EEE-543 - Fall 2023 Mini Project 3: NN, Human Activity Recognition

Salih Deniz Uzel (deniz.uzel@bilkent.edu.tr)
22201382

I. Introduction

A 3 layer fully connected Neural Network (NN) is implemented in python programming language. 3 layer consist of 2 hidden layers and 1 output layer. Model is trained with given hyper-parameters in the assignment for the Human Activity Recognition (HAR) dataset. The dataset has 6 different classes from number 0 to 5 included. It has Training, and Test dataset. Validation dataset created from the 10% of the training dataset with the same class distribution.

In the report, options b, c, d of the assignment are explained together with their implementation details in the Section II. The assignment tasks related to evaluation and interpretation are explained in their corresponding options.

II. Implementation

A. Implementing the Neural Network

The network mainly consist of forward and backward propagation. Partial differential equations given in this section is implemented in the *backward* method of class NN. In order to make weight and bias updates, the change of the weight and bias values in the NN layers are calculated for the Cross Entropy Loss also called Negative Log Likelihood (NLL). The Δ (delta) notation will be used to present amount of change in the corresponding differential equation variable. To denote the change of the a layer's weight following notation is used $\Delta w^{t)} = dw^{(t)} = \frac{\partial L}{\partial w^{(t)}}$. This equation can be solved by using the chain rule. The general representation partial differential equation for calculating the amount of change for weights and biases with respect to (w.r.t.) NLL are given in the equations from (1), (9). The general formulas used for weight and bias updates are given in the (10), and (11). Calculations are implemented in vectorized way.

Vectorized calculations benefit of computer architecture to achieve $AX + b$ computations. All the losses are accumulated due to vectorized implementation. Therefore losses and parameter updates are scaled by the size of samples each epoch.

All the output vectors are encoded as one-hot vectors. For the the hidden layer of the network ReLu activation function, for the output Softmax activation function is used.

$$\frac{\partial L}{\partial a^{(t)}} = \frac{d}{da} NLL \tag{1}$$

$$\frac{dL}{a^{(t)}} = \frac{d}{da}(\frac{1}{n}\sum_{i=1}^{n} y_i(log(a_i))) \tag{2}$$

$$\frac{\partial L}{\partial z^{(t)}} = \frac{\partial \mathcal{L}}{\partial a^{(t)}} \frac{\partial a^{(t)}}{\partial z^{(t)}} \tag{3}$$

$$\frac{\partial L}{\partial w^{(L)}} = \Delta w^{(L)} = \frac{\partial L}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial w^{(L)}} \tag{4}$$

$$\frac{\partial L}{\partial b^{(L)}} = \Delta b^{(L)} = \frac{\partial L}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial b^{(L)}} \tag{5}$$

$$\frac{dL}{a^{(L)}} = \frac{1}{n}\sum_{i=1}^{n}(a_i - y_i) \tag{6}$$

$$\frac{da^{(t)}}{dz^{(t)}} = \frac{d}{dz}(\text{Activation Function}) \tag{7}$$

$$\frac{dz^{(L)}}{dw^{(L)}} = \frac{d}{dw^{(L)}}\left(w^{(L)}a^{(0)} + b^{(L)}\right) = a^{(L-1)} \tag{8}$$

$$\frac{dz^{(L)}}{db^{(L)}} = \frac{d}{db^{(L)}}\left(w^{(L)}a^{(0)} + b^{(L)}\right) = 1 \tag{9}$$

$$W^{(L)} := W^{(L)} - \text{LearningRate} \cdot \Delta w^{(L)} \tag{10}$$

$$b^{(L)} := b^{(L)} - \text{LearningRate} \cdot \Delta b^{(L)} \tag{11}$$

***Activation Functions*:**
The python implementation of activation function for both forward and backward propagation is given below. The backward propagation are given in the equations (3), (4), (5), and (7) implicitly.

1) ReLU

$$\text{Forward:} \quad f(x) = \max(0, x)$$

$$\text{Backward:} \quad \frac{df(x)}{dx} = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x \leq 0 \end{cases}$$

2) Softmax

$$\text{Forward:} \quad f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$$

$$\text{Backward:} \quad \frac{df(x_i)}{dx} = f(x_i) \cdot (1 - f(x_i))$$

***Weight Initialization and Running 16 Different Experiments*:** Model training for each run number was started with 3 different weights due to different numpy seeds. Weights are initialized with a numpy Uniform distribution between [-0.1, 0.1]. [42, 725, 1690] seeds were used for these 3 different initialization. Sample set accuracy calculated by averaging the accuracy results of these 3 different runs. Experiment parameters for each run along with their total training time in seconds, are given in the Table II. Experiment results are given in the Table IV, Table VI, Table VII, Table VIII. Table IV consist of mean and std values of the Top 1,2 and 3 test set accuracies of 3 different runs.

***Loss Function*:**
Cross Entropy Loss or Log Loss function is used for the error function. The Log Loss implemented in within the Loss calculation and back propagation. Both are implemented in vectorized way in Python.

***Learning Coefficient*:**
Learning coefficient [0.001, 0.01] implemented at weight and bias update w.r.t equation (10) and equation (11) during the *backward* method of NN class in the *_Layer* instance.

***Momentum*:**
Momentum also called running average methods makes the nn model less responsive to the last gradient updates by averaging the last gradients updates. This averaging results in dampening the oscillations due to stochastic learning process. The parameter v in the Equation (12)-(15) represents the averaged gradients.

The β coefficient is used to calculate the coefficient that indicates that the last n of the previously calculated gradients will be included in the average. If the $\beta = 0.9$, the gradient updated will be scaled by $1 - 0.9 = \frac{1}{10}$ and previous gradient update will be scaled by $0.9 = \frac{9}{10}$. In other words, new gradient update will contain 10% of the new gradient update and 90% off the previous gradient update.

$$v_{dW}^{(L)} := \beta \cdot v_{dW}^{(L)} + (1 - \beta) \cdot \Delta w^{(L)} \tag{12}$$

$$v_{db}^{(L)} := \beta \cdot v_{db}^{(L)} + (1 - \beta) \cdot \Delta b^{(L)} \tag{13}$$

$$W^{(L)} := W^{(L)} - \text{LearningRate} \cdot v_{dW}^{(L)} \tag{14}$$

$$b^{(L)} := b^{(L)} - \text{LearningRate} \cdot v_{db}^{(L)} \tag{15}$$

***Dropout*:**
Dropout is one of the regularization techniques which involves shutting random neurons. These neurons outputs 0 value do not involve gradient update during the epoch. The aim is to introduce samples to different neurons each epoch in order balance the contribution of each neuron to decision process. In the implementation of dropout, layer activations element-wise multiplied by the matrix of randomly selected 0 and 1 values. The distribution of the zeros and ones defined by dropout value between 0 and 1 included. This Matrix is also called mask. This mask also applied to $\frac{dL}{da}$ during the back propagation. In the assignment implementation, the mask also scaled by $\frac{1}{p}$ in order to maintain the magnitude of the activations. The formula is given in the following equation.

Let $\mathbf{M}$ be a matrix with the same shape as $a^{(L)}$ layer activation a.

$$\text{MASK:} \quad \text{mask} = \begin{cases} \frac{1}{p} & \text{if } M < p \\ 0 & \text{otherwise} \end{cases}$$

$$\text{Forward:} \quad a^{(L)} := a^{(L)} \odot \text{mask}$$

$$\text{Backward:} \quad \frac{dL}{da^{(L)}} := \frac{dL}{da^{(L)}} \odot \text{mask}$$

***Top n Accuracy Metric*:**
Implemented as get_top_n_correct instance method in NN Class.

***Early Stopping*:**
If Top 1 Accuracy does not improve on the Test Set more than $1e-5$ for 20 consecutive epochs training is stopped.

***Input Data*:**
The class distribution of Training, Validation, and Test sets are given in Table I. The amount of samples belonging to the classes is not evenly distributed. Therefore, the model can learn better the classes where it sees more examples.

TABLE I: Training, Validation, Test Set Class Distribution

	Classes					
Data Sets	1	2	3	4	5	6
Training	1103	966	887	1157	1237	1266
Validation	123	107	99	129	137	141
Test	496	471	420	491	532	537

The confusion matrices of the models trained with this distribution are given in Figure 3 and Figure 4. Interpretation is given in the Section IV.

***Processing Output Data*:**
Metrics belong to each run stored as json file. These json files and model weights for best accuracy is provided with the report. Lastly, weights belong to hidden layer, input layer, and output layer are plotted and provided as a heat map. It is possible to observe overly activated or less activated neurons.

Loss and Accuracy plots of each run given in Fig **??** and Fig **??**. Plots are given in the order of Table II. The discussion about Top1, Top2, and Top3 Accuracies are given in Section IV.

III. 3 Run Average Results

All the result given in this section are the average of 3 different runs for the following numpy seeds **[42, 725, 1690]**.

