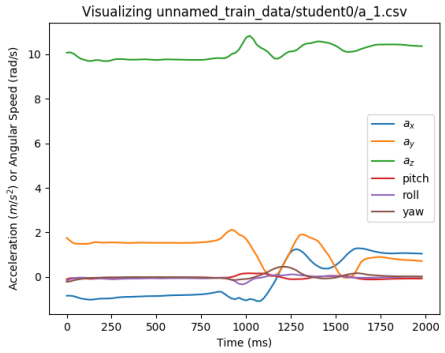
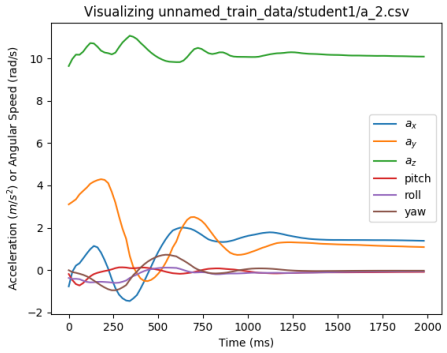
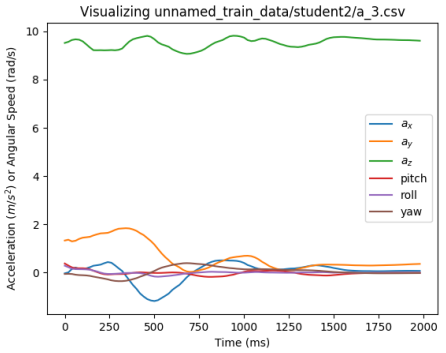
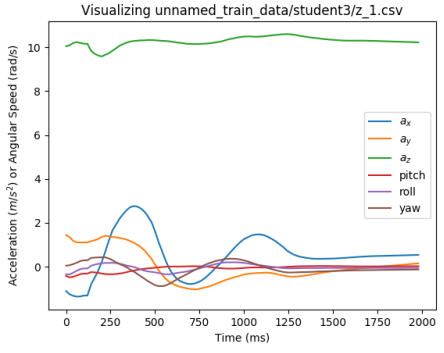
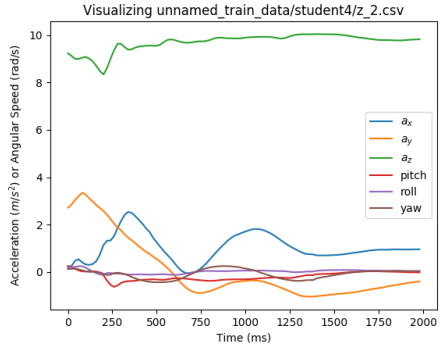
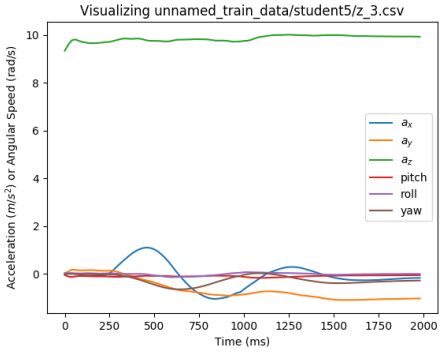
MIE324
Assignment 3: Report Write-up
Ryan Do (1001254117)
October 18, 2018

2.2

You should have 6 graphs in total, three for each gesture. Each plot should have 6 curves corresponding to x-acceleration, y-acceleration, z-acceleration, pitch, roll and yaw. Include these plots in your final report.

<table>
<tr>
<td>Sensor viz for “a”</td>
<td> 
</td>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td>Sensor viz for “z”</td>
<td> 
</td>
<td> 
</td>
<td> 
</td>
</tr>
</table>

1. *(2 points) Can you spot any patterns that distinguish one gesture from the other? Describe them (try to talk about at least 2 curves). Feel free to use informal language like "gesture 1 has 3 bumps on its x-acceleration curve but gesture 2 only has 2" or "both gestures have 2 bumps for the y-acceleration but the bumps occur ealier for gesture 2 than gesture 1".*

   For the "z" gesture, you can clearly see the partial oscillatory patterns in the acceleration graphs for x. There are two bumps in the x channel. However, y acceleration is generally on a steady decline. For the "a" gesture, there is an oscillation in both x and y for acceleration. Each one has two bumps.
   Another noteworthy observation is in the z channel for acceleration. The signal for the "a" gesture in the z acceleration graphs is less steady than for "z". One thing the gestures have in common is that pitch, roll and yaw are approximately 0 as well as z (accounting for gravity).

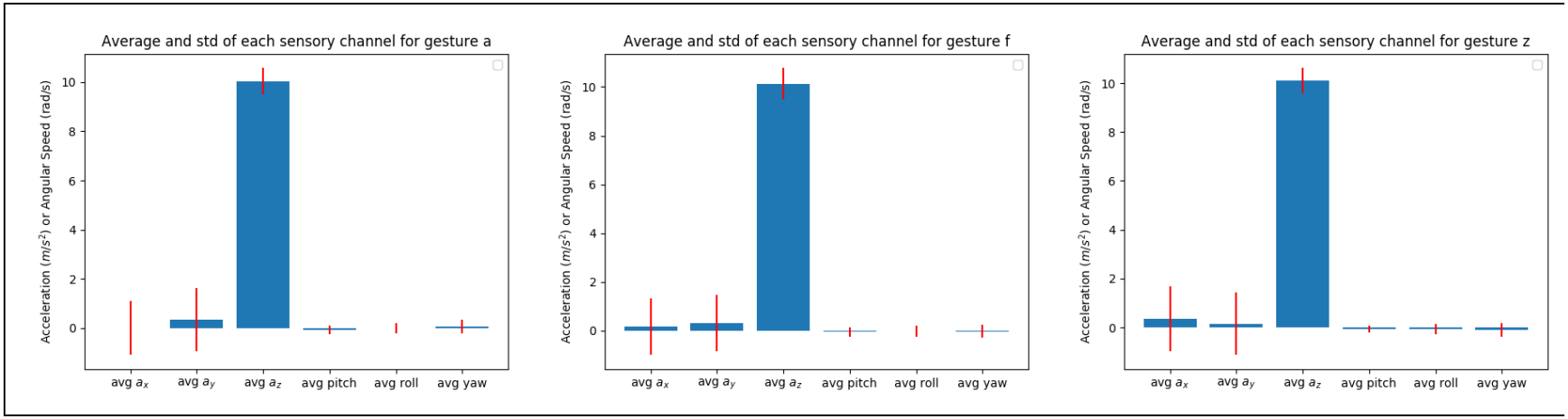
2. *(1 points) Do your observations make sense with the patterns you had to trace on your smartphone for each gesture? Why or why not? (the answer can be \no" as long as you provide some plausible reasons).*

   My observations do make sense with the "a" and "z" patterns. For the lower case letter a, it's comprised of a circle and a vertical line. The circle is represented by the periodic motion in both x and y (i.e. drawing a circle traces a sine graph in x and y). The z pattern y axis motion is purely downwards, therefore the observation with y acceleration steadily declining makes sense. The x axis motion for drawing z is periodic (zig-zag), there the observation with the sinusoidal x makes sense. My explanation for the z-axis acceleration being shakier on "a" is that it's a slightly more complex motion thus the z position might be more difficult to maintain.

3. *(2 points) Visualizations like these are often useful for helping decide the architecture of the neural network. The concept of the receptive field of a CNN arises from this question. The receptive field of a kernel (on any layer of the CNN) is the size of the original input region that the output from the kernel is affected by. The receptive field of a kernel on the first layer is simply the size of the kernel itself. Kernels on subsequent layers indirectly compute on larger regions of the raw input because of things like the pooling layer. What do our graphs suggest about the receptive field that our CNN needs to have in order to classify the gestures?*

   The receptive field size must be large enough to capture the features of the graphs of the 6 sensor channels. For example, for the "a" gesture, the receptive field must be large enough to span the overlapping of "bumps" between the x and y channels. If it is not large large enough, the essential features that distinguishes each gesture will not be captured. If it is too large, the training will require an excessive amount of computational resources.

2.3 *For each gesture, plot the average features in a bar graph. Plot the standard deviation as error bars. Include these plots in your final report.*



1. *(2 points) Looking at your bar graphs, can you classify the gestures by eye? Explain your answer.*

   The two channels that are relevant are x-acceleration and y-acceleration. The other four channels should be approximately constant for all gestures (which is true, shown by the figures). By eye, classifying the gestures only by two values (x and y) would be difficult since there are 26 letters. X and Y averages can be either positive or negative, Y > X or X>Y. As a result, there are only 8 possible combinations. 8 combinations cannot capture the unique features of 26 letters.
   However, we can tell each bar graph corresponds to their respective gestures. Average acceleration for a given axis should be proportional to the displacement in that axis from beginning to end of the gesture. For “a”, the phone ends in the same position as it started in the x component, but not in the y component. The graph reflects this. For z, the phone ends in a different position in both x and y. The graph reflects this.

2. *(2 points) Do you think a neural network could classify the gestures just from the average sensor values rather the original samples? Why or why not?*

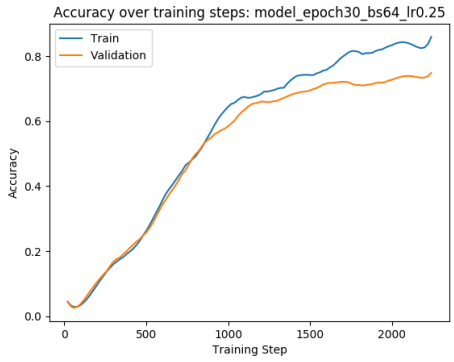
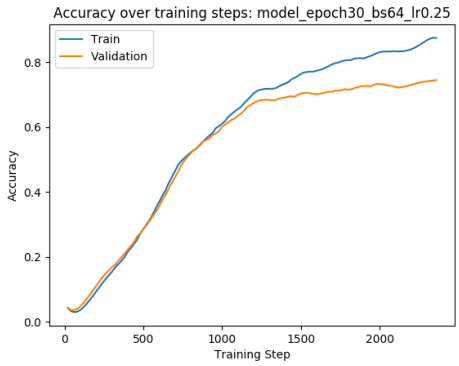
   I do not expect a neural network to be able to classify the gestures just from the average sensor values. The main reason for this is that the variance is too high to be able to generalize all the training data for a given gesture into a single average, let alone training an accurate model to predict gesture. Additionally, from the limited number of features in a couple channels’ average values, an accurate model of each gesture is simply extremely unlikely, if not impossible. Again, there are only 2 features in the pair of bar graphs represented by x and y: positive or negative averages, y > x, or y > x, which leads to only 8 combinations, much lower than 26.

2.5

1. *(2 points) Why is it necessary to evaluate a model on a test set instead of only relying on the validation set?*

   The validation set comes from the same distribution as the training set. Although testing on the validation set is good for checking overall over-fitting/under-fitting and hyper-parameter tuning, a test set is required for a final check on how generalizable the trained model is. The test set may come from a different source therefore having different characteristics/ distribution of values. It is a true test to see if a model can generalize to scenarios outside of the source data.
   Overall, the validation set is used for selecting/tuning a model, and the test set is the final test to evaluate the performance of the model.

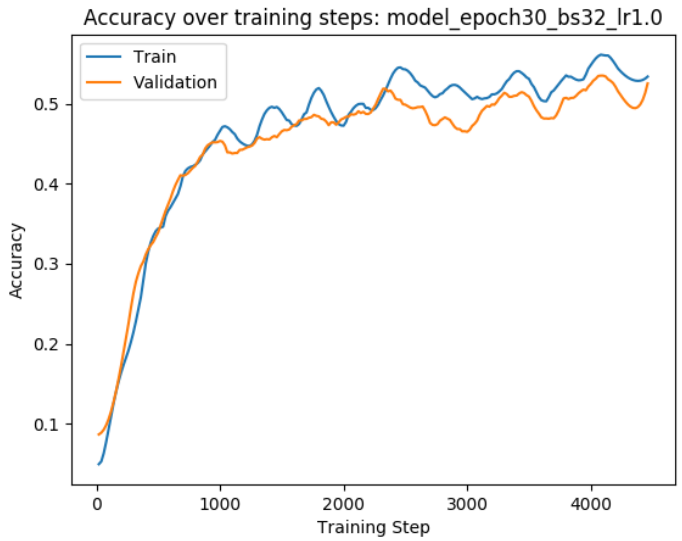
2. *(2 points) Please choose an appropriate data split for this assignment and split your data into a train data.npy and val data.npy. Note that you can come back to this section and change the train to validation ratio later on if you feel that a different setting would work better. In your report, explain how you arrived at your final choice of split.*

