Assignment 3

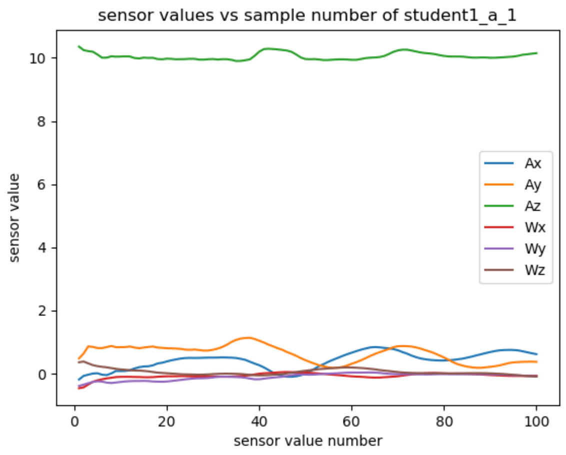
Fang, Cong Yu

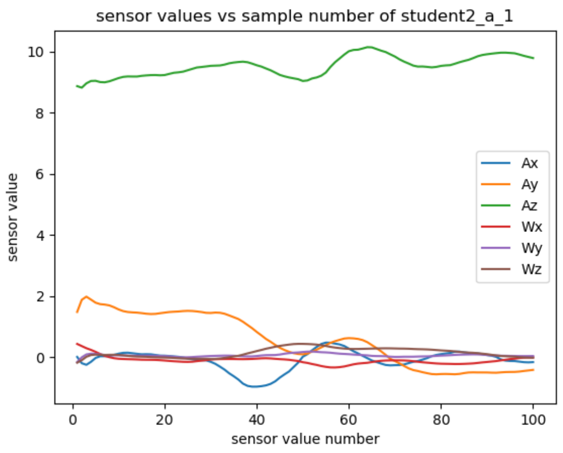
MIE 324

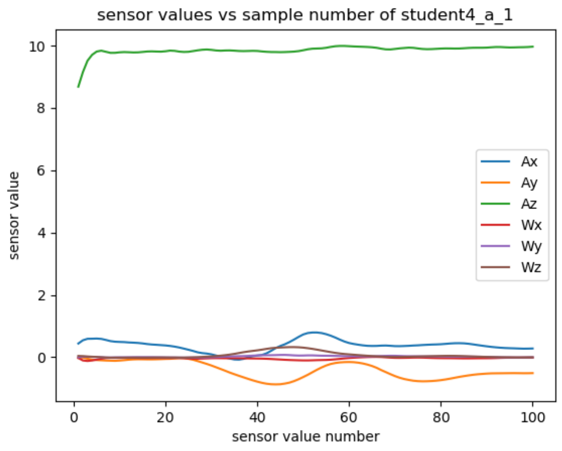
October 20th, 2018

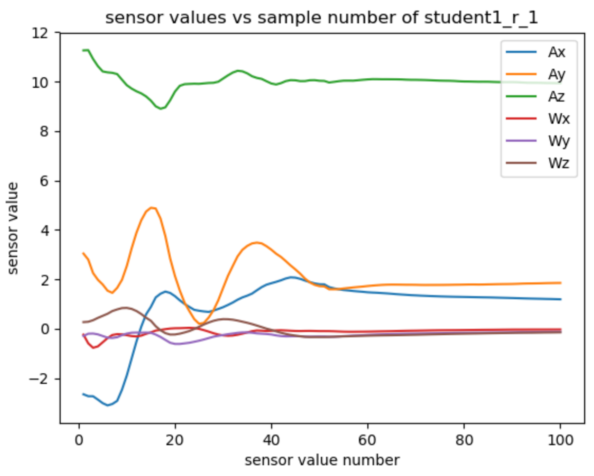
2 Data Pre-processing

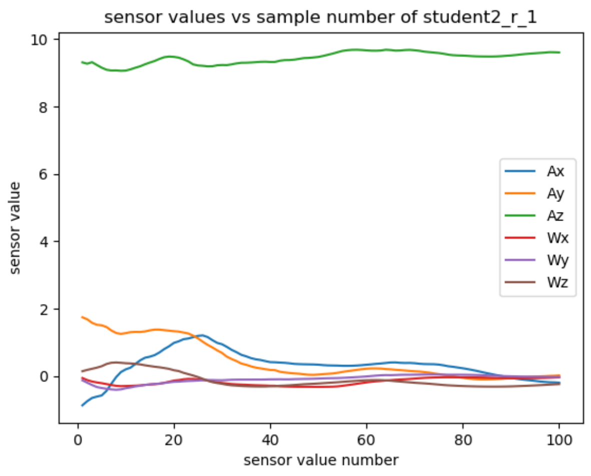
2.2 Understanding the dataset

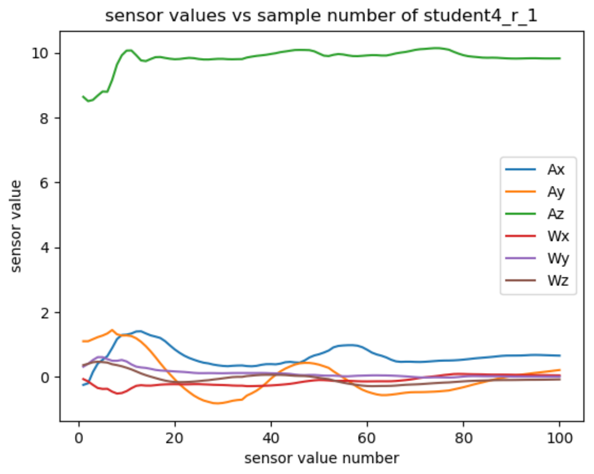








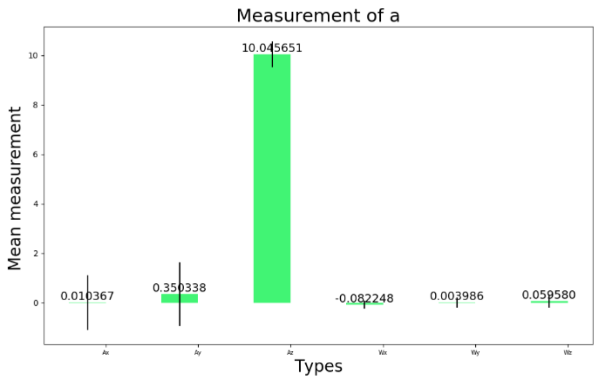


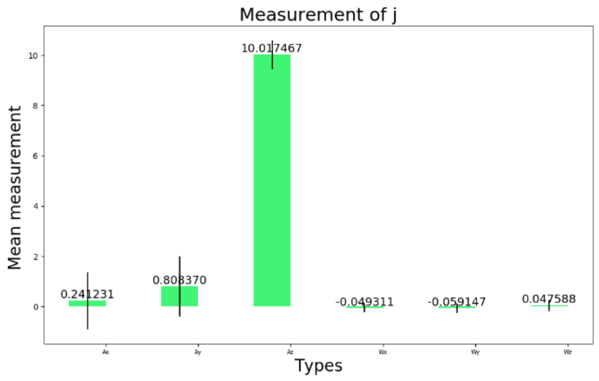


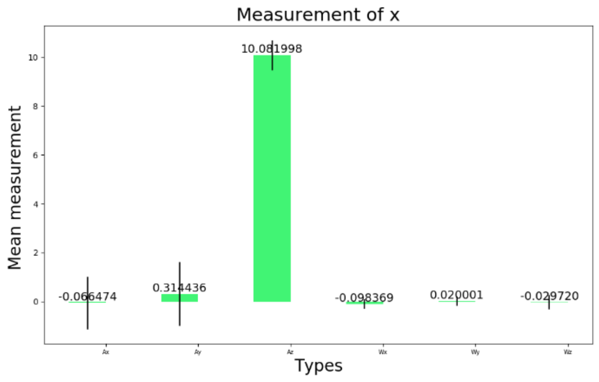
1. Gestures for “a” have two bumps in Ax and the bumps in Ay are at relative same positions as Ax. Wz has a small bump in the middle. The rest of curves are relatively flat. Gestures for “r” have also two bumps in Ax and Ay at relative same positions but they occurred before those bumps of “a”. Wz has two small bumps.