- Table II contains the hyper-parameters, epoch number of the best achieved accuracy, and running time of the 16 different run.
- Table III contains Training, Validation, and Test set accuracy at the weight initialization. It can be seen that all the initial conditions are identical for $N_2 = 100$ and $N_2 = 200$.
- Table IV contains Top 1, Top 2, and Top 3 accuracy of the Test set for the detection of best configuration found.
- Table VI, Table VII, and Table VIII are showing the Training, Validation, and Test set Top 1, Top 2, and Top 3 accuracy respectively
- Table V, contains the Top1 accuracy results of the best performing configuration found in option c for the Training, Validation, and Test sets.
- Figure 1 and Figure 2 are the loss and accuracy plots. Left plot in a row is the loss plot of the Training, Validation, and Test sets. Right plot in a row is the accuracy plots of the Training, Validation, and Test sets. In the accuracy plot; Top 1, Top 2, and Top 3 accuracies are given together. Each row corresponds to Run No, respectively.
- Figure 3 and 4 are the heat maps of the confusion matrices for 16 different runs. Left plot is the confusion matrix of the Training set. Right plot is the confusion matrix of the Test set. Each row corresponds to Run No, respectively.

TABLE II: Model Training Parameters, Running Times, and Epoch of the Best Accuracy for the given 16 Different "Run No".

Run No	Batch Size	N	Learning Rate (α)	Beta (β)	Epochs	Best Epoch	Time (s)
1	50	100	0.001	0	100	99	24.897
2	50	100	0.001	0.9	100	100	32.792
3	50	100	0.01	0	100	14	8.715
4	50	100	0.01	0.9	100	48	23.025
5	50	200	0.001	0	100	100	28.974
6	50	200	0.001	0.9	100	100	37.327
7	50	200	0.01	0	100	11	9.065
8	50	200	0.01	0.9	100	39	22.637
9	1	100	0.001	0	100	1	117.536
10	1	100	0.001	0.9	100	12	321.411
11	1	100	0.01	0	100	1	121.150
12	1	100	0.01	0.9	100	42	612.789
13	1	200	0.001	0	100	1	142.762
14	1	200	0.001	0.9	100	11	333.894
15	1	200	0.01	0	100	1	149.776
16	1	200	0.01	0.9	100	13	354.765

* N is the number of neurons in the second hidden layer.
* Best epoch determined by early stopping condition. If Top 1 Accuracy does not improve on the Test Set more than 1e-5 for 20 consecutive epochs.

IV. Interpretations of the Results

All the models are initialized with the uniform distribution within the given boundaries. After the first gradient update Top1, Top2, and Top3 accuracies updated to almost 16%, 32% and 48% respectively. The reason of seeing these numbers is random initialization. Probability distribution of the random initialization is $1/(number \quad of \quad classes)$ which is $1/6 \approx 16\%$ for Top1 accuracy. $2/6 \approx 33\%$ and $3/6 \approx 48\%$ for Top2, Top3 Accuracy respectively.

TABLE III: Training, Validation, Test Set Average **Top 1** Accuracy of the NN **Initializations** for 16 Different Run

Run	Training		Validation		Test	
No.	μ	σ	μ	σ	μ	σ
1	12.364	0.000	12.092	0.000	11.232	0.000
2	12.364	0.000	12.092	0.000	11.232	0.000
3	12.364	0.000	12.092	0.000	11.232	0.000
4	12.364	0.000	12.092	0.000	11.232	0.000
5	13.271	0.000	13.451	0.000	14.116	0.000
6	13.271	0.000	13.451	0.000	14.116	0.000
7	13.271	0.000	13.451	0.000	14.116	0.000
8	13.271	0.000	13.451	0.000	14.116	0.000
9	12.364	0.000	12.092	0.000	11.232	0.000
10	12.364	0.000	12.092	0.000	11.232	0.000
11	12.364	0.000	12.092	0.000	11.232	0.000
12	12.364	0.000	12.092	0.000	11.232	0.000
13	13.271	0.000	13.451	0.000	14.116	0.000
14	13.271	0.000	13.451	0.000	14.116	0.000
15	13.271	0.000	13.451	0.000	14.116	0.000
16	13.271	0.000	13.451	0.000	14.116	0.000

* 16 run initialization accuracy (%) for numpy seed value 42.

TABLE IV: **Test Set** 3 Run Average Results of **Top 1**, **Top 2**, and **Top 3** Accuracy of the 16 Runs with different configurations.

