1 Data Description

Data was built from recordings of 30 subjects performing daily living activities while carrying a smartphone with embedded accelerometer and gyroscope. The goal is to use these readings to classify an activity according to the three classes described below.

- **Number of Cases:** 5405
- **Classes Description and Sizes**

Walking C_1	Sitting C_2	Standing C_3
1722	1777	1906

Table 1: *Classes for identification task and their number.*

- **Number of Descriptors:** 561
- **Descriptors Information**

  The selected features were computed from the accelerometer and gyroscope triaxial raw signals tAcc-XYZ and tGyro-XYZ. These time domain signals (prefix 't' to denote time) were captured at a constant rate of 50 Hz. Then they were filtered using a median filter and a 3rd order low pass Butterworth filter with a corner frequency of 20 Hz to remove noise. Similarly, the acceleration signal was then separated into body and gravity acceleration signals (tBodyAcc-XYZ and tGravityAcc-XYZ) using another low pass Butterworth filter with a corner frequency of 0.3 Hz.

  Subsequently, the body linear acceleration and angular velocity were derived in time to obtain Jerk signals (tBodyAccJerk-XYZ and tBodyGyroJerk-XYZ). The magnitude of these three-dimensional signals were calculated using the Euclidean norm (tBodyAccMag, tGravityAccMag, tBodyAccJerk-Mag, tBodyGyroMag, tBodyGyroJerkMag).

  A Fast Fourier Transform was applied to some of these signals producing fBodyAcc-XYZ, fBodyAccJerk-XYZ, fBodyGyro-XYZ, fBodyAccJerkMag, fBodyGyroMag, fBodyGyroJerkMag. The prefix 'f' indicates frequency domain signals.

  The descriptors D_i from these signals are:

  - mean(): Mean value
  - std(): Standard deviation
  - mad(): Median absolute deviation
  - max(): Largest value in array
  - min(): Smallest value in array
  - sma(): Signal magnitude area
  - energy(): Energy measure. Sum of the squares divided by the number of values.
  - iqr(): Interquartile range
  - entropy(): Signal entropy

- arCoeff(): Autorregresion coefficients with Burg order equal to 4
- correlation(): correlation coefficient between two signals
- maxInds(): index of the frequency component with largest magnitude
- meanFreq(): Weighted average of the frequency components to obtain a mean frequency
- skewness(): skewness of the frequency domain signal
- kurtosis(): kurtosis of the frequency domain signal
- bandsEnergy(): Energy of a frequency interval within the 64 bins of the FFT of each window.
- angle(): Angle between to vectors.

Additional vectors were obtained by averaging the signals in a signal window sample. These are used on the angle() variable: gravityMean, tBodyAccMean, tBodyAccJerkMean, tBodyGyroMean, tBodyGyroJerkMean. A complete list of these descriptors is found in the Appendix 7.1.

The mean, range, and standard deviation of each of the descriptors are found in Appendix 7.2.

- **Training and Test Sets**

The data was divided into sets, 3886 of which where used for training while the remaning 1519 were designated for testing. Details of the selection were summarized below.

Training Set

Classes	Frequency	Proportion
C_1	1226	0.317698886
C_2	1286	0.333246955
C_3	1374	0.349054159

Test Set

Classes	Frequency	Proportion
C_1	496	0.326530612
C_2	491	0.323238973
C_3	532	0.350230415

Table 2: *Summary of training and data set selection.*

2 MLP Auto-encoder Architecture

Here we describe the structure of the MLP classifier used in this study. We used architecture with three layers L_0, H, L_1, and L_2, of dimensions $p_0 = 561$, h, and $p_1 = 3$ respectively. Additinal two layers of softmax and argmax, both of dimension equal to p_1 and 1 respectively, were introduced to facilitate learning and class coding. See Figure 1 for a visualization.

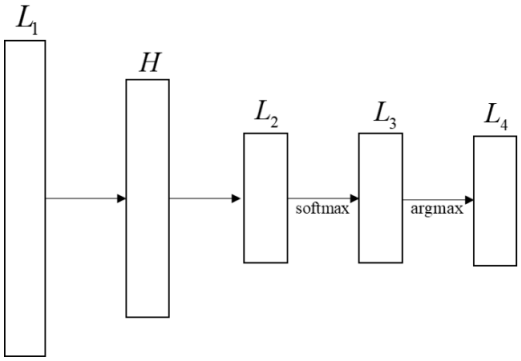


Figure 1: *MLP architecture used in the simulations.*

Each class was coded as either 1, 2, or 3 with the natural assignment. Such is obtained by taking the composition of `argmax` and `softmax` and applying this to layer L_2. The response function used is the RELU function $f(z) = max(z,0)$. Since the data set contains both positive and negative values, minimum at -1, we translated the data by adding 1 to all values thus making all values nonnegative. To facilitate automatic learning, the cross entropy function was applied to layer L_3 and gradient descent was performed on this loss function.

3 Hidden Layer Size Selection

To estimate a small dimension h of the hidden layer H, Principal Component Analysis (PCA) was applied to the input data using MATLAB. Resulting eigenvalues of which are visualized in the Figure 2. Eigenvalues from the PCA reveal that it will take the first 34 eigenvalues to account for 90% of the sum of all eigenvalues. Hence, we take this as one of the primary dimensions of the hidden layer in the training.

PCA was also applied to the input data per class. The number of eigenvalues preserving 90% of the total sum of eigenvalues was taken note and is summarized in the Table 3. As estimate for larger size $h_L > h_{90}$, we take the sum of the eigenvalues per class hence giving $h_L = 167$.

C_1	C_2	C_3
59	55	53

Table 3: *The number of eigenvalues, per class, which preserves 90% of the total sum.*

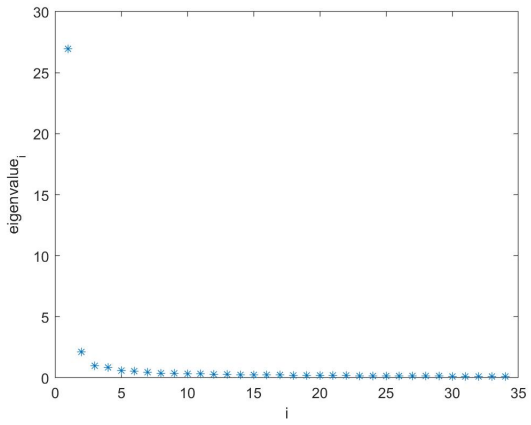


Figure 2: *Plot of the first 34 eigenvalues generated from the PCA of the input data accounting for 90% of the the sum of eigenvalues. The highest eigenvalue is 29.94, while the next two are 2.10 and 0.99 respectively.*

4 MLP Performance Analysis

Automatic learning was implemented using the Tensorflow environment in Python. The learning options used in the implementation in this section are summarized below. Exploration of other learning options and their impact to learning are discussed in Section 5.

1. **Initialization**: The tensorflow initializer `variance_scaling` was used. It is capable of adapting its scale to the shape of weights tensors. Samples are drawn from a normal distribution with mean zero and standard deviation $\sqrt{\frac{scale}{n}}$ where n is the number of input units in the weight tensor.

2. **Batch Learning**: Batches of size 64 are selected, without replacement, at random per epoch. To facilitate the shuffling of input, the function `numpy.random.shuffle` was used.

3. **Learning Rate**: Tensorflow provides a variety of learning options for the gradient descent step sizes. In this section, we used the used `exponential_decay` which applies an exponential decay function, of the form `learning_rate*decay_rate` $^{\texttt{(global\_step/decay\_steps)}}$, to a provided learning rate. The initial learning rate used is 0.0025 while the default decay rate value 0.96 and decay steps 100000 were taken.

We now look at the performance of the MLP by comparing the average cross entropy (ACRE) and classification accuracy for hidden layer dimensions on the training and test sets. Also, we present the batch average cross entropy (BACRE), and gradient and weights norms time plots obtained in the training phase. Plots seen in Figure 3 below were generated using batches of 64 with exponential learning rate for 100 epochs.

ACRE values were computed at the end of each epoch. The values for the hidden layer dimension h_{90} for both training and tests set are so similar in terms of steepness and shape, as characterized by almost overlapping trajectories which follow a downhill trend. As seen in Figure 4, the classification accuracy over time of this hidden layer dimension for both training and test sets also exhibit this familiar pattern. The values from training start off a little lower in accuracy than the test set, but and over time, both curves move together toward the 90% mark.

On the other hand, the ACRE over time for the hidden layer dimension h_L for both data sets are also identical in terms of steepness and shape. Also observed in Figure 4, the classification accuracy of the MLP with this architecture on test and training sets are very similar, as verified by almost perfect overlaps in their curves. Accuracy of the MLP reaches above 90% in both the data sets. In Tensorflow, the confusion matrix function has return value given by $C_{i,j}$, the number of examples with true class i predicted as belonging to class j. We see that the classification accuracy for C1 (walking) is extremely high, while some amount of confusion remains when attempting to classify between C2 (sitting) and C3 (standing). We will see later that this remains a common trend among the various different learning options.

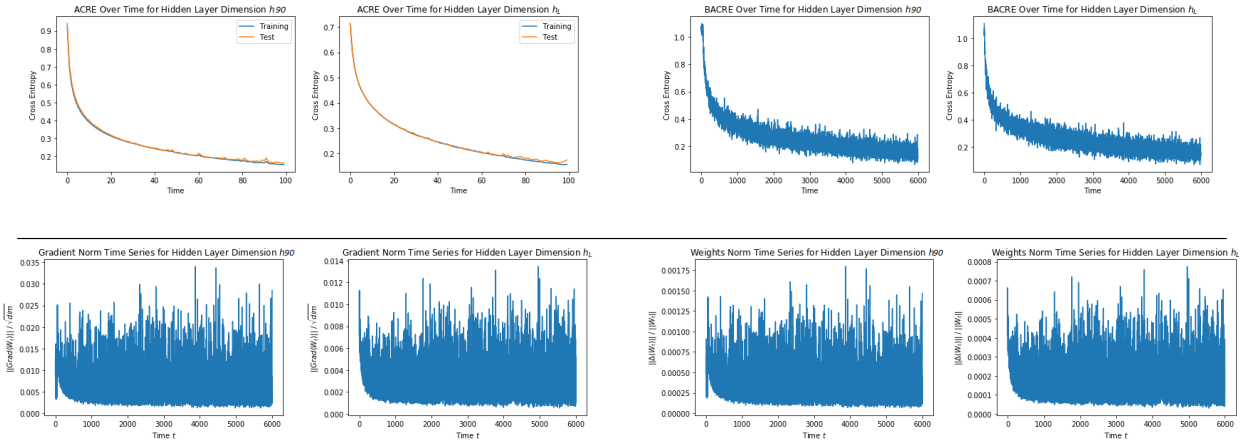
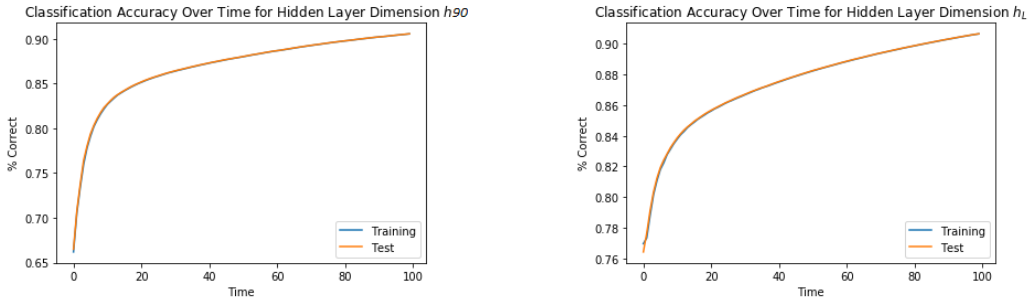


Figure 3: *Varying hidden layer H dimensions. Top left figures show time plots of comparisons of ACRE (per epoch) of the training and test sets, while top left present BACRE time plots for the training set. Bottom left figures show training time series of gradient norms, while bottom right present relative differences in weights norms.*



	C_1	C_2	C_3
$C1$	1226 (496)	0 (0)	0 (0)
$C2$	1 (1)	1146 (422)	139 (68)
$C3$	0 (0)	74 (30)	1300 (502)

	C_1	C_2	C_3
$C1$	1226 (496)	0 (0)	0 (0)
$C2$	1 (0)	1112 (408)	173 (82)
$C3$	0 (0)	52 (28)	1322 (504)

Figure 4: *MLP performance given various hidden layer H dimensions. Top left figures show time plots of accuracy of the MLP while bottom figures show confusion matrices on the training (black) and test (red) sets. Presented from left are results for hidden layer dimension* $h_{90} = 34$ *and* $h_L = 167$.

In Figure 3, we see that the training BACRE values in both cases follow a downward trend ultimately settling to a low of around 0.15. This is an expected behavior as we also observe this pattern in the training ACREs, only that there is a noticeable occurence of controlled oscillations in the BACREs, which can be explained by the fact that it shows more observations. Also, in the two cases, the norm of the gradient and the relative norms of the weights exhibit oscillations, which although do not move toward zero, maintain relatively small amplitudes.

5 Impact of Various Leaning Options

In this section, we look at how various parameter and learning options affect automatic learning. Specifically, we analyze cases for varying batch sizes, initialization schemes, and learning rates. Effects of different hidden layer dimensions on the test set were already presented in Section 4 and will no longer be touched here. Since implementations with h_{90} and h_L as hidden layer sizes showed similar performances as seen in the previous section, further analysis done here were carried out with the smaller h_{90} in the architecture.

5.1 Batch Size

Here, we look at the effects of different batch sizes to automatic learning. All simulations were done using the following specifications:

- dimension of hidden layer: $h_{90} = 34$
- learning rate: tensorflow exponential
- number of epochs: 100
- initializer: variance scaling

In terms of ACRE, as seen in Figure 5, all time plots exhibit a downward trend in both the training and test sets. Performing the best is batch size 64 having terminal training ACRE of 0.1531 and test ACRE of 0.1697. Increasing the batch sizes seem to bring diminished accuracy as the training (test) ACRE slightly increased to 0.2580 (0.2509) for batch size 128, and largely rose to 0.6786 (0.6571) for batch size 256. This is further supported by the accuracy level that the MLPs achieved which, starting with 91% correct test and training set classification of batch size 64, follows a downward pattern as the batch size is increased. Batch size 128 and 256 respectively registered 85% and 78% accuracy for both the training and test sets.

Examination of the confusion matrices (refer to Figure 6) show that the MLP for any batch size generally classifies class 1 (walking) the best, even performing perfect classification in the case of batch sizes 64 and 128 in both the training and test sets. It also seems that there is quite a difficulty in the separation of classes 2 (sitting) and 3 (standing) in all three cases. This is quite anticipated since walking is fundamentally different than the other two in terms of bodily movement.

Expectedly, training of the MLP with batch size 64 took the longest finishing after 48.6 seconds, followed by 128 with 32.46 seconds, and last by 256 which terminated after 25.18 seconds.

