In [1]:
```
import numpy as np
import pandas as pd
from tqdm import tqdm
import collections
import time
import random
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_curve, auc
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay
```

In [3]:
```
def set_seeds(seed):
    np.random.seed(seed)
    random.seed(seed)
    tf.random.set_seed(seed)

# Call this function before creating and training your model
set_seeds(42)
```

In [4]:
```
# CONSTANT
LEARNING_RATE_VAL = 0.0001
BATCH_SIZE_VAL = 32
VALIDATION_SPLIT_VAL = 0.2
EPOCH_NUMBER = 50
```

In [5]:
```
early_stopping = EarlyStopping(
    monitor='val_loss',       # Metric to monitor
    patience=1,              # Number of epochs to wait for improvement
    restore_best_weights=True # Restore model weights from the epoch with the best value
)
```

load dataset

In [6]:
```
def load_np_array(file_name):
    X_array = np.load('data/X_' + file_name + '_array.npy')
    y_array = np.load('data/y_' + file_name + '_array.npy')
    return X_array, y_array

X_train_fall, y_train_fall = load_np_array("train_fall")
X_train_notfall, y_train_notfall = load_np_array("train_notfall")
X_test_fall, y_test_fall = load_np_array("test_fall")
X_test_notfall, y_test_notfall = load_np_array("test_notfall")
```

In [7]:
```
print(X_train_fall.shape)
print(y_train_fall.shape)
print(X_train_notfall.shape)
print(y_train_notfall.shape)
```

```
(2912, 40, 3)
(2912,)
(2912, 40, 3)
(2912,)
```

In [8]:
```
print(X_test_fall.shape)
print(y_test_fall.shape)
print(X_test_notfall.shape)
print(y_test_notfall.shape)
```

```
(1456, 40, 3)
(1456,)
(1456, 40, 3)
(1456,)
```

In [9]:
```
y_test_fall
```

Out[9]:
```
array([1, 1, 1, ..., 1, 1, 1])
```

In [10]:
```
# Combine fall and non-fall data for training
X_train = np.concatenate((X_train_fall, X_train_notfall), axis=0)
y_train = np.concatenate((y_train_fall, y_train_notfall), axis=0)
```

```
# Combine fall and non-fall data for testing
X_test = np.concatenate((X_test_fall, X_test_notfall), axis=0)
y_test = np.concatenate((y_test_fall, y_test_notfall), axis=0)

# Shuffle the training data
train_indices = np.arange(X_train.shape[0])
np.random.shuffle(train_indices)
X_train = X_train[train_indices]
y_train = y_train[train_indices]

# Shuffle the testing data
test_indices = np.arange(X_test.shape[0])
np.random.shuffle(test_indices)
X_test = X_test[test_indices]
y_test = y_test[test_indices]

print("Combined training data shape:", X_train.shape, y_train.shape)
print("Combined testing data shape:", X_test.shape, y_test.shape)
```

```
Combined training data shape: (5824, 40, 3) (5824,)
Combined testing data shape: (2912, 40, 3) (2912,)
```

In [11]:

```
# Define the LSTM model
model = Sequential()
model.add(LSTM(256, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dense(32, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))

# Compile the model
model.compile(optimizer=Adam(learning_rate=LEARNING_RATE_VAL), loss='binary_crossentropy', metrics=['accura
```

```
2024-08-05 00:06:12.618695: I metal_plugin/src/device/metal_device.cc:1154] Metal device set to: Apple M1 P
ro
2024-08-05 00:06:12.618717: I metal_plugin/src/device/metal_device.cc:296] systemMemory: 16.00 GB
2024-08-05 00:06:12.618722: I metal_plugin/src/device/metal_device.cc:313] maxCacheSize: 5.33 GB
2024-08-05 00:06:12.618737: I tensorflow/core/common_runtime/pluggable_device/pluggable_device_factory.cc:3
05] Could not identify NUMA node of platform GPU ID 0, defaulting to 0. Your kernel may not have been built
with NUMA support.
2024-08-05 00:06:12.618745: I tensorflow/core/common_runtime/pluggable_device/pluggable_device_factory.cc:2
71] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 0 MB memory) -> physical P
luggableDevice (device: 0, name: METAL, pci bus id: <undefined>)
/opt/miniconda3/envs/tensorflow/lib/python3.10/site-packages/keras/src/layers/rnn/rnn.py:204: 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)
```