Run	Test					
	Top 1		Top 2		Top 3	
No.	μ	σ	μ	σ	μ	σ
1	91.483	0.289	99.446	0.058	100.000	0.000
2	91.641	0.424	99.468	0.070	100.000	0.000
3	88.949	3.463	99.468	0.195	99.966	0.048
4	94.661	0.943	99.525	0.100	99.955	0.016
5	92.671	0.340	99.548	0.131	100.000	0.000
6	92.965	0.598	99.570	0.139	100.000	0.000
7	88.836	3.272	99.480	0.153	99.955	0.064
8	94.774	1.034	99.683	0.125	99.932	0.000
9	84.052	2.978	98.055	0.530	99.943	0.032
10	95.068	0.623	99.604	0.178	99.932	0.028
11	18.222	0.000	36.274	0.000	52.935	0.000
12	95.668	0.647	99.695	0.194	99.887	0.032
13	85.579	3.048	98.733	0.513	99.932	0.096
14	95.329	0.764	99.695	0.139	99.910	0.032
15	18.222	0.000	36.274	0.000	52.935	0.000
16	**95.329**	**0.514**	**99.627**	**0.073**	**99.943**	**0.042**

* 16 run test set accuracy (%) for numpy seed value 42.
* 16th run has the best configuration that yields the best accuracy (%).

A. *Effects of the N_2*

Effects of the N_2 on the accuracies can be observed by comparing the results within the same batch size runs. In the Table VI Run No (1,5), (2,6), (3,7), and (4,8) pairs for the given batch size 50 and Run No (9,13), (10,14), (11,15), and (12,16) pairs for the given batch size 1 are almost equal. It can deduced that the using 100 and 200 Neurons for second hidden layer did not improved the overall accuracy. This could be the indication of the neural model with $N_2 = 100$ is complex enough to learn 6 different classes and increasing the complexity with $N_2 = 200$ does not make significant improvement.

TABLE V: Training, Validation, Test Set **Top 1** Accuracy of the Run No: 16 Best Performance Configuration.

Acc.	Training	Validation	Test
Top1	98.176	98.098	95.329
Top2	100.000	100.000	99.943
Top3	100.000	100.000	99.627

TABLE VI: 3 Run Average of **Top 1** Accuracy Metric Training, Validation, and Test Sets

Run	Training		Validation		Test	
No.	μ	σ	μ	σ	μ	σ
1	93.425	0.065	93.659	0.524	91.483	0.289
2	93.339	0.036	93.614	0.444	91.641	0.424
3	92.019	1.664	91.893	0.833	88.949	3.463
4	98.227	0.579	98.143	0.390	94.661	0.943
5	94.458	0.156	94.520	0.547	92.671	0.340
6	94.448	0.168	94.293	0.333	92.965	0.598
7	91.107	1.826	91.168	1.386	88.836	3.272
8	98.075	0.768	97.826	0.384	94.774	1.034
9	85.530	3.223	86.141	2.599	84.052	2.978
10	98.549	0.706	97.917	0.500	95.068	0.623
11	19.135	0.000	19.158	0.000	18.222	0.000
12	98.655	0.225	98.460	0.448	95.668	0.647
13	86.865	3.372	87.998	2.862	85.579	3.048
14	98.453	0.752	98.098	0.400	95.329	0.764
15	19.135	0.000	19.158	0.000	18.222	0.000
16	98.176	0.477	98.098	0.384	95.329	0.514

B. Effects of the Batch Size

By comparing the Figure 1 and Figure 2 it can be deduced that the batch size had an extreme effect on on the prediction accuracy of the models. These effects can be examined by comparing the loss-accuracy graph along with the Best Epoch, and Time results and momentum parameter given in the Table II in the following subsections.

1) Time

Running times of the algorithms determined by the early stopping mention in Section II. Training stopped if the test accuracy did not improved more than $1e-5$ for 20 consecutive epochs.

The model that is trained on 50 batch size were converged faster. This is due to modern-day computer architectures. $Ax + b$ operations are optimized with the usage of caches, fast memory accesses. Also bigger batch sizes results in more averaged gradient updates towards minima therefore models converges faster. However, this minima could be a local minima.

C. Effects of the Momentum (β)

Models trained with batch size 1 is the stochastic gradient descent where model makes gradient updates only for 1 sample. This is computationally more expensive and time consuming but increases the chance of finding global minima. It also prevent the model from *Exploding Gradients* and *Vanishing Gradients* problem. For the cases where there is no momentum, model weights are either getting very close to zero or infinity IV.

