Assignment #3 Report

Student Information

- **Name**: Tehami Azmat
- **Roll Number**: 24i7616
- **Course**: Deep Learning
- **Course Instructor** : Dr Sajjid Ali Khan
- **Assignment#3**: Hybrid Deep Learning Models

Introduction

In this report, we explore the development and evaluation of two different deep learning models on a classification task, followed by the design of a hybrid model combining the strengths of both. The goal is to assess performance improvements and understand the contribution of individual components through an ablation study. This approach is common in modern deep learning research where ensemble or hybrid architectures often outperform standalone models. This project focuses on building a **Hybrid Deep Learning Model for Human Activity Recognition Using Wearable Sensor Data**. The goal is to classify human activities (like walking, sitting, running, etc.) using time-series data collected from wearable sensors. We design and evaluate two independent deep learning models (CNN and LSTM), and then combine them into a hybrid model that leverages both spatial and temporal feature extraction. Finally, an ablation study is performed to analyze the contribution of each component to the overall performance.

Theoretical Background:

- **Convolutional Neural Networks (CNNs)** are highly effective at extracting spatial patterns and local dependencies from multivariate data.
- **Long Short-Term Memory (LSTM) Networks** excel at capturing temporal dependencies and long-range patterns in sequence data.
- **Hybrid CNN-LSTM Models** combine the advantages of both architectures, first using CNN layers to extract local features, then feeding these into LSTM layers to model temporal relationships.
- **Ablation Study** is a systematic approach where individual components of a system are removed or altered to evaluate their impact on the system's performance.

Dataset Selection and Preprocessing

- **Dataset**: Synthetic dataset with 1,000 samples, 128 timesteps, 9 features, and 6 classes (simulating something like human activity recognition).
- **Preprocessing**: Data split into training (80%) and test (20%) sets; labels one-hot encoded using `to_categorical()`; features normalized as random floats (for demonstration).

Model 1 Implementation (CNN)

- **Architecture**:
  - Conv1D layer (64 filters, kernel size 3, ReLU)
  - MaxPooling1D (pool size 2)
  - Flatten
  - Dense (100 units, ReLU)
  - Output Dense (softmax, 6 classes)
- **Results**:
  - Final Training Accuracy: ~49%
  - Test Accuracy: **18.5%**

Model 2 Implementation (LSTM)

- **Architecture**:
  - LSTM layer (64 units)
  - Dense (100 units, ReLU)
  - Output Dense (softmax, 6 classes)
- **Results**:
  - Final Training Accuracy: ~14–20%
  - Test Accuracy: **15.5%**

Hybrid Model Construction (CNN + LSTM)

- **Architecture**:
  - Conv1D layer (64 filters, kernel size 3, ReLU)
  - MaxPooling1D (pool size 2)
  - LSTM layer (64 units)
  - Dense (100 units, ReLU)
  - Output Dense (softmax, 6 classes)
- **Results**:
  - Final Training Accuracy: ~18–21%
  - Test Accuracy: **19.5%**

Ablation Study

Model Type	Test Accuracy
CNN Only	18.5%
LSTM Only	15.5%
CNN + LSTM Hybrid	19.5%

- **Observation**: The hybrid model slightly outperforms the individual CNN and LSTM models, but overall performance is low, likely due to synthetic/random data.

Report and Conclusion

- **Summary**: We implemented two deep learning models (CNN and LSTM) and combined them into a hybrid architecture. Although the hybrid model improved accuracy slightly, the dataset quality and model tuning heavily affect performance.

- **Challenges**:
  - The synthetic/random dataset limits meaningful learning.
  - Low class-wise performance (as seen in the classification report) suggests model underfitting.
- **Next Steps**:
  - Use a real-world dataset (e.g., UCI HAR dataset or similar time series dataset).
  - Tune hyperparameters (more epochs, learning rates, batch sizes).
  - Add dropout or batch normalization for better regularization.

