03. PyTorch Computer Vision Exercise

```
# Check for GPU
!nvidia-smi
```

```
Thu Jan  4 06:21:29 2024
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.104.05             Driver Version: 535.104.05   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  Tesla T4                       Off | 00000000:00:04.0 Off |                    0 |
| N/A   54C    P8              11W /  70W |      0MiB / 15360MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+---------------------------------------------------------------------------------------+
| Processes:                                                                            |
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory |
|        ID   ID                                                             Usage      |
|=======================================================================================|
|  No running processes found                                                           |
+---------------------------------------------------------------------------------------+
```

kode tersebut digunakan untuk mengecek informasi mengenai GPU (Graphics Processing Unit) yang tersedia pada sistem. outputnya memberikan informasi tentang spesifikasi GPU yang terpasang, seperti nama GPU, penggunaan memori, dan informasi lainnya.

```
# Import torch
import torch

# Exercises require PyTorch > 1.10.0
print(torch.__version__)

# Setup device agnostic code
device = "cuda" if torch.cuda.is_available() else "cpu"
device
```

```
2.1.0+cu121
'cuda'
```

output yang dihasilkan adalah 'cuda' yang artinya perangkat akan diatur ke GPU

1. What are 3 areas in industry where computer vision is currently being used?

1. Self-driving cars, such as Tesla using computer vision to percieve what's happening on the road. See Tesla AI day for more - https://youtu.be/j0z4FweCy4M
2. Healthcare imaging, such as using computer vision to help interpret X-rays. Google also uses computer vision for detecting polyps in the intenstines - https://ai.googleblog.com/2021/08/improved-detection-of-elusive-polyps.html
3. Security, computer vision can be used to detect whether someone is invading your home or not - https://store.google.com/au/product/nest_cam_battery?hl=en-GB

2. Search "what is overfitting in machine learning" and write down a sentence about what you find.

Overfitting is like memorizing for a test but then you can't answer a question that's slightly different.

In other words, if a model is overfitting, it's learning the training data *too well* and these patterns don't generalize to unseen data.

3. Search "ways to prevent overfitting in machine learning", write down 3 of the things you find and a sentence about each.

> **Note:** there are lots of these, so don't worry too much about all of them, just pick 3 and start with those.

See this article for some ideas: https://elitedatascience.com/overfitting-in-machine-learning

3 ways to prevent overfitting:

1. **Regularization techniques** - You could use dropout on your neural networks, dropout involves randomly removing neurons in different layers so that the remaining neurons hopefully learn more robust weights/patterns.
2. **Use a different model** - maybe the model you're using for a specific problem is too complicated, as in, it's learning the data too well because it has so many layers. You could remove some layers to simplify your model. Or you could pick a totally different model altogether, one that may be more suited to your particular problem. Or... you could also use transfer learning (taking the patterns from one model and applying them to your own problem).
3. **Reduce noise in data/cleanup dataset/introduce data augmentation techniques** - If the model is learning the data too well, it might be just memorizing the data, including the noise. One option would be to remove the noise/clean up the dataset or if this doesn't, you can introduce artificial noise through the use of data augmentation to artificially increase the diversity of your training dataset.

4. Spend 20-minutes reading and clicking through the CNN Explainer website.

- Upload your own example image using the "upload" button on the website and see what happens in each layer of a CNN as your image passes through it.

The CNN explainer website is a great insight into all of the nuts and bolts of a convolutional neural network.

5. Load the `torchvision.datasets.MNIST()` train and test datasets.

```
import torchvision
from torchvision import datasets

from torchvision import transforms
```

Kode tersebut mengimpor modul torchvision dan beberapa submodul serta fungsi dari modul

```
# Get the MNIST train dataset
train_data = datasets.MNIST(root=".",
                            train=True,
                            download=True,
                            transform=transforms.ToTensor()) # do we want to transform the data as we download it?

# Get the MNIST test dataset
test_data = datasets.MNIST(root=".",
                           train=False,
                           download=True,
                           transform=transforms.ToTensor())
```

```
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz to ./MNIST/raw/train-images-idx3-ubyte.gz
100%|██████████| 9912422/9912422 [00:00<00:00, 91202606.15it/s]
Extracting ./MNIST/raw/train-images-idx3-ubyte.gz to ./MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz to ./MNIST/raw/train-labels-idx1-ubyte.gz
100%|██████████| 28881/28881 [00:00<00:00, 35945309.74it/s]
Extracting ./MNIST/raw/train-labels-idx1-ubyte.gz to ./MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz to ./MNIST/raw/t10k-images-idx3-ubyte.gz
100%|██████████| 1648877/1648877 [00:00<00:00, 114902912.44it/s]Extracting ./MNIST/raw/t10k-images-idx3-ubyte.gz to ./MNIST/raw


Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz to ./MNIST/raw/t10k-labels-idx1-ubyte.gz
100%|██████████| 4542/4542 [00:00<00:00, 6038202.46it/s]
Extracting ./MNIST/raw/t10k-labels-idx1-ubyte.gz to ./MNIST/raw
```

kode digunakan untuk mendapatkan dataset MNIST (Modified National Institute of Standards and Technology) menggunakan modul 'datasets' dari 'torchvision' .

```
train_data, test_data
```

```
(Dataset MNIST
     Number of datapoints: 60000
     Root location: .
```

```
        Split: Train
        StandardTransform
    Transform: ToTensor(),
    Dataset MNIST
        Number of datapoints: 10000
        Root location: .
        Split: Test
        StandardTransform
    Transform: ToTensor())
```

kode dieksekusi dan output mencantumkan informasi tentang objek 'train_data' dan 'test_data' termasuk jenis dataset, jumlah sampel, dan bentuk tensor untuk setiap sampel.

```
len(train_data), len(test_data)
```

```
(60000, 10000)
```

kode akan menghasilkan output jumlah data points dari train_data dan test_data. train_data memiliki datapoints 6000 sementara test_data memiliki datapoints 1000

```
# Data is in tuple form (image, label)
img = train_data[0][0]
label = train_data[0][1]
print(f"Image:\n {img}")
print(f"Label:\n {label}")
```

```
          0.0000, 0.0000, 0.0000, 0.0000],
         [0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.1922,
          0.9333, 0.9922, 0.9922, 0.9922, 0.9922, 0.9922, 0.9922, 0.9922,
          0.9922, 0.9843, 0.3647, 0.3216, 0.3216, 0.2196, 0.1529, 0.0000,
          0.0000, 0.0000, 0.0000, 0.0000],
         [0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0706,
          0.8588, 0.9922, 0.9922, 0.9922, 0.9922, 0.9922, 0.7765, 0.7137,
          0.9686, 0.9451, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
          0.0000, 0.0000, 0.0000, 0.0000],
         [0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
          0.3137, 0.6118, 0.4196, 0.9922, 0.9922, 0.8039, 0.0431, 0.0000,
          0.1686, 0.6039, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
          0.0000, 0.0000, 0.0000, 0.0000],
         [0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
          0.0000, 0.0549, 0.0039, 0.6039, 0.9922, 0.3529, 0.0000, 0.0000,
          0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
          0.0000, 0.0000, 0.0000, 0.0000],
         [0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
          0.0000, 0.0000, 0.0000, 0.5451, 0.9922, 0.7451, 0.0078, 0.0000,
          0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
          0.0000, 0.0000, 0.0000, 0.0000],
         [0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
          0.0000, 0.0000, 0.0000, 0.0431, 0.7451, 0.9922, 0.2745, 0.0000,
          0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
          0.0000, 0.0000, 0.0000, 0.0000],
         [0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
          0.0000, 0.0000, 0.0000, 0.0000, 0.1373, 0.9451, 0.8824, 0.6275,
          0.4235, 0.0039, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
          0.0000, 0.0000, 0.0000, 0.0000],
         [0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
          0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.3176, 0.9412, 0.9922,
          0.9922, 0.4667, 0.0980, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
          0.0000, 0.0000, 0.0000, 0.0000],
         [0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
          0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.1765, 0.7294,
          0.9922, 0.9922, 0.5882, 0.1059, 0.0000, 0.0000, 0.0000, 0.0000,
          0.0000, 0.0000, 0.0000, 0.0000],
         [0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
          0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0627,
          0.3647, 0.9882, 0.9922, 0.7333, 0.0000, 0.0000, 0.0000, 0.0000,
          0.0000, 0.0000, 0.0000, 0.0000],
         [0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
          0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
          0.0000, 0.9765, 0.9922, 0.9765, 0.2510, 0.0000, 0.0000, 0.0000,
          0.0000, 0.0000, 0.0000, 0.0000],
         [0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
          0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.1804, 0.5098,
          0.7176, 0.9922, 0.9922, 0.8118, 0.0078, 0.0000, 0.0000, 0.0000,
          0.0000, 0.0000, 0.0000, 0.0000],
         [0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
          0.0000, 0.0000, 0.0000, 0.0000, 0.1529, 0.5804, 0.8980, 0.9922,
          0.9922, 0.9922, 0.9804, 0.7137, 0.0000, 0.0000, 0.0000, 0.0000,
          0.0000, 0.0000, 0.0000, 0.0000],
         [0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
          0.0000, 0.0000, 0.0941, 0.4471, 0.8667, 0.9922, 0.9922, 0.9922,
          0.9922, 0.7882, 0.3059, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
          0.0000, 0.0000, 0.0000, 0.0000],
         [0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
          0.0902, 0.2588, 0.8353, 0.9922, 0.9922, 0.9922, 0.9922, 0.7765,
```

kode mengekstrak informasi dari satu sampel dalam dataset pelatihan MNIST. oputput yang dihasilkan yaitu label kelas dari sampel pertama sampai label kelas sampel ke 5

```
# Check out the shapes of our data
print(f"Image shape: {img.shape} -> [color_channels, height, width] (CHW)")
print(f"Label: {label} -> no shape, due to being integer")
```

```
     Image shape: torch.Size([1, 28, 28]) -> [color_channels, height, width] (CHW)
     Label: 5 -> no shape, due to being integer
```

kode akan mencetak atau mengeluarkan informasi tentang bentuk (shape) dari tensor gambar dan label pada satu sampel dari dataset pelatihan MNIST. output yang dihasilkan merupakan informasi tentang struktur tensor gambar dan label pada satu sampel dari dataset MNIST.

Note: There are two main agreed upon ways for representing images in machine learning:

1. Color channels first: [color_channels, height, width] (CHW) -> PyTorch default (as of April 2022)
2. Color channels last: [height, width, color_channels] (HWC) -> Matplotlib/TensorFlow default (as of April 2022)

```
# Get the class names from the dataset
class_names = train_data.classes
class_names
```

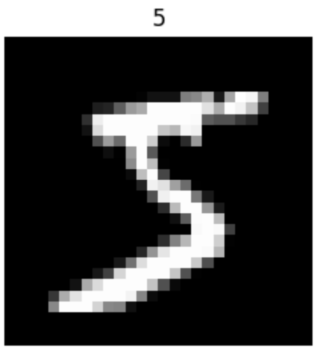
```
     ['0 - zero',
      '1 - one',
      '2 - two',
      '3 - three',
      '4 - four',
      '5 - five',
      '6 - six',
      '7 - seven',
      '8 - eight',
      '9 - nine']
```

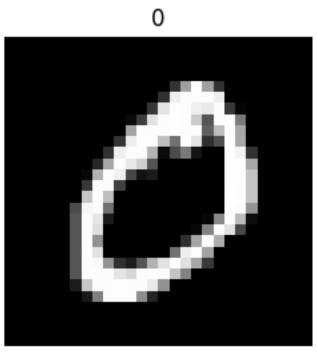
kode tersebut mengakses atribut classes dari objek 'train_data' pada dataset MNIST. Atribut classes dari objek dataset menyediakan daftar nama kelas yang mewakili digit dari 0 hingga 9 pada dataset MNIST.

