Deep Learning Lab Experiment - 2

Soil Type Classification Using PSA Data

In [1]:

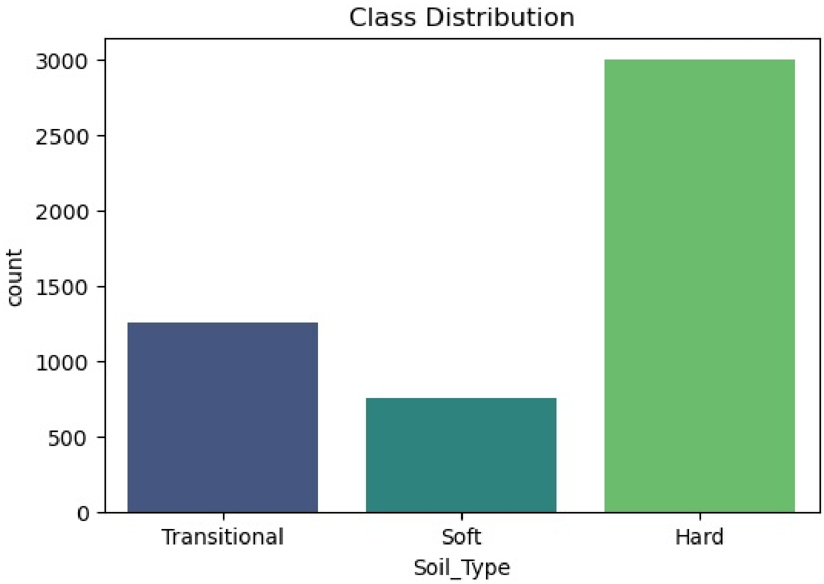
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.decomposition import PCA
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.utils import to_categorical

# Task A: Generate Synthetic Data
def generate_synthetic_psa_data(num_samples=5000, num_features=20):
    np.random.seed(42)
    magnitudes = np.random.uniform(3.0, 8.0, num_samples)  # Earthquake magnitude
    depths = np.random.uniform(5, 100, num_samples)  # Earthquake depth in km
    spectral_ratios = np.random.normal(1.0, 0.3, (num_samples, num_features))  # Simulated spectral ratio data
    soil_classes = np.random.choice(["Hard", "Transitional", "Soft"], num_samples, p=[0.6, 0.25, 0.15])  # Clas

    # Combine into a DataFrame
    data = pd.DataFrame(spectral_ratios, columns=[f"Spectral_Ratio_{i+1}" for i in range(num_features)])
    data["Magnitude"] = magnitudes
    data["Depth"] = depths
    data["Soil_Type"] = soil_classes
    return data

# Generate the synthetic dataset
num_features = 1000  # High-dimensional data for full SR
df = generate_synthetic_psa_data(num_features=num_features)
print(df.head())

# Task A: Data Exploration and Preprocessing
# EDA
print(df.describe())
print(df["Soil_Type"].value_counts())

# Visualize class distribution
plt.figure(figsize=(6, 4))
sns.countplot(data=df, x="Soil_Type", palette="viridis")
plt.title("Class Distribution")
plt.show()

# Data Splitting and Scaling
features = [col for col in df.columns if "Spectral_Ratio" in col] + ["Magnitude", "Depth"]
X = df[features]
y = df["Soil_Type"]

# Encode labels
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Train-test split
X_train, X_temp, y_train, y_temp = train_test_split(X, y_encoded, test_size=0.3, stratify=y_encoded, random_sta
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, stratify=y_temp, random_state=42

# Normalize data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_val = scaler.transform(X_val)
X_test = scaler.transform(X_test)

# Optional PCA (to reduce dimensionality)
pca = PCA(n_components=100)  # Reduce to 100 components
X_train = pca.fit_transform(X_train)
X_val = pca.transform(X_val)
X_test = pca.transform(X_test)

# Convert labels to one-hot encoding for NN
y_train_oh = to_categorical(y_train)
y_val_oh = to_categorical(y_val)
y_test_oh = to_categorical(y_test)

# Task B: Model Architecture & Implementation
model = Sequential([
    Dense(128, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.3),
    Dense(64, activation='relu'),
```

```python
    Dropout(0.3),
    Dense(32, activation='relu'),
    Dense(3, activation='softmax')  # 3 output classes (Hard, Transitional, Soft)
])

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(X_train, y_train_oh, validation_data=(X_val, y_val_oh), epochs=20, batch_size=32, verbose=1

