Assignment 3

June 13, 2025

Assignment-3

#Task1

Code:

```
# Import necessary libraries
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.utils import to_categorical
# Step 1: Load the Iris dataset
iris = load_iris()
X = iris.data                    # Input features (150 samples, 4 features)
y = iris.target                  # Target labels (0: Setosa, 1: Versicolor, 2: Virg
# Step 2: Split the dataset (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size=0.2, random_state=42, stratify=y
)
# Step 3: Feature scaling (standardization)
```

```
scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)

# Step 4: One-hot encoding of target labels
```

$\mathbf{y}_train_encoded = to_categorical(y_train)$
$\mathbf{y}_test_encoded = to_categorical(y_test)$

```
# Print shapes to verify

print("X_train_scaled shape:", X_train_scaled.shape)
```

print("$\mathbf{y}_train_encoded shape:", y_train_encoded.shape$)

```
print("X_test_scaled shape:", X_test_scaled.shape)
```

print("$\mathbf{y}_test_encoded shape:", y_test_encoded.shape$)

Output

```
OUTPUT:

X_train_scaled shape: (120, 4)
```

$\mathbf{y}_train_encoded shape : (120, 3)$

```
X_test_scaled shape: (30, 4)
```

$\mathbf{y}_test_encoded shape : (30, 3)$

Task2:

```
# Import necessary libraries

import tensorflow as tf

from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import Dense
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
```

from sklearn.preprocessing import OneHotEncoder, StandardScaler

```
# Load the Iris dataset
iris = load_iris()
X = iris.data  # 4 input features
y = iris.target.reshape(-1, 1)  # labels
```

One-hot encode the target labels (for softmax output)

encoder = OneHotEncoder(sparse$_output = False)$

$\mathbf{y}_encoded = encoder.fit_transform(y)$

```
# Split into train and test sets
```

$\mathbf{X}_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size = 0.2, random_state = 42)$

```
# Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
# Build the neural network model
model = Sequential()
model.add(Dense(8, input_shape=(4,), activation='relu'))  # Hidden layer with 8
model.add(Dense(3, activation='softmax'))  # Output layer for 3 classes
```

Task 3:

```
# Compile the model
model.compile(optimizer='adam',                    # Adam optimizer
loss='categorical_crossentropy', # Loss for multi-class classification
metrics=['accuracy'])                    # Track accuracy
# Train the model
model.fit(X_train, y_train, epochs=100, batch_size=5, verbose=1)
```

Task-4:

```
# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
# Print the test accuracy
print(f"Test Accuracy: {accuracy * 100:.2f}%")
```

Output

```
Test Accuracy : 96.67%
```

Problem Set 2 :

```
# Install required packages
!pip install annoy --quiet
import torch
import torchvision.models as models
import torchvision.transforms as transforms
from PIL import Image
import os
```

```
from annoy import AnnoyIndex
import numpy as np
import matplotlib.pyplot as plt
from google.colab import files
import shutil
# 1. Upload images
uploaded = files.upload()
os.makedirs("dataset_images", exist_ok=True)
```

for file in uploaded.keys():

```
shutil.move(file, os.path.join("dataset_images", file))
# 2. Upload a query image
query_file = files.upload()
query_image_path = list(query_file.keys())[0]
# 3. Load pretrained ResNet18 and remove last layer
model = models.resnet18(pretrained=True)
model.eval()
feature_extractor = torch.nn.Sequential(*list(model.children())[:-1])
# 4. Define transform
transform = transforms.Compose([
transforms.Resize((224, 224)),
transforms.ToTensor()
])
# 5. Build feature index using Annoy
ann_index = AnnoyIndex(512, 'angular')
image_map = {}
image_list = os.listdir("dataset_images")
```

for idx, image$_{n}ameinenumerate(image_{l}ist)$:

```
img_path = os.path.join("dataset_images", image_name)
img = Image.open(img_path).convert("RGB")
tensor = transform(img).unsqueeze(0)
```

with torch.no$_{g}rad()$:

```
vector = feature_extractor(tensor).squeeze().numpy()
ann_index.add_item(idx, vector)
image_map[idx] = image_name
ann_index.build(10)
# 6. Feature extraction for the query image
query_img = Image.open(query_image_path).convert("RGB")
query_tensor = transform(query_img).unsqueeze(0)
```

with torch.no$_{g}rad()$:

```
query_vector = feature_extractor(query_tensor).squeeze().numpy()
# 7. Find similar images
similar_indices = ann_index.get_nns_by_vector(query_vector, 5)
# 8. Display results
plt.imshow(query_img)
plt.title("Query Image")
plt.axis("off")
plt.show()
```

for idx in similar$_i ndices$:

```
sim_img = Image.open(os.path.join("dataset_images", image_map[idx]))

plt.imshow(sim_img)

plt.title(f"Similar Image: {image_map[idx]}")

plt.axis("off")

plt.show()
```

What I Learnt from this Assignment?

While working out the assignment I understood building a simple neural network to classify flowers from the Iris Dataset.

I also understood searching similar images using PyTorch.