In [12]:
```
# Train the model
history = model.fit(X_train, y_train, epochs=EPOCH_NUMBER, batch_size=BATCH_SIZE_VAL, validation_split=VAL
```

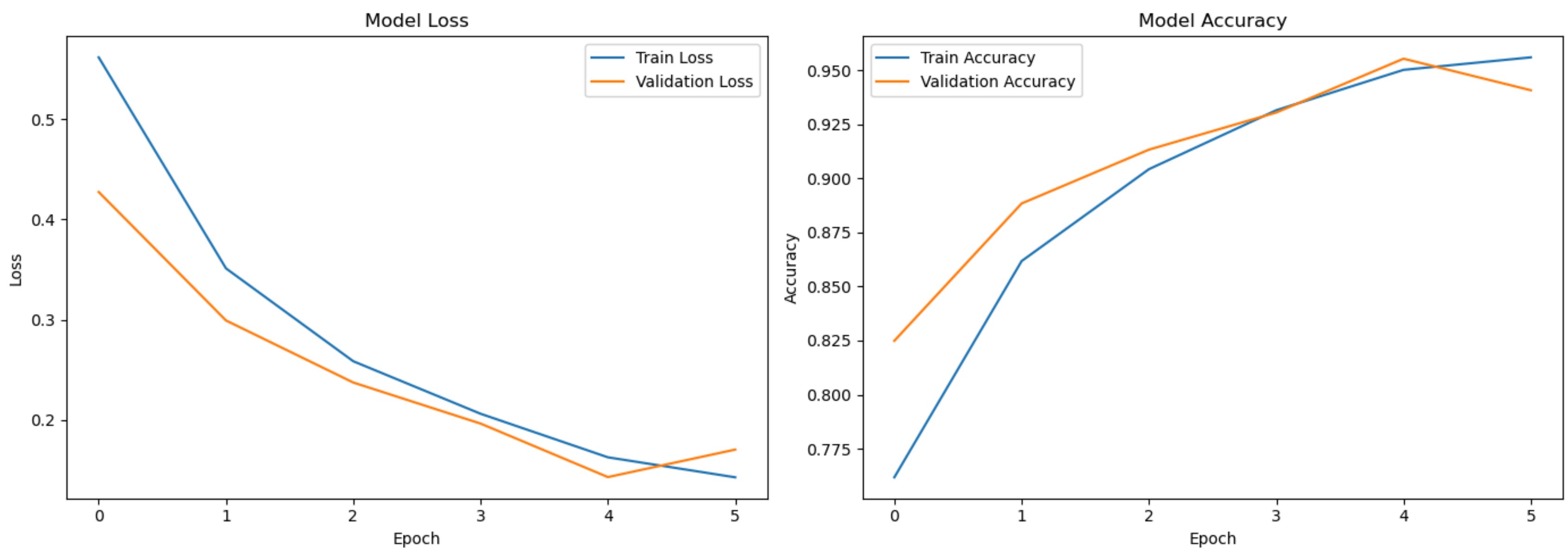
```
Epoch 1/50
2024-08-05 00:06:13.144137: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:117] P
lugin optimizer for device_type GPU is enabled.
146/146 ━━━━━━━━━━━━━━━━━━━━ 4s 19ms/step - accuracy: 0.7234 - loss: 0.6170 - val_accuracy: 0.8249 - val_lo
ss: 0.4272
Epoch 2/50
146/146 ━━━━━━━━━━━━━━━━━━━━ 2s 17ms/step - accuracy: 0.8423 - loss: 0.3851 - val_accuracy: 0.8884 - val_lo
ss: 0.2991
Epoch 3/50
146/146 ━━━━━━━━━━━━━━━━━━━━ 3s 17ms/step - accuracy: 0.8999 - loss: 0.2717 - val_accuracy: 0.9133 - val_lo
ss: 0.2373
Epoch 4/50
146/146 ━━━━━━━━━━━━━━━━━━━━ 3s 17ms/step - accuracy: 0.9283 - loss: 0.2149 - val_accuracy: 0.9305 - val_lo
ss: 0.1963
Epoch 5/50
146/146 ━━━━━━━━━━━━━━━━━━━━ 2s 17ms/step - accuracy: 0.9531 - loss: 0.1642 - val_accuracy: 0.9554 - val_lo
ss: 0.1430
Epoch 6/50
146/146 ━━━━━━━━━━━━━━━━━━━━ 3s 17ms/step - accuracy: 0.9588 - loss: 0.1345 - val_accuracy: 0.9408 - val_lo
ss: 0.1704
```