# Plot training history
plt.figure(figsize=(10, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Val Accuracy')
plt.legend()
plt.title("Accuracy")
plt.grid()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.legend()
plt.title("Loss")
plt.grid()
plt.show()

# Task C: Evaluation and Interpretation
# Evaluate on the test set
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)

print("\nClassification Report:")
print(classification_report(y_test, y_pred_classes, target_names=label_encoder.classes_))

print("\nConfusion Matrix:")
cm = confusion_matrix(y_test, y_pred_classes)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=label_encoder.classes_, yticklabels=label_encode
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()
```

```
   Spectral_Ratio_1  Spectral_Ratio_2  Spectral_Ratio_3  Spectral_Ratio_4  \
0          0.553641          0.662444          1.116646          0.647838   
1          0.965029          1.064251          0.707014          1.048081   
2          1.280396          1.110978          0.664802          0.987553   
3          0.722046          0.839937          0.545676          1.164610   
4          1.189775          0.627139          0.319277          0.706878   

   Spectral_Ratio_5  Spectral_Ratio_6  Spectral_Ratio_7  Spectral_Ratio_8  \
0          1.333790          0.978664          1.025678          0.916488   
1          0.872850          0.770317          1.220232          0.538688   
2          1.291147          0.622521          1.190966          1.696783   
3          0.975246          0.488438          0.749551          1.100980   
4          0.927200          1.774088          1.162977          0.457723   

   Spectral_Ratio_9  Spectral_Ratio_10  ...  Spectral_Ratio_994  \
0          1.231854           1.234948  ...            0.677232   
1          1.169433           1.205346  ...            0.935074   
2          1.318545           1.424209  ...            1.157375   
3          0.837093           0.790918  ...            0.916403   
4          0.793103           0.984036  ...            0.768700   

   Spectral_Ratio_995  Spectral_Ratio_996  Spectral_Ratio_997  \
0            0.663041            1.613696            1.159103   
1            0.959073            0.766507            0.866971   
2            0.899180            1.080641            0.546836   
3            0.883753            0.925849            0.900628   
4            0.895460            1.298201            0.987320   

   Spectral_Ratio_998  Spectral_Ratio_999  Spectral_Ratio_1000  Magnitude  \
0            1.091348            0.682746             1.187003   4.872701   
1            0.918919            1.483861             1.283985   7.753572   
2            0.793665            0.796897             0.887683   6.659970   
3            1.083772            0.486947             1.311151   5.993292   
4            0.944156            0.625401             1.235645   3.780093   

       Depth     Soil_Type  
0  42.395374  Transitional  
1  49.976388  Transitional  
2  86.182002          Soft  
3  37.300417          Hard  
```

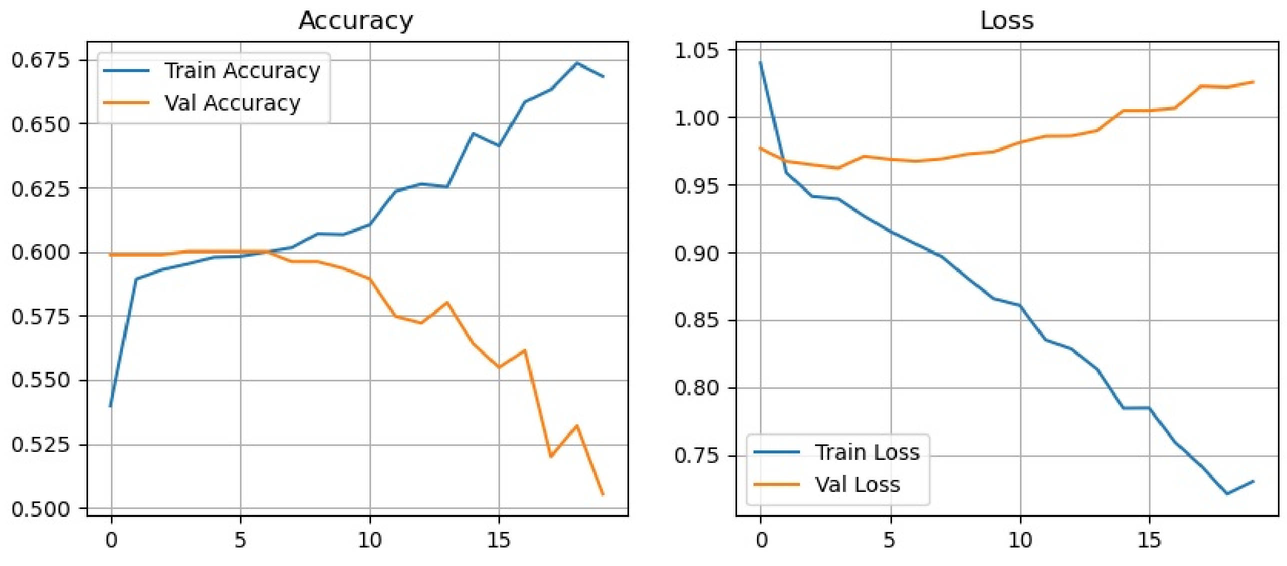
```
4  87.616720          Hard