2. It makes sense when drawing an “a”, we first drew a circle and then we changed direction and drew the tail hence there is acceleration in both x and y direction as well as changes in angular velocity on the x-y plane. Also, in similar manner, drawing the letter “r” involves changes in directions in x and y directions as well Wz. The rest of the curves are relatively flat since we didn’t have major motions in z direction or rotations in x or y directions.
3. In order to classify the gestures, the receptive fields will need to cover all sensor values so that the model can learn all the information recorded by the sensors.

   The receptive field needs to be large enough in the *time* dimension to be able to detect the bumps in the different features

2.3 Basic statistics







1. It is hard to distinguish different letters from the bar graphs since all values except for Az are really small. But we can still see some differences by eye. For example, Ax and Ay for 'j' are the largest; those for 'a' are moderate and those for 'x' are the smallest.
2. It might be possible but I don't think the neural network would be able to classify the gestures accurately based on these average values since average values are not representative. For example, we need to know the relative positions of the bumps (from part 2.2) rather than an average over time and instances. The positive and negative values will end up cancelling each other out after taking the sum.

2.5 Train-validation split

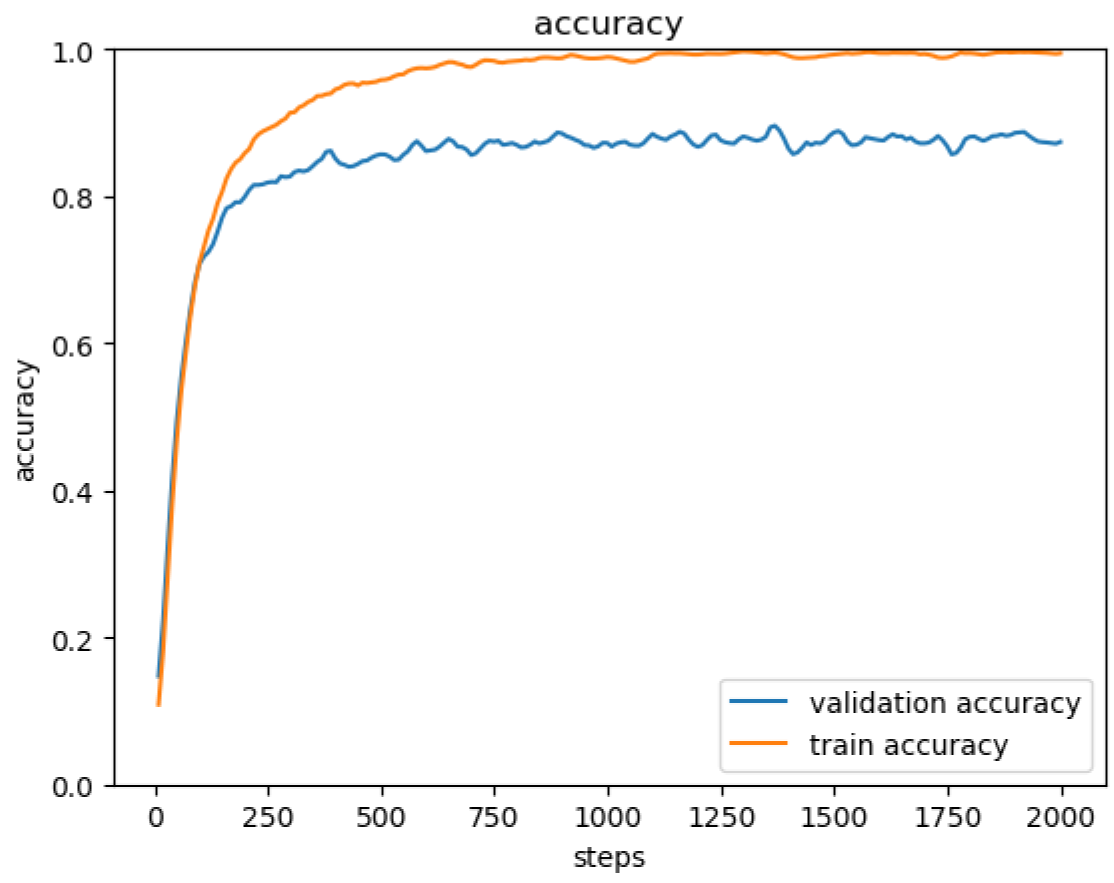
1. Having a test set prevents us from overfitting the hyperparameters. We would like to see how the model perform on unseen dataset (i.e. the test set).

