DATA PRE-PROCESSING:

In order to normalise the data, we took the 1st joint as the origin and ensured that the second joint was at a distance of 1 unit from the first joint. As per the diagram, the second joint is at the chest and the first one is near the pelvic region, and hence, we should get a roughly equal distribution of points with positive and negative coordinates (in the x, y and z dimensions).

Also, the data included one more dimension for a second person, but to start with we consider only one person and removed the data corresponding to the last 11 classes which involved 2 person activities.

The data is such that if the length of the video is 300 frames, there are 0s padded at the end. We worked with this data to begin with, and later modified it such that the same repeating frames are repeated after the action ends, until the length is 300 frames.

Broadly, we have tries two approaches as yet:

1, TREATING THE VIDEO LIKE AN IMAGE WITH ONE DIMENSION BEING TIME

1. We took the input image to be 300\*25\*3 and used the following model:

Conv2D(32, (3,3))

Conv2D(32, (3,3))

Conv2D(64, (3,3))

MaxPooling2D(pool\_size = (2,2)))

Conv2D(64, (3,3))

Conv2D(64, (3,3))

MaxPooling2D(pool\_size=(2,2)))

Conv2D(64, (3,3))

Conv2D(64, (3,3))

Flatten ()

Dense(128)

Dense(64)

Dense(49)

This lead to learning curves that stayed flat for some time and then fell and then again plateaued. It overfit on taking 300, 3000 and 8000 samples, but the validation accuracy was at 2%, ie, absolutely no change! For the entire data there was no change in loss.

b. Same input but different model:

Conv2D(32, (3,3))

Conv2D(32, (3,3))

Conv2D(64, (3,3))

MaxPooling2D(pool\_size = (2,2)))

Conv2D(64, (3,3))

Conv2D(64, (3,3))

MaxPooling2D(pool\_size = (2,2)))

MaxPooling2D(pool\_size=(2,2)))

Conv2D(64, (3,3))

Conv2D(64, (3,3))

Flatten ()

Dense(512)

Dense(128)

Dense(49)

For 300 samples, the loss curve is as follows:

We are unable to understand why this sort of a loss is coming. One error that we have been committing is that we are performing convolutions in both dimensions, when there is no correlation between some adjoining joints in some cases, but that should still not lead to a model that generalises so badly!!!

c. TCN ResNet

We used the code and the model from the paper, where they managed to get an accuracy of 74%. We were unable to replicate this too, and we spoke to sir about this yesterday and he advised us to start very small (like just focus on one joint and use it to differentiate between two very different classes) in order to determine where the error is- in the data and its processing or in our code. Hence, this will be our next order of business.

The loss for this was as follows:

2. USING RECURRENT NETWORKS (LSTMS)

We used the following model:

Linear (75->70)

Linear (70->64)

LSTM

SoftMax

This model gave the following loss curve:

As we discussed, the first dimension is batch size in the input.

One error I discovered is that we did not give it a num\_layers argument, which by default is 1.

However, as per <https://discuss.pytorch.org/t/example-of-many-to-one-lstm/1728/2> , the first dimension is the length of the input sequence. So now, the input should be num\_layers\*batch\_size\*input\_size?