[5 rows x 1003 columns]
       Spectral_Ratio_1  Spectral_Ratio_2  Spectral_Ratio_3  Spectral_Ratio_4  \
count       5000.000000       5000.000000       5000.000000       5000.000000   
mean           0.999977          0.994361          0.993150          1.011166   
std            0.296223          0.301537          0.300723          0.298434   
min           -0.109789         -0.114431         -0.062554         -0.188176   
25%            0.795829          0.789352          0.789893          0.811696   
50%            0.995803          0.999918          0.993878          1.010109   
75%            1.194240          1.189641          1.193709          1.208912   
max            2.509712          2.062604          2.039908          2.089326   

       Spectral_Ratio_5  Spectral_Ratio_6  Spectral_Ratio_7  Spectral_Ratio_8  \
count       5000.000000       5000.000000       5000.000000       5000.000000   
mean           1.005590          1.001378          1.001662          0.997998   
std            0.301862          0.303413          0.302296          0.295670   
min           -0.052411         -0.109786         -0.142847         -0.242816   
25%            0.804297          0.795059          0.800381          0.799767   
50%            1.006238          1.001247          1.002392          0.993906   
75%            1.212249          1.211692          1.204181          1.196104   
max            2.119253          2.151534          2.510142          2.191137   

       Spectral_Ratio_9  Spectral_Ratio_10  ...  Spectral_Ratio_993  \
count       5000.000000        5000.000000  ...         5000.000000   
mean           0.993018           1.009585  ...            0.989993   
std            0.300113           0.301466  ...            0.303659   
min           -0.324555          -0.082240  ...           -0.086681   
25%            0.791671           0.805814  ...            0.782113   
50%            0.997973           1.008644  ...            0.992796   
75%            1.200695           1.220272  ...            1.197773   
max            2.127435           2.052733  ...            1.949363   

       Spectral_Ratio_994  Spectral_Ratio_995  Spectral_Ratio_996  \
count         5000.000000         5000.000000         5000.000000   
mean             1.002469            0.996621            1.000747   
std              0.300647            0.306510            0.301665   
min             -0.079650           -0.033461           -0.004647   
25%              0.800295            0.785923            0.792222   
50%              0.999575            0.997004            1.000641   
75%              1.204511            1.205941            1.206547   
max              2.037501            2.131030            2.307170   

       Spectral_Ratio_997  Spectral_Ratio_998  Spectral_Ratio_999  \
count         5000.000000         5000.000000         5000.000000   
mean             0.999766            0.987985            0.998090   
std              0.296711            0.297273            0.302160   
min              0.031277           -0.130544           -0.107033   
25%              0.799827            0.795473            0.793046   
50%              1.001291            0.990075            0.995310   
75%              1.201209            1.185036            1.206081   
max              1.950706            2.152775            2.017117   

       Spectral_Ratio_1000    Magnitude        Depth  
count          5000.000000  5000.000000  5000.000000  
mean              0.996545     5.484160    51.691277  
std               0.302621     1.448168    27.133568  
min              -0.107505     3.000058     5.005019  
25%               0.794485     4.219314    28.478861  
50%               0.991353     5.500043    51.167300  
75%               1.196008     6.740504    74.670302  
max               2.134068     7.998588    99.952993  

[8 rows x 1002 columns]
Soil_Type
Hard            2999
Transitional    1251
Soft             750
Name: count, dtype: int64
```

```
C:\Users\shubh\AppData\Local\Temp\ipykernel_23744\2069158784.py:40: FutureWarning: 

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(data=df, x="Soil_Type", palette="viridis")
```

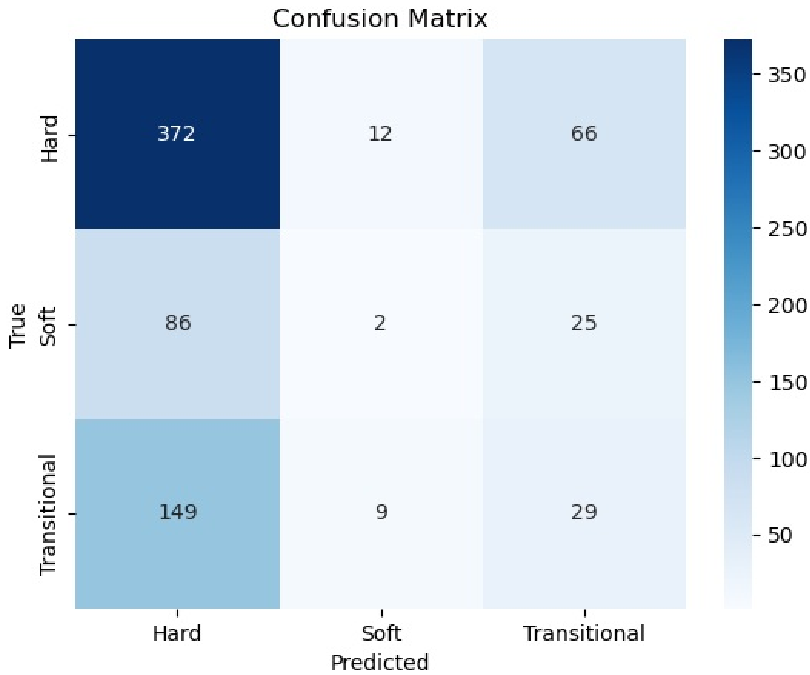



```
C:\Users\shubh\anaconda3\Lib\site-packages\keras\src\layers\core\dense.py:87: UserWarning: Do not pass an `input
_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as
the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
Epoch 1/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 6s 13ms/step - accuracy: 0.4684 - loss: 1.1418 - val_accuracy: 0.5987 - val_loss: 0
.9765
Epoch 2/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - accuracy: 0.5958 - loss: 0.9466 - val_accuracy: 0.5987 - val_loss: 0
.9669
Epoch 3/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.5753 - loss: 0.9591 - val_accuracy: 0.5987 - val_loss: 0.
9643
Epoch 4/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - accuracy: 0.5761 - loss: 0.9550 - val_accuracy: 0.6000 - val_loss: 0
.9619
Epoch 5/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.5954 - loss: 0.9294 - val_accuracy: 0.6000 - val_loss: 0.
9706
Epoch 6/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.6021 - loss: 0.9168 - val_accuracy: 0.6000 - val_loss: 0.
9684
Epoch 7/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.6137 - loss: 0.8887 - val_accuracy: 0.6000 - val_loss: 0.
9670
Epoch 8/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.6032 - loss: 0.8921 - val_accuracy: 0.5960 - val_loss: 0.
9687
Epoch 9/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.5988 - loss: 0.8789 - val_accuracy: 0.5960 - val_loss: 0.
9723
Epoch 10/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.5987 - loss: 0.8707 - val_accuracy: 0.5933 - val_loss: 0.
9740
Epoch 11/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.6056 - loss: 0.8652 - val_accuracy: 0.5893 - val_loss: 0.
9809
Epoch 12/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.6346 - loss: 0.8200 - val_accuracy: 0.5747 - val_loss: 0.
9856
Epoch 13/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.6130 - loss: 0.8410 - val_accuracy: 0.5720 - val_loss: 0.
9858
Epoch 14/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.6472 - loss: 0.7858 - val_accuracy: 0.5800 - val_loss: 0.
9898
Epoch 15/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.6519 - loss: 0.7790 - val_accuracy: 0.5640 - val_loss: 1.
0044
Epoch 16/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.6336 - loss: 0.7877 - val_accuracy: 0.5547 - val_loss: 1.

0044
Epoch 17/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.6732 - loss: 0.7408 - val_accuracy: 0.5613 - val_loss: 1.
0062
Epoch 18/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.6544 - loss: 0.7437 - val_accuracy: 0.5200 - val_loss: 1.
0225
Epoch 19/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.6686 - loss: 0.7230 - val_accuracy: 0.5320 - val_loss: 1.
0217
Epoch 20/20
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.6713 - loss: 0.7272 - val_accuracy: 0.5053 - val_loss: 1.
0255
```



```
24/24 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step

Classification Report:
              precision    recall  f1-score   support

        Hard       0.61      0.83      0.70       450
        Soft       0.09      0.02      0.03       113
Transitional       0.24      0.16      0.19       187

    accuracy                           0.54       750
   macro avg       0.31      0.33      0.31       750
weighted avg       0.44      0.54      0.47       750


Confusion Matrix:
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js