-1ubuntu0.2 [267 kB]  
 Get:37 <http://archive.ubuntu.com/ubuntu> jammy/universe amd64 libzip-0-13 amd64 0.13.72+dfsg.1-1.1 [27.0 kB]  
 Get:38 <http://archive.ubuntu.com/ubuntu> jammy/main amd64 xfonts-encodings all 1:1.0.5-0ubuntu2 [578 kB]  
 Get:39 <http://archive.ubuntu.com/ubuntu> jammy/main amd64 xfonts-utils amd64 1:7.7+6build2 [94.6 kB]  
 Get:40 <http://archive.ubuntu.com/ubuntu> jammy/universe amd64 lmodern all 2.004.5-6.1 [9,471 kB]  
 Get:41 <http://archive.ubuntu.com/ubuntu> jammy/universe amd64 pandoc-data all 2.9.2.1-3ubuntu2 [81.8 kB]  
 Get:42 <http://archive.ubuntu.com/ubuntu> jammy/universe amd64 pandoc amd64 2.9.2.1-3ubuntu2 [20.3 MB]  
 Get:43 <http://archive.ubuntu.com/ubuntu> jammy/universe amd64 preview-latex-style all 12.2-1ubuntu1 [185 kB]  
 Get:44 <http://archive.ubuntu.com/ubuntu> jammy/main amd64 t1utils amd64 1.41-4build2 [61.3 kB]  
 Get:45 <http://archive.ubuntu.com/ubuntu> jammy/universe amd64 teckit amd64 2.5.11+ds1-1 [699 kB]  
 Get:46 <http://archive.ubuntu.com/ubuntu> jammy/universe amd64 tex-gyre all 20180621-3.1 [6,209 kB]  
 Get:47 <http://archive.ubuntu.com/ubuntu> jammy-updates/universe amd64 texlive-binaries amd64 2021.20210626.59705-1ubuntu0.2 [9,860 kB]  
 Get:48 <http://archive.ubuntu.com/ubuntu> jammy/universe amd64 texlive-base all

```

2021.20220204-1 [21.0 MB]
Get:49 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-fonts-
recommended all 2021.20220204-1 [4,972 kB]
Get:50 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-latex-base
all 2021.20220204-1 [1,128 kB]
Get:51 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-latex-
recommended all 2021.20220204-1 [14.4 MB]
Get:52 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive all
2021.20220204-1 [14.3 kB]
Get:53 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libfontbox-java all
1:1.8.16-2 [207 kB]
Get:54 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libpdfbox-java all
1:1.8.16-2 [5,199 kB]
Get:55 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-pictures
all 2021.20220204-1 [8,720 kB]
Get:56 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-latex-extra
all 2021.20220204-1 [13.9 MB]
Get:57 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-plain-
generic all 2021.20220204-1 [27.5 MB]
Get:58 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tipa all 2:1.3-21
[2,967 kB]
Get:59 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-xetex all
2021.20220204-1 [12.4 MB]
Fetched 202 MB in 8s (26.4 MB/s)
Extracting templates from packages: 100%
Preconfiguring packages ...
Selecting previously unselected package fonts-droid-fallback.
(Reading database ... 123630 files and directories currently installed.)
Preparing to unpack .../00-fonts-droid-fallback_1%3a6.0.1r16-1.1build1_all.deb
...
Unpacking fonts-droid-fallback (1:6.0.1r16-1.1build1) ...
Selecting previously unselected package fonts-lato.
Preparing to unpack .../01-fonts-lato_2.0-2.1_all.deb ...
Unpacking fonts-lato (2.0-2.1) ...
Selecting previously unselected package poppler-data.
Preparing to unpack .../02-poppler-data_0.4.11-1_all.deb ...
Unpacking poppler-data (0.4.11-1) ...
Selecting previously unselected package tex-common.
Preparing to unpack .../03-tex-common_6.17_all.deb ...
Unpacking tex-common (6.17) ...
Selecting previously unselected package fonts-urw-base35.
Preparing to unpack .../04-fonts-urw-base35_20200910-1_all.deb ...
Unpacking fonts-urw-base35 (20200910-1) ...
Selecting previously unselected package libgs9-common.
Preparing to unpack .../05-libgs9-common_9.55.0~dfsg1-0ubuntu5.10_all.deb ...
Unpacking libgs9-common (9.55.0~dfsg1-0ubuntu5.10) ...
Selecting previously unselected package libidn12:amd64.
Preparing to unpack .../06-libidn12_1.38-4ubuntu1_amd64.deb ...

