**Model architecture is as follows:**

**Input 2**

Emg data from right hand (hand 1)

Has shape (num\_points(1), 15(2), 8(3))

**Input 1**

Emg data from left hand (hand 0)

Has shape (num\_points(1), 15(2), 8(3))

1D Convolutional Layer\*

(input size = 15, output size = 15, kernel size = 1, stride = 1, groups = 15)

Has shape (num\_points, 15, 8\*2)

Concatenate both hands

Has shape (num\_points(1), 15(2), 8(3))

Has shape (num\_points(1), 15(2), 8(3))

LSTM Layer

(input size = 8, hidden size=8, num layers = 1)

Has shape (num\_points(1), 15(2), 8(3))

Has shape (num\_points(1), 15(2), 8(3))

1D Max Pool Layer

(kernel size = 1, stride = 1)

Shape (num\_points(1), 15(2), 8(3))

Shape (num\_points(1), 15(2), 8(3))

Flatten all dimensions except batch size

Has shape (num\_points, 15\*8\*2=240)

First 1D Linear Layer\*

(input size = 15 \*8\*2, output size = 70)

Has shape (num\_points(1), 15(2), 8(3))

Has shape (num\_points(1), 15(2), 8(3))

Has shape (num\_points(1), 15(2), 8(3))

Has shape (num\_points(1), 15(2), 8(3))

Has shape (num\_points, 70)

Second 1D Linear Layer

(input size = 70, output size = 5)

Has shape (num\_points(1), 5)

**Note on dimensions**

(1) Depends on the batch size

(2)An interaction was 15s, Each interaction was split into 15 second sub windows

(3)Emg has 8 channels, the spatial relationship was preserved

**Note on activation functions**

\* ReLU was applied as the activation function after the 1d convolutional layer and after the first 1D Linear Layer

Further, softmax was used as the activation function after the second 1D Linear Layer

**In general:**

* The model would output an array of shape (num\_points,5).
* A row in this output array represents the probabilities that the datapoint belongs to each property. For example, if a row in the output array is [0.2, 0.1, 0.1, 0.2, 0.4]., this means that the datapoint belongs to 4 (softness)
* Classes:
  + 0 - Smoothness
  + 1 - Thickness
  + 2 - Warmth
  + 3 - Flexibility
  + 4 - Softness
* The criterion used to calculate loss was cross entropy. The y labels entered to the model were **not** one hot encoded because the target labels should be indices (<https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html> )

Graphical user interface, text, application, email

Description automatically generated



**Methodology used to conduct LOSOCV (Leave One Sock Out CV):**

* Nested 5 fold cross validation was conducted. The inner CV tuned the hyper parameters - learning rate and number of epochs.
* For a given sock, the training data consisted of emg data of all socks but the chosen sock and the testing data was the emg data corresponding to the chosen sock. Therefore, training data has 150 rows (6 people \* 5 socks \* 5 properties) and testing data had 30 rows (6 people \* 1 sock \* 5 properties).
* The training data was then further split into training and validation sets
* If I the model was trained for 100 epochs, we have 100 different models. How did I select the best model?
  + For every epoch, the model was trained using the training set and the training loss was recorded. Then, the model was fit to the validation set and the validation loss was calculated
  + A weighted average loss (i.e. (5/6)\*training loss + (1/6)\*validation loss) was recorded
  + The best model was the model that had the lowest average weighted loss
* How were the hyper parameters chosen?
  + The values tried for the hyper parameters were - Learning rate = [0.001, 0.01, 0.1] and number of epochs = [5,10,20,50]
  + For each combination of learning rates and number of epochs I trained a model using the training data, evaluated the model using the validation data and recorded the weighted average loss
  + The hyper parameters were chosen so that they minimised the weighted average loss

Results of testing:

All results were the same as chance

Text

Description automatically generated

Text

Description automatically generated

**Average loss plots when sock 1 was left out,**

Note: the average losses were similar when other socks were left out as well

**Chart, line chart

Description automatically generated**

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Text

Description automatically generated with low confidence

Chart, line chart

Description automatically generated

Graphical user interface, application

Description automatically generated

Chart, line chart

Description automatically generated

Graphical user interface, application

Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated