**Chapter 5**

**Problem 5: EMG Signal**

The brain sends an electrical signal to specific muscles in order to perform a physical action. This signal is transmitted throughout the body by the nervous system. Each specific action has its own unique electrical signal. EMG signals are biomedical signals that determine the currents in muscles. The specifications of this signal are entirely determined by the type of physical activity, the destination muscle, and the intensity of the action.

Since EMG detectors are placed on the surface of the skin, the signal contains noise as a result of receiving signals from different sources, passing through multiple layers of tissue, and detector error. This noise complicates the processing of the signal.

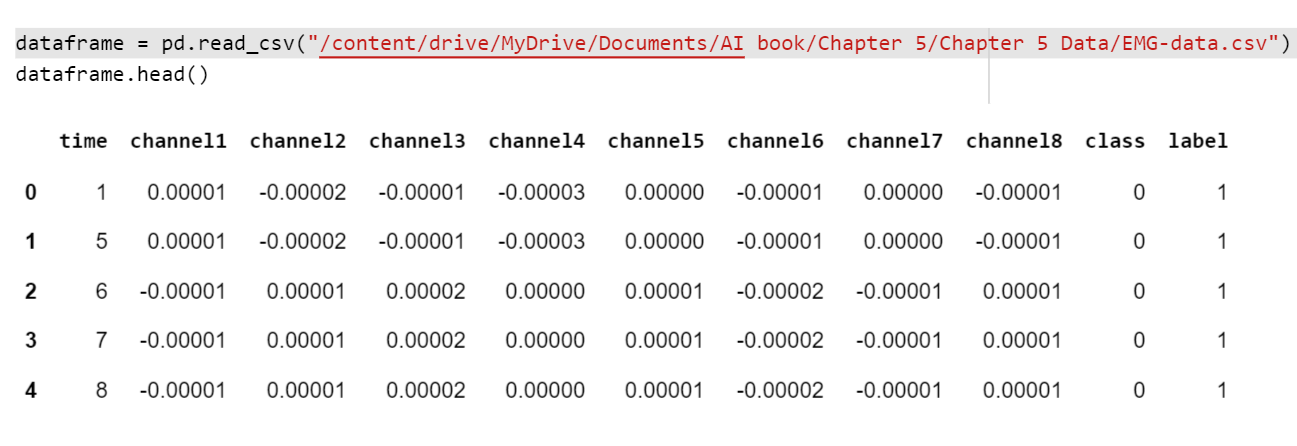
Signal processing of EMG signals is primarily used in rehabilitation, human-machine interaction, and medical diagnosis. Signals contain hidden properties and characteristics that are crucial to the understanding of neural systems. Signals emitted by the brain during sleep time provide insight into the quality of sleep, while signals received from heart activity provide insight into cardiovascular disease. In order to rehabilitate disabled people using robotic arms, we must detect signals emitted by neurons.

**About Dataset**

A CSV file is a file containing data with numbers and values separated by commas. The EMG signal information has been stored in CSV format. There are many ways in which to read this type of file. However, using Pandas to convert a CSV file into a data frame is one of the most efficient methods. In data frames, each column represents a particular feature and each row represents a sample or record. Data frames are fast and reliable data structures consisting of specific columns.

As can be seen, the first five samples of data will be displayed when calling the head instance of a data frame. As it is apparent, the first column represents the number of samples in the data frame. The second one represents the time of recording which is not particularly relevant, and the third to tenth ones depict the value of electrical current in different channels. The final two columns indicate the class of the sample and the label of the sample. This label indicates the identity of those participating in the EMG signal sampling, which is not very meaningful, and the class value describes the type of activity involved during sampling time, as follows:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Class**  **Value** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| Type of Activity | Unmarked | Hand at rest | Clenched in a fist | Wrist flexion | Wrist extension | Radial deviations | Ulnar deviation | Extended pain |

****

**­­Introduction**

Unlike typical computer programs, Machine Learning techniques will literally learn from data. Machine Learning algorithms can actually find insights and data even if they are specifically instructed on what to look for in that data, and that's what separates a Machine Learning algorithm from a typical computer program. You're just giving the Machine Learning algorithm a set of rules to follow. Instead of actually telling it what to look for, it will find the insights on its own.

**Why do we use Machine Learning to solve mechanical problems?**

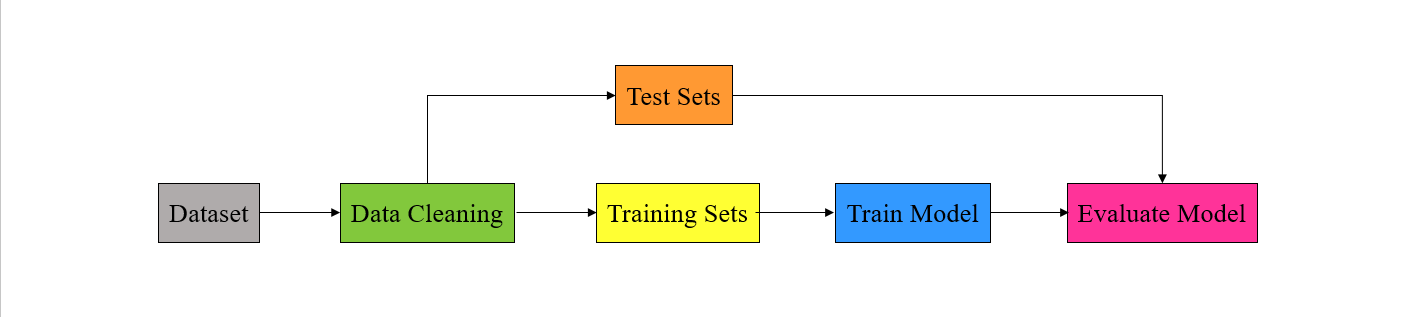
Machine learning is a method for predicting behavior or classifying data sets that, unlike common methods in mechanics, instead of being based on an intuitive model, uses a mathematical model and arbitrary functions to describe and predict the behavior of systems. In other words, machine learning is a search in the space of algorithms and parameters in such a way that it infers a model from the data (data-driven model) and based on that, predicts or categorizes the studied system. For example, using neural networks as one of the methods of traditional machines, I can perform a set of inputs based on an arbitrary number of intermediate hidden layers to the output image results. In the input and output data that are entered quantities, the middle layers do not necessarily have meanings and other adverbial expressions on them. For this reason, I can choose the number of intermediate layers and the number of nodes in each layer at will, and this approach is completely acceptable in the input to the output image. In particular, the relationship between the data is so complex that the models created with a limited number of adjustable settings express this relationship with sufficient accuracy, the efficiency of the methods using machines can be very important.

**Classification Problems**

High-level machine learning consists of supervised and unsupervised learning. Supervised learning means that we label historical data and use it to inform our model. We call that label or something that we want to predict as a target. So, in supervised learning, we have a specific goal (target) for that past information, and in unsupervised learning, we don't have a specific goal. In supervised learning, we have classification and regression. Classification problems are problems where our goal is a category (that is, we want to see what category it belongs to. It's usually True or False, but it can be multiple categories). Regression problems are those where our target is a numerical value.

**What are we going to do in this Chapter?**

We have a dataset from the Kaggle website and then clean that data. After that, we split our data into two groups (train and test). Then we train our model on the training set and after that, we evaluate our model with the test set we have.



***Figure 3-3.***

**Tensorflow**

This open-source library uses tensors to support machine learning and numerical computation. TensorFlow can be developed even by people without any programming experience and can be used in a variety of programming languages, such as Python, JavaScript, and C++.

A major application of TensorFlow is the construction of neural networks such as CNNs and RNNs. TensorFlow, since it is based on graphs, can be executed on multiple processors such as GPUs much more efficiently.



**Pandas**

Using Pandas, one can easily and intuitively work with relational or labeled data due to its fast, flexible, and expressive data structures. This course is intended to serve as a high-level foundation for conducting practical, real-world data analysis in Python. A further objective of the project is to become the most robust and flexible open-source data analysis/manipulation tool available in any language. It is already well on its way to achieving this goal.



**Copy**

When we copy a variable content to another variable in Python, we simply assign equality (=) to the new variable. Python creates a shortcut to the same memory space in this case. Therefore, whenever the copied variable is changed, the original variable is also changed. In order to prevent the original variable from being changed, we must use Python's deep copy module. With this module, a new memory space is allocated for a new variable in order to protect the original variable from alteration.



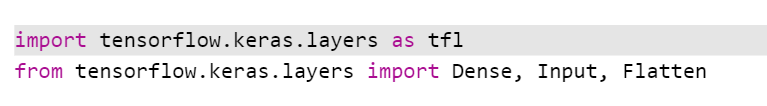
**Keras layers**

The Keras models, whether sequential or functional, require layers to be defined. Each layer has various parameters that need to be defined. The following layers are used in this notebook:

1. **Input:** The input layer, as its name suggests, should be the first layer to receive input. There is only one parameter, input shape, which indicates the shape of input tensors except the batch size.

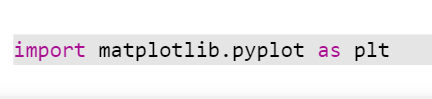
2. **Flatten:** Using this layer, every multidimensional tensor is converted to a one-dimensional tensor.

3. **Dense:** The dense layer indicates a simple neural network with neurons fully connected to one another. Two parameters must be set for this layer, including the number of neurons and the activation function.