```

```

Unpacking libidn12:amd64 (1.38-4ubuntu1) ...
Selecting previously unselected package libijs-0.35:amd64.
Preparing to unpack .../07-libijs-0.35_0.35-15build2_amd64.deb ...
Unpacking libijs-0.35:amd64 (0.35-15build2) ...
Selecting previously unselected package libjbig2dec0:amd64.
Preparing to unpack .../08-libjbig2dec0_0.19-3build2_amd64.deb ...
Unpacking libjbig2dec0:amd64 (0.19-3build2) ...
Selecting previously unselected package libgs9:amd64.
Preparing to unpack .../09-libgs9_9.55.0~dfsg1-0ubuntu5.10_amd64.deb ...
Unpacking libgs9:amd64 (9.55.0~dfsg1-0ubuntu5.10) ...
Selecting previously unselected package libkpathsea6:amd64.
Preparing to unpack .../10-libkpathsea6_2021.20210626.59705-1ubuntu0.2_amd64.deb
...
Unpacking libkpathsea6:amd64 (2021.20210626.59705-1ubuntu0.2) ...
Selecting previously unselected package libwoff1:amd64.
Preparing to unpack .../11-libwoff1_1.0.2-1build4_amd64.deb ...
Unpacking libwoff1:amd64 (1.0.2-1build4) ...
Selecting previously unselected package dvisvgm.
Preparing to unpack .../12-dvisvgm_2.13.1-1_amd64.deb ...
Unpacking dvisvgm (2.13.1-1) ...
Selecting previously unselected package fonts-lmodern.
Preparing to unpack .../13-fonts-lmodern_2.004.5-6.1_all.deb ...
Unpacking fonts-lmodern (2.004.5-6.1) ...
Selecting previously unselected package fonts-noto-mono.
Preparing to unpack .../14-fonts-noto-mono_20201225-1build1_all.deb ...
Unpacking fonts-noto-mono (20201225-1build1) ...
Selecting previously unselected package fonts-texgyre.
Preparing to unpack .../15-fonts-texgyre_20180621-3.1_all.deb ...
Unpacking fonts-texgyre (20180621-3.1) ...
Selecting previously unselected package libapache-pom-java.
Preparing to unpack .../16-libapache-pom-java_18-1_all.deb ...
Unpacking libapache-pom-java (18-1) ...
Selecting previously unselected package libcmark-gfm0.29.0.gfm.3:amd64.
Preparing to unpack .../17-libcmark-gfm0.29.0.gfm.3_0.29.0.gfm.3-3_amd64.deb ...
Unpacking libcmark-gfm0.29.0.gfm.3:amd64 (0.29.0.gfm.3-3) ...
Selecting previously unselected package libcmark-gfm-
extensions0.29.0.gfm.3:amd64.
Preparing to unpack .../18-libcmark-gfm-
extensions0.29.0.gfm.3_0.29.0.gfm.3-3_amd64.deb ...
Unpacking libcmark-gfm-extensions0.29.0.gfm.3:amd64 (0.29.0.gfm.3-3) ...
Selecting previously unselected package libcommons-parent-java.
Preparing to unpack .../19-libcommons-parent-java_43-1_all.deb ...
Unpacking libcommons-parent-java (43-1) ...
Selecting previously unselected package libcommons-logging-java.
Preparing to unpack .../20-libcommons-logging-java_1.2-2_all.deb ...
Unpacking libcommons-logging-java (1.2-2) ...
Selecting previously unselected package libfontenc1:amd64.
Preparing to unpack .../21-libfontenc1_1%3a1.1.4-1build3_amd64.deb ...

```

```

Unpacking libfontenc1:amd64 (1:1.1.4-1build3) ...
Selecting previously unselected package libptexenc1:amd64.
Preparing to unpack .../22-libptexenc1_2021.20210626.59705-1ubuntu0.2_amd64.deb
...
Unpacking libptexenc1:amd64 (2021.20210626.59705-1ubuntu0.2) ...
Selecting previously unselected package rubygems-integration.
Preparing to unpack .../23-rubygems-integration_1.18_all.deb ...
Unpacking rubygems-integration (1.18) ...
Selecting previously unselected package ruby3.0.
Preparing to unpack .../24-ruby3.0_3.0.2-7ubuntu2.8_amd64.deb ...
Unpacking ruby3.0 (3.0.2-7ubuntu2.8) ...
Selecting previously unselected package ruby-rubygems.
Preparing to unpack .../25-ruby-rubygems_3.3.5-2_all.deb ...