In [13]:
```
# Plot training & validation loss values
plt.figure(figsize=(14, 5))
```

```
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(loc='upper right')

# Plot training & validation accuracy values
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='upper left')

plt.tight_layout()
plt.show()
```



In [14]:
```
# Predict on the test set
y_pred = (model.predict(X_test) > 0.5).astype("int32")
```

```python
# Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1 Score: {f1:.4f}")
```

```
91/91 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
Accuracy: 0.9361
Precision: 0.9562
Recall: 0.9141
F1 Score: 0.9347
```

In [15]:

```python
# Generate and print classification report
report = classification_report(y_test, y_pred, target_names=['Not Fall', 'Fall'])
print("\nClassification Report:\n", report)
```

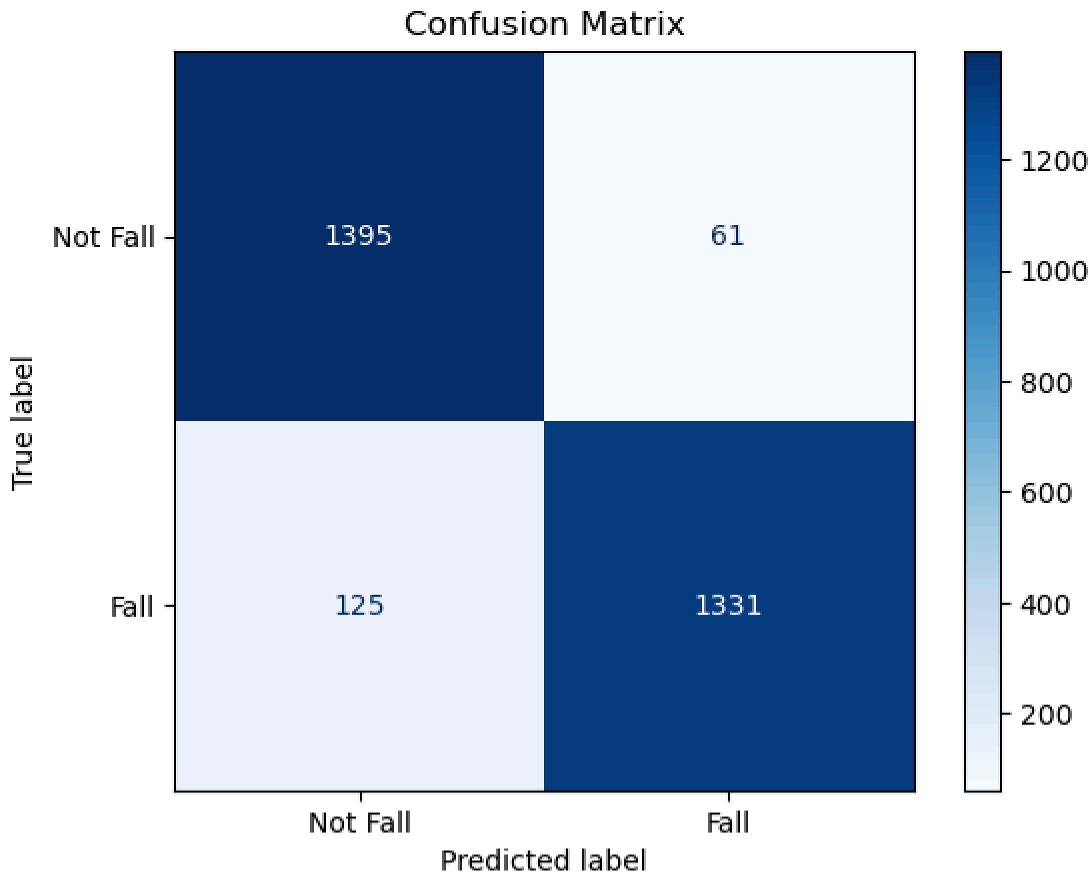
```
Classification Report:
               precision    recall  f1-score   support

    Not Fall       0.92      0.96      0.94      1456
        Fall       0.96      0.91      0.93      1456

    accuracy                           0.94      2912
   macro avg       0.94      0.94      0.94      2912
weighted avg       0.94      0.94      0.94      2912
```

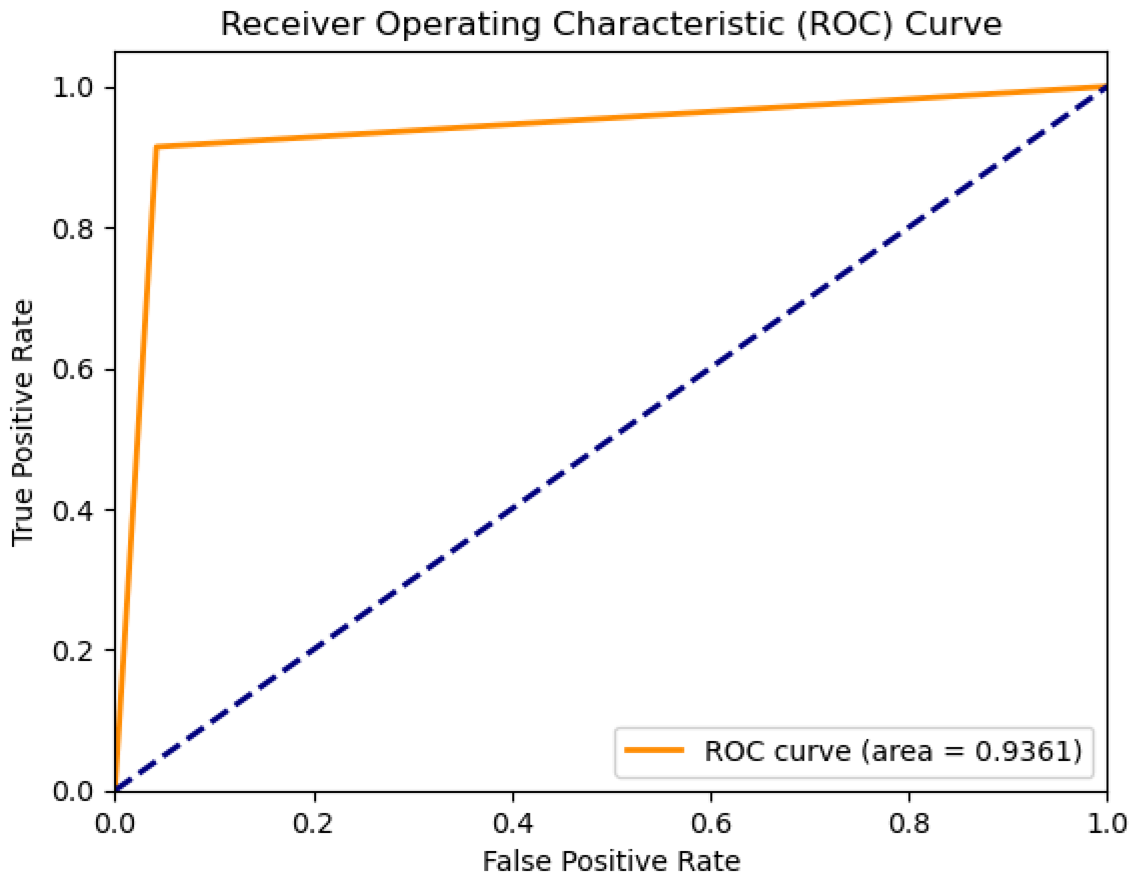
In [16]:

```python
# Compute and plot the confusion matrix
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Not Fall', 'Fall'])
disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.show()
```



In [17]:
```
# Plot ROC curve
fpr, tpr, _ = roc_curve(y_test, y_pred)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.4f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```


Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
ROC curve (area = 0.9361)
0.0
0.2
0.4
0.6
0.8
1.0

In [ ]: