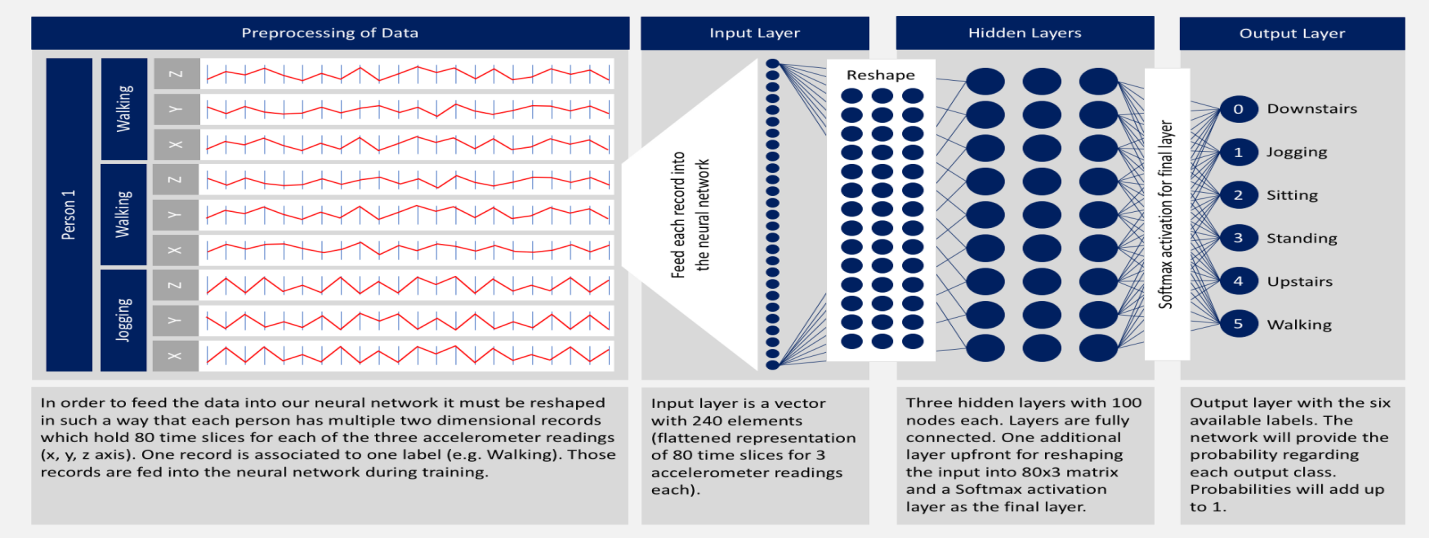
**Human Activity Recognition Using Accelerometer Data**

Problem statement:-

In this project we are going to use accelometer  data to train the model so that it can predict the human activity. We are going to use 2D Convolutional Neural Networks to build the model.



Dataset Link: <http://www.cis.fordham.edu/wisdm/dataset.php> or <https://github.com/laxmimerit/Human-Activity-Recognition-Using-Accelerometer-Data-and-CNN>

This WISDM dataset contains data collected through controlled, laboratory conditions. The total number of examples is 1,098,207. The dataset contains six different labels (Downstairs, Jogging, Sitting, Standing, Upstairs, Walking).

Here we are importing the necessary libraries. We will be using tensorflow-keras to build the CNN. We are also importing the necessary layers required to build the CNN.

* pandas is used to read the dataset.
* numpy is used to perform basic array operations.
* pyplot from matplotlib is used to visualize the results.
* train\_test\_split from sklearn is used split the data into training and testing dataset.
* LabelEncoder from sklearn is used to encode target labels with value between 0 and number of classes-1.
* StandardScaler from sklearn is used to bring all the data in the same scale.

Load and process the Dataset

If we try to read this data directly using pd.read\_csv() we will get an error because this data is not pre-processed properly. So we will have to read this data into a native python file and then pre-process it.

Using open() we will first open the file. Then we will read all the lines of the file into the read variable. Now we will consider all the lines one by one using a for loop. For each line the following operations will be performed-

* line = line.split(',') splits the line wherever there is a comma and returns an array of separated elements.
* last = line[5].split(';')[0] removes the semicolon after the last element in the array.
* last = last.strip() removes any extra space.
* Then if last is not empty we copy all the elements into temp.
* Now that the line is ready we append it to processedList

try and except is used for error handling. In this process if we get an error, the number of the line which is throwing that error is displayed.

### 2D CNN Model

A Sequential() model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor.

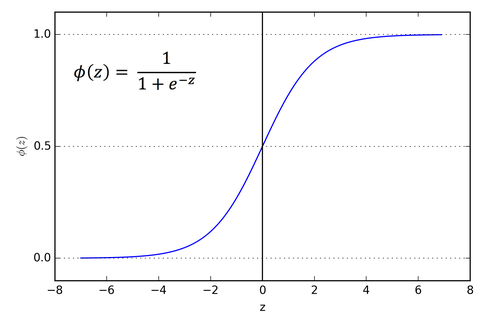
Conv2D() is a 2D Convolution Layer, this layer creates a convolution kernel that is wind with layers input which helps produce a tensor of outputs. In image processing kernel is a convolution matrix or masks which can be used for blurring, sharpening, embossing, edge detection, and more by doing a convolution between a kernel and an image. In the first Conv2D() layer we are learning a total of 16 filters each having size (2,2). We will be using ReLu activation function. The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero.



Dropout layer is used to by randomly set the outgoing edges of hidden units to 0 at each update of the training phase. The value passed in dropout specifies the probability at which outputs of the layer are dropped out.

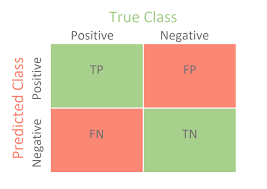
Flatten() is used to convert the data into a 1-dimensional array for inputting it to the next layer.

Dense layer is the regular deeply connected neural network layer with 64 neurons. The output layer is also a dense layer with 6 neurons for the 6 classes. The activation function used is softmax. Softmax converts a real vector to a vector of categorical probabilities. The elements of the output vector are in range (0, 1) and sum to 1. Softmax is often used as the activation for the last layer of a classification network because the result could be interpreted as a probability distribution.



### Confusion Matrix

* A confusion matrix is a table that is often used to describe the performance of a classification model (or “classifier”) on a set of test data for which the true values are known.
* Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class (or vice versa)
* The name stems from the fact that it makes it easy to see if the system is confusing two classes (i.e. commonly mislabeling one as another).
* All correct predictions are located in the diagonal of the table, so it is easy to visually inspect the table for prediction errors, as they will be represented by values outside the diagonal. For two classes the confusion matrix looks like this-



where:TP = True Positive; FP = False Positive; TN = True Negative; FN = False Negative.

Detailed video is available here: <https://youtu.be/SToqP9V9y7Q>

To calculate the confusion matrix we will use confusion\_matrix from sklearn. We will be using mlxtend to plot the confusion matrix. You can install it using the command or from the link mentioned.

pip install mlxtend -> [http://rasbt.github.io/mlxtend/installation/](https://rasbt.github.io/mlxtend/installation/)