Unpacking ruby-rubygems (3.3.5-2) ...
Selecting previously unselected package ruby.
Preparing to unpack .../26-ruby_1%3a3.0~exp1_amd64.deb ...
Unpacking ruby (1:3.0~exp1) ...
Selecting previously unselected package rake.
Preparing to unpack .../27-rake_13.0.6-2_all.deb ...
Unpacking rake (13.0.6-2) ...
Selecting previously unselected package ruby-net-telnet.
Preparing to unpack .../28-ruby-net-telnet_0.1.1-2_all.deb ...
Unpacking ruby-net-telnet (0.1.1-2) ...
Selecting previously unselected package ruby-webrick.
Preparing to unpack .../29-ruby-webrick_1.7.0-3ubuntu0.1_all.deb ...
Unpacking ruby-webrick (1.7.0-3ubuntu0.1) ...
Selecting previously unselected package ruby-xmlrpc.
Preparing to unpack .../30-ruby-xmlrpc_0.3.2-1ubuntu0.1_all.deb ...
Unpacking ruby-xmlrpc (0.3.2-1ubuntu0.1) ...
Selecting previously unselected package libruby3.0:amd64.
Preparing to unpack .../31-libruby3.0_3.0.2-7ubuntu2.8_amd64.deb ...
Unpacking libruby3.0:amd64 (3.0.2-7ubuntu2.8) ...
Selecting previously unselected package libsyntax2:amd64.
Preparing to unpack .../32-libsyntax2_2021.20210626.59705-1ubuntu0.2_amd64.deb
...
Unpacking libsyntax2:amd64 (2021.20210626.59705-1ubuntu0.2) ...
Selecting previously unselected package libteckit0:amd64.
Preparing to unpack .../33-libteckit0_2.5.11+ds1-1_amd64.deb ...
Unpacking libteckit0:amd64 (2.5.11+ds1-1) ...
Selecting previously unselected package libtexlua53:amd64.
Preparing to unpack .../34-libtexlua53_2021.20210626.59705-1ubuntu0.2_amd64.deb
...
Unpacking libtexlua53:amd64 (2021.20210626.59705-1ubuntu0.2) ...
Selecting previously unselected package libtexluajit2:amd64.
Preparing to unpack
.../35-libtexluajit2_2021.20210626.59705-1ubuntu0.2_amd64.deb ...
Unpacking libtexluajit2:amd64 (2021.20210626.59705-1ubuntu0.2) ...
Selecting previously unselected package libzip-0-13:amd64.

```

```

Preparing to unpack .../36-libzip-0-13_0.13.72+dfsg.1-1.1_amd64.deb ...
Unpacking libzip-0-13:amd64 (0.13.72+dfsg.1-1.1) ...
Selecting previously unselected package xfonts-encodings.
Preparing to unpack .../37-xfonts-encodings_1%3a1.0.5-0ubuntu2_all.deb ...
Unpacking xfonts-encodings (1:1.0.5-0ubuntu2) ...
Selecting previously unselected package xfonts-utils.
Preparing to unpack .../38-xfonts-utils_1%3a7.7+6build2_amd64.deb ...
Unpacking xfonts-utils (1:7.7+6build2) ...
Selecting previously unselected package lmodern.
Preparing to unpack .../39-lmodern_2.004.5-6.1_all.deb ...
Unpacking lmodern (2.004.5-6.1) ...
Selecting previously unselected package pandoc-data.
Preparing to unpack .../40-pandoc-data_2.9.2.1-3ubuntu2_all.deb ...
Unpacking pandoc-data (2.9.2.1-3ubuntu2) ...
Selecting previously unselected package pandoc.
Preparing to unpack .../41-pandoc_2.9.2.1-3ubuntu2_amd64.deb ...
Unpacking pandoc (2.9.2.1-3ubuntu2) ...
Selecting previously unselected package preview-latex-style.
Preparing to unpack .../42-preview-latex-style_12.2-1ubuntu1_all.deb ...
Unpacking preview-latex-style (12.2-1ubuntu1) ...
Selecting previously unselected package t1utils.
Preparing to unpack .../43-t1utils_1.41-4build2_amd64.deb ...
Unpacking t1utils (1.41-4build2) ...
Selecting previously unselected package teckit.
Preparing to unpack .../44-teckit_2.5.11+ds1-1_amd64.deb ...
Unpacking teckit (2.5.11+ds1-1) ...
Selecting previously unselected package tex-gyre.
Preparing to unpack .../45-tex-gyre_20180621-3.1_all.deb ...