TABLE VII: 3 Run Average of **Top2** Accuracy Metric Training, Validation, and Test Sets

Run	Training		Validation		Test	
No.	μ	σ	μ	σ	μ	σ
1	99.773	0.044	99.774	0.064	99.446	0.058
2	99.768	0.031	99.774	0.064	99.468	0.070
3	99.854	0.050	99.819	0.064	99.468	0.195
4	99.980	0.007	99.955	0.064	99.525	0.100
5	99.854	0.014	99.728	0.111	99.548	0.131
6	99.849	0.044	99.728	0.111	99.570	0.139
7	99.879	0.021	99.819	0.064	99.480	0.153
8	99.980	0.007	99.955	0.064	99.683	0.125
9	98.977	0.216	99.321	0.111	98.055	0.530
10	99.995	0.007	100.000	0.000	99.604	0.178
11	37.833	0.000	37.772	0.000	36.274	0.000
12	100.000	0.000	100.000	0.000	99.695	0.194
13	99.345	0.147	99.457	0.192	98.733	0.513
14	99.990	0.014	100.000	0.000	99.695	0.139
15	37.833	0.000	37.772	0.000	36.274	0.000
16	100.000	0.000	100.000	0.000	99.627	0.073

TABLE VIII: 3 Run Average of **Top3** Accuracy Metric Training, Validation, and Test Sets

Run	Training		Validation		Test	
No.	μ	σ	μ	σ	μ	σ
1	99.995	0.007	100.000	0.000	100.000	0.000
2	99.995	0.007	100.000	0.000	100.000	0.000
3	100.000	0.000	100.000	0.000	99.966	0.048
4	100.000	0.000	100.000	0.000	99.955	0.016
5	100.000	0.000	100.000	0.000	100.000	0.000
6	100.000	0.000	100.000	0.000	100.000	0.000
7	100.000	0.000	100.000	0.000	99.955	0.064
8	100.000	0.000	100.000	0.000	99.932	0.000
9	99.995	0.007	99.955	0.064	99.943	0.032
10	100.000	0.000	100.000	0.000	99.932	0.028
11	55.320	0.000	55.299	0.000	52.935	0.000
12	100.000	0.000	100.000	0.000	99.887	0.032
13	99.995	0.007	99.955	0.064	99.932	0.096
14	100.000	0.000	100.000	0.000	99.910	0.032
15	55.320	0.000	55.299	0.000	52.935	0.000
16	100.000	0.000	100.000	0.000	99.943	0.042

Updating for a sample creates oscillations during the learning and it can be seen from the loss and accuracy plots. Figure 1 where models have $batchsize = 50$ results in smooth loss and accuracy plots. Meanwhile, the oscillations can be observed on the loss and accuracy plots of the models that have $batchsize = 1$ in the Figure 2.

The oscillations for the model that have $batchsize = 1$ can be problematic for the cases where there is no momentum. Table IV shows that the trainings are converging in only 1 epochs for Run No: (9, 11, 13, 15). However, implementation of momentum with $\beta = 0.9$ removes the negative side of the stochastic gradient descent, averages last gradient updates and causes a smoothing effect on the gradient updates, thus learning process for Run No: (10, 12, 14, 16). Momentum can also improve models' convergence time.

D. Effects of the Learning Rate(α)

Overall, learning rate effect results in 1-3 % difference on the test set accuracy. It can be seen by comparing Run No: $\{(1, 3), (2, 4), (5, 7), (6, 8), (10, 12), (14, 16)\}$ pairs.

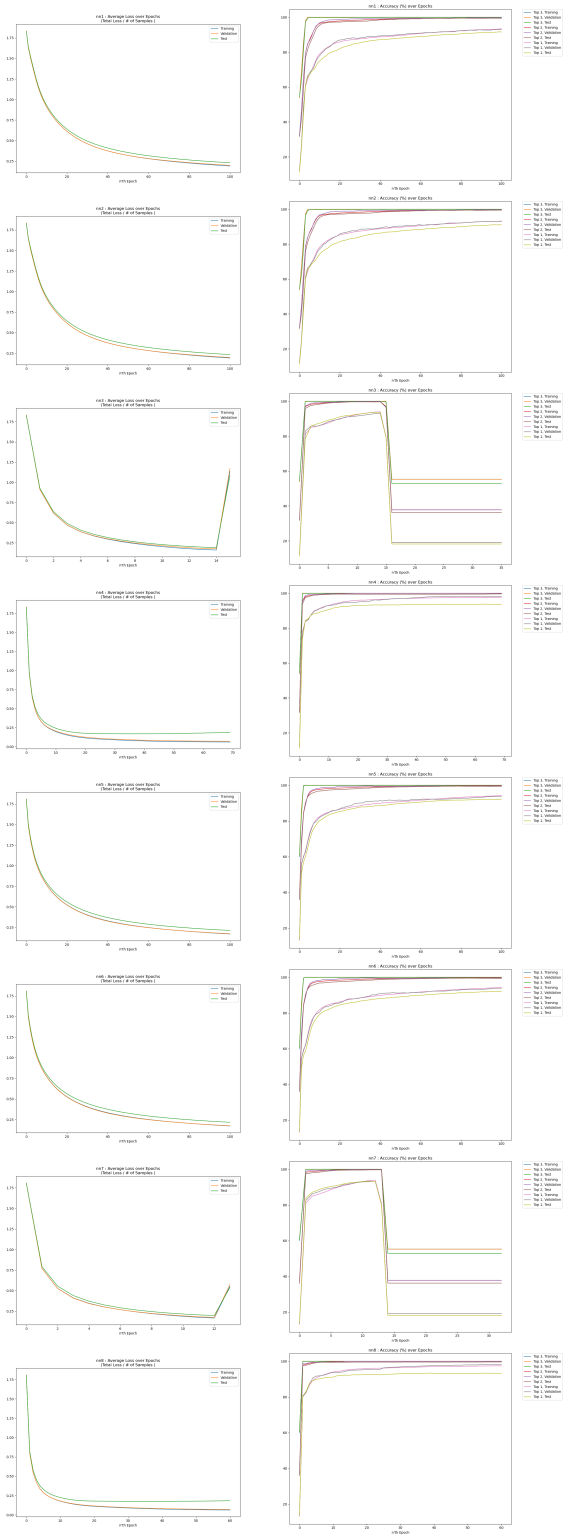

Fig. 1: Top1, Top2, and Top3 Accuracy of Run No: 1-8 Plots of Each Run for Seed Number 42. Loss(left), Accuracy(right) Pairs.

Plots are given in the order of Table II and IV. From top to bottom.

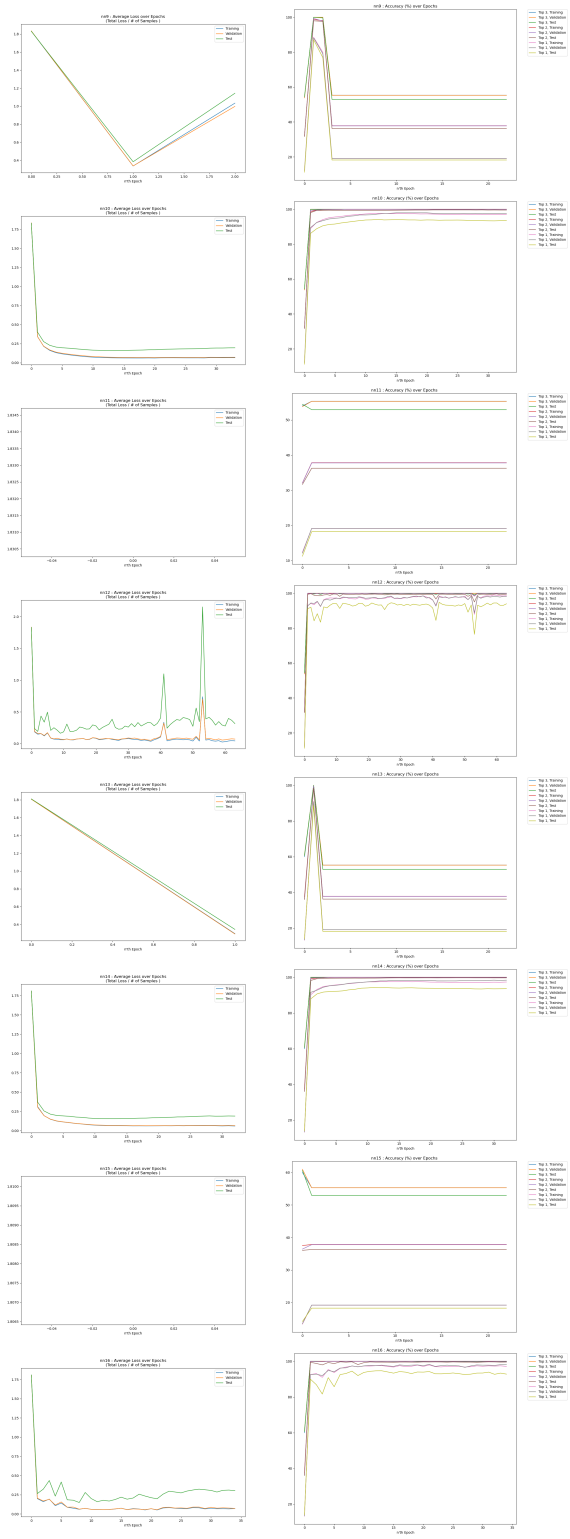

Fig. 2: Top1, Top2, and Top3 Accuracy of Run No: 9-16 Plots of Each Run for Seed Number 42. Loss(left), Accuracy(right) Pairs.

Plots are given in the order of Table II and IV from top to bottom.

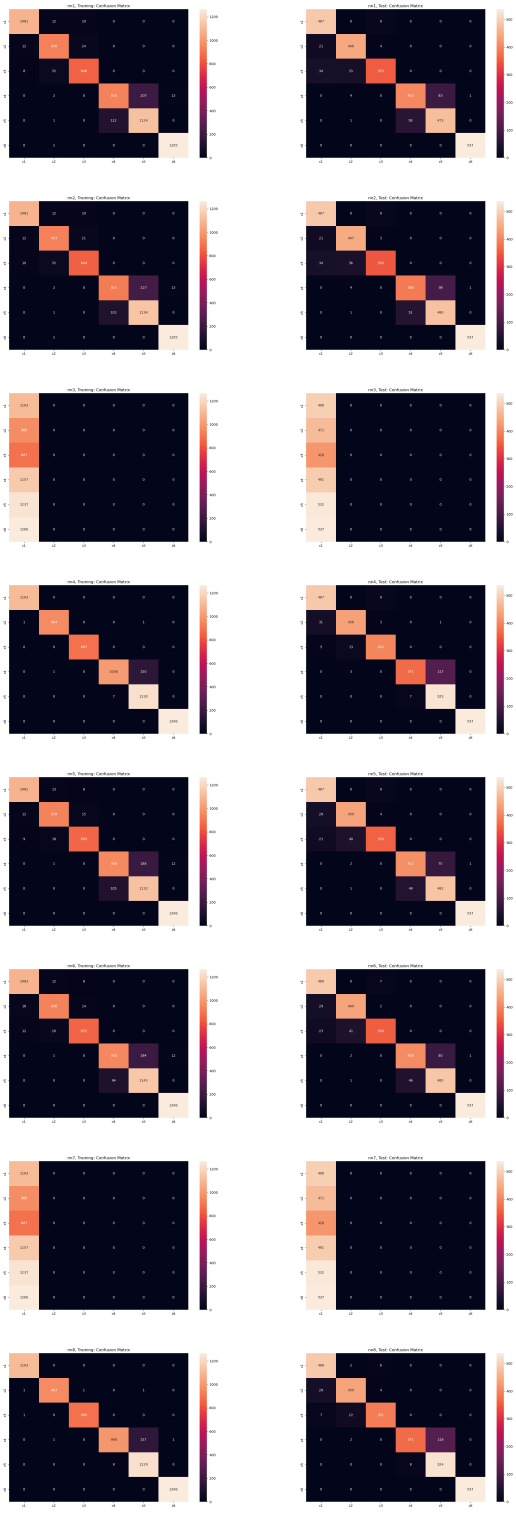

Fig. 3: Training and Test Set Confusion Matrices of Run No: 1-8 for Seed Number 42. Training(left), Test(right) Pairs.

Plots are given in the order of Table II and VI from top to bottom.

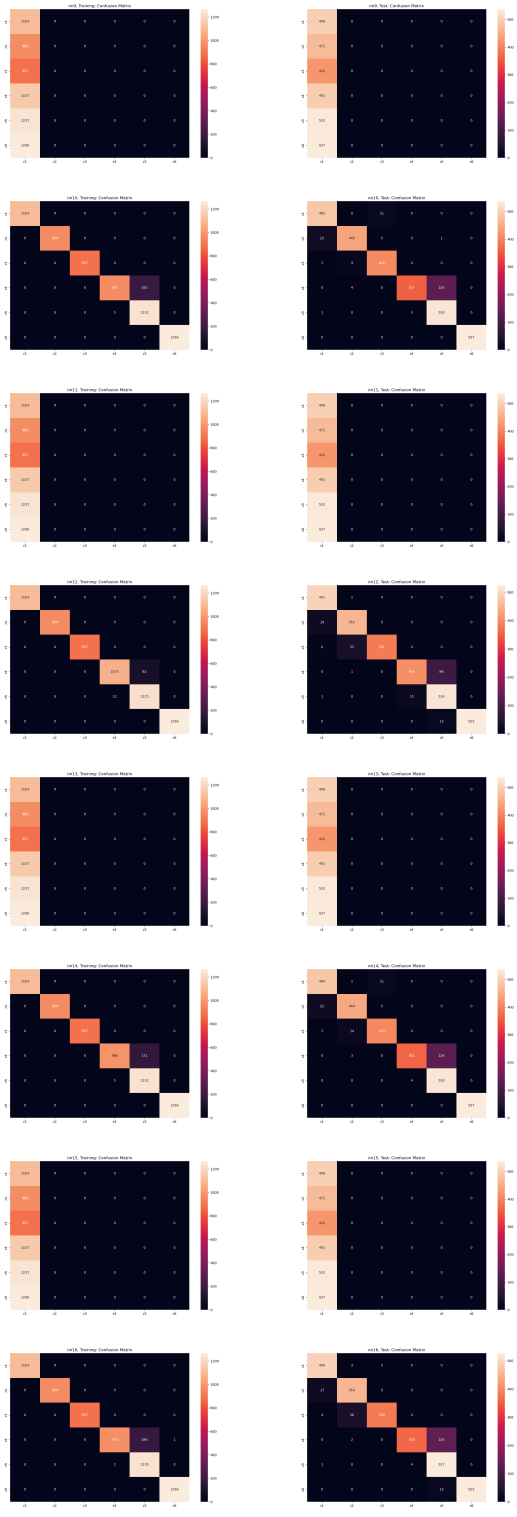

Fig. 4: Training and Test Set Confusion Matrices of Run No: 9-16 for Seed Number 42.

Plots are given in the order of Table II and VI from left to right, top to bottom.

*E. **Overall Result Interpretation for Option c**, Including Effects of the Training, Validation, Test Set Class Distributions*

As seen in the training data set (Table I), the most samples are in Class 6. It seems that fewer mistakes were made in the class 6 in the confusion matrices of the training set 3 and 4. The error rate increased slightly in the 2nd and 3rd classes, where there were fewer examples.

One important inference can be made from the predictions for the class 4 and class 5. Although there are more samples than the 2nd and 3rd classes, in some cases class 5 is predicted as 4 and class 4 is predicted as 5. This may indicate that there is a **correlation** between the 4th and 5th classes in the HAR dataset. For example, when considering sitting and standing, the position of the body can give the same measurements when standing/sitting upright. Therefore, top 2, top 3 accuracy metrics will help us to make more accurate comments. For the Top 2 and Top 3 accuracy metric, results are more than 99 percent. As a result, the model has learned to successfully predict human activities, including classes 4 and 5.

*F. **Overall Result Interpretation for Option d***

100 epoch training was run with 3 different seeds used in option b, with a dropout of 0.5 in the second hidden layer. The results are calculated as a 3 run average and given in Table IX and X. It can be seen from the IX and X that the overall test set accuracy of the model trained with dropout increased to over 96.5 percent for different initializations. During the training , oscillation on the loss and accuracy is observed Figure 5. This may be due to the learning rate being 0.01. If the learning rate is high, it will cause overshooting and cause the accuracy to oscillate around 95 percent and not increase further. Overall, dropout regularization improved the prediction capability of the model (Figure 6) that has the best configuration among the 16 run.

TABLE IX: **Test Set** 3 Run Average Results of **Top 1**, **Top 2**, and **Top 3** Accuracy of the Run No:16 with $p = 0.5$ Dropout on Second Hidden Layer.

	Test					
Run	Top 1		Top 2		Top 3	
No.	μ	σ	μ	σ	μ	σ
16	95.657	0.639	99.774	0.042	99.955	0.016

TABLE X: 3 Run Average of Option d Accuracy Metric Training, Validation, and Test Sets

Run	Training		Validation		Test	
Acc.	μ	σ	μ	σ	μ	σ
Top 1	98.463	0.385	98.007	0.558	95.657	0.639
Top 2	100.000	0.000	99.909	0.128	99.774	0.042
Top 3	100.000	0.000	100.000	0.000	99.955	0.016

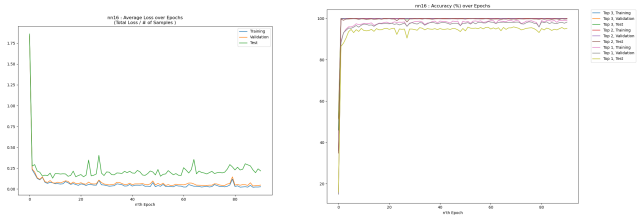

Fig. 5: Top1, Top2, and Top3 Accuracy of Run No: 16 with Dropout Plots of Each Run for Seed Number 725. Loss (top left), Accuracy (top right) Pairs.

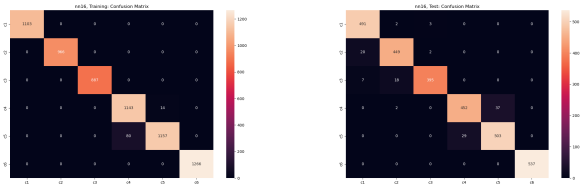

Fig. 6: Confusion Matrices: Training (bottom left), Test (bottom right)