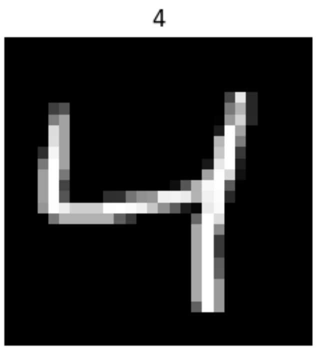
6. Visualize at least 5 different samples of the MNIST training dataset.

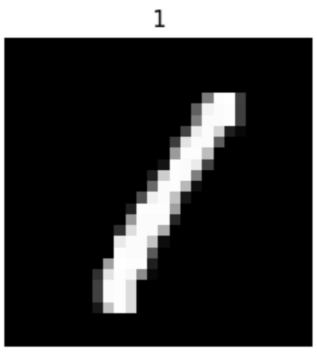
```
import matplotlib.pyplot as plt
for i in range(5):
  img = train_data[i][0]
  print(img.shape)
  img_squeeze = img.squeeze()
  print(img_squeeze.shape)
  label = train_data[i][1]
  plt.figure(figsize=(3, 3))
  plt.imshow(img_squeeze, cmap="gray")
  plt.title(label)
  plt.axis(False);
```

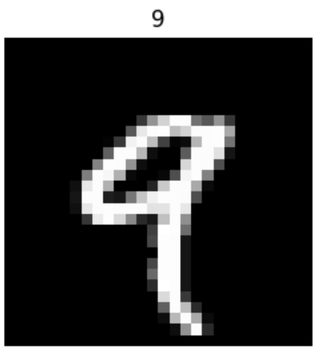
```
torch.Size([1, 28, 28])
torch.Size([28, 28])
torch.Size([1, 28, 28])
torch.Size([28, 28])
torch.Size([1, 28, 28])
torch.Size([28, 28])
torch.Size([1, 28, 28])
torch.Size([28, 28])
torch.Size([1, 28, 28])
torch.Size([28, 28])
```











kode akan melakukan iterasi sebanyak 5 kali melalui lima sampel pertama dalam dataset pelatihan MNIST. hasil output tersebut adalah plot dari lima sampel pertama dalam dataset pelatihan MNIST, menampilkan gambar digit bersama dengan label kelasnya.

7. Turn the MNIST train and test datasets into dataloaders using `torch.utils.data.DataLoader`, set the `batch_size=32`.

```
# Create train dataloader
from torch.utils.data import DataLoader

train_dataloader = DataLoader(dataset=train_data,
                              batch_size=32,
                              shuffle=True)

test_dataloader = DataLoader(dataset=test_data,
                             batch_size=32,
                             shuffle=False)
```

Kode tersebut menggunakan DataLoader dari PyTorch untuk membuat dataloader (pemuat data) untuk dataset pelatihan dan pengujian MNIST. dengan memuat data dalam batch dan dapat mengacak urutan data untuk setiap epoch pelatihan, membantu meningkatkan keberagaman dan keacakan dalam pelatihan model. Dengan menggunakan dataloader akan memudahkan mengakses batch data saat melatih atau menguji model.

```
train_dataloader, test_dataloader
```

```
(<torch.utils.data.dataloader.DataLoader at 0x7ad5e37c0f40>,
 <torch.utils.data.dataloader.DataLoader at 0x7ad5e37c11b0>)
```

train_dataloader dan test_dataloader adalah instance dari DataLoader dari PyTorch.

```
for sample in next(iter(train_dataloader)):
  print(sample.shape)
```

```
torch.Size([32, 1, 28, 28])
torch.Size([32])
```

kode memberikan informasi tentang bentuk dari elemen dalam batch pertama dari 'train_dataloader'. next(iter(train_dataloader)) digunakan untuk mengambil satu batch pertama dari train_dataloader untuk keperluan demonstrasi.

```
len(train_dataloader), len(test_dataloader)
```

```
(1875, 313)
```

kode tersebut memberikan informasi jumlah batch yang tersedia dalam dataloader pelatihan dan dataloader pengujian. batch ini akan digunakan selama satu epoch dari evaluasi model pada dataset pengujian. Jumlah batch dalam dataloader bergantung pada ukuran dataset dan ukuran batch yang ditentukan saat membuat dataloader. Semakin besar dataset dan semakin kecil ukuran batch, semakin banyak batch yang akan tersedia. Sebaliknya, semakin kecil dataset dan semakin besar ukuran batch, semakin sedikit batch yang akan tersedia.

8. Recreate `model_2` used in notebook 03 (the same model from the CNN Explainer website, also known as TinyVGG) capable of fitting on the MNIST dataset.

```
from torch import nn
class MNIST_model(torch.nn.Module):
  """Model capable of predicting on MNIST dataset.
  """
  def __init__(self, input_shape: int, hidden_units: int, output_shape: int):
    super().__init__()
    self.conv_block_1 = nn.Sequential(
      nn.Conv2d(in_channels=input_shape,
                out_channels=hidden_units,
                kernel_size=3,
                stride=1,
                padding=1),
      nn.ReLU(),
      nn.Conv2d(in_channels=hidden_units,
                out_channels=hidden_units,
                kernel_size=3,
                stride=1,
                padding=1),
      nn.ReLU(),
      nn.MaxPool2d(kernel_size=2)
    )
    self.conv_block_2 = nn.Sequential(
      nn.Conv2d(in_channels=hidden_units,
                out_channels=hidden_units,
                kernel_size=3,
                stride=1,
                padding=1),
      nn.ReLU(),
      nn.Conv2d(in_channels=hidden_units,
                out_channels=hidden_units,
                kernel_size=3,
                stride=1,
                padding=1),
      nn.ReLU(),
      nn.MaxPool2d(kernel_size=2)
    )
    self.classifier = nn.Sequential(
      nn.Flatten(),
      nn.Linear(in_features=hidden_units*7*7,
                out_features=output_shape)
    )

  def forward(self, x):
    x = self.conv_block_1(x)
    # print(f"Output shape of conv block 1: {x.shape}")
    x = self.conv_block_2(x)
    # print(f"Output shape of conv block 2: {x.shape}")
    x = self.classifier(x)
    # print(f"Output shape of classifier: {x.shape}")
    return x
```

model akan menggunakan arsitektur konvolusi dengan dua blok konvolusi dan lapisan linear untuk menghasilkan prediksi pada dataset MNIST

```
device
```

```
'cuda'
```

device yang tersedia yaitu 'GPU'

```
model = MNIST_model(input_shape=1,
                    hidden_units=10,
                    output_shape=10).to(device)
model
```

```
MNIST_model(
  (conv_block_1): Sequential(
    (0): Conv2d(1, 10, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): Conv2d(10, 10, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU()
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (conv_block_2): Sequential(
    (0): Conv2d(10, 10, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): Conv2d(10, 10, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU()
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (classifier): Sequential(
    (0): Flatten(start_dim=1, end_dim=-1)
```

```
          (1): Linear(in_features=490, out_features=10, bias=True)
        )
      )
```

output dari kode menunjukkan representasi string dari model, termasuk informasi tentang setiap lapisan dan parameter yang digunakan dalam model.

```
# Check out the model state dict to find out what patterns our model wants to learn
# model.state_dict()
```

```
# Try a dummy forward pass to see what shapes our data is
dummy_x = torch.rand(size=(1, 28, 28)).unsqueeze(dim=0).to(device)
# dummy_x.shape
model(dummy_x)
```

```
     tensor([[ 0.0521,  0.0058,  0.0138,  0.0235, -0.0566, -0.0056, -0.0691,  0.0350,
              -0.0887,  0.0046]], device='cuda:0', grad_fn=<AddmmBackward0>)
```

output yang dihasilkan oleh model dari forward pass ini adalah gambaran tentang bentuk output yang dihasilkan oleh model ketika diberikan data dengan dimensi tertentu. Output tersebut mencerminkan dimensi dari lapisan-lapisan dalam model.

```
dummy_x_2 = torch.rand(size=([1, 10, 7, 7]))
dummy_x_2.shape
```

```
     torch.Size([1, 10, 7, 7])
```

kode tersebut digunakan untuk membuat tensor dummy_x_2 dengan dimensi [1, 10, 7, 7]. Jadi, dummy_x_2 mencerminkan dimensi yang diharapkan dari output model ketika menerima input dari blok konvolusi kedua dan lapisan aktivasinya.

```
flatten_layer = nn.Flatten()
flatten_layer(dummy_x_2).shape
```

```
     torch.Size([1, 490])
```

kode tersebut yaitu menggunakan lapisan nn.Flatten() untuk meratakan (flatten) tensor dummy_x_2 yang memiliki dimensi [1, 10, 7, 7]. Setelah dilakukan flattening, dimensinya menjadi [1, 490] yang kemudian digunakan sebagai input untuk lapisan linear

9. Train the model you built in exercise 8. for 5 epochs on CPU and GPU and see how long it takes on each.

```
%%time
from tqdm.auto import tqdm

# Train on CPU
model_cpu = MNIST_model(input_shape=1,
                        hidden_units=10,
                        output_shape=10).to("cpu")

# Create a loss function and optimizer
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model_cpu.parameters(), lr=0.1)

### Training loop
epochs = 5
for epoch in tqdm(range(epochs)):
  train_loss = 0
  for batch, (X, y) in enumerate(train_dataloader):
    model_cpu.train()

    # Put data on CPU
    X, y = X.to("cpu"), y.to("cpu")

    # Forward pass
    y_pred = model_cpu(X)

    # Loss calculation
    loss = loss_fn(y_pred, y)
    train_loss += loss

    # Optimizer zero grad
    optimizer.zero_grad()
    # Loss backward
    loss.backward()

    # Step the optimizer
    optimizer.step()

  # Adjust train loss for number of batches
  train_loss /= len(train_dataloader)

  ### Testing loop
  test_loss_total = 0

  # Put model in eval mode
  model_cpu.eval()

  # Turn on inference mode
  with torch.inference_mode():
    for batch, (X_test, y_test) in enumerate(test_dataloader):
      # Make sure test data on CPU
      X_test, y_test = X_test.to("cpu"), y_test.to("cpu")
      test_pred = model_cpu(X_test)
      test_loss = loss_fn(test_pred, y_test)

      test_loss_total += test_loss

    test_loss_total /= len(test_dataloader)

  # Print out what's happening
  print(f"Epoch: {epoch} | Loss: {train_loss:.3f} | Test loss: {test_loss_total:.3f}")
```

```
100%                                              5/5 [03:18<00:00, 39.36s/it]
Epoch: 0 | Loss: 0.317 | Test loss: 0.060
Epoch: 1 | Loss: 0.065 | Test loss: 0.049
Epoch: 2 | Loss: 0.053 | Test loss: 0.038
Epoch: 3 | Loss: 0.044 | Test loss: 0.037
Epoch: 4 | Loss: 0.038 | Test loss: 0.038
CPU times: user 3min 10s, sys: 1.61 s, total: 3min 12s
Wall time: 3min 18s
```

selanjutnya dilakukan pelatihan model pada CPU untuk beberapa epoch

```
%%time
from tqdm.auto import tqdm

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

# Train on GPU
model_gpu = MNIST_model(input_shape=1,
                        hidden_units=10,
                        output_shape=10).to(device)

# Create a loss function and optimizer
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model_gpu.parameters(), lr=0.1)

# Training loop
epochs = 5
for epoch in tqdm(range(epochs)):
  train_loss = 0
  model_gpu.train()
  for batch, (X, y) in enumerate(train_dataloader):
    # Put data on target device
    X, y = X.to(device), y.to(device)

    # Forward pass
    y_pred = model_gpu(X)

    # Loss calculation
    loss = loss_fn(y_pred, y)
    train_loss += loss

    # Optimizer zero grad
    optimizer.zero_grad()

    # Loss backward
    loss.backward()

    # Step the optimizer
    optimizer.step()

  # Adjust train loss to number of batches
  train_loss /= len(train_dataloader)

  ### Testing loop
  test_loss_total = 0
  # Put model in eval mode and turn on inference mode
  model_gpu.eval()
  with torch.inference_mode():
    for batch, (X_test, y_test) in enumerate(test_dataloader):
      # Make sure test data on target device
      X_test, y_test = X_test.to(device), y_test.to(device)

      test_pred = model_gpu(X_test)
      test_loss = loss_fn(test_pred, y_test)

      test_loss_total += test_loss

    # Adjust test loss total for number of batches
    test_loss_total /= len(test_dataloader)

  # Print out what's happening
  print(f"Epoch: {epoch} | Loss: {train_loss:.3f} | Test loss: {test_loss_total:.3f}")
```

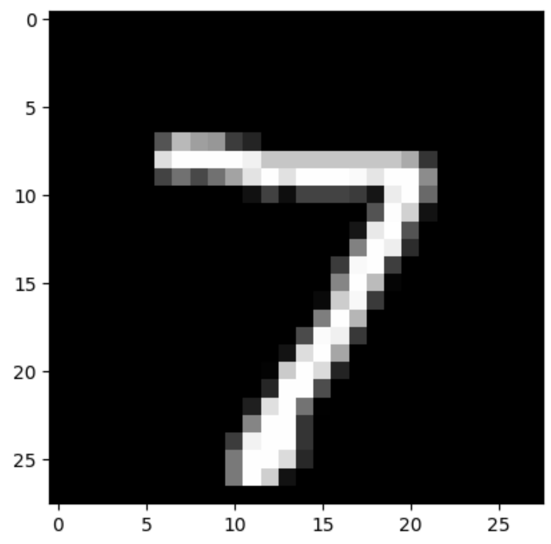
```
100%                                        5/5 [01:00<00:00, 12.06s/it]
Epoch: 0 | Loss: 0.309 | Test loss: 0.069
Epoch: 1 | Loss: 0.074 | Test loss: 0.058
Epoch: 2 | Loss: 0.059 | Test loss: 0.052
Epoch: 3 | Loss: 0.050 | Test loss: 0.040
Epoch: 4 | Loss: 0.045 | Test loss: 0.034
CPU times: user 57.9 s, sys: 660 ms, total: 58.6 s
Wall time: 1min
```

langkah selanjutnya pada kode tersebut dilakukan pelatihan model pada GPU untuk beberapa epoch. langkah-langkahnya mirip dengan pelatihan pada CPU. hanya saja model ditempatkan di GPU dan data dipindahkan ke GPU sebelum melakukan komputasi.

10. Make predictions using your trained model and visualize at least 5 of them comparing the prediciton to the target label.

```
# Make predictions with the trained model
plt.imshow(test_data[0][0].squeeze(), cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7ad5e146dc60>
```



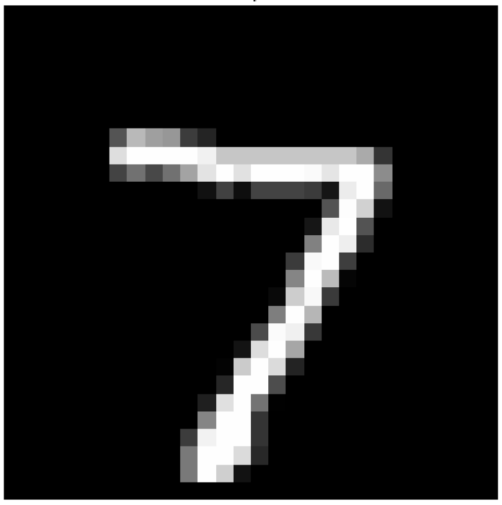
output yang dihasilkan merupakan gambar pertama angka 7 dari dataset uji ditampilkan dalam bentuk matriks piksel hitam-putih. dilakukan visualisasi gambar dari dataset uji (test_data).

```
# Logits -> Prediction probabilities -> Prediction labels
model_pred_logits = model_gpu(test_data[0][0].unsqueeze(dim=0).to(device)) # make sure image is right shape + on right device
model_pred_probs = torch.softmax(model_pred_logits, dim=1)
model_pred_label = torch.argmax(model_pred_probs, dim=1)
model_pred_label
```

```
tensor([7], device='cuda:0')
```

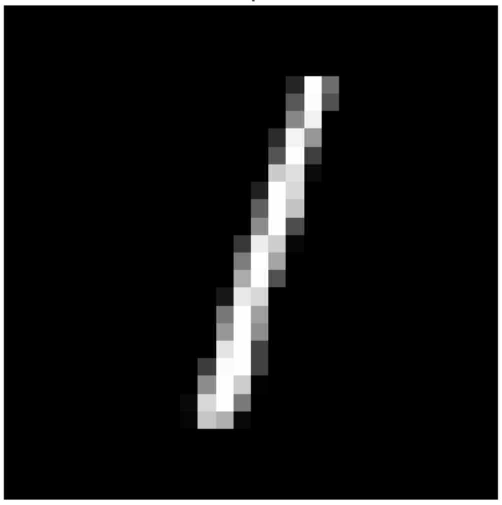
```
num_to_plot = 5
for i in range(num_to_plot):
  # Get image and labels from the test data
  img = test_data[i][0]
  label = test_data[i][1]

  # Make prediction on image
  model_pred_logits = model_gpu(img.unsqueeze(dim=0).to(device))
  model_pred_probs = torch.softmax(model_pred_logits, dim=1)
  model_pred_label = torch.argmax(model_pred_probs, dim=1)

  # Plot the image and prediction
  plt.figure()
  plt.imshow(img.squeeze(), cmap="gray")
  plt.title(f"Truth: {label} | Pred: {model_pred_label.cpu().item()}")
  plt.axis(False);
```

Truth: 7 | Pred: 7

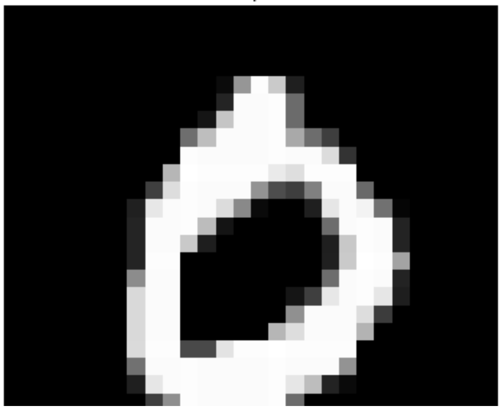


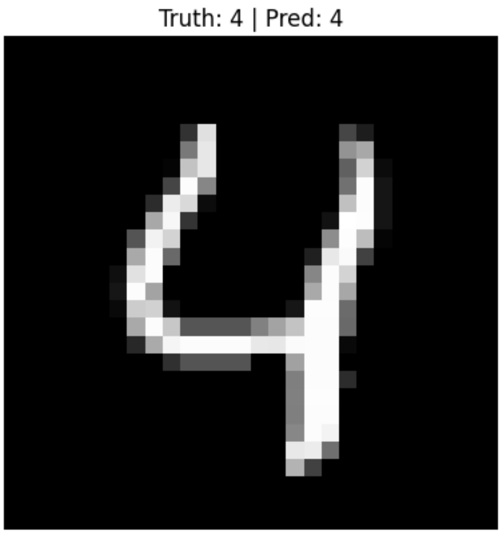
Truth: 2 | Pred: 2

Truth: 1 | Pred: 1



Truth: 0 | Pred: 0





Hasil prediksi berupa label kelas yang diperoleh dari model untuk gambar pertama dari dataset uji.

11. Plot a confusion matrix comparing your model's predictions to the truth labels.

```
# See if torchmetrics exists, if not, install it
try:
    import torchmetrics, mlxtend
    print(f"mlxtend version: {mlxtend.__version__}")
    assert int(mlxtend.__version__.split(".")[1]) >= 19, "mlxtend verison should be 0.19.0 or higher"
except:
    !pip install -q torchmetrics -U mlxtend # <- Note: If you're using Google Colab, this may require restarting the runtime
    import torchmetrics, mlxtend
    print(f"mlxtend version: {mlxtend.__version__}")
```

```
        ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 806.1/806.1 kB 8.8 MB/s eta 0:00:00
        ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.4/1.4 MB 45.1 MB/s eta 0:00:00
 mlxtend version: 0.23.0
```

kode dilakukan untuk pengecekan apakah modul torchmetrics dan mlxtend telah diinstal. Jika belum, instalasi dilakukan menggunakan perintah pip. Pengecekan ini memastikan bahwa versi terbaru dari kedua modul tersebut diinstal.

```
# Import mlxtend upgraded version
import mlxtend
print(mlxtend.__version__)
assert int(mlxtend.__version__.split(".")[1]) >= 19 # should be version 0.19.0 or higher
```

```
 0.23.0
```

modul mlxtend yang telah diinstal sebelumnya diimpor kembali. Kemudian, versi mlxtend diperiksa untuk memastikan bahwa versi yang diinstal setidaknya adalah 0.19.0 atau versi yang lebih tinggi.

```
# Make predictions across all test data
from tqdm.auto import tqdm
model_gpu.eval()
y_preds = []
with torch.inference_mode():
  for batch, (X, y) in tqdm(enumerate(test_dataloader)):
    # Make sure data on right device
    X, y = X.to(device), y.to(device)
    # Forward pass
    y_pred_logits = model_gpu(X)
    # Logits -> Pred probs -> Pred label
    y_pred_labels = torch.argmax(torch.softmax(y_pred_logits, dim=1), dim=1)
    # Append the labels to the preds list
    y_preds.append(y_pred_labels)
  y_preds=torch.cat(y_preds).cpu()
len(y_preds)
```

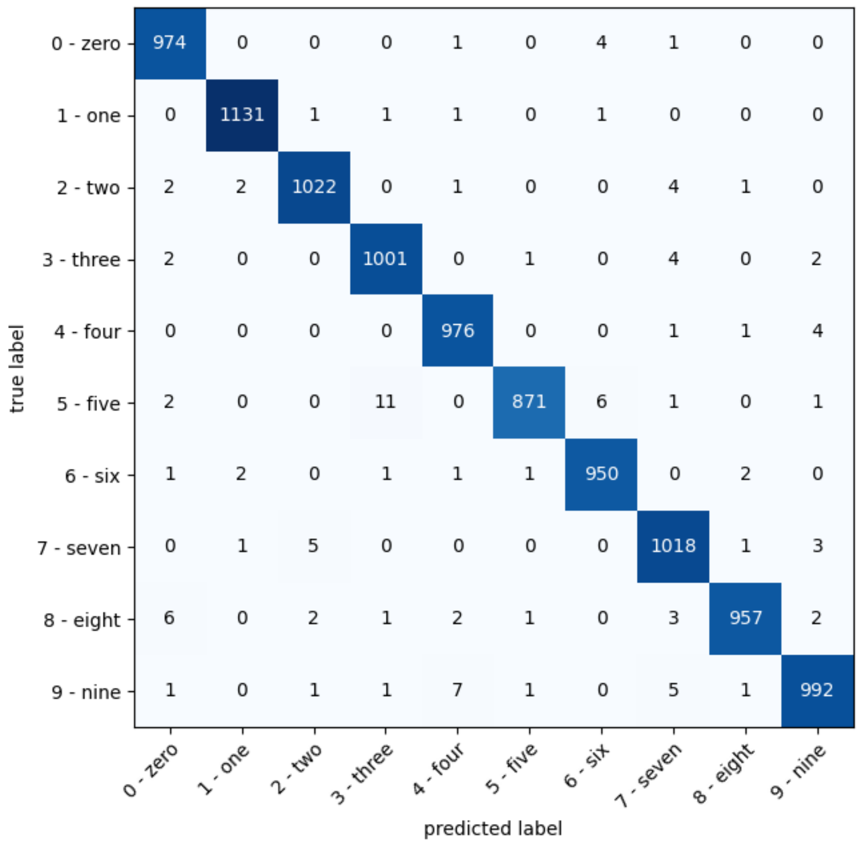
```
    313/? [00:01<00:00, 187.93it/s]
   10000
```

kode di atas dilakukan prediksi menggunakan model pada seluruh data uji (test_data).

```
test_data.targets[:10], y_preds[:10]

    (tensor([7, 2, 1, 0, 4, 1, 4, 9, 5, 9]),
     tensor([7, 2, 1, 0, 4, 1, 4, 9, 5, 9]))
```

kode di atas dilakukan pencocokan antara label sebenarnya (test_data.targets) dengan label prediksi (y_preds) untuk 10 sampel pertama dalam data uji.

```
from torchmetrics import ConfusionMatrix
from mlxtend.plotting import plot_confusion_matrix

# Setup confusion matrix
confmat = ConfusionMatrix(task="multiclass", num_classes=len(class_names))
confmat_tensor = confmat(preds=y_preds,
                         target=test_data.targets)

# Plot the confusion matrix
fix, ax = plot_confusion_matrix(
    conf_mat=confmat_tensor.numpy(),
    class_names=class_names,
    figsize=(10, 7)
)
```



	0 - zero	1 - one	2 - two	3 - three	4 - four	5 - five	6 - six	7 - seven	8 - eight	9 - nine
0 - zero	974	0	0	0	1	0	4	1	0	0
1 - one	0	1131	1	1	1	0	1	0	0	0
2 - two	2	2	1022	0	1	0	0	4	1	0
3 - three	2	0	0	1001	0	1	0	4	0	2
4 - four	0	0	0	0	976	0	0	1	1	4
5 - five	2	0	0	11	0	871	6	1	0	1
6 - six	1	2	0	1	1	1	950	0	2	0
7 - seven	0	1	5	0	0	0	0	1018	1	3
8 - eight	6	0	2	1	2	1	0	3	957	2
9 - nine	1	0	1	1	7	1	0	5	1	992

kode tersebut menghasilkan 'confusionMatrix' yang memberikan informasi tentang seberapa baik model dapat mengklasifikasikan instance ke dalam kelas yang benar

12. Create a random tensor of shape `[1, 3, 64, 64]` and pass it through a `nn.Conv2d()` layer with various hyperparameter settings (these can be any settings you choose), what do you notice if the `kernel_size` parameter goes up and down?

```
random_tensor = torch.rand([1, 3, 64, 64])
random_tensor.shape
```

```
    torch.Size([1, 3, 64, 64])
```

hasil output merupakan tensor acak dengan bentuk (1, 3, 64, 64), yang sesuai dengan representasi batch dari gambar berwarna dengan ukuran 64x64 piksel. [1, 3, 64, 64]: Ini adalah bentuk (shape) dari tensor yang akan dibuat. Secara spesifik, ini adalah tensor dengan dimensi 4, di mana:

Dimensi pertama (1): Menunjukkan bahwa kita memiliki satu batch dari data. Dimensi kedua (3): Merupakan jumlah channel warna. Dalam hal ini, kita memiliki 3 channel warna (RGB). Dimensi ketiga (64): Merupakan tinggi dari gambar. Dimensi keempat (64): Merupakan lebar dari gambar.

```
conv_layer = nn.Conv2d(in_channels=3,
                       out_channels=64,
                       kernel_size=3,
                       stride=2,
                       padding=1)

print(f"Random tensor original shape: {random_tensor.shape}")
random_tensor_through_conv_layer = conv_layer(random_tensor)
print(f"Random tensor through conv layer shape: {random_tensor_through_conv_layer.shape}")

    Random tensor original shape: torch.Size([1, 3, 64, 64])
    Random tensor through conv layer shape: torch.Size([1, 64, 32, 32])
```

kode tersebut digunakan untuk memberikan output dari layer konvolusi

13. Use a model similar to the trained `model_2` from notebook 03 to make predictions on the test `torchvision.datasets.FashionMNIST` dataset.

- Then plot some predictions where the model was wrong alongside what the label of the image should've been.
- After visualing these predictions do you think it's more of a modelling error or a data error?
- As in, could the model do better or are the labels of the data too close to each other (e.g. a "Shirt" label is too close to "T-shirt/top")?

```
# Download FashionMNIST train & test
from torchvision import datasets
from torchvision import transforms

fashion_mnist_train = datasets.FashionMNIST(root=".",
                                            download=True,
                                            train=True,
                                            transform=transforms.ToTensor())

fashion_mnist_test = datasets.FashionMNIST(root=".",
                                           train=False,
                                           download=True,
                                           transform=transforms.ToTensor())

len(fashion_mnist_train), len(fashion_mnist_test)

    Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-images-idx3-ubyte.gz
    Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-images-idx3-ubyte.gz to ./FashionMNIST/raw/train-images
    100%|██████████| 26421880/26421880 [00:02<00:00, 10453616.03it/s]
    Extracting ./FashionMNIST/raw/train-images-idx3-ubyte.gz to ./FashionMNIST/raw

    Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-labels-idx1-ubyte.gz
    Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-labels-idx1-ubyte.gz to ./FashionMNIST/raw/train-label
    100%|██████████| 29515/29515 [00:00<00:00, 170093.94it/s]
    Extracting ./FashionMNIST/raw/train-labels-idx1-ubyte.gz to ./FashionMNIST/raw

    Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-images-idx3-ubyte.gz
    Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-images-idx3-ubyte.gz to ./FashionMNIST/raw/t10k-images-
    100%|██████████| 4422102/4422102 [00:01<00:00, 3272319.90it/s]
    Extracting ./FashionMNIST/raw/t10k-images-idx3-ubyte.gz to ./FashionMNIST/raw

    Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-labels-idx1-ubyte.gz
    Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-labels-idx1-ubyte.gz to ./FashionMNIST/raw/t10k-labels-
    100%|██████████| 5148/5148 [00:00<00:00, 14048325.95it/s]Extracting ./FashionMNIST/raw/t10k-labels-idx1-ubyte.gz to ./FashionMNIST/


    (60000, 10000)
```

pada kode tersebut menunjukkan perintah untuk mengunduh dataset FashionMNIST menggunakan PyTorch's torchvision.datasets. Hasil cetakan len(fashion_mnist_train), len(fashion_mnist_test) menunjukkan jumlah sampel data dalam dataset pelatihan dan pengujian FashionMNIST.

```
# Get the class names of the Fashion MNIST dataset
fashion_mnist_class_names = fashion_mnist_train.classes
fashion_mnist_class_names
```

```
['T-shirt/top',
 'Trouser',
 'Pullover',
 'Dress',
 'Coat',
 'Sandal',
 'Shirt',
 'Sneaker',
 'Bag',
 'Ankle boot']
```

kode tersebut digunakan untuk mendapatkan dataset fasihon MNIST untuk melatih model yang terdiri dari dataset gambar yang terdiri dari 10 kategori pakaian yang berbeda

```
# Turn FashionMNIST datasets into dataloaders
from torch.utils.data import DataLoader

fashion_mnist_train_dataloader = DataLoader(fashion_mnist_train,
                                            batch_size=32,
                                            shuffle=True)

fashion_mnist_test_dataloader = DataLoader(fashion_mnist_test,
                                           batch_size=32,
                                           shuffle=False)

len(fashion_mnist_train_dataloader), len(fashion_mnist_test_dataloader)
```

```
(1875, 313)
```

langkah selanjutnya pada kode tersebut yaitu mengubah dataset fashion MNIST menjadi dataloaders yang akan mempermudah proses pelatihan dan pengujian model dengan memungkinkan penggunaan batch data. output yang dihasilkan adalah jumlah batch dari DataLoader untuk dataset pelatihan dan pengujian

```
# model_2 is the same architecture as MNIST_model
model_2 = MNIST_model(input_shape=1,
                      hidden_units=10,
                      output_shape=10).to(device)
model_2
```

```
MNIST_model(
  (conv_block_1): Sequential(
    (0): Conv2d(1, 10, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): Conv2d(10, 10, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU()
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (conv_block_2): Sequential(
    (0): Conv2d(10, 10, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): Conv2d(10, 10, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU()
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (classifier): Sequential(
    (0): Flatten(start_dim=1, end_dim=-1)
    (1): Linear(in_features=490, out_features=10, bias=True)
  )
)
```

kode tersebut digunakan untuk melatih atau menguji model dengan menggunakan model_2. model_2 ini merupakan objek model yang dibuat dari kelas yang sesuai

```
# Setup loss and optimizer
from torch import nn
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model_2.parameters(), lr=0.01)
```

kode dilakukan untuk menyiapkan fungsi kerugian (loss) dan pengoptimal (optimizer) untuk pelatihan model model_2.