85:15 training validation split	90:10 training validation split
	

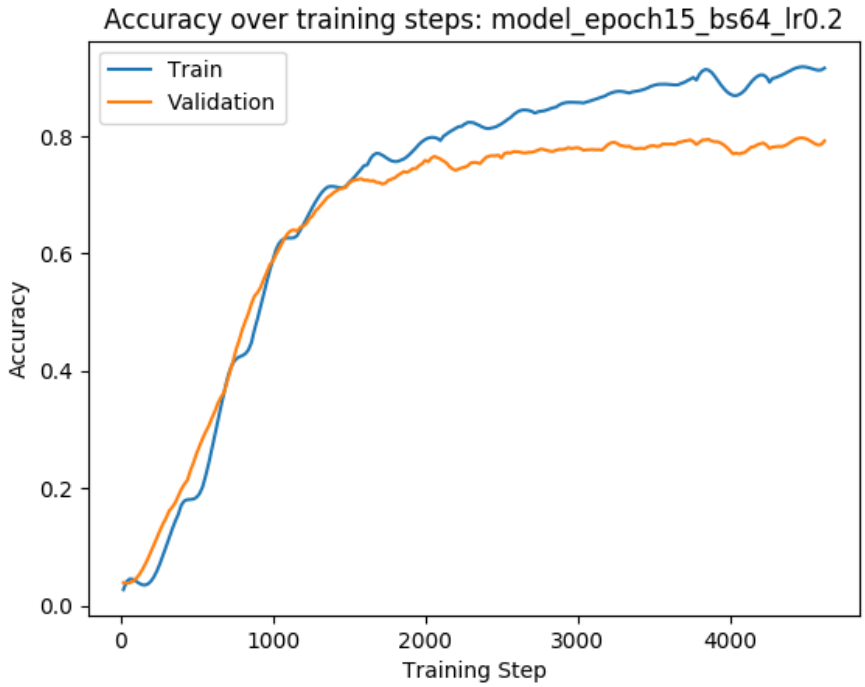
I ended up using a 90:10 training validation split. There is still enough of a validation sample size (559) to minimize variation in validation test results, but more importantly the diversity of training data is maximized with the greater proportion of training data.

3.5

1. *Include a plot of the training and validation accuracy of your model as a function of training steps. Include these plots in your report. Report the best validation accuracy the model achieves. Apply smoothing as needed.*



The first set of hyperparameters tried produced a model that achieves a 53.6% training accuracy and a 52.5% validation accuracy.



My best model achieved a validation accuracy of 80.5%.

List of hyper parameters:
batch size = 64
learning rate = 0.2
num_epochs = 13
validation-training data ratio: 10:90
evaluating validation data every 20 steps
3 convolutional layers, 3 linear layers:

```
self.conv1 = nn.Conv1d(in_channels=6, out_channels=10, kernel_size=11)
self.conv2 = nn.Conv1d(in_channels=10, out_channels=15, kernel_size=11)
self.conv3 = nn.Conv1d(in_channels=15, out_channels=20, kernel_size=11)
self.fc1 = nn.Linear(in_features=20*70, out_features=64)
self.dropout = nn.Dropout(p=0.2)
self.fc2 = nn.Linear(in_features=64, out_features=32)
self.fc3 = nn.Linear(in_features=32, out_features=26)
```

tanh act functions on the convolutional layers. relu, sigmoid on first two linear layers.

data augmentation:
time warp ratio: 4/11
time slice size: 11

4.2
Describe how you achieved the accuracy that you did - provide a clear description of each technique you used, and what it individually achieved. More points will be awarded for more attempts to improve.

I used a range of techniques to achieve my highest accuracy. The strategy was to improve accuracy through trivial methods first (which yield the most impact) then moving onto more advanced techniques. The techniques I used are as follows, in order:

1. Devineau, G., Xi, W., Moutarde, F., & Yang, J. (2018, June). Convolutional Neural Networks for Multivariate Time Series Classification using both Inter-and Intra-Channel Parallel Convolutions. In *Reconnaissance des Formes, Image, Apprentissage et Perception (RFIAP'2018)*. https://hal-mines-paristech.archives-ouvertes.fr/hal-01737771/document

   This paper by (Devineau, 2018) describes the use of a convolutional neural net to identify hand gestures captured by 3-dimensional rotation data at each finger join. The neural net architecture describe in the paper was used as reference for a starting point (3 cnn layers and 3 fc layers).

2. Tuning learning rate, batch size, epochs using grid search
   Gridsearch, or parameter sweep was used to pick optimal values for each of these hyper parameters by exploring the entire space. Learning rate must not be too low to stretch out training time, but not too high to overshoot the min of the loss function. 0.2 worked well here. A batch size of 64 allowed quicker convergence to a minimum while providing enough randomness to jump out of local minima. A corresponding number of epochs maximized validation accuracy while not being so long as to take a long time training and/or overfit.

3. Selection of activation function
   Choice of activation function seemed to have the highest impact on the model accuracy.
   Tanh performs well on the convolutional layers. Relu on the fully connected hidden layers prevents gradient fading over training steps (The extreme ends of the sigmoid function have nearly 0 gradient thus weights may be stuck at those values over the training process).

Sigmoid on the output layer gave best results.

4. Changing validation/training split to 90:10. There is still enough of a validation sample size (559) to minimize variation in validation test results, but more importantly the range of training data is maximized.