Unpacking tex-gyre (20180621-3.1) ...
Selecting previously unselected package texlive-binaries.
Preparing to unpack .../46-texlive-
binaries_2021.20210626.59705-1ubuntu0.2_amd64.deb ...
Unpacking texlive-binaries (2021.20210626.59705-1ubuntu0.2) ...
Selecting previously unselected package texlive-base.
Preparing to unpack .../47-texlive-base_2021.20220204-1_all.deb ...
Unpacking texlive-base (2021.20220204-1) ...
Selecting previously unselected package texlive-fonts-recommended.
Preparing to unpack .../48-texlive-fonts-recommended_2021.20220204-1_all.deb ...
Unpacking texlive-fonts-recommended (2021.20220204-1) ...
Selecting previously unselected package texlive-latex-base.
Preparing to unpack .../49-texlive-latex-base_2021.20220204-1_all.deb ...
Unpacking texlive-latex-base (2021.20220204-1) ...
Selecting previously unselected package texlive-latex-recommended.
Preparing to unpack .../50-texlive-latex-recommended_2021.20220204-1_all.deb ...
Unpacking texlive-latex-recommended (2021.20220204-1) ...
Selecting previously unselected package texlive.
Preparing to unpack .../51-texlive_2021.20220204-1_all.deb ...
Unpacking texlive (2021.20220204-1) ...

```

```

Selecting previously unselected package libfontbox-java.
Preparing to unpack .../52-libfontbox-java_1%3a1.8.16-2_all.deb ...
Unpacking libfontbox-java (1:1.8.16-2) ...
Selecting previously unselected package libpdfbox-java.
Preparing to unpack .../53-libpdfbox-java_1%3a1.8.16-2_all.deb ...
Unpacking libpdfbox-java (1:1.8.16-2) ...
Selecting previously unselected package texlive-pictures.
Preparing to unpack .../54-texlive-pictures_2021.20220204-1_all.deb ...
Unpacking texlive-pictures (2021.20220204-1) ...
Selecting previously unselected package texlive-latex-extra.
Preparing to unpack .../55-texlive-latex-extra_2021.20220204-1_all.deb ...
Unpacking texlive-latex-extra (2021.20220204-1) ...
Selecting previously unselected package texlive-plain-generic.
Preparing to unpack .../56-texlive-plain-generic_2021.20220204-1_all.deb ...
Unpacking texlive-plain-generic (2021.20220204-1) ...
Selecting previously unselected package tipa.
Preparing to unpack .../57-tipa_2%3a1.3-21_all.deb ...
Unpacking tipa (2:1.3-21) ...
Selecting previously unselected package texlive-xetex.
Preparing to unpack .../58-texlive-xetex_2021.20220204-1_all.deb ...
Unpacking texlive-xetex (2021.20220204-1) ...
Setting up fonts-lato (2.0-2.1) ...
Setting up fonts-noto-mono (20201225-1build1) ...
Setting up libwoff1:amd64 (1.0.2-1build4) ...
Setting up libtexlua53:amd64 (2021.20210626.59705-1ubuntu0.2) ...
Setting up libijs-0.35:amd64 (0.35-15build2) ...
Setting up libtexluaajit2:amd64 (2021.20210626.59705-1ubuntu0.2) ...
Setting up libfontbox-java (1:1.8.16-2) ...
Setting up rubygems-integration (1.18) ...
Setting up libzip-0-13:amd64 (0.13.72+dfsg.1-1.1) ...
Setting up fonts-urw-base35 (20200910-1) ...
Setting up poppler-data (0.4.11-1) ...
Setting up tex-common (6.17) ...
update-language: texlive-base not installed and configured, doing nothing!
Setting up libfontenc1:amd64 (1:1.1.4-1build3) ...
Setting up libjbig2dec0:amd64 (0.19-3build2) ...
Setting up libteckit0:amd64 (2.5.11+ds1-1) ...
Setting up libapache-pom-java (18-1) ...
Setting up ruby-net-telnet (0.1.1-2) ...
Setting up xfonts-encodings (1:1.0.5-0ubuntu2) ...
Setting up t1utils (1.41-4build2) ...
Setting up libidn12:amd64 (1.38-4ubuntu1) ...
Setting up fonts-texgyre (20180621-3.1) ...
Setting up libkpathsea6:amd64 (2021.20210626.59705-1ubuntu0.2) ...
Setting up ruby-webrick (1.7.0-3ubuntu0.1) ...
Setting up libcmark-gfm0.29.0.gfm.3:amd64 (0.29.0.gfm.3-3) ...
