Problem Statement 2 : Building a new Employee Security System

Animesh Singh
animeshs@iitbhilai.ac.in

April 6, 2024

Introduction to the Problem

- Currently the employees use a physical keycard for entry into the building of the company.
- New idea suggests that the employees use smartphone and machine learning to provide a contactless system.
- When an employee enters the firm's territory, his or her smartphone connects to the server and transmits data from the employee smartphone sensor data like the accelerometer's data.
- The server performs the calculations and determines this person as one of the employees using Gait analysis.
- Design and develop a system that will perform the gait analysis.

The Dataset

- The dataset provided is Human Activity Recognition using Smartphones.
- It contains acceleremoter and gyroscope signal data of 30 subjects doing 6 human activities, namely : WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING.
- The dataset includes a rich set of 561 features which have been extracted from signal data.
- The dataset does not have null/missing values and is already split in train and test, although, not useful for our task.
- The dataset also has names of features, activity mapping and subjects who performed the activity mentioned.

Possible Solution Approach : Traditional Machine Learning

- Perform extensive feature engineering to extract relevant features indicative of an individual's gait.
- Utilize machine learning classifiers such as Support Vector Machines (SVM), Random Forest, and Gradient Boosting to build a predictive model.
- Train these models using the engineered features and evaluate their performance based on standard classification metrics.
- Advantages:
  - Well-established techniques with interpretable results.
  - Can handle high-dimensional feature spaces effectively.
  - Generally faster to train and deploy compared to deep learning models.

Selected Solution Approach : Time Series Based Model

- Utilize the raw signal data from the accelerometer and gyroscope sensors.
- Train a time series based model such as Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs), etc.
- These models will learn patterns directly from the sequential nature of the sensor data.
- Evaluate the performance of these models on unseen data.
- Advantages:
    - Can capture complex temporal dependencies present in the sensor data.
    - May require less manual feature engineering compared to traditional methods.
    - Suitable for handling sequential data directly without feature extraction.

Preprocessing Steps Taken

Clean and format the raw sensor data for further analysis.

- **Data Cleaning & Normalization**:
  - Dataset was already not having any null/missing values. Also, values of signals as well as features were already normalized in the dataset
- **Handling Duplicate Features**:
  - Some features had duplicate names and were repeated thrice.
  - It was concluded that this duplication was due to missing axis labels (X, Y, Z) for accelerometer and gyroscope data.
  - Resolved by adding axis labels to the feature names to distinguish between them effectively.

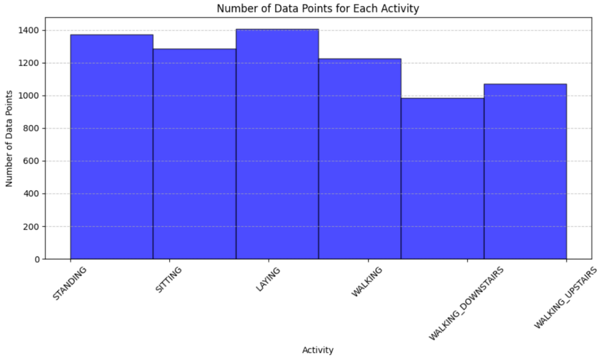
```
features_set=set()
unique_features=[]
for i,v in enumerate(features):
    if v not in features_set:
        features_set.add(v)
        unique_features.append(v+'-X')
    elif v+'-Y' not in features_set:
        features_set.add(v+'-Y')
        unique_features.append(v+'-Y')
    else:
        features_set.add(v+'-Z')
        unique_features.append(v+'-Z')

print(len(unique_features))
```

Preprocessing Steps Taken

- **Data Splitting**:
  - The dataset was initially split into train and test sets for Human Activity Recognition.
  - However, since our aim is gait analysis, this split was not applicable.
  - The test set contained data from 9 out of 30 people, whose data was completely absent from the train set.
  - Concatenated the train and test sets, shuffled them to remove any sequential bias, and then performed the split.
- **Label Encoding**:
  - As the data was categorical in nature, the labels were one hot encoded to represent different activities.
  - This encoding enables the model to understand and learn from the categorical data effectively.

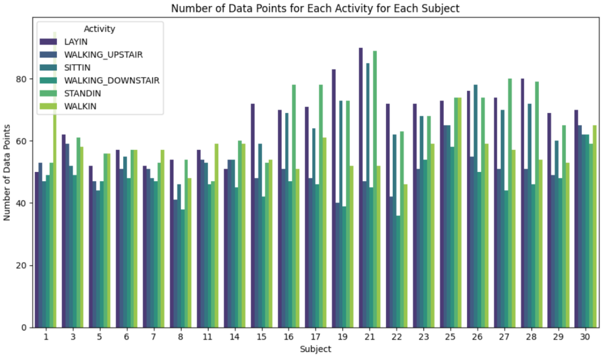
Number of Data Points vs Activity

- To ensure no activity bias, a graph was plotted showing the distribution of data points across different activities.
- This helps in assessing if there's an imbalance in the dataset, which could affect the model's performance.