5. Selecting number of hidden layers (convolutional and fully connected):
   Simply adding more convolutional layers doesn't necessarily work better (3->6) – it decreased validation (overfitting) accuracy from 70 to 50. This also slowed training process by multiple times. 3 convolutional layers and 3 fully connected layers seemed to work best.

6. Adding momentum to the stochastic gradient descent optimizer had a negative effect.

7. Removed gyroscope channels to prevent overfitting? Doesn't work – it decreased validation accuracy. Perhaps the gyroscope data contains key features for the gestures.

8. Used batch normalization: increased training speed and reduced likelihood for overfitting.

9. Tried Adam optimizer, doesn't work as well – lower training and validation accuracy.

10. Use of dropout: learns the most robust neural connections and drops the rest. This prevents overfitting, but may cause mis-training of the model. Dropout also causes the training process to take a lot longer. Dropout didn't provide any benefits so it was not included in the model.

11. data augmentation for the time series sensor data

See augment_data.py: Two new functions were created for data augmentation: timewarp() and jitter(). This took a significant amount of time but it paid off.

a. The timewarp() function takes a set of training data (many instances of 100x6 acceleration data) and distorts the timescale, outputting a novel synthesized dataset. A fixed time slice size is fed in as an argument and alternating time slices are sped up and slowed by the same factor. Since the gesture is really only recognizable from position data, the double integral of this warped acceleration time series is a fairly different but still recognizable/valuable set of data. The end result is a multiplied quantity of data that is more diverse than where it came from.

   The idea to time-warp the sensor data came from the insight that the root source of variation in data was distortions in the shapes of the letters in terms of proportion. By distorting the acceleration data (and gyroscope data) at regular intervals, different variations of the gestures were synthesized. However, precaution was taken to not distort the acceleration data too much since the gestures are very sensitive to it (the acceleration data would be integrated twice to get the gesture shapes).

   This timewarping augmentation technique improved the model accuracy significantly. It brought the validation accuracy from low-mid 70% to 80%. It also introduces a set of new hyperparameters to tune (which I didn't have the computing resources or time to do extensively): time slice length and speed factor.

b. Jittering
   The jitter() function adds noise to a duplicated training set and adds a portion of it to the main training set. Noise is sampled from a normal distribution with mean 0 and arbitrary

standard deviation 0.01. The jittering emulates vibrations/noise in the sensor data due to unsteadiness, random perturbations, etc. It didn't seem to make too much of a difference but it didn't harm the training set either.

- Additional data augmentation ideas (time did not permit):
    - I could take different combinations of accelerometer /gyroscope channel data from different samples. Essentially mixing and matching x,y,z channels from different instances of the same gesture, this is somewhat similar to timewarp where it is another way to distort the traced letters.
    - Changing the magnitudes of the acceleration data by multiplying a scalar replicates variation in size of the gestures, as well as overall speed.

Describe your experience with assignment 3:

(a) How much time did you spend on assignment 3?

I spent approximately 30 hours total.

(b) What did you find challenging?

Since this was the first assignment done mostly from scratch, it required more critical thinking than the previous assignments. It was quite smooth until the creation of the model and training loop, where I ran into some issues with shapes of tensors, dataset format and model structure (# of inputs, outputs, etc.) though nothing was really unresolvable.
The time warping data augmentation technique worked in the end but took a decent amount of trial and error and bug resolving to get to.
This assignment also took a long time, which was especially challenging since it was due during the same week as two other proposals and a midterm among other things.

(c) What did you enjoy?

I enjoyed the feeling of accomplishment after building and training a model with a fairly high accuracy prediction (~80%). I overcame some hurdles in getting the trainer and model to work, so seeing it run and train for the first time was exciting. Additionally, I decided to attempt data augmentation with a method I thought of on my own which was to alternate speed up and slow down periods on the sensor data. This took a little while to get right but paid off since it improved my model accuracy by over 5%. I may try other methods in the future when I have time.

(d) What did you find confusing?

Trying to achieve a high accuracy was at times pure trial and error. It's often difficult to predict how the performance of the model will respond to changing hyper-parameters such as optimizer choice, loss function, activation functions, number of layers/ kernels, etc. Small changes in parameters like the choice of activation functions can make a ridiculously large difference (20%+ in test accuracy).

(e) What was helpful?

Getting to learn the debugger and being fluent with it really sped things up in terms of error resolving and in general. The previous two assignments (1 and 2) were good inspiration and experience in doing assignment 3 more smoothly. The lectures on convolutional neural nets gave a good high level background on how they should be implemented.