Hybrid Deep Learning Model for Human Activity Recognition Using Wearable Sensor Data

```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, LSTM, Dense,
Flatten, Dropout
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

# Load the dataset
# For simplicity, we simulate data here (you should replace with
actual loading code)
num_samples = 1000
num_timesteps = 128
num_features = 9
num_classes = 6

X = np.random.rand(num_samples, num_timesteps, num_features)
y = np.random.randint(0, num_classes, num_samples)
y_cat = to_categorical(y, num_classes)

X_train, X_test, y_train, y_test = train_test_split(X, y_cat,
test_size=0.2, random_state=42)

# Model 1: CNN
cnn_model = Sequential([
    Conv1D(64, 3, activation='relu', input_shape=(num_timesteps,
num_features)),
    MaxPooling1D(2),
    Flatten(),
    Dense(100, activation='relu'),
    Dense(num_classes, activation='softmax')
])

cnn_model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
```

```
cnn_model.fit(X_train, y_train, epochs=5, batch_size=32, verbose=1)

cnn_score = cnn_model.evaluate(X_test, y_test, verbose=0)
print(f"CNN Test Accuracy: {cnn_score[1]:.4f}")
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: 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 1/5
25/25 ━━━━━━━━━━━━━━━━━━━━ 2s 13ms/step - accuracy: 0.1899 - loss:
1.8942
Epoch 2/5
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.2136 - loss:
1.7704
Epoch 3/5
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.4041 - loss:
1.7074
Epoch 4/5
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.3720 - loss:
1.6375
Epoch 5/5
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.4898 - loss:
1.5132
CNN Test Accuracy: 0.1850
```

```
# Model 2: LSTM
lstm_model = Sequential([
    LSTM(64, input_shape=(num_timesteps, num_features)),
    Dense(100, activation='relu'),
    Dense(num_classes, activation='softmax')
])

lstm_model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
lstm_model.fit(X_train, y_train, epochs=5, batch_size=32, verbose=1)

lstm_score = lstm_model.evaluate(X_test, y_test, verbose=0)
print(f"LSTM Test Accuracy: {lstm_score[1]:.4f}")
```

```
Epoch 1/5

/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/
rnn.py:200: 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__(**kwargs)
```

```
25/25 ━━━━━━━━━━━━━━━━━━━━ 6s 60ms/step - accuracy: 0.1739 - loss:
1.8006
Epoch 2/5
25/25 ━━━━━━━━━━━━━━━━━━━━ 2s 44ms/step - accuracy: 0.1775 - loss:
1.7901
Epoch 3/5
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 53ms/step - accuracy: 0.1841 - loss:
1.7903
Epoch 4/5
25/25 ━━━━━━━━━━━━━━━━━━━━ 2s 77ms/step - accuracy: 0.2037 - loss:
1.7847
Epoch 5/5
25/25 ━━━━━━━━━━━━━━━━━━━━ 2s 45ms/step - accuracy: 0.1483 - loss:
1.7906
LSTM Test Accuracy: 0.1550
```

```python
# Hybrid Model: CNN + LSTM
hybrid_model = Sequential([
    Conv1D(64, 3, activation='relu', input_shape=(num_timesteps,
num_features)),
    MaxPooling1D(2),
    LSTM(64),
    Dense(100, activation='relu'),
    Dense(num_classes, activation='softmax')
])

hybrid_model.compile(optimizer='adam',
loss='categorical_crossentropy', metrics=['accuracy'])
hybrid_model.fit(X_train, y_train, epochs=5, batch_size=32, verbose=1)

hybrid_score = hybrid_model.evaluate(X_test, y_test, verbose=0)
print(f"Hybrid Model Test Accuracy: {hybrid_score[1]:.4f}")
```

```
Epoch 1/5
25/25 ━━━━━━━━━━━━━━━━━━━━ 4s 35ms/step - accuracy: 0.1516 - loss:
1.8015
Epoch 2/5
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.1984 - loss:
1.7912
Epoch 3/5
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.2122 - loss:
1.7887
Epoch 4/5
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 50ms/step - accuracy: 0.1843 - loss:
1.7898
Epoch 5/5
25/25 ━━━━━━━━━━━━━━━━━━━━ 2s 63ms/step - accuracy: 0.1801 - loss:
1.7882
Hybrid Model Test Accuracy: 0.1950
```

```
# Ablation Study Summary
print("\nAblation Study Results:")
print(f"CNN Only Accuracy: {cnn_score[1]:.4f}")
print(f"LSTM Only Accuracy: {lstm_score[1]:.4f}")
print(f"CNN + LSTM Hybrid Accuracy: {hybrid_score[1]:.4f}")
```

```
Ablation Study Results:
CNN Only Accuracy: 0.1850
LSTM Only Accuracy: 0.1550
CNN + LSTM Hybrid Accuracy: 0.1950
```

```
# Classification report for hybrid model
y_pred = hybrid_model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = np.argmax(y_test, axis=1)
print(classification_report(y_true, y_pred_classes))
```

```
7/7 ━━━━━━━━━━━━━━━━━━━━ 1s 55ms/step
              precision    recall  f1-score   support

           0       0.21      0.42      0.28        38
           1       0.00      0.00      0.00        30
           2       0.00      0.00      0.00        26
           3       0.19      0.58      0.29        33
           4       0.17      0.11      0.13        37
           5       0.00      0.00      0.00        36

    accuracy                           0.20       200
   macro avg       0.09      0.18      0.12       200
weighted avg       0.10      0.20      0.12       200


/usr/local/lib/python3.11/dist-packages/sklearn/metrics/
_classification.py:1565: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classificatio
n.py:1565: UndefinedMetricWarning: Precision is ill-defined and being
set to 0.0 in labels with no predicted samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classificatio
n.py:1565: UndefinedMetricWarning: Precision is ill-defined and being
set to 0.0 in labels with no predicted samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```