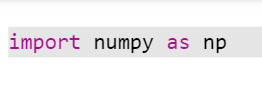
**Matplotlib**

The Matplotlib package provides static and dynamic visualization in Python. Pyplot is a Matplotlib module that creates a separate figure and makes changes to this figure to generate a plot.



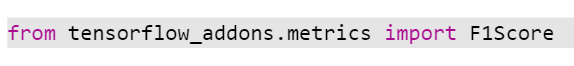
**NumPy**

The NumPy library contains functions that deal with arrays, which have the advantages of being processed faster than lists, consuming less memory, and reserving a static portion of memory.



**TensorFlow Addons**

TensorFlow addons is a library that provides additional helper functions for TensorFlow and Keras models, including additional layers, optimizers, losses, metrics, etc.



**Seaborn**

An attractive and informative statistical graphic can be generated using Seaborn, a Python data visualization library based on matplotlib.

****

**Scikit-Learn**

In Python, there are several libraries that implement machine learning algorithms. Scikit-Learn is one of the best known, providing efficient versions of numerous common algorithms. There’s a good online documentation and a clean, uniform API with Scikit-Learn.

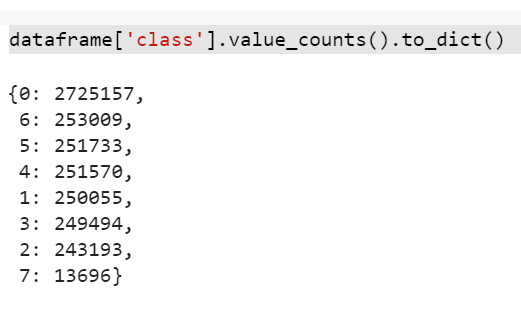


**Step 1. Preprocessing**

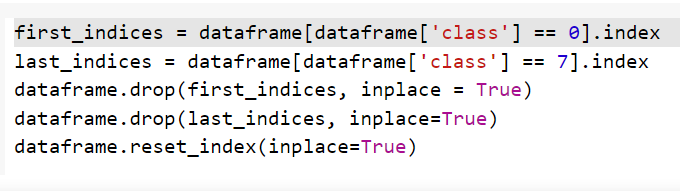
At first data in machine learning are not usable often, so needed to be cleaned like filling missed values, separating features and labels, normalizing or scaling and so on. These kinds of processes named preprocessing to prepare data for feed into the model.

**Removing Useless Data**

Let’s look at the distribution of classes in data:



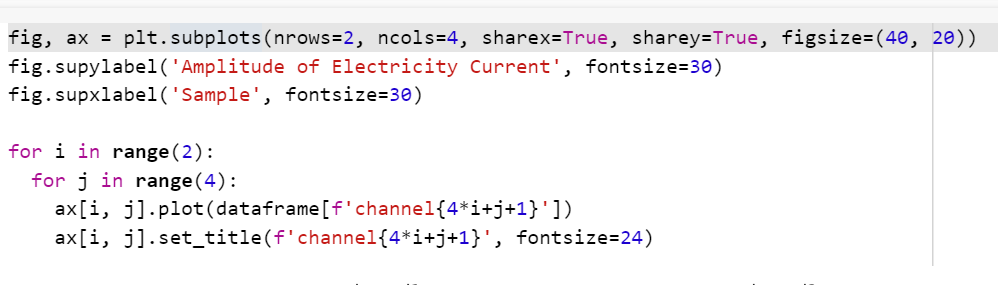
As it is clear the number of unmarked is around 67 percent of data and is useless because don’t add meaningful idea to model. Also the proportion of last class is so tiny relative to others. In order to improve performance we need to eliminate these two classes from data frame.

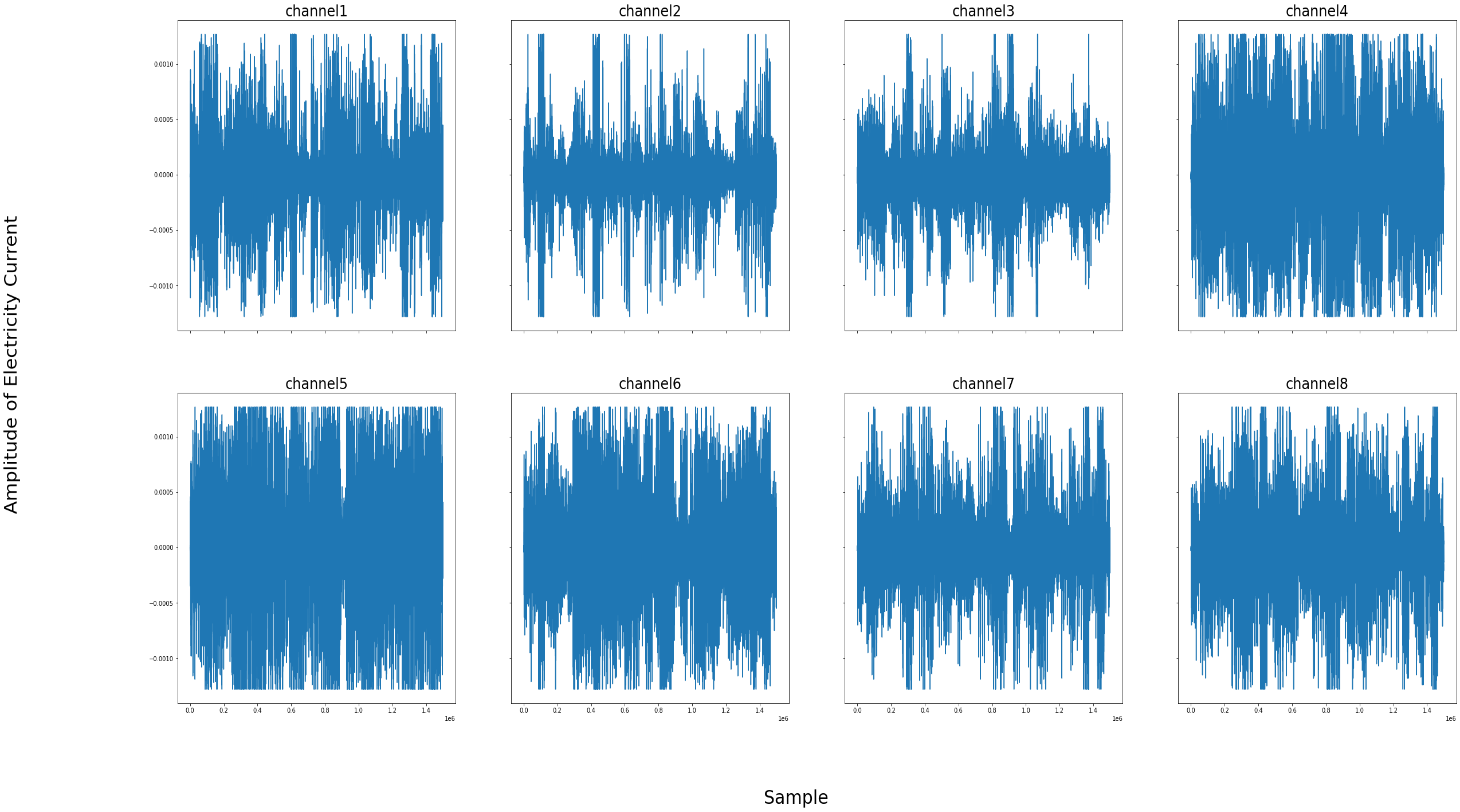


**Data Visualization**

It is necessary to illustrate trends in data points by time in order to gain a deeper understanding of time series data properties. As is evident from the data, the EMG signal is too noisy. The data contains too many peaks and valleys. These noises must be removed in order to improve the performance of the model and reduce error.

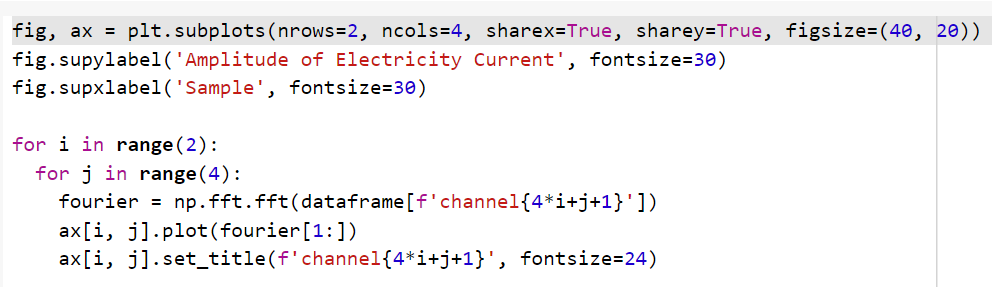
This dataset contains approximately 1.5 billion samples. The amplitude of electricity current in neurons from different channels fluctuates between -0.001 and 0.001 Ampere.

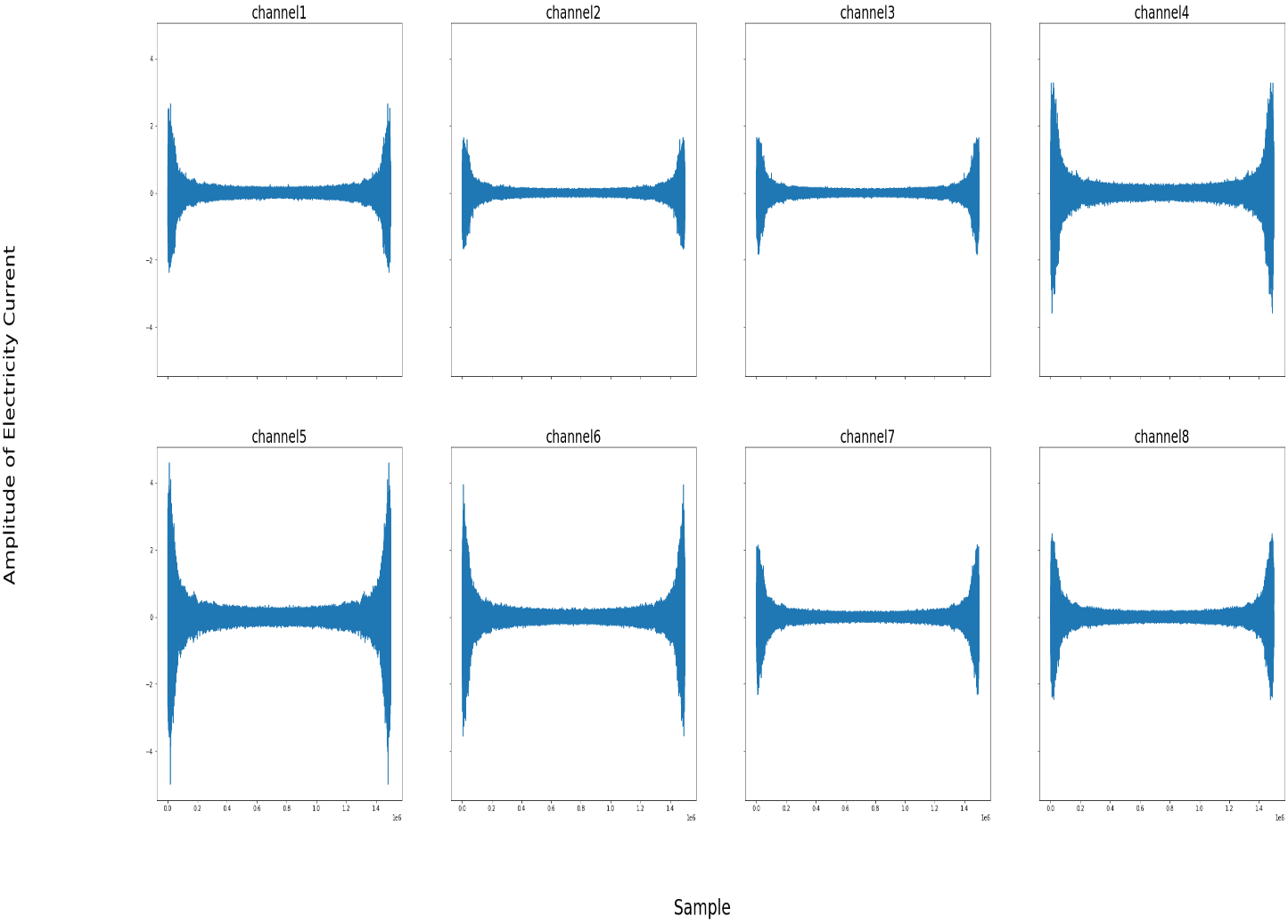




**FFT**

Using FFT (Fast Fourier Transform) of the signal, it is possible to see the frequency distribution hidden in the signal. This figure illustrates that the specifications and properties of frequencies collected from different channels are similar. This similarity provides a more straightforward solution to preprocessing the data.





To remove noises we need to perform these 3 tasks:

1. Subtracting mean of data

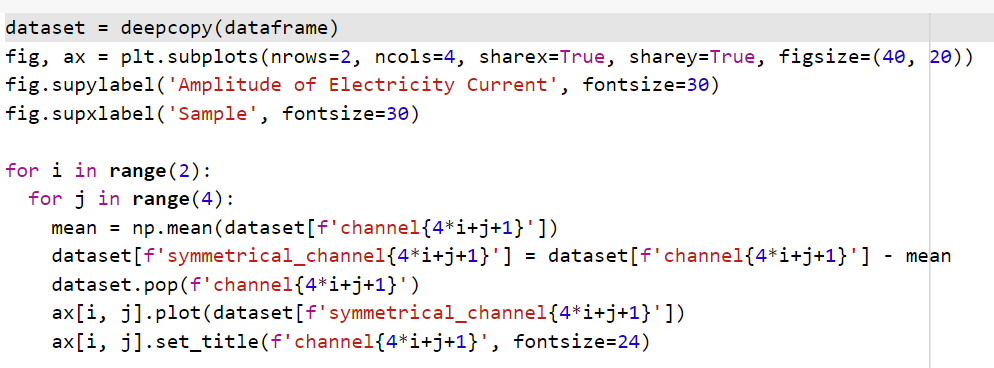
2. Take absolute value

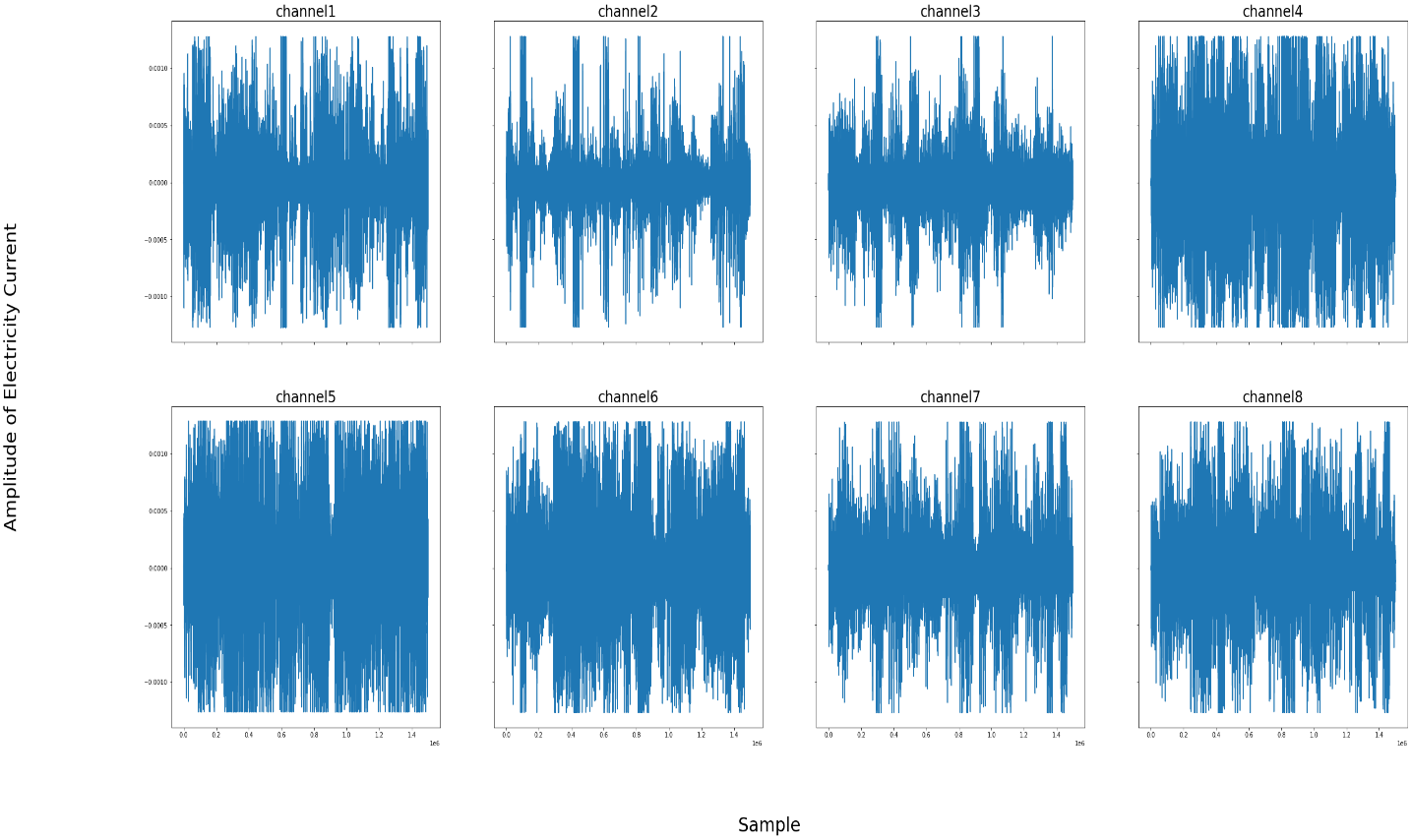
3. Calculate moving average

**Mean Subtraction**

The electric signals emitted by neurons are symmetric around zero. However, due to noise, EMG signals are not symmetric around zero. To remove the noise, the mean of the data is calculated and subtracted from the data.

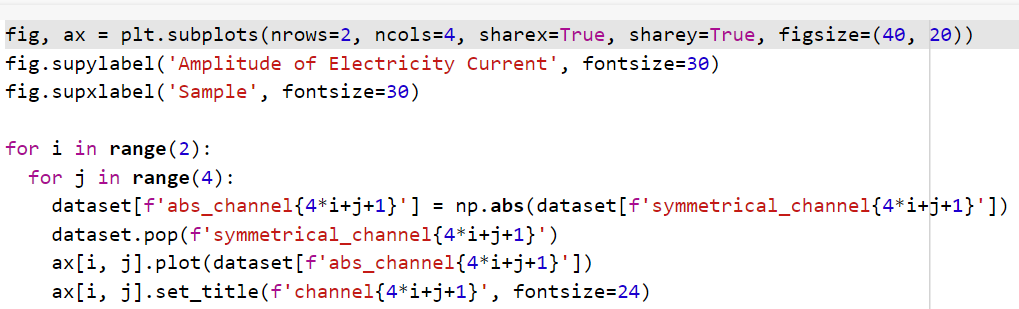
|  |  |  |  |
| --- | --- | --- | --- |
| **X** | **Mean of data** | **X – Mean** | **Mean of new data** |
| 3 | 5 | -2 | 0 |
| 10 | 5 | 5 | 0 |
| 0 | 5 | -5 | 0 |
| 5 | 5 | 0 | 0 |
| 7 | 5 | 2 | 0 |

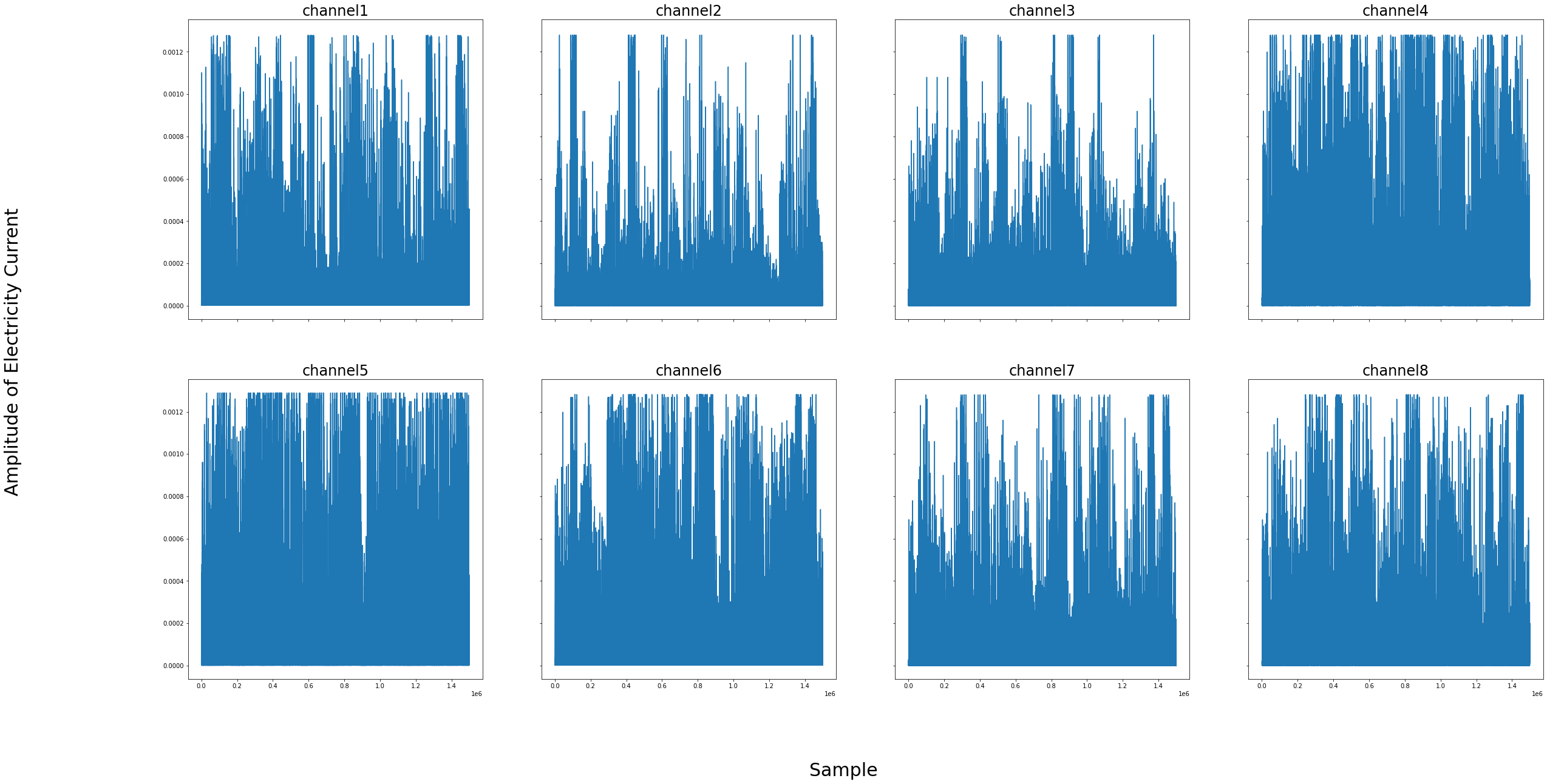


****

**Rectification**

A negative value cannot be defined for electric signals with an amplitude below zero. In order to perform better on electric signals, the absolute value of the signal must be considered. Therefore, there is no difference between a negative and a positive value. We must calculate the absolute value of the signal.



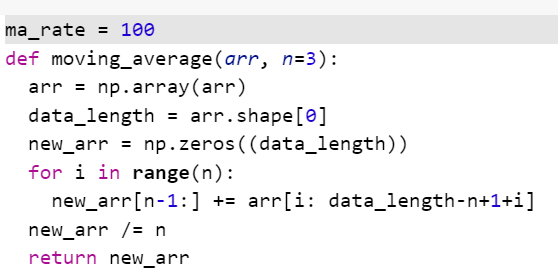


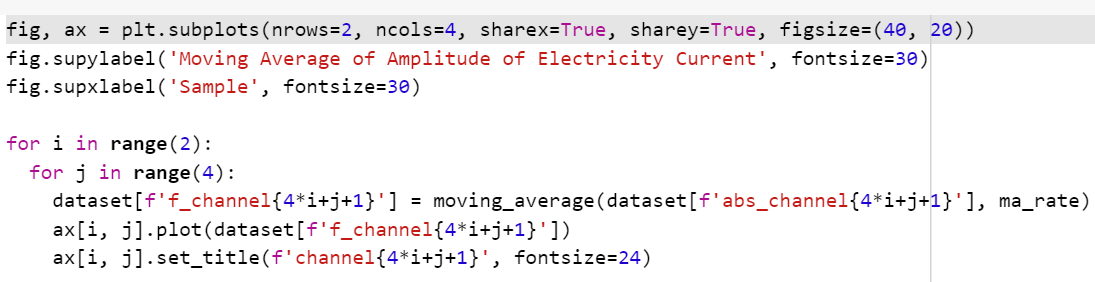
**Moving Average**

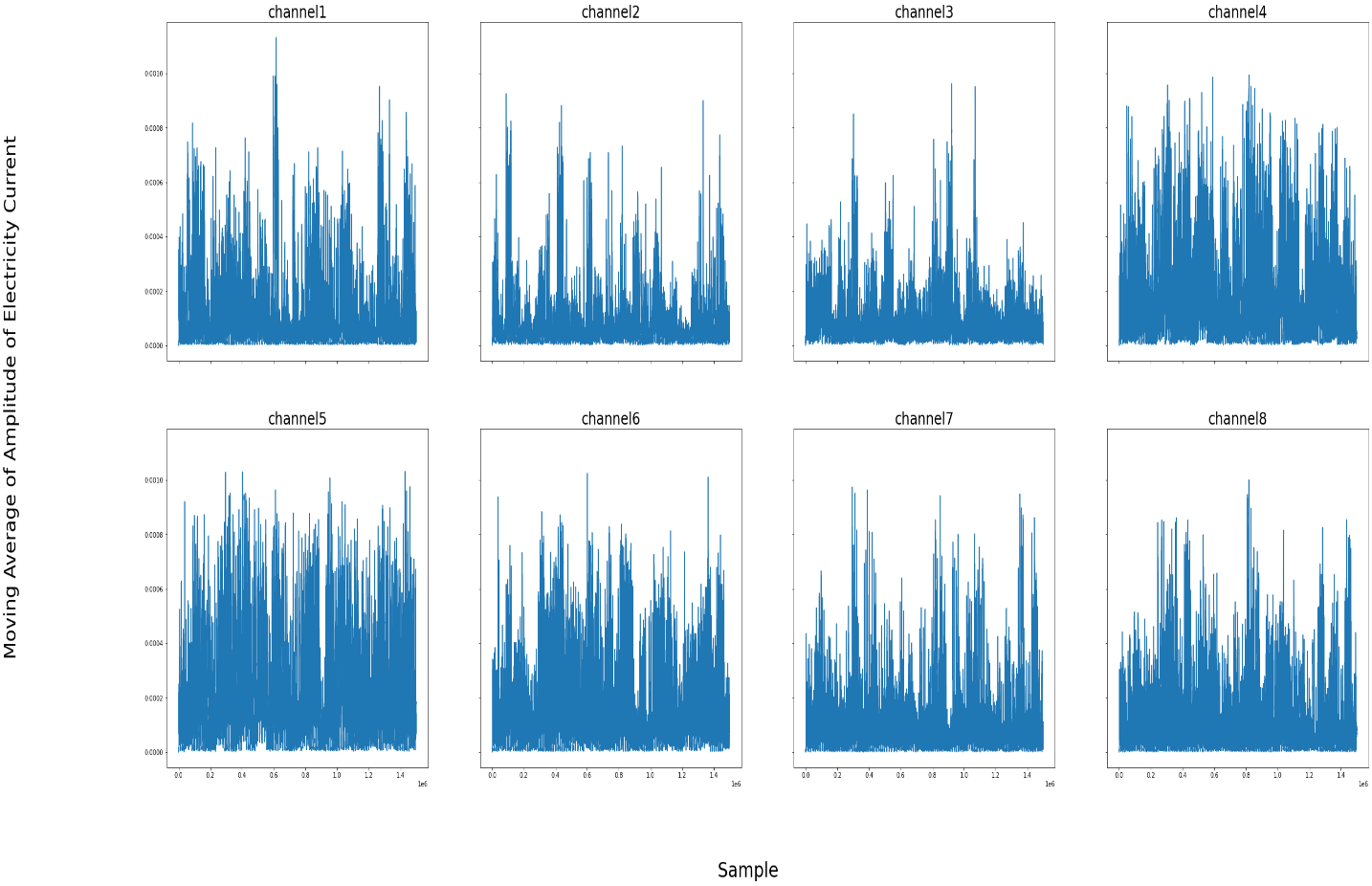
By creating a series of average of data points from different subsets of the full data set, a moving average can be used to analyze data points. As part of time series analysis, moving averages are commonly used to remove noise, highlight long-term trends and cycles, and smooth out short-term fluctuations. The application determines one of the parameters of the moving average, which is the period of calculation.

For instance, assume that this series below are the dataset. To calculate moving average by period of 3 times, we perform like below:

|  |  |  |
| --- | --- | --- |
| **X** | **Time Frames** | **Mean** |
| 3 | - | - |
| 10 | - | - |
| 0 | 3,10,0 | 4.33 |
| 5 | 10,0,5 | 5 |
| 7 | 0,5,7 | 4 |

****

****

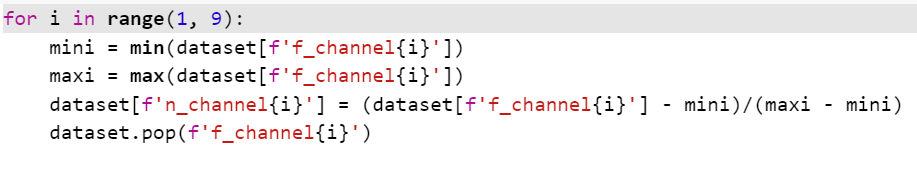
****

**Normalization of Data**

For faster and more stable training, numerical values in machine learning must have the same range, such as between 0 and 1. This type of analysis is called normalizing, since all values are scaled between 0 and 1.

As it’s shown in the table below, there is some random data between 0 and 10, to normalize this data, subtract each number from minimum (0) and divide by data range (10 – 0) .

|  |  |  |
| --- | --- | --- |
| *X* |  |  |
| 4 | (4 - 0)/(10 - 0) | 0.4 |
| 10 | (10 - 0)/(10 - 0) | 1 |
| 0 | (0 - 0)/(10 - 0) | 0 |
| 5 | (5 - 0)/(10 - 0) | 0.5 |
| 7 | (7 - 0)/(10 - 0) | 0.7 |

****

**Why Do We Need to Normalize Data in Python?**

One of the most important topics in the field of Machine Learning and Data Mining, especially in the Data Preparation section, is the topic of Re-scaling of data, which is usually done by There are two methods of Standardization and Normalization. The meaning of normalization is to transform the data into the domain [0 and 1]. Each of the data recorded in the dataset will change to a range between zero and one. This makes the data fall under a shorter domain and the model is trained better.

**Step 2. Load Data**

Signal amplitudes are now ready to be integrated into the model. In order to load a dataset containing predefined data, the Inputs and Labels matrices must be defined. The Inputs matrix is composed of all data samples from the eight channels. As a result, it is a 2-dimensional matrix, with rows equal to samples and columns equal to channels. As with inputs, labels matrix is a 2D matrix as well. The number of rows equals the number of samples and the number of columns equals the number of classes. All elements of the Labels matrix are one-hot encoded, meaning all elements except one are zero.

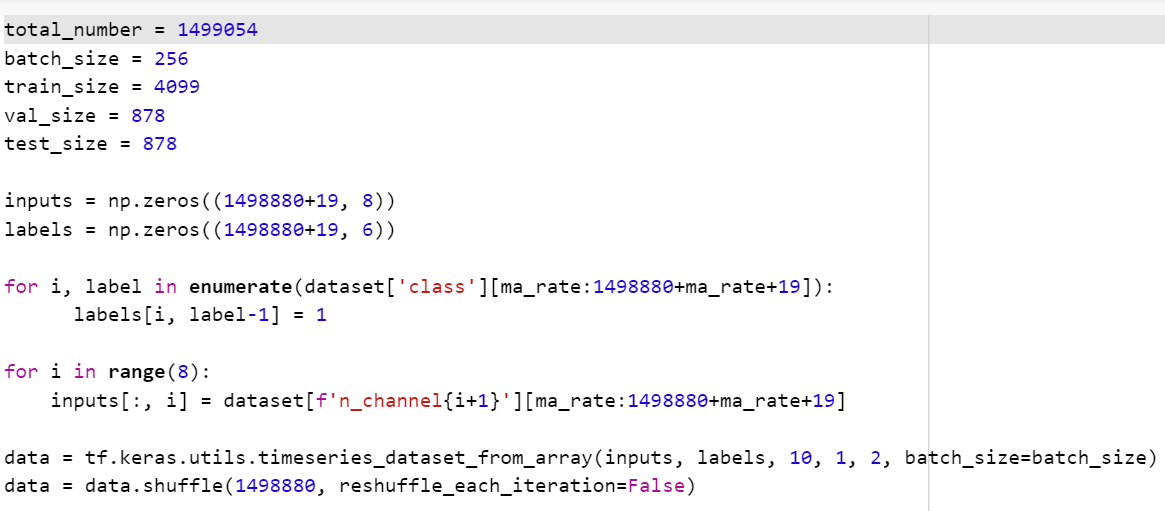
After defining the Inputs and Labels matrixes in TensorFlow, the timeseries dataset from array library converts arrays of values to timeseries sequences. Inputs, labels, number of samples in each sequence, stride, sampling rate, batch size, and sampling rate are all taken into consideration by this library. A stride is defined as the difference between the start index of each sequence, while a sampling rate is defined as the amount of time between each sampling. For example, suppose that our data comprises an array of natural numbers ranging from 0 to 99. Assuming a sequence length of 5, stride of 2, and sampling rate of 3, we have:

First sequence: [0 3 6 9 12]

Second sequence: [2 5 8 11 14]

Third sequence: [4 7 10 13 16]

...

Last sequence: [86 89 92 95 98] 

**Batch**

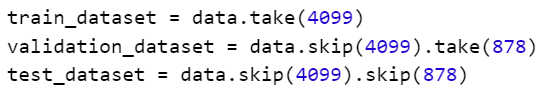
It is much more computationally efficient to compute loss on a small portion of data and calculate gradient descent based on that rather than computing loss on the whole data. A batch is a set of data that is used to calculate loss once a time and a batch size is the number of data in each batch. Batch size is the first element of the shape in Tensorflow and it is saved as a None object since it is independent of the model architecture.

**Step 6. Split Training and Testing Datasets**

Data is the key to training all machine learning models, and parameters are changed based on the available data. Therefore, more data results in better results. Using gradient descent, the model calculates training data loss and attempts to decrease the loss value. However, does a decreasing training data loss mean better results on real-world data? Not necessarily. Real-world data often include data that the model has never encountered previously. Thus, for evaluating results, loss values on training data are not sufficient, and some additional data, such as validation, must be collected.

It is very critical to determine the number and size of validation data, and this selection is highly dependent upon the number and size of data. Despite the fact that it is not a principle, it is acceptable to split 20 percent of the data between validation and training. The distribution of validation data must be the same as the distribution of actual data in order to achieve better results.

To divide a dataset into training, validation, and testing, I use the take and skip methods. In this example, 70 percent of the data is set aside for training, 15 percent for validation, and 15 percent for testing. The Take method accepts a number that determines the number of batches to split for data. Skip method takes the same number in order to ignore the first batches as much as you specify.

****

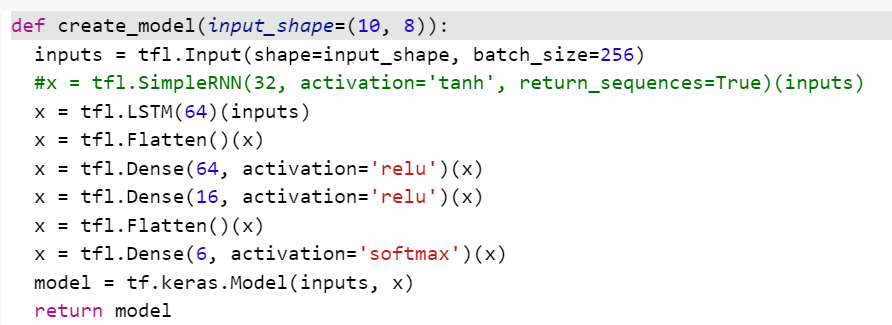
**Step 7. Training**

**Create Model**

A neural network architecture is created by this function. The first layer of the network is the input layer. The input shape of (10,8) refers to 10 time series of 8 dimensions. The batch size must be defined in the same manner as in the data loader.

In the second layer, LSTM is used, which is described in detail on the following page. After quitting the LSTM cells, the data passes through the Flatten layer, which converts multidimensional data into one-dimensional data. Dense layers have been applied to classify series in the last layers. As a consequence of the multiclass classification problem, 6 neurons (total number of classes) are required in the last layer and the activation function must be set to SoftMax.

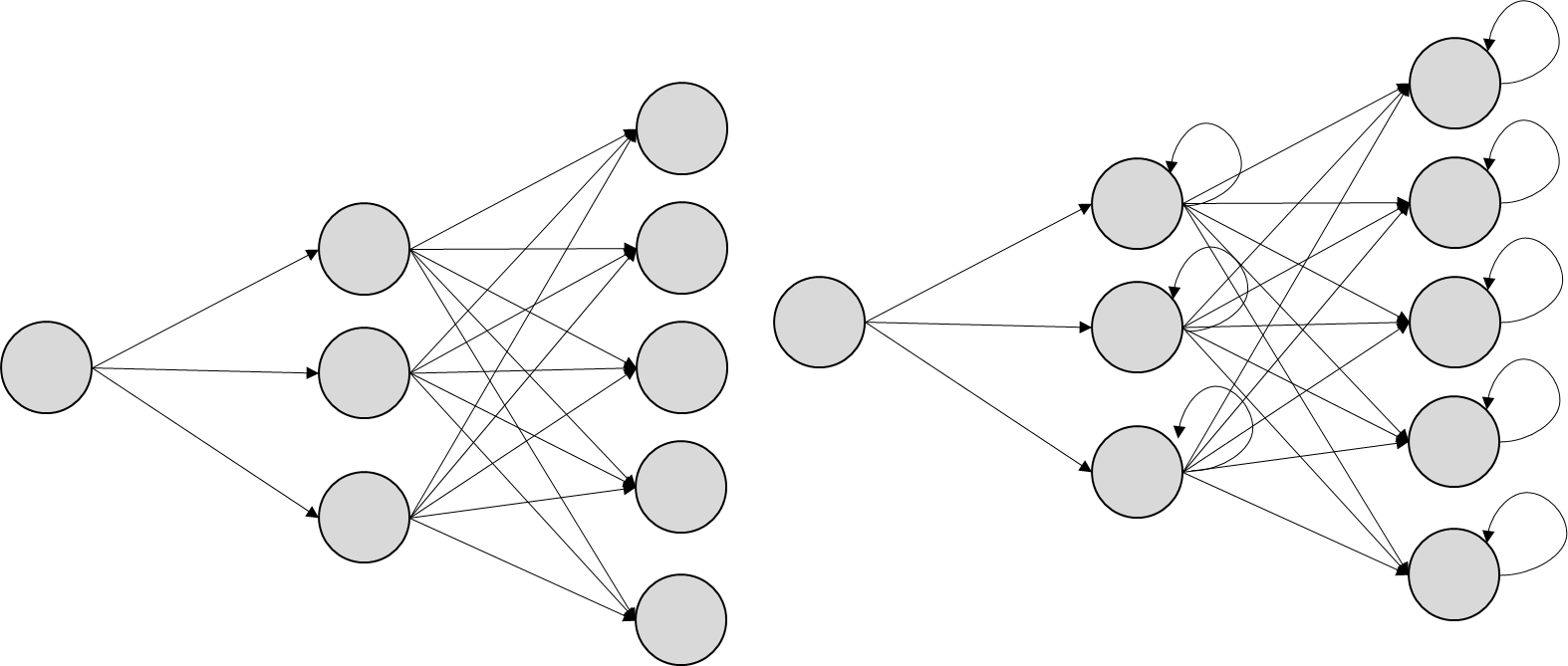
Call the Model method at the end of the process and specify the inputs and outputs of the model in order to create an instance of the model.



**RNN**

As a class of artificial neural networks, recurrent neural networks (RNNs) are characterized by connections between nodes that can lead to a cycle where output from one node can affect input to another node in the future. Since RNNs are derived from feedforward neural networks, they can be used to process a variable length sequence of inputs using their internal state (memory).

The output of recurrent neural networks depends on the prior elements within the sequence. This is compared to traditional deep neural networks which assume inputs and outputs are independent of each other. The output of a sequence can also be predicted based on future events, but unidirectional recurrent neural networks are not able to take these events into account.

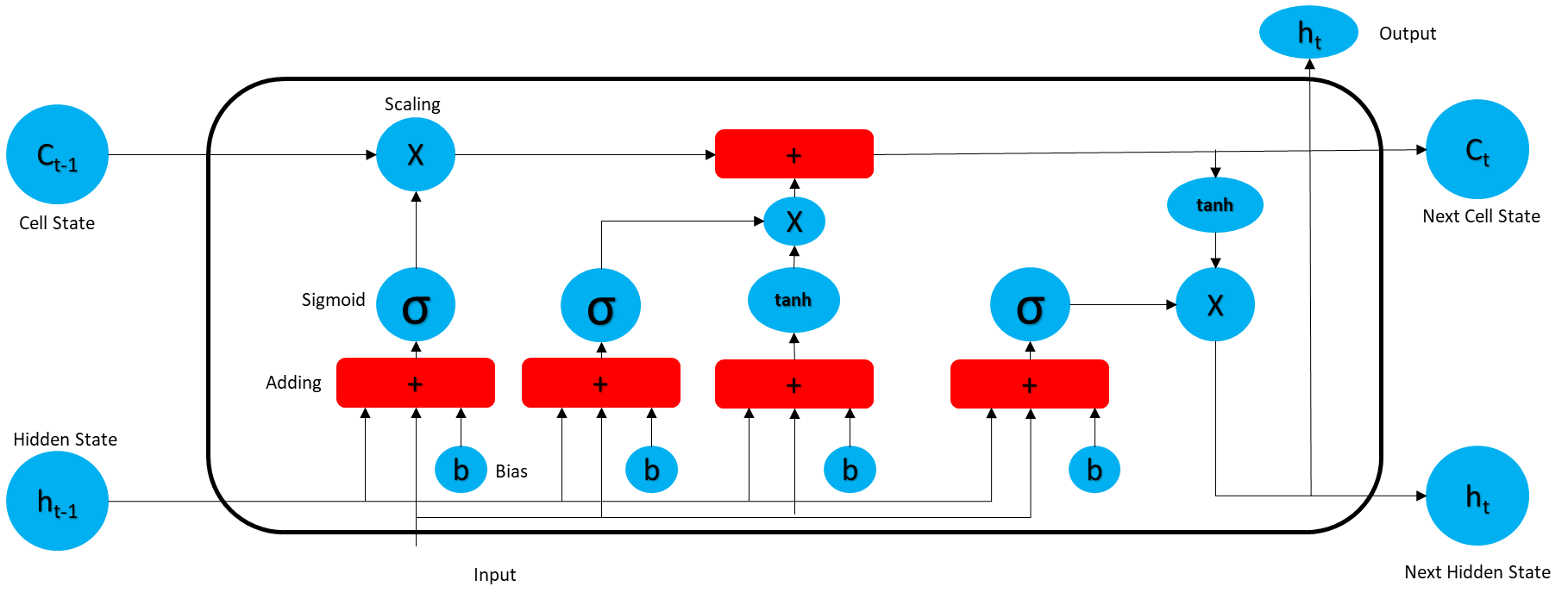


**FNN RNN**

**LSTM**

Long-Short Term Memory is a type of artificial neural network with feedback connections. Because of this feature, LSTMs are capable of processing a variety of data sequences effectively. An LSTM unit consists of a cell, an input, an output, and a forget gate, and it performs better on sequences of data. In terms of vanishing gradients, LSTMs perform better than traditional RNNs.

As a rule, LSTMs keep information away from the normal flow of RNNs within their cells. Input gates, output gates, and forget gates may be used to read, write, or delete this information. These gates, like filters, adjust some weights with gradient descent in order to determine how much information must pass.



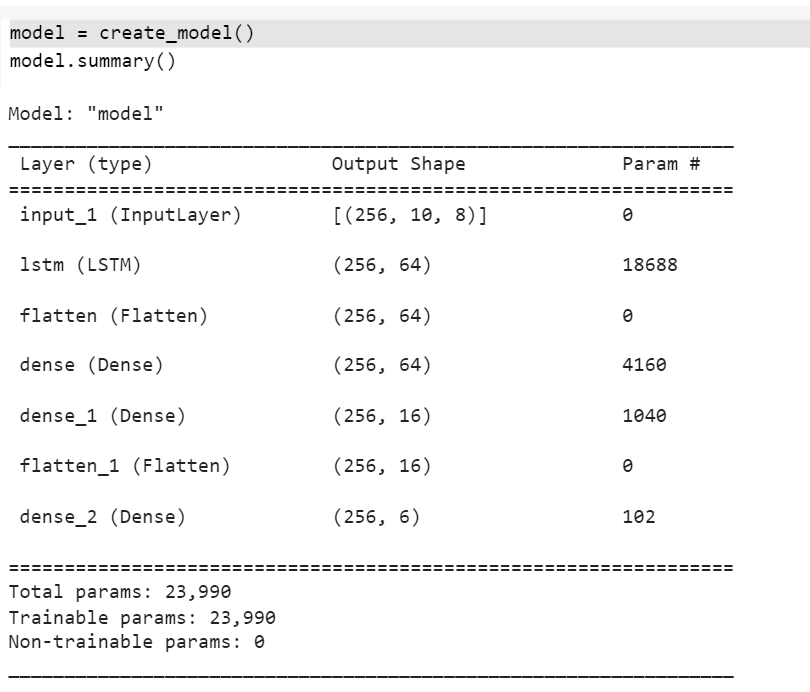
**SoftMax**

The SoftMax function converts a vector of numbers into their probability distributions. Neural networks deal with numbers on different scales. In multi-class classification, the model outputs a vector of numbers with a dimension equal to the number of classes. Our model will determine which class is predicted by converting these numbers into a probability distribution, which means the sum of all numbers must equal one. The SoftMax function suggests a way to convert these numbers.

**Summary**

Using the model summary, we can see the output and number of parameters for each layer, which gives a better overview of the model. Approximately 75 percent of the model parameters pertain to the LSTM model. The LSTM model output has the shape (256, 64).

The summary of the model indicates that no parameters are non-trainable, and all parameters are trainable. Trainable parameters are parameters that update during training. As the model continues to learn, it updates its values using gradient descent, however non-trainable parameters remain fixed during training. Setting more trainable parameters requires more time and computation, but it does not complete the learning process.



**Compile Model**

Compile model means to define 3 sets of parameters, loss function, optimizer and metrics. Loss function is a function that compute loss. In this example has been set to custom loss function that defined previously.

****

**Categorical Cross Entropy**

Categorical cross entropy is a loss function that compute logarithmic loss of data as mentioned below.

This loss function computes large loss every time there is difference between true label (y) and prediction (p(y)).

**Adam**

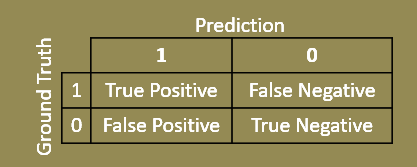
Optimizer is a function that can be used to compute gradient descent. It is set to Adam in this example. The SGD algorithm is very noisy, and it does not descend well on curves. To decrease noise in steps moving averages, a new parameter called Momentum is defined. The SGD algorithm with Momentum performs better on curves and requires fewer steps to converge. This method employs two momentum variables, a first-order momentum, and a second-order momentum, as well as an epsilon value that prevents division by zero. Adam is extremely efficient and useful for the convergence of local minima.

**Accuracy**

Metrics has been set to accuracy means that how much model predict accurately. The accuracy formula showed in below and as it suggests to evaluate our model by computing number of times that model predict right over by number of times that model predicts.

**F1 Score**

Often, more detailed metrics are needed to evaluate model performance than accuracy, as accuracy determines the difference between prediction and ground truth. In terms of classification, the model prediction and ground truth labels are vectors consisting of zeros and ones. There are four options which can be used to determine the difference between prediction and ground truth:



The number of times the model correctly predicts a positive label over a number of positive labels is referred to as Recall, whereas the number of times the model correctly predicts a positive label true over a number of positive predictions is referred to as Precision. The combination of these two metrics results in F1 Score, which is a more intuitive way to evaluate model performance.

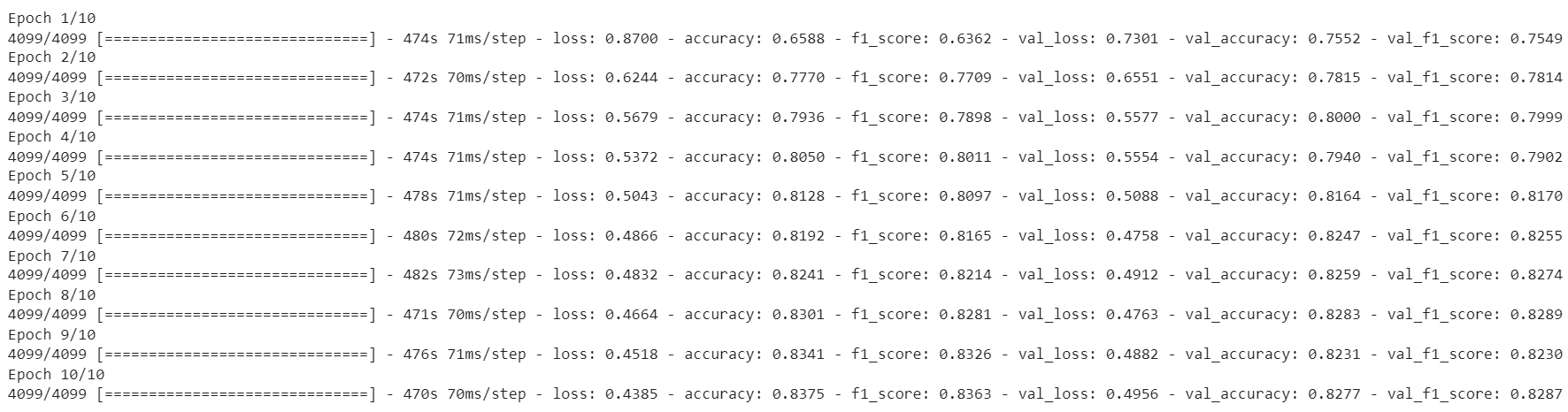
**Fit Model**

After all previous steps were completed to prepare the model for training, all that remains is to fit the model over the data. In order for the training model to be fit over the data, training and validation data need to be determined, as well as a number of epochs, where an epoch is the number of times the model was trained.

**Epoch**

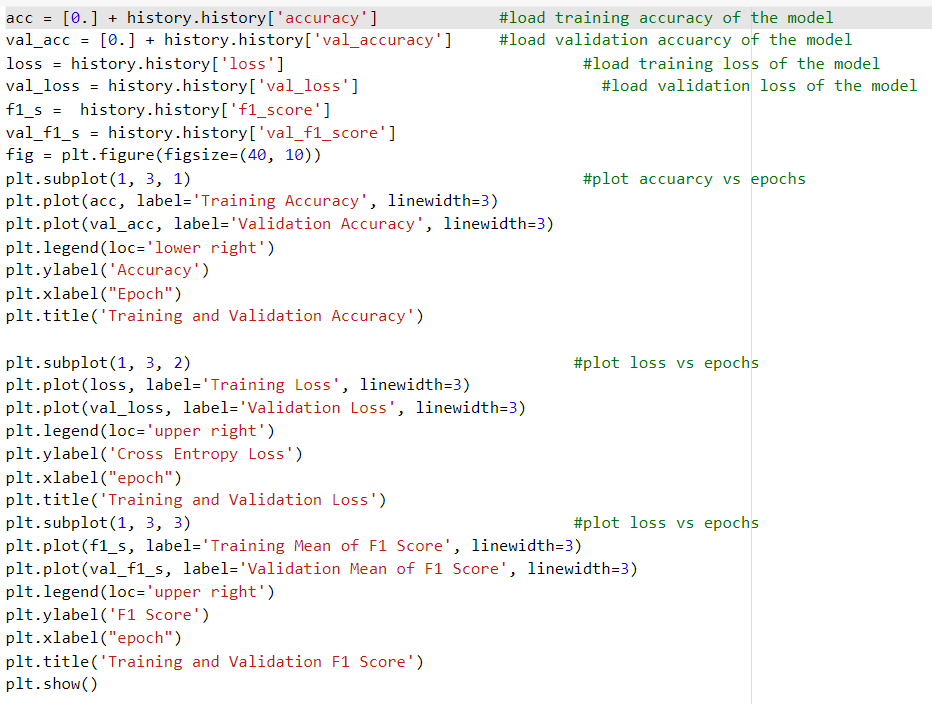
In training, the training process passes through all training data, so the number of training epochs indicates how many times the training process has been passed through all training data. When dealing with batched datasets, the number of times that training algorithm has performed is not important. The number of epochs is an important parameter to control the model since it indicates how many times the loss was computed on the entire data set.

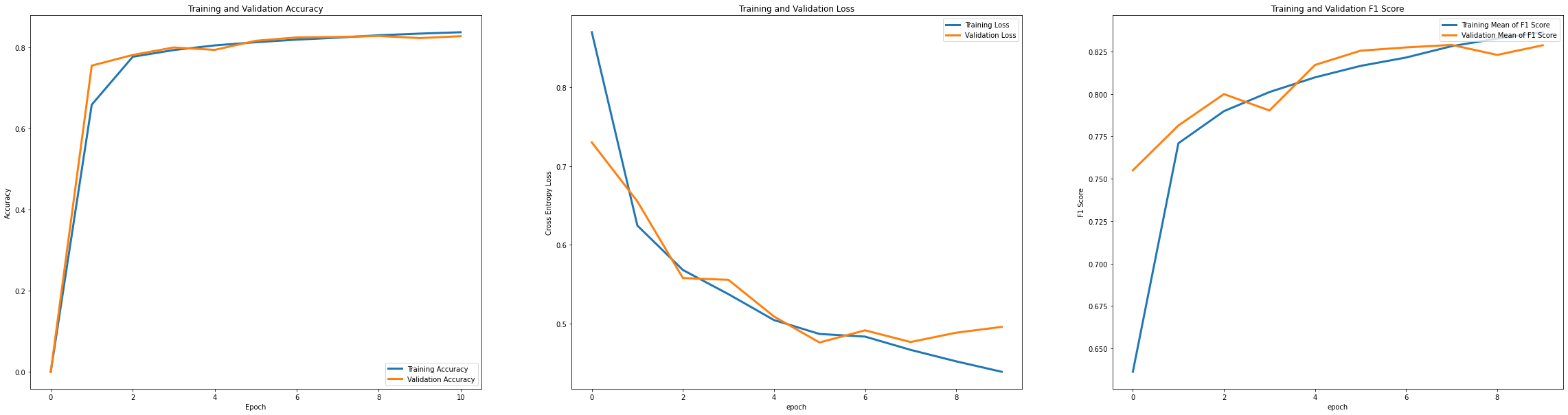




**Analysis**

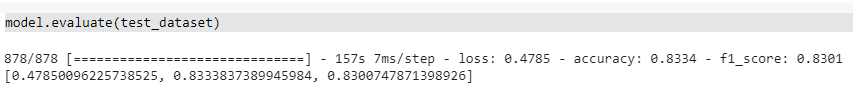
Using training as the number of epochs, it is possible to inspect the parameters of loss and accuracy. It is evident from the loss and accuracy plot that approximately 80% accuracy is acceptable for both validation and training.



