Nama : Andhika Aria Pratama N

NIM : 1103202121

1 PyTorch Workflow Fundamentals

- Data (preparing and loading)

Dalam tahap persiapan data untuk pengolahan machine learning, data yang digunakan dapat berupa struktur atau tidak terstruktur, tergantung pada kebutuhan spesifik proyek. Setelah data dipersiapkan, langkah selanjutnya adalah membaginya menjadi dua subset, yaitu data pelatihan (train) dan data pengujian (test). Pembagian ini umumnya dilakukan dengan rasio 80% untuk data pelatihan dan 20% untuk data pengujian. Hal ini bertujuan untuk melatih model menggunakan sebagian besar data dan menguji kinerjanya pada bagian yang tidak digunakan selama pelatihan

- Pembuatan Model

Dalam proses ini, mengembangkan model regresi linear menggunakan PyTorch sebagai kelas Python yang mengimplementasikan fungsi regresi linear sederhana. Model tersebut didefinisikan dengan dua parameter, yakni weight dan bias, yang awalnya diinisialisasi dengan nilai acak selama pembuatan model. Pada dasarnya, kami memanfaatkan modul torch.nn yang menyediakan blok bangunan untuk grafik komputasi, seperti nn.Module, nn.Parameter, dan torch.optim. Model regresi linear yang telah dibuat bertujuan untuk menyesuaikan nilai weight dan bias selama pelatihan. Hasil yang didapatkan masih belum akurat karena model masih menggunakan parameter acak pada tahap awal.

- Pelatihan Model

Dalam langkah-langkah pelatihan model regresi linear menggunakan PyTorch, dimulai dengan inisialisasi model, pembuatan fungsi kerugian, dan optimizer. Training loop diimplementasikan untuk melakukan iterasi, prediksi, perhitungan loss, backpropagation, dan pembaharuan parameter model selama 100 epoch. Proses evaluasi dilakukan setiap 10 epoch pada data pengujian. Visualisasi kurva loss membantu memahami performa model dari waktu ke waktu. Hasil pelatihan menunjukkan model yang dapat memprediksi lebih baik, dengan parameter mendekati nilai sebenarnya, mengindikasikan pembelajaran yang baik dari data. Pemahaman konsep pelatihan, evaluasi model, dan pengaruh hyperparameter seperti jumlah epoch dan learning rate adalah kunci dalam mengembangkan model machine learning yang efektif menggunakan PyTorch.

- Membuat prediksi dengan pelatihan model pytorch

Dalam melakukan prediksi (inferensi) dengan model PyTorch, terdapat tiga hal yang penting:

- Atur model dalam mode evaluasi (model.eval()):

  Mode evaluasi menonaktifkan elemen-elemen yang tidak diperlukan untuk inferensi, sehingga meningkatkan kecepatan perhitungan.

- Gunakan konteks manajer inference mode (with torch.inference_mode(): ...):

Melakukan prediksi dalam konteks manajer inference mode memastikan pengaturan yang tidak diperlukan selama inferensi dinonaktifkan, meningkatkan efisiensi perhitungan.

- Semua prediksi sebaiknya dilakukan dengan objek pada perangkat yang sama:

  Pastikan data dan model berada pada perangkat yang sama, entah itu GPU atau CPU, untuk menghindari kesalahan cross-device.

Kesimpulannya, ketiga langkah ini memastikan bahwa selama inferensi, model beroperasi dengan efisien tanpa melakukan perhitungan atau pengaturan yang tidak perlu, yang dapat meningkatkan kecepatan komputasi. Selain itu, memastikan bahwa data dan model berada pada perangkat yang sama mencegah terjadinya kesalahan yang dapat timbul akibat perbedaan perangkat.

- Menyimpan model dan memuat model pytorch

Model yang telah dilatih dapat disimpan dengan menggunakan `torch.save` untuk menyimpan `state_dict()`. Selanjutnya, jika menggunakan model tersebut kembali, model baru dapat dibuat, dan `state_dict()` dari model yang disimpan dimuat ke model baru dengan menggunakan `load_state_dict`. Untuk menyimpan model, langkah-langkahnya melibatkan pembuatan direktori ("models"), pembuatan jalur file untuk menyimpan model, dan penggunaan `torch.save(obj, f)` di mana obj adalah `state_dict()` dari model target, dan f adalah nama file untuk menyimpan model. Untuk memuat model, dilakukan pembuatan instance baru dari model yang akan dimuat, dan `state_dict()` dari model yang telah disimpan dimuat menggunakan `torch.load(f)`, kemudian dimasukkan ke instance baru model dengan `loaded_model.load_state_dict()`.

