Musconet

Ayush Choudhary[1],Mahesh Babu Vanapartrhi[2]
University of Colorado Denver[1], University of Colorado Denver[2]

Abstract

This project focuses on studying electromyography (EMG) data of the lower limb from individuals having normal and abnormal knee conditions. We have used autoencoders, an unsupervised learning approach, to extract distinct features from the EMG signals of both the groups. By leveraging these extracted features, we aim to develop methods to differentiate between individuals with normal knee function and those that could potentially have knee abnormalities. The analysis and findings from this study provide valuable insights into the potential application of autoencoders in identifying and distinguishing lower limb EMG patterns associated with different knee conditions.

1. Introduction

Electromyography is a great way to study the working of muscles during various movements. In our project, we study EMG data from individuals with normal knee function and those having certain knee abnormalities. The objective is to use a semi-labelled dataset and then utilize autoencoders, an unsupervised learning strategy, to extract distinctive features from the EMG signals of these two groups.

Autoencoders are a class of neural networks that are mainly used for dimensionality reduction and feature extraction. By training an autoencoder on the EMG data from both normal and abnormal knee conditions, we aim to uncover latent representations or features that capture the underlying patterns specific to each group. These extracted features can serve as discriminative markers to differentiate between individuals with normal knee function and those with knee abnormalities.The subsequent analysis of these extracted features will allow us to gain insights into the distinguishing characteristics of EMG signals associated with different knee conditions. We plan to develop robust methods to differentiate between people having a normal knee and one that has issues.

We aim to contribute to the advancement of diagnostic techniques and treatment strategies for knee-related disorders.The human knee is a complex joint responsible for various movements, such as extension, flexion, and rotation. Knee-related conditions, such as ligament injuries, muscle imbalances, or degenerative diseases, can significantly affect an individual's quality of life and mobility. Accurate and early detection of these abnormalities is crucial for effective diagnosis and intervention.

In this report, we present our approach to extracting and analyzing features from lower limb EMG data using autoencoders. We discuss the methodology, data collection process, model training, and evaluation techniques employed in this study. Additionally, we present and interpret the results obtained from comparing the extracted features of individuals with normal knee function and those with knee abnormalities.

2. Methodology

The different phases of the project are explained below-

- **Data Collection** - I took the dataset from UC Irvine’s Machine Learning Repository. The dataset is named EMG dataset in lower limb data set. It has been donated by Dr. Oscar Fernando Aviles Sanchez, Universidad MIlitar Nueva Granada, Colombia.

- **Data Preprocessing** - The data I collected was in txt files. I created a script that took in all the txt files, removed the header, added the header that we required and then converted the files to csv. All this is being done in the script csv_Converter.py[1]. Then column 5 was dropped as well for simplicity but we will use that for future work.

- **Autoencoder architecture and model training** - We are using an overcomplete autoencoder, which is diverging first then converging to create a bottleneck and then it diverges again. The input and output dimensions are equal, so that the model tries to recreate the input again. I then extract the first layer of the encoder part of the autoencoder (64-neuron layer) and save it as a model. All this is done in autoEnc_Normal_Knee.ipynb[2] and autoEnc_Abnormal_Knee.ipynb[3].

- **Feature extraction and analysis** - Here we are extracting features using the layer that we had stored from the encoder part of the autoencoder. I have then compared the features (gathered from the readings of the biceps femoris and rectus femoris in one case and biceps femoris and Semitendinosus for the second case). I have plotted graphs comparing good knee march v/s abnormal knee march for the first case, i.e comparison between the same exercise and then I have compared Good Knee march v/s Abnormal Knee extension in the second case, i.e comparison between two different exercises. All of this is being done in the two codes marching_Extension_Comparion_Rf_Ss.ipynb[4] and marching_feature_Comparion_Rf_Bf.ipynb[5] respectively.

3. Implementation Discussion

The file csv_Converter.py is converting the txt files to csv files and also giving a column header to each column.

```
column_names = ['RF', 'BF', 'VM', 'ST', 'FX']

# Prepare the output CSV file
csv_file = txt_file.replace('.txt', '.csv')
csv_path = os.path.join(output_dir, csv_file)
```

Fig.1 - Snippet of code from csv_Converter.py. The column headers are short for the muscle names.

The script file_Divider.py is creating separate folders for the 11 patients and then placing their 3 csv files into them.

```
for file_name in os.listdir(source_path):
    prefix = file_name.split("A")[0]
    folder_name = prefix
    folder_path = os.path.join(source_path, folder_name)
    os.makedirs(folder_path, exist_ok=True)
```

```
# Move files to respective subfolders based on their starting number
for file_name in os.listdir(source_path):
    prefix = file_name.split("A")[0]
    folder_name = prefix
    source_file_path = os.path.join(source_path, file_name)
    destination_folder_path = os.path.join(source_path, folder_name)
    shutil.move(source_file_path, destination_folder_path)
```

Fig. 2 - Snippet on left shows folder creation and that on the right is placing the files into the folders. Done in file_Divider.py[6].

Now, explanation for autoEnc_Normal_Knee.ipynb and autoEnc_Abnormal_Knee.ipynb -

```
patient_folders = [folder for folder in os.listdir(main_directory)]

march_data = []  # Initialize empty data frame for march data
leg_extension_data = []  # Initialize empty data frame for leg extension data
knee_flexion_data = []  # Initialize empty data frame for knee flexion data

for patient_folder in patient_folders:
    patient_path = os.path.join(main_directory, patient_folder)  # Path to the patient folder

    march_files = [file for file in os.listdir(patient_path) if file.__contains__("mar")]
    march_patient_data = pd.concat([pd.read_csv(os.path.join(patient_path, file)) for file in march_files]) # Load CSV
    march_data.append(march_patient_data)  # Append patient data to march data frame

    leg_extension_files = [file for file in os.listdir(patient_path) if file.__contains__("pie")]
    leg_extension_patient_data = pd.concat([pd.read_csv(os.path.join(patient_path, file)) for file in leg_extension_fi
    leg_extension_data.append(leg_extension_patient_data)  # Append patient data to leg extension data frame

    knee_flexion_files = [file for file in os.listdir(patient_path) if file.__contains__("sen")]
    knee_flexion_patient_data = pd.concat([pd.read_csv(os.path.join(patient_path, file)) for file in knee_flexion_file
    knee_flexion_data.append(knee_flexion_patient_data)  # Append patient data to knee flexion data frame

march_data = pd.concat(march_data)  # Concatenate all march data frames
leg_extension_data = pd.concat(leg_extension_data)  # Concatenate all leg extension data frames
knee_flexion_data = pd.concat(knee_flexion_data)  # Concatenate all knee flexion data frames
```

Fig. 3 - The folders have been mounted on my G drive. I then just extract all of the data for the 3 exercises into 3 respective Pandas data frames.

```
column_index = 4
if column_index < len(g_march_data.columns):
    g_march_data = g_march_data.drop(g_march_data.columns[column_index], axis=1)

if column_index < len(g_leg_extension_data.columns):
    g_leg_extension_data = g_leg_extension_data.drop(g_leg_extension_data.columns[column_index], axis=1)

if column_index < len(g_knee_flexion_data.columns):
    g_knee_flexion_data = g_knee_flexion_data.drop(g_knee_flexion_data.columns[column_index], axis=1)

g_march_data_train,g_march_data_test=train_test_split(g_march_data,test_size=0.2)
g_leg_extension_data_train,g_leg_extension_data_test=train_test_split(g_leg_extension_data,test_size=0.2)
g_knee_flexion_data_train,g_knee_flexion_data_test=train_test_split(g_knee_flexion_data,test_size=0.2)
```

Fig. 4 - Fourth column is dropped and the training and testing data is split on a 80-20 split.

```
num_features=4

input_shape = (num_features,)

input_data = Input(shape=input_shape)
encoded = Dense(64, activation='elu')(input_data)
encoded = Dense(32, activation='elu')(encoded)
encoded = Dense(16, activation='elu')(encoded)


decoded = Dense(32, activation='elu')(encoded)
decoded = Dense(64, activation='elu')(decoded)
decoded = Dense(num_features, activation='linear')(decoded)

# Create the autoencoder model
autoencoder = Model(input_data, decoded)

# Compile the model
autoencoder.compile(optimizer='adam', loss='mse')

# Train the autoencoder using your good EMG data
autoencoder.fit(g_march_data_train, g_march_data_train, epochs=10, batch_size=32)

# Create a new model for feature extraction
encoder_model = Model(inputs=autoencoder.input, outputs=autoencoder.get_layer('dense_60').output)

encoder_model.save('encoder_model.h5')
files.download('encoder_model.h5')
```

Fig. 5 - Architecture of the autoencoder, training and saving of the model.

Given below is the explanation for the two files marching_Extension_Comparion_Rf_Ss.ipynb and marching_Feature_Comparison_Rf_Bf.ipynb -

```
model_normal_knee_march = load_model("/content/drive/MyDrive/Models/encoder_model_march.h5")
model_normal_knee_extension = load_model("/content/drive/MyDrive/Models/encoder_model_Leg_Extension.h5")
model_normal_knee_flexion = load_model("/content/drive/MyDrive/Models/encoder_model_Knee_flexion.h5")

model_abnormal_knee_march = load_model("/content/drive/MyDrive/Models/encoder_model_bad_march.h5")
model_abnormal_knee_extension = load_model("/content/drive/MyDrive/Models/encoder_model_bad_leg_extension.h5")
model_abnormal_knee_flexion = load_model("/content/drive/MyDrive/Models/encoder_model_bad_knee_flexion.h5")
```

Fig. 6 - Loading the stored models.

```
encoder_model = Model(inputs=model_normal_knee_march.input, outputs=model_normal_knee_march.layers[1].output)
features_normal_knee_march = encoder_model.predict(g_march_data_test)

encoder_model_2 = Model(inputs=model_normal_knee_extension.input, outputs=model_normal_knee_extension.layers[1].output)
features_normal_knee_extension = encoder_model_2.predict(g_leg_extension_data_test)

encoder_model_3 = Model(inputs=model_normal_knee_flexion.input, outputs=model_normal_knee_flexion.layers[1].output)
features_normal_knee_flexion = encoder_model_3.predict(g_knee_flexion_data_test)

encoder_model_4 = Model(inputs=model_abnormal_knee_march.input, outputs=model_abnormal_knee_march.layers[1].output)
features_abnormal_knee_march = encoder_model_4.predict(b_march_data_test)

encoder_model_5 = Model(inputs=model_abnormal_knee_extension.input, outputs=model_abnormal_knee_extension.layers[1].output)
features_abnormal_knee_extension = encoder_model_5.predict(b_leg_extension_data_test)

encoder_model_6 = Model(inputs=model_abnormal_knee_flexion.input, outputs=model_abnormal_knee_flexion.layers[1].output)
features_abnormal_knee_flexion = encoder_model_6.predict(b_knee_flexion_data_test)
```

Fig. 7 - Feature extraction.

```
plt.scatter(features_normal_knee_march[:, 0], features_normal_knee_march[:, 3],label='Good Knee March')
plt.scatter(features_abnormal_knee_flexion[:, 0], features_abnormal_knee_flexion[:, 3],label='Abnormal Knee extension')
plt.xlabel('Feature of Rectus Femoris')
plt.ylabel('Feature of Semitendinosus')
plt.title('Comparison of Features: Good Knee March vs. Abnormal Knee extension')
plt.legend()
plt.show()
```

Fig. 8 - Creating plots and comparisons.

4. Results

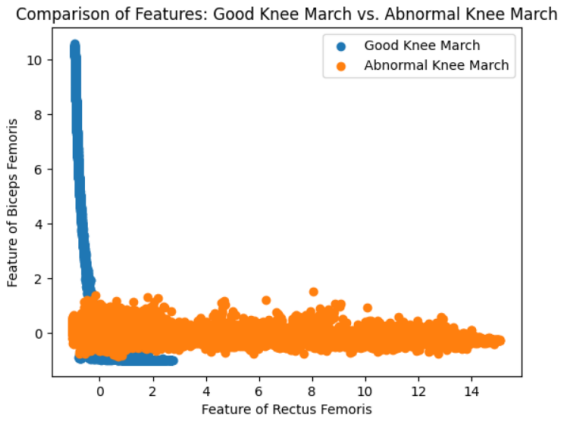

Fig. 9 - Comparison of Good Knee march and the abnormal knee march.

Here, we can see that there's two separate clusters that are getting created for the two groups. These clusters are clearly distinct and help us differentiate between the groups of patients.

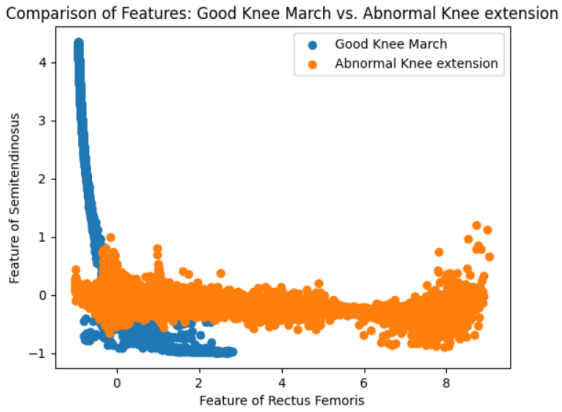

Fig. 10 - Comparison of Good knee march with abnormal knee extension.

Here again we can see two different clusters that are being created for the two groups. From this we can infer that our model is able to differentiate between the two groups of patients even while comparing different exercises.

5. Limitations

- **Dataset related limitations** - The dataset that we used had 5-channel data for the limb, which limited our study to very few comparisons. Also, the dataset wasn't properly labeled, which created issues while training and limited our approach. The number of patients were only 22, which could be increased to have more diverse data that could in turn develop more robust models.
- **Dependency on feature extraction and autoencoders** - In our proposed model, we have used autoencoders. Instead, we can use different machine learning approaches and can compare them to filter out the optimum model. This would have created a better model overall.
- **Limited scope of analysis** - We are analyzing the patients over only 3 exercises, if we had a pool of more exercises, we would have had a more diverse dataset and would have built a more robust model.
- **Lack of Ground Truth Validation** - The project's findings are based on unsupervised learning using autoencoders. While the extracted features can differentiate between healthy and abnormal patients, there might be a lack of ground truth validation or external criteria to verify the accuracy of the classification.

6. Future Work

- **Extension to multiple joints and body parts** - We want to extend our project to multiple joints and movements in the future.
- **Use of multiple machine learning models** - For future work, we want to use multiple machine learning strategies and compare their results to figure out the best one.
- **Fine-tuning** - Using the most optimized hyper-parameters, implementing different architectures, implementing techniques like dropouts is a part of future work in order to build robust models.

7. Conclusion

In summary, this project utilized autoencoders to compare and analyze EMG data from healthy individuals and those with knee abnormalities. By extracting features from the EMG data using the encoder part of the autoencoder models, we were able to capture essential patterns and characteristics in the signals. These extracted features provided valuable insights into the differences between the two groups, enabling us to develop potential ways to differentiate healthy and abnormal patients based on their EMG data.

While the project demonstrated promising results, it is important to acknowledge its limitations. The project was conducted using a limited dataset, which may restrict the generalizability of the findings. Additionally, the classification of abnormalities relied on subjective criteria and further refinement is needed to ensure robustness and accuracy. Interpretability of the extracted features and their clinical relevance also pose challenges that warrant further investigation.

Despite these limitations, this project serves as a foundation for future research in the analysis of EMG data and the application of autoencoders for feature extraction. By addressing the limitations and refining the methodology, researchers can build upon our findings to develop more advanced and accurate methods for diagnosing and understanding knee-related abnormalities. The project's outcomes hold great potential for improving clinical decision-making, treatment planning, and rehabilitation strategies for individuals with knee issues.

8. Code Repository

1.https://github.com/rippedtuco/Musconet/blob/20e54e381d33e77bf02a350e3083a41b9a7f7aae/csv_Converter.py
2.https://github.com/rippedtuco/Musconet/blob/20e54e381d33e77bf02a350e3083a41b9a7f7aae/autoENC_Normal_Knee.ipynb
3.https://github.com/rippedtuco/Musconet/blob/20e54e381d33e77bf02a350e3083a41b9a7f7aae/autoEnc_Abnormal_Knee.ipynb
4.https://github.com/rippedtuco/Musconet/blob/20e54e381d33e77bf02a350e3083a41b9a7f7aae/marching_Extension_Comparion_Rf_Ss.ipynb
5.https://github.com/rippedtuco/Musconet/blob/20e54e381d33e77bf02a350e3083a41b9a7f7aae/marching_Feature_Comparison_Rf_Bf.ipynb
6.https://github.com/rippedtuco/Musconet/blob/20e54e381d33e77bf02a350e3083a41b9a7f7aae/file_Divider.py