Setting up fonts-lmodern (2.004.5-6.1) ...
Setting up libcmark-gfm-extensions0.29.0.gfm.3:amd64 (0.29.0.gfm.3-3) ...

```



```

Setting up fonts-droid-fallback (1:6.0.1r16-1.1build1) ...
Setting up pandoc-data (2.9.2.1-3ubuntu2) ...
Setting up ruby-xmlrpc (0.3.2-1ubuntu0.1) ...
Setting up libsyntax2:amd64 (2021.20210626.59705-1ubuntu0.2) ...
Setting up libgs9-common (9.55.0~dfsg1-0ubuntu5.10) ...
Setting up teckit (2.5.11+ds1-1) ...
Setting up libpdfbox-java (1:1.8.16-2) ...
Setting up libgs9:amd64 (9.55.0~dfsg1-0ubuntu5.10) ...
Setting up preview-latex-style (12.2-1ubuntu1) ...
Setting up libcommons-parent-java (43-1) ...
Setting up dvisvgm (2.13.1-1) ...
Setting up libcommons-logging-java (1.2-2) ...
Setting up xfonts-utils (1:7.7+6build2) ...
Setting up libptexenc1:amd64 (2021.20210626.59705-1ubuntu0.2) ...
Setting up pandoc (2.9.2.1-3ubuntu2) ...
Setting up texlive-binaries (2021.20210626.59705-1ubuntu0.2) ...
update-alternatives: using /usr/bin/xdvi-xaw to provide /usr/bin/xdvi.bin
(xdvi.bin) in auto mode
update-alternatives: using /usr/bin/bibtex.original to provide /usr/bin/bibtex
(bibtex) in auto mode
Setting up lmodern (2.004.5-6.1) ...
Setting up texlive-base (2021.20220204-1) ...
/usr/bin/ucfr
/usr/bin/ucfr
/usr/bin/ucfr
/usr/bin/ucfr
mktexlsr: Updating /var/lib/texmf/ls-R-TEXLIVEDIST...
mktexlsr: Updating /var/lib/texmf/ls-R-TEXMFMAIN...
mktexlsr: Updating /var/lib/texmf/ls-R...
mktexlsr: Done.
tl-paper: setting paper size for dvips to a4:
/var/lib/texmf/dvips/config/config-paper.ps
tl-paper: setting paper size for dvipdfmx to a4:
/var/lib/texmf/dvipdfmx/dvipdfmx-paper.cfg
tl-paper: setting paper size for xdvi to a4: /var/lib/texmf/xdvi/XDvi-paper
tl-paper: setting paper size for pdftex to a4: /var/lib/texmf/tex/generic/tex-
ini-files/pdftexconfig.tex
Setting up tex-gyre (20180621-3.1) ...
Setting up texlive-plain-generic (2021.20220204-1) ...
Setting up texlive-latex-base (2021.20220204-1) ...
Setting up texlive-latex-recommended (2021.20220204-1) ...
Setting up texlive-pictures (2021.20220204-1) ...
Setting up texlive-fonts-recommended (2021.20220204-1) ...
Setting up tipa (2:1.3-21) ...
Setting up texlive (2021.20220204-1) ...
Setting up texlive-latex-extra (2021.20220204-1) ...
Setting up texlive-xetex (2021.20220204-1) ...
Setting up rake (13.0.6-2) ...

```

```

Setting up libruby3.0:amd64 (3.0.2-7ubuntu2.8) ...
Setting up ruby3.0 (3.0.2-7ubuntu2.8) ...
Setting up ruby (1:3.0~exp1) ...
Setting up ruby-rubygems (3.3.5-2) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for fontconfig (2.13.1-4.2ubuntu5) ...
Processing triggers for libc-bin (2.35-0ubuntu3.4) ...
/sbin/ldconfig.real: /usr/local/lib/libur_loader.so.0 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtcm_debug.so.1 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_5.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc_proxy.so.2 is not a symbolic
link

/sbin/ldconfig.real: /usr/local/lib/libur_adapter_opengl.so.0 is not a symbolic
link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_0.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbb.so.12 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtcm.so.1 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libur_adapter_level_zero.so.0 is not a
symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc.so.2 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libhwloc.so.15 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libumf.so.0 is not a symbolic link

Processing triggers for tex-common (6.17) ...
Running updmap-sys. This may take some time... done.
Running mktexlsr /var/lib/texmf ... done.
Building format(s) --all.
    This may take some time... done.

```