- Keseluruhan

```
# Import PyTorch and matplotlib
import torch
from torch import nn # nn contains all of PyTorch's building blocks for neural networks
import matplotlib.pyplot as plt

# Check PyTorch version
torch.__version__
```

```
'2.1.0+cu121'
```

Diawali dengan mengimport library yang digunakan yaitu torch dengan versi 2.1 dan matpotlib.

```
# Setup device agnostic code
device = "cuda" if torch.cuda.is_available() else "cpu"
print(f"Using device: {device}")
```

```
Using device: cuda
```

Mengecek device yang digunakan seperti didatas. Dihasilkan device menggunakan GPU karena hasil menunjukan cuda bukan cpu.

```
# Create weight and bias
weight = 0.7
bias = 0.3

# Create range values
start = 0
end = 1
step = 0.02

# Create X and y (features and labels)
X = torch.arange(start, end, step).unsqueeze(dim=1) # without unsqueeze, errors will happen later on (shapes
y = weight * X + bias
X[:10], y[:10]
```

```
(tensor([[0.0000],
         [0.0200],
         [0.0400],
         [0.0600],
         [0.0800],
         [0.1000],
         [0.1200],
         [0.1400],
         [0.1600],
         [0.1800]]),
 tensor([[0.3000],
         [0.3140],
         [0.3280],
         [0.3420],
         [0.3560],
         [0.3700],
         [0.3840],
         [0.3980],
         [0.4120],
         [0.4260]]))
```

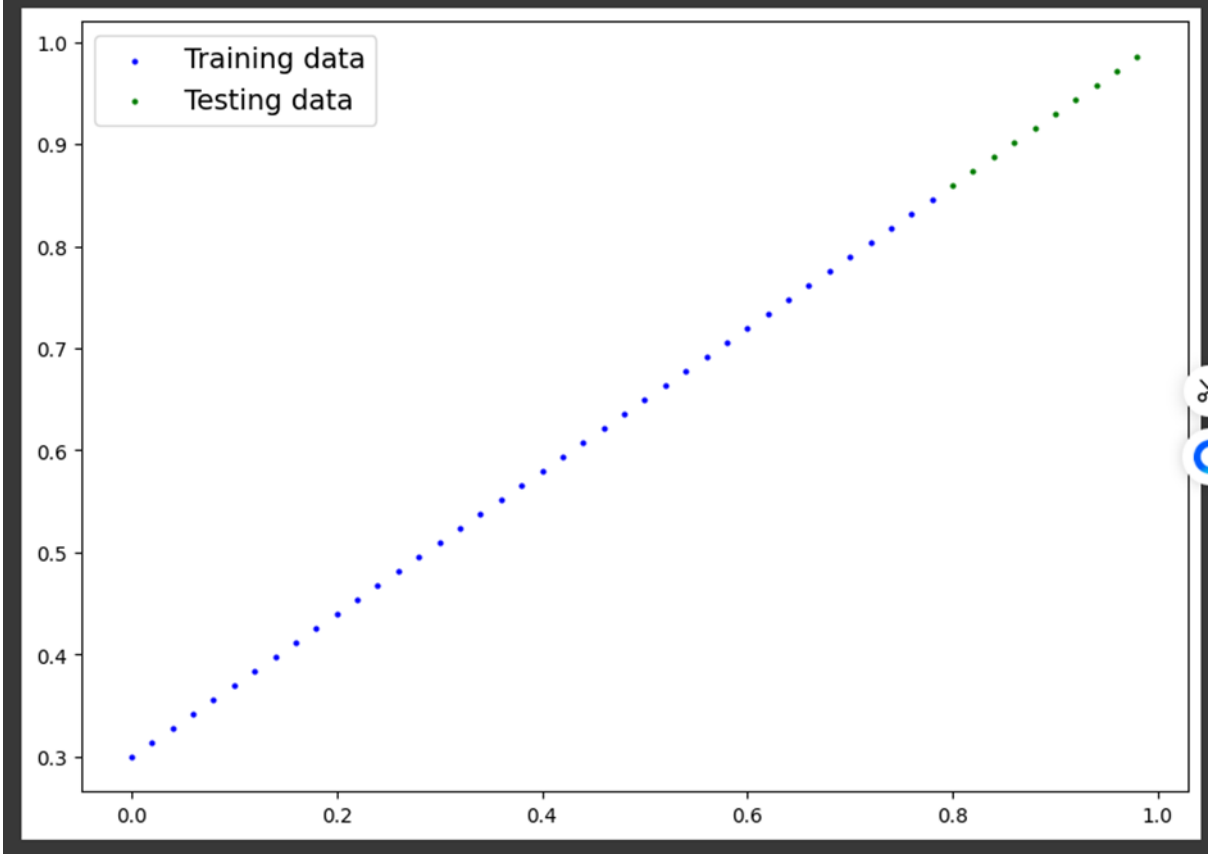
Kode ini membuat data latihan untuk model regresi linear dengan menghasilkan tensor X yang berisi nilai dalam rentang dari 0 hingga 1 dengan selang 0.02, kemudian menghitung tensor y sebagai hasil dari fungsi regresi linear sederhana dengan bobot 0.7 dan bias 0.3. Hasil output menunjukkan sepuluh nilai pertama dari tensor X dan tensor y yang telah dibuat.

```
# Split data
train_split = int(0.8 * len(X))
X_train, y_train = X[:train_split], y[:train_split]
X_test, y_test = X[train_split:], y[train_split:]

len(X_train), len(y_train), len(X_test), len(y_test)
```

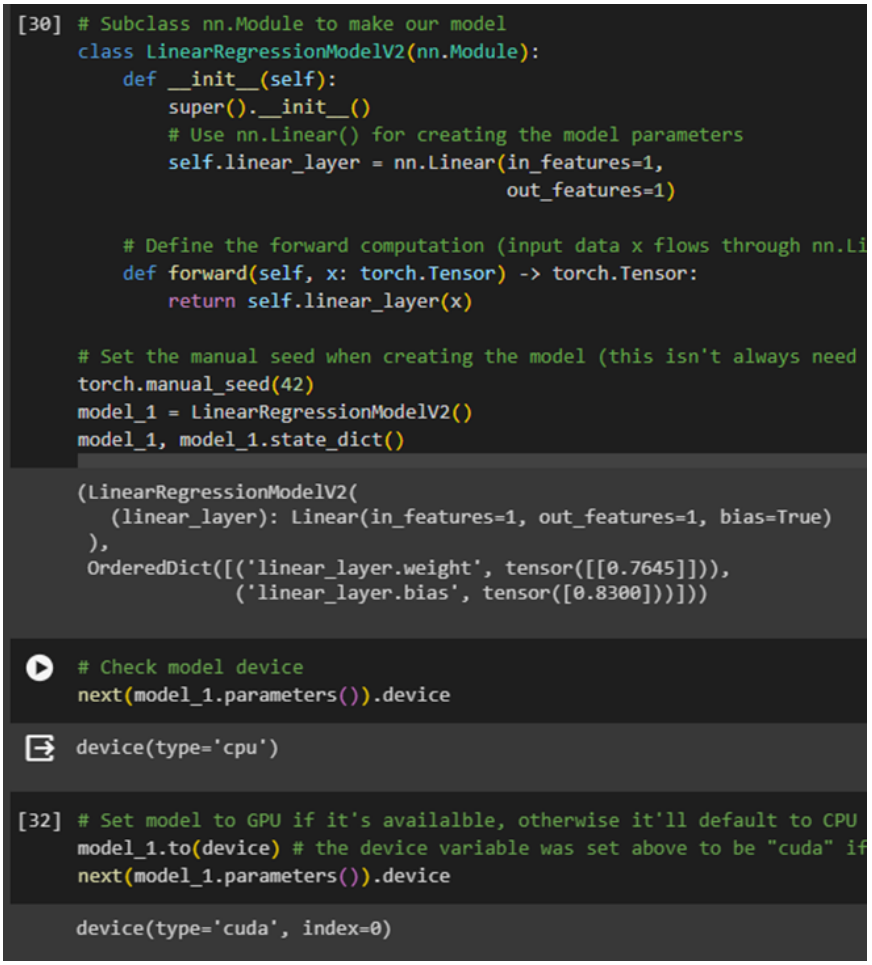
```
(40, 40, 10, 10)
```

Membagi data untuk train dan test masing masing sebesar 80% dan 20%. Dari total 50 data, masing-masing train mendapatkan 40 data dan test mendapatkan sebanyak 10 data.

```
# Note: If you've reset your runtime, this function won't work,
# you'll have to rerun the cell above where it's instantiated.
plot_predictions(X_train, y_train, X_test, y_test)
```