I started with 20% of data as validation data since this was used in previous assignments. Eventually I switched to 10% since we have relatively little data to work with this time. I didn't use any smaller fraction since if the validation set is too small, the validation results might not be accurate enough.

3 Training

3.5 Plotting

Best validation accuracy: 89.80% (epoch 46)



4 Test performance

4.2 Explanation/Source of Accuracy

I used 2 layers of 'conv1d' followed by 3 layers of 'Linear'. It takes input as a tensor with size [batch_size, 100, 6]. '6' represents the 6 different sensor measurements (Ax, Ay Az, Wx, Wy, Wz) and '100' is the number of time steps of each measurement. The first CNN layer takes input_channel = 6, kernel_size = 14, kernel_num = 64, stride = 1, padding = 7. The second layer input_channel = 64, kernel_size = 14, kernel_num = 64, stride = 1, padding = 7. Each of the two convolution layers are fed into relu and Maxpool1d. Then the outputs from CNN are fed into BatchNorm1d, Dropout, then then into the input layer of the MLP and followed by relu. Then the data are passed to the hidden layer with 256 neurons and relu, and then a 26-neuron output. The output is then passed into log_softmax so that it represents the probability of being one of the 26 letters with size [batch_size, 26]. It is then passed into CrossEntropy loss function.

2 CNN layers yield better results than a single-layer CNN because the kernels at the second layer were able to have a larger inception field to see a larger pattern. Introducing padding to the model increased the accuracy because it helps to keep the information at the edges. Pooling was used to reduce the size of the output from CNN. Dropout with small p-value is used to regularize and prevent co-adaptation of feature detectors. BatchNorm1d was also used as another way to regularize. The model trained using normalized data or unnormalized data yields similar results. Unnormalized data yields slightly higher accuracy. The models trained with or without Az, Wx or Wy yield similar results. Relu is compared against with other activation function (i.e. tanh). The results are similar but relu gives slightly higher accuracy. Different optimizers are considered (Adam and SGD) and Adam performed better with low learning rate. A moderate batch size (260) gave better results than extremely big or small batch sizes. One or multiple hidden layers of MLP gave similar results. Different loss functions were considered (CrossEntropyLoss and NLLLoss) and CrossEntropyLoss gave higher accuracy.

batch size = 260, lr = 0.001, epochs = 100, op= Adam, hidden size= 256, hidden_layer_num=1, act_func= relu, kernel_size=14, kernel_num=64, conv_layer_num=2, loss_fnc=CE, seed=0, pooling = 4, unnormalized data

5 Feedback

1. I spent about 25 hours
2. Testing out different hyperparameters and build a good model
3. Seeing the model eventually getting better
4. N/A
5. The recommendations are helpful