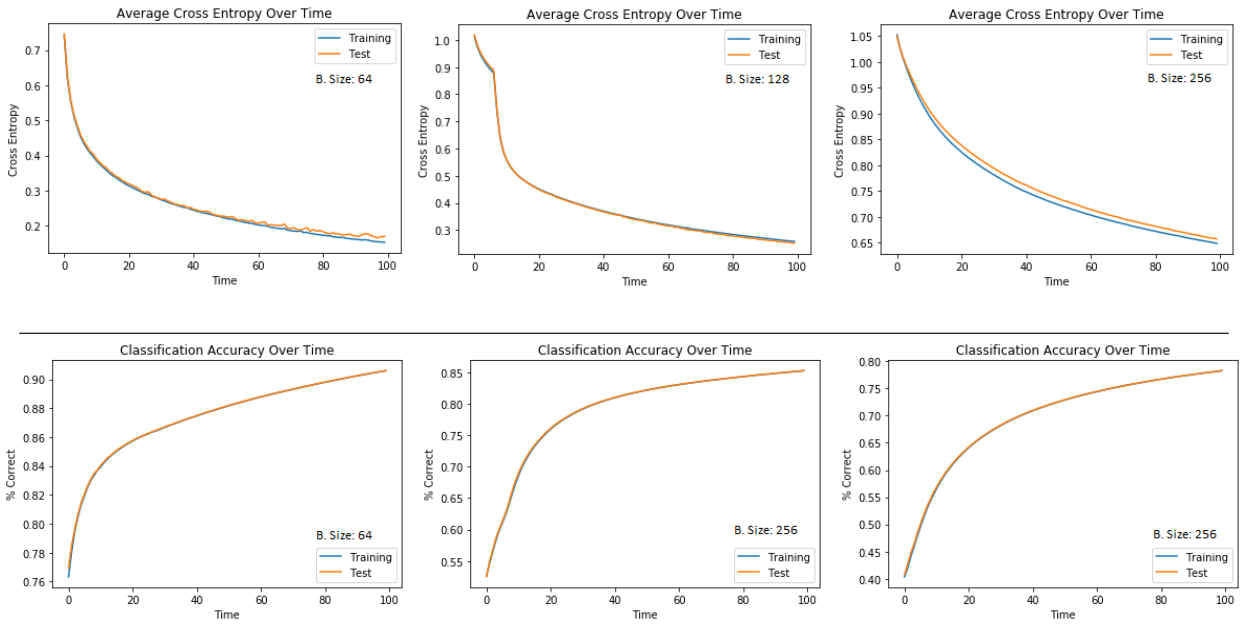


Figure 5: *Varying batch sizes. Top figures show time plots of comparisons of ACRE (per epoch) of the training and test sets. Bottom figures show time series of MLP accuracy in terms of percentage of correct classification. Presented from left are plots for batches of size 64, 128, and 256 respectively.*

batch size: 64

	C_1	C_2	C_3
$C1$	1226	0	0
$C2$	1	1149	136
$C3$	0	67	1307

batch size: 128

	C_1	C_2	C_3
$C1$	1226	0	0
$C2$	1	1060	225
$C3$	0	196	1178

batch size: 256

	C_1	C_2	C_3
$C1$	1143	32	51
$C2$	0	1009	277
$C3$	0	196	1178

	C_1	C_2	C_3
$C1$	496	0	0
$C2$	1	419	71
$C3$	0	31	501

	C_1	C_2	C_3
$C1$	496	0	0
$C2$	1	415	75
$C3$	0	57	475

	C_1	C_2	C_3
$C1$	432	45	19
$C2$	0	385	106
$C3$	0	59	473

Figure 6: *Confusion matrix of the MLPs with different batch sizes. Top figures are the matrices for the training set, while bottom figures are for the test set. Presented from left are tables for batches of size 64, 128, and 256 respectively.*

5.2 Initialization

Next, we examine effects of varying weights and threshholds initialization schemes to the learning process. Since it was found out in Section 5.1 that the MLP performed the best when fed with input in batches of size 64, such is used in the implementations in this subsection. All the simulations were done using the following parameters:

- dimension of hidden layer: h_{90}=34
- number of epochs: 100
- batch size: 64
- learning rate: tensorflow exponential

We observe learning for the three initialization schemes given below:

1. *Variance Scaling* - A tensor flow initializer that is capable of adapting its scale to the shape of weights tensors. Samples are drawn from a normal distribution with mean zero and standard deviation $\sqrt{\frac{scale}{n}}$ where n is the number of input units in the weight tensor.
2. *Ones* - All values are initialized to 1.
3. *Uniform Random* - Samples are drawn from a uniform distribution between -1 and 1.

The test and training sets results are comparable in the sense that the ACRE and accuracy values always almost overlap. As seen in Figure 7, in all three cases, we see a downward pattern in the ACRE on both the training and test sets, which is consistent with the upward trend in the time series of classification accuracy. Variance scaling initialization outperforms the other two, which on the onset already registered a relatively low ACRE (≈ 0.76) and high accuracy ($\approx 77\%$), that ultimately transitioned to an ACRE of approximately 0.76 and an accuracy of around 91%.

Initial ACRE values for both the ones and random uniform schemes are extremely high. Although we see a downhill trend in the plots for these two cases, gradient descent minimization only managed to bring the final values for both the training and test sets down to around 5 in the case of the ones scheme, and 3 for the random uniform method. An analogous pattern can be seen in the accuracy of the MLPs as ones and uniform random schemes registered values around 57% and 79% respectively. We also observe large oscillations in the early phase of the ACRE plots which in turn are translated to slower improvement of accuracy as seen in the correctness plot.

Deeper analysis through the confusion matrix, as presented in Figure 8, reveals that while the ones and random uniform schemes performed a near perfect classification of class 1 in both the training and test sets, they performed rather poorly in classifying the other classes. Worst-performing is the MLP with ones initialization which showed zero success in classifying class 3.

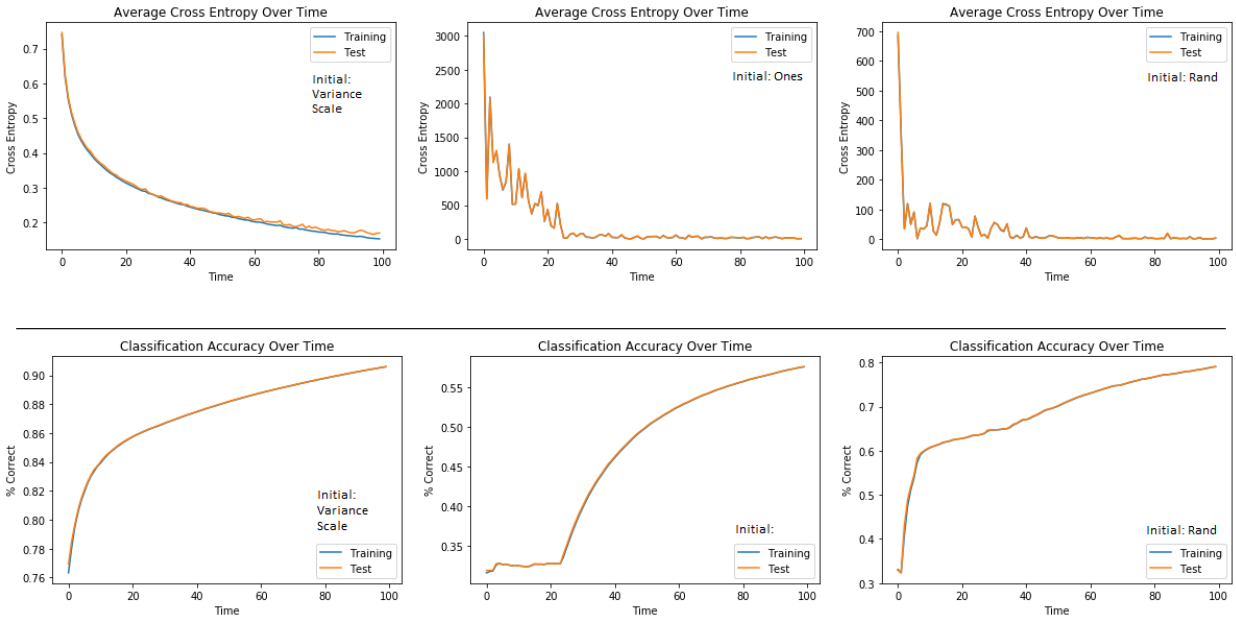


Figure 7: *Varying initialization schemes. Top figures show time plots of comparisons of average cross entropy (per epoch) of the training and test sets. Bottom figures show time series of MLP accuracy in terms of percentage of correct classification. Presented from left are plots for initialization schemes variance scaling, ones, and uniform random respectively.*

variance scaling

	C_1	C_2	C_3
$C1$	1226	0	0
$C2$	1	1149	136
$C3$	0	67	1307

ones

	C_1	C_2	C_3
$C1$	1226	0	0
$C2$	2	1284	0
$C3$	1	1373	0

uniform random

	C_1	C_2	C_3
$C1$	1225	1	0
$C2$	0	1278	8
$C3$	0	517	857

	C_1	C_2	C_3
$C1$	496	0	0
$C2$	1	419	71
$C3$	0	31	501

	C_1	C_2	C_3
$C1$	496	0	0
$C2$	4	487	0
$C3$	4	528	0

	C_1	C_2	C_3
$C1$	496	0	0
$C2$	0	490	1
$C3$	0	167	365

Figure 8: *Confusion matrix of the MLPs with different initialization schemes. Top figures are the matrices for the training set, while bottom figures are for the test set. Presented from left are plots for initialization schemes variance scaling, ones, and uniform random respectively.*

5.3 Gradient Step Size

In this subsection, we look into the effects of different learning rates or gradient step sizes to the MLP training. We consider learning rates of the form $\frac{c}{t}$, for $c = 0.1$ and 0.01, and compare them to the default exponential decay learning rate with parameter 0.0025. All simulations were done with the following parameters.

- dimension of hidden layer: h_{90}=34
- number of epochs: 100
- batch size: 64
- initialization: variance scaling

In Figure 9, we observe the familiar downward trend in the ACRE and upward pattern in the accuracy in all the cases. As the test and training sets results are very similar, we focus the description on the training set. In both the ACRE and accuracy categories, the exponential learning rate outperforms the other two exhibiting a final ACRE value of 0.1530 and accuracy of 91%. On the other hand, for $c = 0.1$ the terminal ACRE value achieved is 0.5447. Further decreasing c to 0.01 sets the final ACRE to be 1.002. Accuracy for $c = 0.1$ is acceptable at 82.22%, but for $c = 0.01$ we observe a dismal correct classification of 45%. This may indicate that a longer training time is required when these learning rates are employed as improvements in the perfomance indicators are still observable.

Similar patterns are seen in the confusion matrices (refer to Figure 10). Exponential decay performs best and followed by $c = 0.1$, both perfectly classifying class 1 cases. It is remarkable that decreasing c to 0.01 leads the MLP to misclassify class 1 cases to class 3, which is something unobserved in any other analysis done so far.

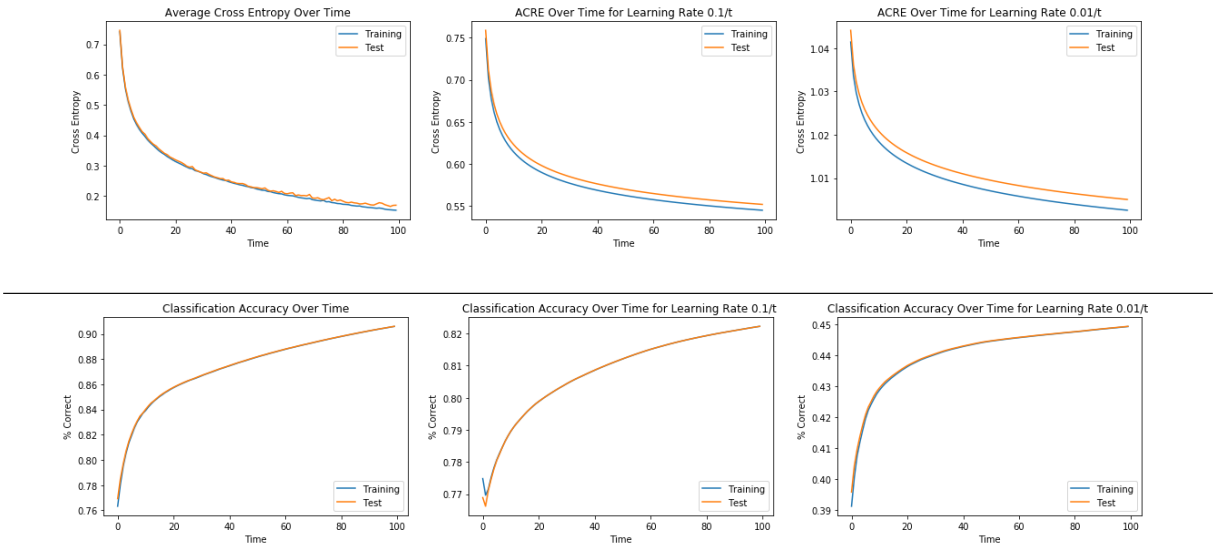


Figure 9: *Varying learning rates. Top figures show time plots of comparisons of average cross entropy (per epoch) of the training and test sets. Bottom figures show time series of MLP accuracy in terms of percentage of correct classification. Presented from left are tables for learning rates exponential decay,* $\frac{0.1}{t}$*, and* $\frac{0.01}{t}$*.*

exponential decay

	C_1	C_2	C_3
$C1$	1226	0	0
$C2$	1	1149	136
$C3$	0	67	1307

$\frac{0.1}{t}$

	C_1	C_2	C_3
$C1$	1226	0	0
$C2$	5	840	441
$C3$	6	166	1202

$\frac{0.01}{t}$

	C_1	C_2	C_3
$C1$	43	15	1168
$C2$	0	951	335
$C3$	0	202	330

	C_1	C_2	C_3
$C1$	496	0	0
$C2$	1	419	71
$C3$	0	31	501

	C_1	C_2	C_3
$C1$	496	0	0
$C2$	5	294	192
$C3$	8	63	461

	C_1	C_2	C_3
$C1$	8	2	486
$C2$	0	380	111
$C3$	0	202	330

Figure 10: *Confusion matrix of the MLPs with different learning rates. Top figures are the matrices for the training set, while bottom figures are for the test set. Presented from left are tables for learning rates exponential decay,* $\frac{0.1}{t}$*, and* $\frac{0.01}{t}$*.*

6 Post-learning Analysis of Hidden Layer Behavior

Here we give a profile of the different classes by looking at the hidden layer activity when presented with data of the same type. Hidden layer size was taken to be $h_{90} = 34$, as this provides a much simpler structure while maintaining desirable accuracy levels. As seen in Sections 4 and 5, we do not observe signs of overfitting of parameters, and hence we take the global step size m to be the final step number in the training.

First we performed PCA on the hidden layer output and it was found out that only three eigenvalues are needed to keep 99% of the total sum (see Figure 11 for visualization). This is an indication that the data can be classified properly into the three classifications as intended in this study, and an addition of hidden layers is no longer necessary.

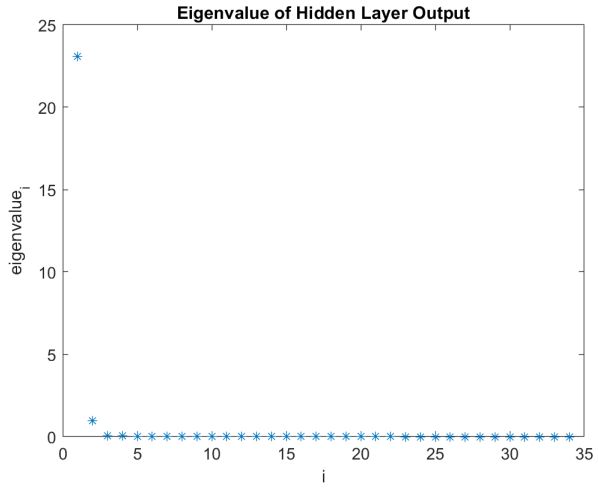


Figure 11: *Plot of the 34 eigenvalues generated from the PCA of the hidden layer output. The first three eigenvalues account for 99% of the the sum of eigenvalues*

To describe neuron activity per class, we first took the average neuron activity across the data set and set the mean values as threshhold. If class-wise average activity is above the threshhold computed, then the neuron is considered active, and is inactive otherwise. Figure 12 summarizes the activity profiles per class. Active neurons (coded as 1) are colored orange while the inactive neurons (coded as 0) are colored blue.

	CL1	CL2	CL3
H1	1	0	0
H2	0	1	1
H3	0	0	0
H4	1	0	0
H5	1	0	0
H6	1	0	0
H7	1	0	0
H8	0	0	1
H9	1	0	0
H10	0	1	0
H11	1	0	0
H12	0	0	0
H13	0	0	1
H14	1	0	1
H15	1	0	0
H16	0	0	0
H17	0	1	1

	CL1	CL2	CL3
H18	0	1	1
H19	0	1	0
H20	1	0	0
H21	0	0	1
H22	1	0	0
H23	0	0	1
H24	0	0	0
H25	0	0	0
H26	0	0	0
H27	0	1	1
H28	0	1	1
H29	0	1	1
H30	0	1	0
H31	0	1	1
H32	0	1	1
H33	0	1	1
H34	0	0	0

Figure 12: *Neuron activity profiles of each of the classes. Colored orange are active neurons while colored blue are inactive ones.*

Below we do a comparison of class profiles.

- $CL1$ vs $CL2$: We observe that for the first half of hidden neurons (i.e. $H1$ to $H17$) class 1 has more active neurons than class 2, with ratio of 3:1. Meanwhile, looking at the last half of the hidden layer neurons (i.e. $H18$ to $H34$) the situation reverses. We take note that whenever a neuron is active in one class, the other class is inactive, which means that the active neurons are characteristically associated to the class in focus.

- $CL1$ vs $CL3$: We see a similar pattern as in $CL1$ vs $CL3$, that is, there are more active neurons in first half of the set in class 1 and the situation reverses as we move to the next half. Also, we take note that except for $H14$ high activity for class 1 means inactivity in the other, and vice versa.

- $CL2$ vs $CL3$: The two profiles are very similar in terms of number of active neurons. Activity of hidden layer neurons 10, 19, and 30 is a characteristic of class 2, while activity of 8, 13, 14, 21, and 23 is of class 3.

- $CL1$ vs $CL2$ vs $CL3$: We see that in general $CL3$ has most number of active neurons with 14, followed by $CL1$ with 11, and by $CL2$ with 12. Seven hidden neurons(3, 12, 16, 24, 25, 26 and 34) are dormant as we have no active neurons in all three classes. We can also see that $CL1$ (i.e walking) is clearly different from the remaining two classes which is expected as this is the only class representing a kinetic action while the remaining two classes are stationary actions (sitting and standing). Active neurons unique to class 1 are 1, 4, 5, 5, 7, 9, 11, 15, 20, and 22. For class 2, unique are neuron 10, 19, and 30, while for class 3 we have 8, 13, 21, and 23.

Note that several neurons were never tagged as active, so further explorations may consider their deletion.