Membuat plot scatter dengan regresi dari data x dan y pada train dan test data.



```
[30] # Subclass nn.Module to make our model
     class LinearRegressionModelV2(nn.Module):
         def __init__(self):
             super().__init__()
             # Use nn.Linear() for creating the model parameters
             self.linear_layer = nn.Linear(in_features=1,
                                           out_features=1)

         # Define the forward computation (input data x flows through nn.Li
         def forward(self, x: torch.Tensor) -> torch.Tensor:
             return self.linear_layer(x)

     # Set the manual seed when creating the model (this isn't always need
     torch.manual_seed(42)
     model_1 = LinearRegressionModelV2()
     model_1, model_1.state_dict()

     (LinearRegressionModelV2(
        (linear_layer): Linear(in_features=1, out_features=1, bias=True)
      ),
      OrderedDict([('linear_layer.weight', tensor([[0.7645]])),
                   ('linear_layer.bias', tensor([0.8300]))]))

     # Check model device
     next(model_1.parameters()).device

     device(type='cpu')

[32] # Set model to GPU if it's availalble, otherwise it'll default to CPU
     model_1.to(device) # the device variable was set above to be "cuda" if
     next(model_1.parameters()).device

     device(type='cuda', index=0)
```

Pembuatan linear model menggunakan pytorch. Kode ini mendefinisikan model regresi linear sederhana (`LinearRegressionModelV2`) sebagai subkelas dari `nn.Module` dalam PyTorch. Model menggunakan `nn.Linear()` untuk membuat lapisan linear dengan satu fitur input dan satu fitur output. Pengaturan seed manual (`torch.manual_seed(42)`) digunakan untuk reproduktibilitas. Setelah membuat instansi model (`model_1`) lalu metode `state_dict()` digunakan untuk menampilkan parameter awal model termasuk nilai bobot dan bias yang diinisialisasikan berdasarkan seed yang ditentukan.

Hasil output menunjukkan arsitektur model, terdiri dari lapisan linear dengan satu fitur input dan satu fitur output. Parameter awal model (bobot dan bias) ditampilkan dalam state_dict sebagai tensor. Dalam contoh ini, bobot diinisialisasi sekitar 0.7645, dan bias diinisialisasi sebesar 0.8300.

```
# Create loss function
loss_fn = nn.L1Loss()

# Create optimizer
optimizer = torch.optim.SGD(params=model_1.parameters(), # optimize newly created m
                            lr=0.01)
```

```
torch.manual_seed(42)

# Set the number of epochs
epochs = 1000

# Put data on the available device
# Without this, error will happen (not all model/data on device)
X_train = X_train.to(device)
X_test = X_test.to(device)
y_train = y_train.to(device)
y_test = y_test.to(device)

for epoch in range(epochs):
    ### Training
    model_1.train() # train mode is on by default after construction

    # 1. Forward pass
    y_pred = model_1(X_train)

    # 2. Calculate loss
    loss = loss_fn(y_pred, y_train)

    # 3. Zero grad optimizer
    optimizer.zero_grad()

    # 4. Loss backward
    loss.backward()

    # 5. Step the optimizer
    optimizer.step()

    ### Testing
    model_1.eval() # put the model in evaluation mode for testing (inference)
    # 1. Forward pass
    with torch.inference_mode():
        test_pred = model_1(X_test)

        # 2. Calculate the loss
        test_loss = loss_fn(test_pred, y_test)

    if epoch % 100 == 0:
        print(f"Epoch: {epoch} | Train loss: {loss} | Test loss: {test_loss}")
```

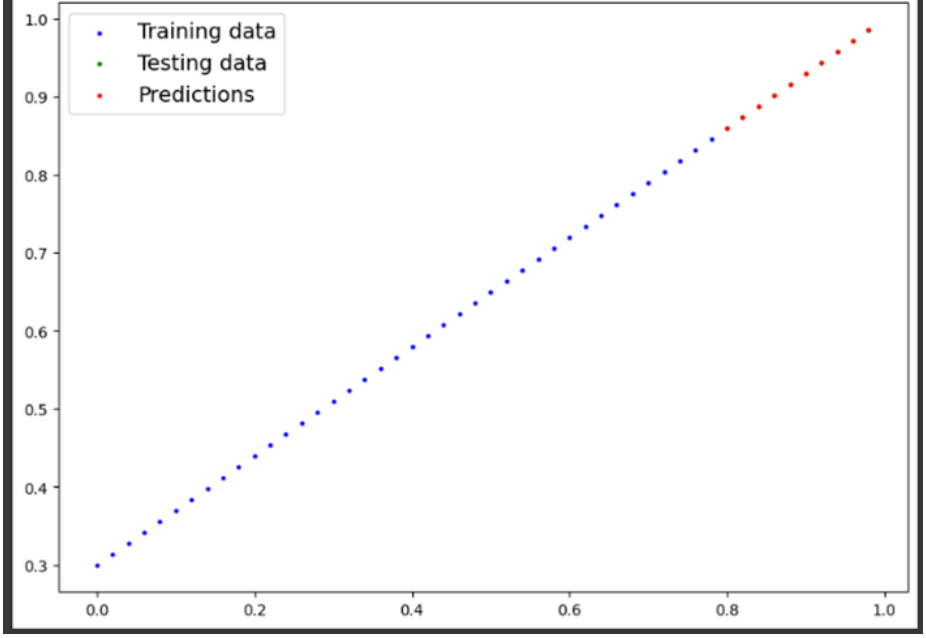
Melakukan training data dengan beberapa percobaan pada epoch.

```
# Turn model into evaluation mode
model_1.eval()

# Make predictions on the test data
with torch.inference_mode():
    y_preds = model_1(X_test)
y_preds
```

```
tensor([[0.8600],
        [0.8739],
        [0.8878],
        [0.9018],
        [0.9157],
        [0.9296],
        [0.9436],
        [0.9575],
        [0.9714],
        [0.9854]], device='cuda:0')
```

Kode ini menunjukkan langkah-langkah membuat prediksi menggunakan model regresi linear (`model_1`) yang telah dibuat sebelumnya. Model diatur ke dalam mode evaluasi dengan `model_1.eval()`, dan prediksi pada data uji (`X_test`) dilakukan menggunakan `torch.inference_mode()`. Hasil prediksi (`y_preds`) ditampilkan sebagai tensor, dengan nilai-nilai yang mewakili output model untuk data uji.

```
# plot_predictions(predictions=y_preds) # -> won't work... data not on CPU

# Put data on the CPU and plot it
plot_predictions(predictions=y_preds.cpu())
```



Hasil pelatihan yang divisualisasikan dengan plot.

```python
from pathlib import Path

# 1. Create models directory
MODEL_PATH = Path("models")
MODEL_PATH.mkdir(parents=True, exist_ok=True)

# 2. Create model save path
MODEL_NAME = "01_pytorch_workflow_model_1.pth"
MODEL_SAVE_PATH = MODEL_PATH / MODEL_NAME

# 3. Save the model state dict
print(f"Saving model to: {MODEL_SAVE_PATH}")
torch.save(obj=model_1.state_dict(), # only saving the state_dict() only sa
           f=MODEL_SAVE_PATH)
```

```
Saving model to: models/01_pytorch_workflow_model_1.pth
```

```python
# Instantiate a fresh instance of LinearRegressionModelV2
loaded_model_1 = LinearRegressionModelV2()

# Load model state dict
loaded_model_1.load_state_dict(torch.load(MODEL_SAVE_PATH))

# Put model to target device (if your data is on GPU, model will have to be
loaded_model_1.to(device)

print(f"Loaded model:\n{loaded_model_1}")
print(f"Model on device:\n{next(loaded_model_1.parameters()).device}")
```

```
Loaded model:
LinearRegressionModelV2(
  (linear_layer): Linear(in_features=1, out_features=1, bias=True)
)
Model on device:
cuda:0
```

```python
# Evaluate loaded model
loaded_model_1.eval()
with torch.inference_mode():
    loaded_model_1_preds = loaded_model_1(X_test)
y_preds == loaded_model_1_preds
```

```
tensor([[True],
        [True],
        [True],
        [True],
        [True],
        [True],
```

Kode ini menggunakan PyTorch untuk membuat model regresi linear sederhana dengan bobot dan bias tertentu. Selain itu, kode tersebut juga mencakup langkah-langkah untuk menyimpan, memuat, dan mengevaluasi model tersebut.