****

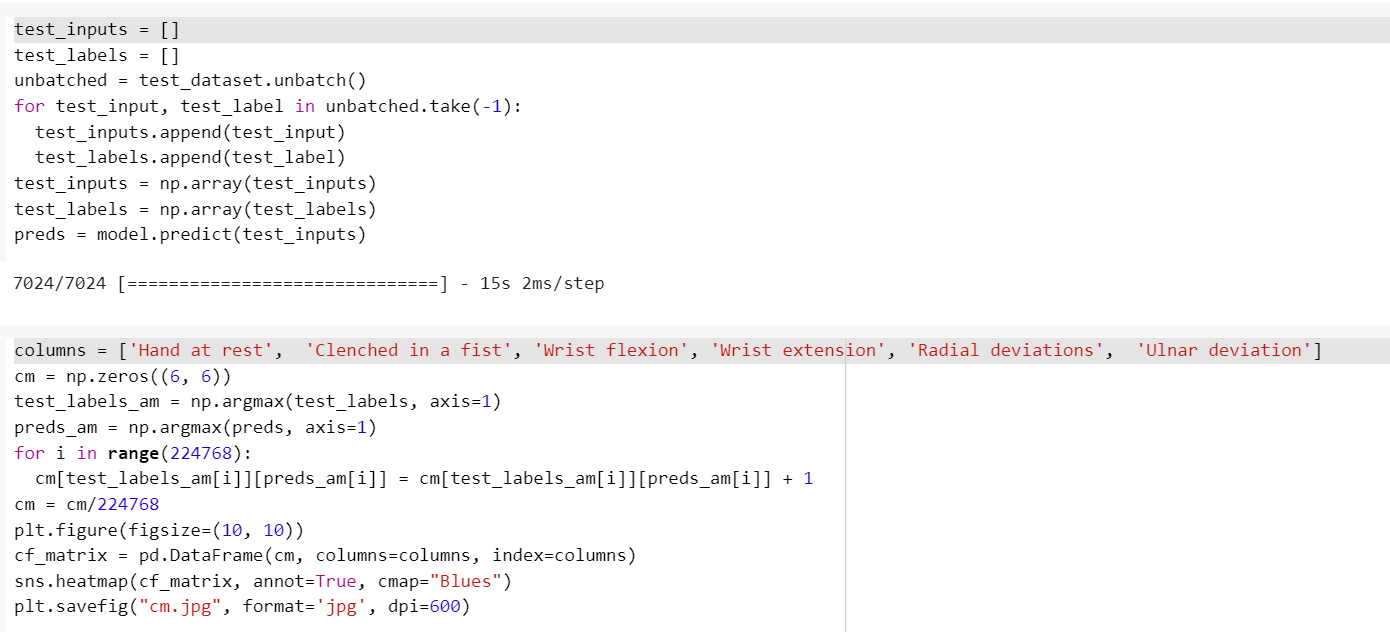
**Evaluate**

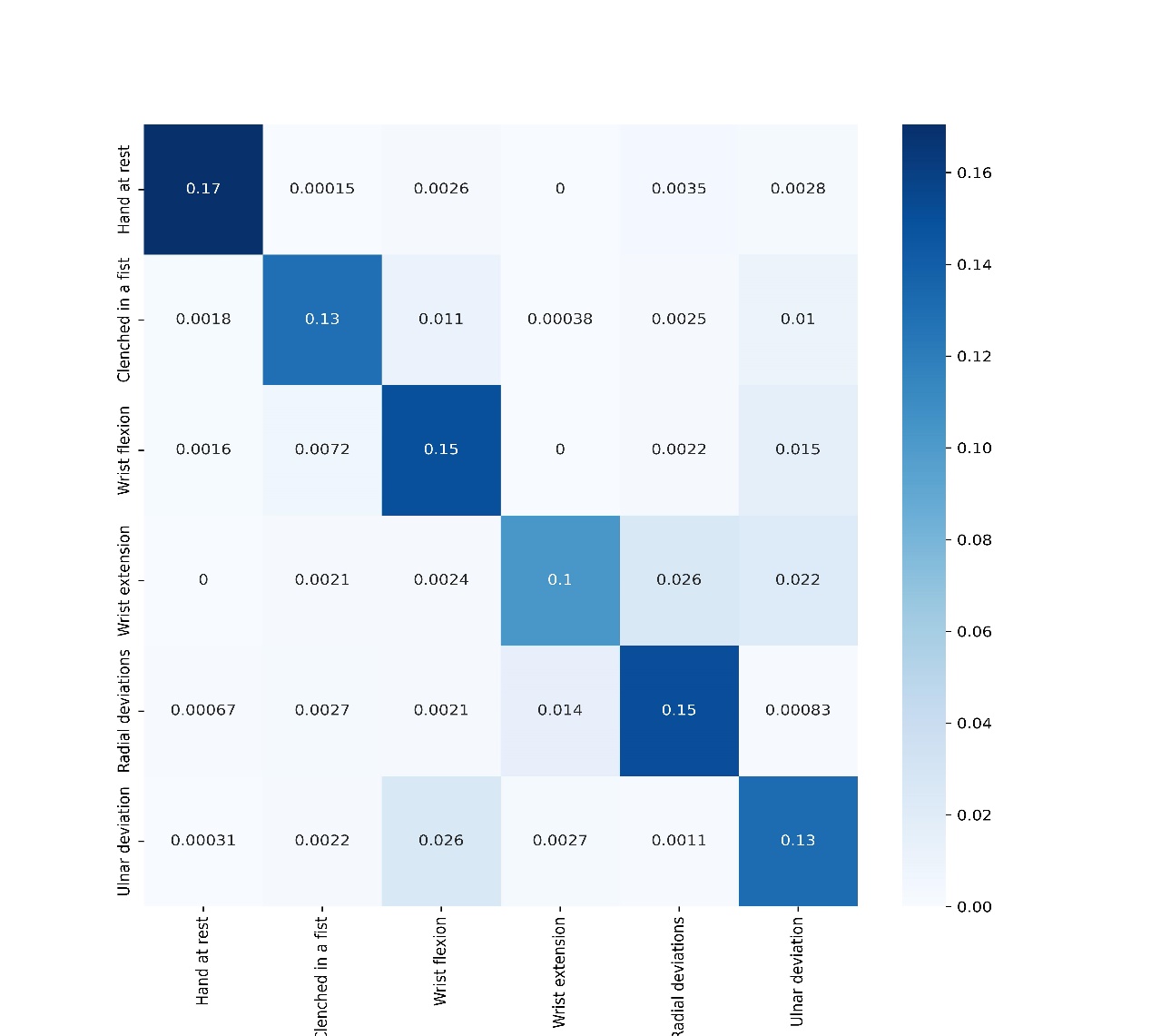
Evaluate the model's performance on the test dataset by calling the evaluate method. This method computes loss, accuracy and other predefined metrics for the model on the test dataset and reports them.

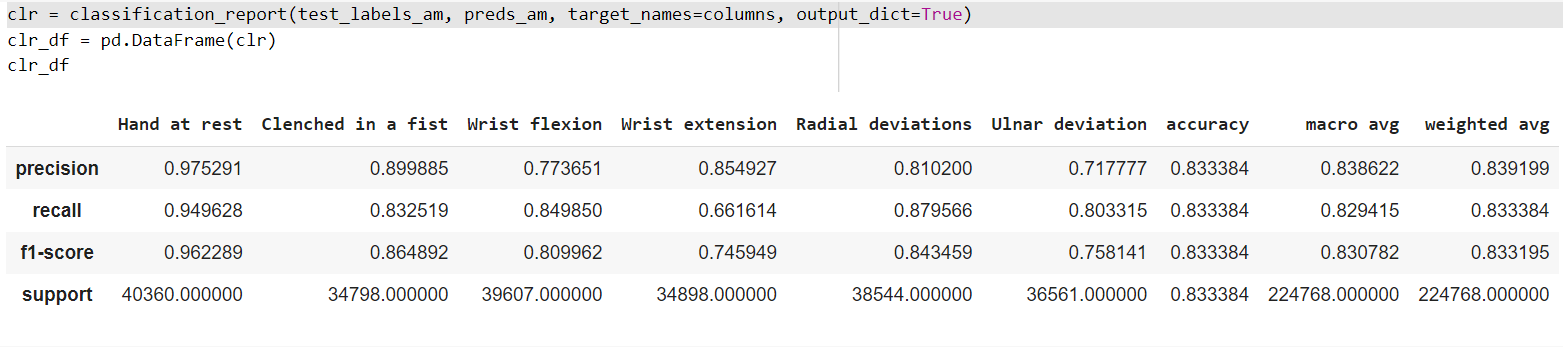


Evaluate the model's performance on the test dataset by calling the evaluate method. This method computes loss, accuracy and other predefined metrics for the model on the test dataset and reports them.

As part of our analysis of model performance on test data, we must report the F1 score of classes separately. To accomplish this, we use the classification report provided by scikit-learn and draw confusion matrix too.

****

****

****

**References**

[1] [Research Gate](https://www.researchgate.net/figure/portrays-the-VGG16-model-for-ImageNet-40-It-has-13-convolutional-layers-and-three_fig2_331562880)