24/07/25

LAB-1

EXPLORING DEEP LEARNING PLATFORMS. AND FRAMEWORKS

1. Various Deep Learning Platforms

> Google Colab :

* Creator / Organization : Google Research
* Main Features : Cloud-based notebooks, free GPU, TPU, auto save to Google Drive.
* Popular Use-case : Prototyping, student projects, fast testing.

> Jupyter Notebook :

* Creator / Organization : Project Jupyter.
* Main Features : Interactive coding, supports multiple languages via kernels.
* Popular Use case : Data analysis, education, ML/DL coding.

> Kaggle Kernels :

* Creator / Organization : Kaggle (Google)
* Main Features : Online notebooks with public dataset & competitions
* Popular Use-case : ML competitions, code sharing, fast training.

2. Various Deep Learning Frameworks

> TensorFlow :
* Creator / Organization : Google Brain
* Main Features : Static and dynamic graphs, TensorBoard, TF lite, TF serving
* Popular Use-case : NLP, image classification, production apps.

> PyTorch :
* Creator / Organization : Facebook AI Reasearch (FAIR)
* Main Features : Dynamics graphs, easy debugging
* Popular Use Cases : Research, computer vision, NLP

> KERAS :
* Creator / Organization : Google (TF wrapper)
* Main Features : High-level API over TensorFlow
* Popular use case : Quick model building, education

⇒ Basic Deep Learning Script
> PyTorch $y = 2x$

```
import torch
import torch.nn as nn
import torch.optim as optim

model = nn.Linear(1,1)
Criterion = nn.MSELoss()
Optimizer = optim.SGD(model.parameters(),
                      lr = 0.01)
```

Expected output

Trained weight : tensor([[2.0000]])
Trained bias : tensor([0.0000])

x = torch.tensor([[
y = torch.tensor([[

for epoch in range

y_pred = model(
loss = criterion
optimizer.zero_g
loss.backward()
optimizer.step()

print("Trained weight
print("Trained bias

> TensorFlow y = 2

import tensorFlow
import numpy as

x = np.array([[1.0]
y = np.array([[2.0],

model = tf.keras.S
tf.keras.layers
])

model.compile(op

model.fit(x,y,

```
x = torch.tensor([[1.0], [2.0], [3.0]])
y = torch.tensor([[2.0], [4.0], [6.0]])

for epoch in range(100):
    y_pred = model(x)
    loss = criterion(y_pred, y)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

print("Trained weight:", model.weight.data)
print("Trained bias:", model.bias.data)
```

> TensorFlow $y = 2x$

```
import tensorFlow as tf
import numpy as np

x = np.array([[1.0], [2.0], [3.0]], dtype=np.float32)
y = np.array([[2.0], [4.0], [6.0]], dtype=np.float32)

model = tf.keras.Sequential([
    tf.keras.layers.Dense(units=1, input_shape=[1])
])

model.compile(optimizer='sgd', loss='mean_squared_error')
model.fit(x, y, epochs=100, verbose=0)
```

Expected Output:

Trained weight : [[2.]]
Trained bias : [0.]

```
weights = model.layers[0].get_weights()
print("Trained weight:", weights[0])
print("Trained bias:", weights[1])
```

Conclusion:

Both models successfully learned the linear relationship between x and y.

31/8/26