Subject-wise Data Distribution:

- Another graph was plotted to visualize the distribution of data points per activity for each subject.
- This analysis helps identify if certain subjects have disproportionately more data for specific activities, which could introduce bias in the model.



Model Definition & Parameters

- Model parameters such as the number of hidden units in LSTM model, learning rate, batch size and epochs were decided based on experimentation with different values.
- Tuning these parameters is crucial to achieve optimal performance of the time series model.

```
num_classes = 30
num_epoch= 100
batch= 32
lr=0.005
```

```
model = Sequential([
    LSTM(128, input_shape=(128, 9), kernel_initializer=initializer),
    Dense(num_classes, activation='softmax', kernel_initializer=initializer)
])

optimizer = Adam(learning_rate=lr)
model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
lr_scheduler_callback = LearningRateScheduler(lr_scheduler)
```

Initializer & Learning Rate Scheduler

- Glorot initializer was used to initialize the model parameters.
- Glorot initializer helps in maintaining the variance of activations and gradients throughout the training process, preventing issues like vanishing or exploding gradients.
- Learning rate scheduler was employed to dynamically adjust the learning rate during training.
- It helps in controlling the speed and stability of the learning process.
- In our case, the learning rate was decreased by 0.7 times its value every 10 epochs to gradually fine-tune the model as it converges.

```
initializer = GlorotUniform()

def lr_scheduler(epoch, lr):
    if epoch % 10 == 0 and epoch != 0:
        return lr * 0.7
    else:
        return lr
```

Accuracy and Metrics

- The model achieved an accuracy of 94% in both the training and test sets, indicating robust performance.
- Additionally, the following metrics were calculated:
  - Precision: Measure of the model's ability to correctly identify positive cases.
  - Recall: Measure of the model's ability to capture all positive cases.
  - F1 Score: Harmonic mean of precision and recall, providing a balanced measure of the model's performance.

Accuracy and Metrics

```
loss, accuracy = model.evaluate(X_train, y_train)
print(f'Training Loss: {loss}, Training Accuracy: {accuracy}')
```

```
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.9454 - loss: 0.1824
Training Loss: 0.19289769232273102, Training Accuracy: 0.9445967674255371
```

```
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Test Loss: {loss}, Test Accuracy: {accuracy}')
```

```
49/49 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9403 - loss: 0.1979
Test Loss: 0.18010137975215912, Test Accuracy: 0.9430420994758606
```


```
loss, accuracy = model.evaluate(X_train, y_train)
print(f'Training Loss: {loss}, Training Accuracy: {accuracy}')
```

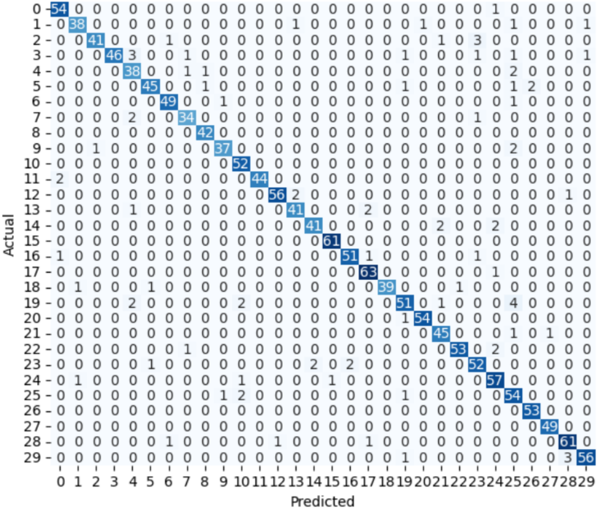
```
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.9454 - loss: 0.1824
Training Loss: 0.19289769232273102, Training Accuracy: 0.9445967674255371
```

```
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Test Loss: {loss}, Test Accuracy: {accuracy}')
```

```
49/49 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9403 - loss: 0.1979
Test Loss: 0.18010137975215912, Test Accuracy: 0.9430420994758606
```

Confusion Matrix

- It offers insights into the model's ability to accurately classify different classes.



Conclusion

- The deep learning model for development of a gait analysis system using smartphone sensor data was successfully implemented.
- By leveraging raw sensor data and employing long short-term memory networks (LSTMs), we were able to capture complex temporal dependencies present in gait patterns.
- The model demonstrated robust performance with an accuracy of 94% on both the training and test sets, indicating its reliability in real-world scenarios.
- Some steps need to be done to make this a fully fledged security system :
    1. Develop APIs or endpoints to receive sensor data from employees' smartphones.
    2. Implement real-time data pre-processing before feeding the data to the trained model, as employees enter the company's premises.
    3. Integrate it with the trained model and thoroughly test the entire system before deployment.