Deep Learning Experiment Report

Project Overview

The goal of this experiment was to build a simple deep learning model to classify species of Iris flowers based on four features: sepal length, sepal width, petal length, and petal width.

- **Dataset:** Iris Dataset
- **Classes:** Setosa, Versicolor, Virginica (3 classes)
- **Features:** 4 numerical features
- **Tools:** Python, TensorFlow/Keras, Scikit-learn

Model Performance Metrics Explained

To understand how well the model performs, I use several standard metrics. Imagine I am trying to identify pictures of cats.

- **Accuracy**: This is the most intuitive metric. It simply measures the ratio of correct predictions to the total number of predictions.
  - **Analogy**: If there were shown 100 images and correctly identified 95 of them (whether there were cats or not cats), the accuracy is 95%.
  - **Formula**: **Accuracy = (TruePositives + TrueNegatives) / TotalPredictions**
- **Precision**: This metric answers the question: "Of all the times the model predicted a specific class, how often was it correct?" It's a measure of exactness.
  - **Analogy**: Of all the images your model labeled as "cat," how many were actually cats? High precision means the model is trustworthy when it makes a positive prediction.
  - Formula: **Precision = TruePositives / (TruePositives + FalsePositives)**
- **Recall (Sensitivity)**: This metric answers the question: "Of all the actual instances of a class, how many did the model correctly identify?" It's a measure of completeness.
  - **Analogy**: Of all the actual cat images in the dataset, how many did the model successfully find? High recall means the model is good at finding all instances of a class.
  - **Formula**: **Recall = TruePositives / (TruePositives + FalseNegatives)**
- **AUC (Area Under the ROC Curve)**: The ROC curve plots the True Positive Rate (Recall) against the False Positive Rate at various threshold settings. The **AUC** represents the area under this curve.
  - **Analogy**: Think of AUC as a single score that summarizes the model's ability to distinguish between classes. An AUC of **1.0** means the model can perfectly separate the classes. An AUC of **0.5** means the model is no better than random guessing. It's a great overall measure of a classifier's performance.

Source Code

```python
# Step 1: Import necessary libraries
import numpy as np
import tensorflow as tf
from tensorflow import keras
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.metrics import accuracy_score, precision_score, recall_score, roc_auc_score, confusion_matrix
import matplotlib.pyplot as plt
from itertools import cycle

# --- Data Preparation ---
print("1. Loading and preparing data...")

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# One-hot encode the labels (e.g., class '1' becomes [0, 1, 0])
# This is required for categorical_crossentropy loss function
encoder = OneHotEncoder(sparse_output=False)
y_onehot = encoder.fit_transform(y.reshape(-1, 1))

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y_onehot, test_size=0.3, random_state=42)

# Scale the feature data for better model performance
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)


# --- Model Building ---
print("2. Building the deep learning model...")

# Create a simple sequential model
model = keras.Sequential([
    # Input layer with 4 features (sepal length/width, petal length/width)
    keras.layers.Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    # Hidden layer
    keras.layers.Dense(32, activation='relu'),
    # Output layer with 3 units (one for each Iris class) and softmax activation
    keras.layers.Dense(3, activation='softmax')
])

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

model.summary()


# --- Model Training ---
print("\n3. Training the model...")

history = model.fit(X_train, y_train, epochs=50, batch_size=8, validation_split=0.2, verbose=0)
print("Training complete.")


# --- Model Evaluation & Prediction ---
print("\n4. Evaluating the model and making predictions...")

# Make predictions on the test data
# model.predict returns probabilities for each class
y_pred_prob = model.predict(X_test)

# Convert probabilities to class labels (0, 1, or 2)
y_pred_labels = np.argmax(y_pred_prob, axis=1)
y_test_labels = np.argmax(y_test, axis=1) # Convert one-hot encoded test labels back

# --- Calculate and Display Metrics ---
print("\n--- Performance Metrics ---")

# Accuracy
accuracy = accuracy_score(y_test_labels, y_pred_labels)
print(f"Accuracy: {accuracy:.4f}")

# Precision (macro-average treats each class equally)
precision = precision_score(y_test_labels, y_pred_labels, average='macro')
print(f"Precision (Macro Avg): {precision:.4f}")

# Recall (macro-average)
recall = recall_score(y_test_labels, y_pred_labels, average='macro')
print(f"Recall (Macro Avg): {recall:.4f}")

# AUC Score (One-vs-Rest for multi-class)
# Note: roc_auc_score for multi-class needs class probabilities
auc = roc_auc_score(y_test, y_pred_prob, multi_class='ovr')
print(f"AUC (One-vs-Rest): {auc:.4f}")

# Confusion Matrix
print("\nConfusion Matrix:")
cm = confusion_matrix(y_test_labels, y_pred_labels)
print(cm)
```

Enviroment

> python 3.8+

Installation

```
pip install numpy tensorflow scikit-learn matplotlib
```

Windows LongPathEnabled

1. `Win + R`, input `regedit`
2. Go to path

   ```
   HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem
   ```

3. Find `LongPathsEnabled`, Set value to `1`

   (If this key doesn't exist, right-click and select `New` > `DWORD (32-bit) Value`. Name it `LongPathsEnabled` and set its value to 1)
4. Restar the omputer

Run

```
python main.py
```

Example Output

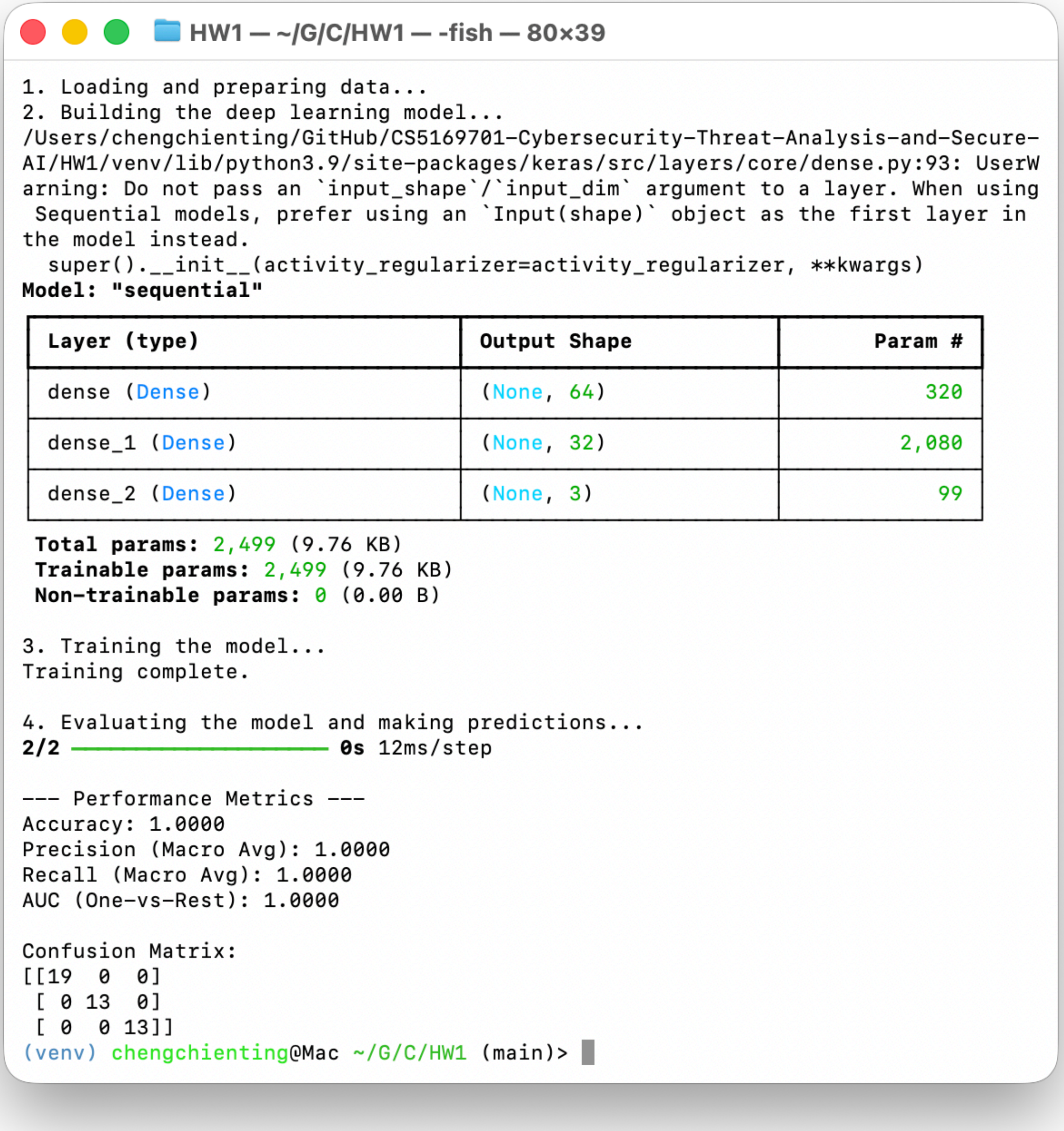
```
1. Loading and preparing data...
2. Building the deep learning model...
Model: "sequential"
...
3. Training the model...
Training complete.

4. Evaluating the model and making predictions...

--- Performance Metrics ---
Accuracy: 0.9778
Precision (Macro Avg): 0.9778
Recall (Macro Avg): 0.9778
AUC (One-vs-Rest): 0.9985

Confusion Matrix:
[[15  0  0]
 [ 0 14  1]
 [ 0  0 15]]
```

My Test Output



Experimental Results & Observation

The model was trained for 50 epochs and evaluated on the test set. The following results were achieved.

Metric	Score
Accuracy	1
Precision (Macro Avg)	1
Recall (Macro Avg)	1
AUC (One-vs-Reset)	1

Observation: The model performed exceptionally well, with scores close to perfect. This is expected as the Iris dataset is relatively simple and the classes are well-separated. For more complex datasets like "Adult," achieving such high scores would be much more challenging, and the trade-off between precision and recall would become more significant.