Hands-on with PyTorch/TensorFlow

🔥 PyTorch

1. Define Model

```python
import torch
import torch.nn as nn

class NeuralNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(784, 128)
        self.fc2 = nn.Linear(128, 10)
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.relu(self.fc1(x))
        x = self.fc2(x)
        return x

model = NeuralNet()
```

2. Training Loop

```python
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters())

for epoch in range(epochs):
    for x_batch, y_batch in train_loader:
```

🧮 TensorFlow/Keras

1. Define Model

```python
import tensorflow as tf
from tensorflow import keras

model = keras.Sequential([
    keras.layers.Dense(128, activation='relu',
                          input_shape=(784,)),
    keras.layers.Dense(10, activation='softmax')
])

model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

2. Training

```python
# Simple training with fit()
history = model.fit(
    x_train, y_train,
    batch_size=32,
    epochs=10,
    validation_split=0.2
)
```

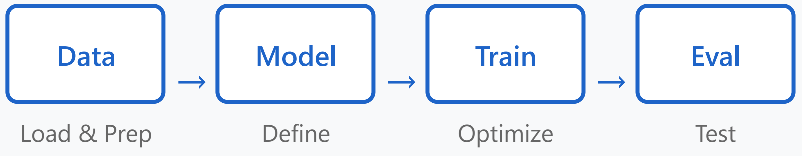
```
        # Forward pass
        outputs = model(x_batch)
        loss = criterion(outputs, y_batch)

        # Backward pass
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

```
# Evaluation
test_loss, test_acc = model.evaluate(
    x_test, y_test
)
```

Key Differences

Style	Pythonic	Keras API
Paradigm	Define-by-run	Sequential
Debugging	Native Python	TensorBoard
Deployment	TorchScript	TF Serving
Learning Curve	Moderate	Easy

Common Workflow



Which to Choose?

PyTorch: Research, experimentation, custom operations, more control

TensorFlow: Production deployment, mobile/edge, high-level API, easier prototyping