Eray Gündoğdu 22101962

EEE 443 2024-25 Fall Mini Project

Q1) The aim of this question is to preprocess the RGB natural images properly and train an autoencoder with these preprocessed natural images so that the model learn the features of natural images. Moreover influence of hyperparamters such as weight decay paramter and number of hidden units on the model performance is observed

a)The implementation of part a is straightforward and done by following the instructions in the manual. Initially to understand the data structure, the shape of the data is observed, there are 10240 16x16 RGB patches in the dataset which is consistent with the manual. The RGB patches are converted to gray scale patches with the given equation in the convert_rgb method. After this process mean pixel value is substracted from the data and it is clipped to ±3 std and mapped to value range of [0.1, 0.9] with the help of numpy library. After completing the desired preprocess of data, RGB and grayscale versions of randomly selected 200 patches are visualized.

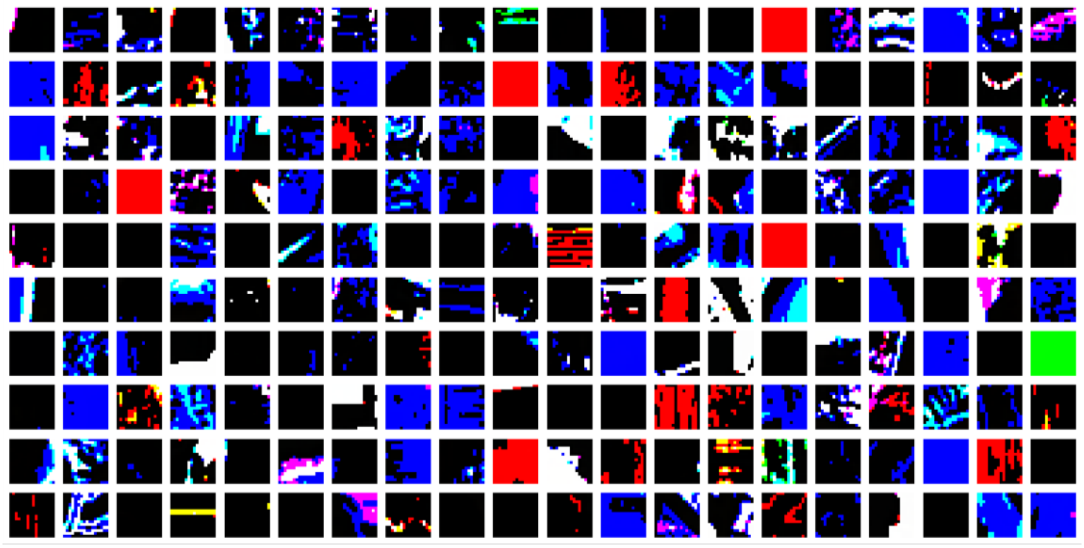


Figure 1 200 Randomly selected 16x16 RGB patches

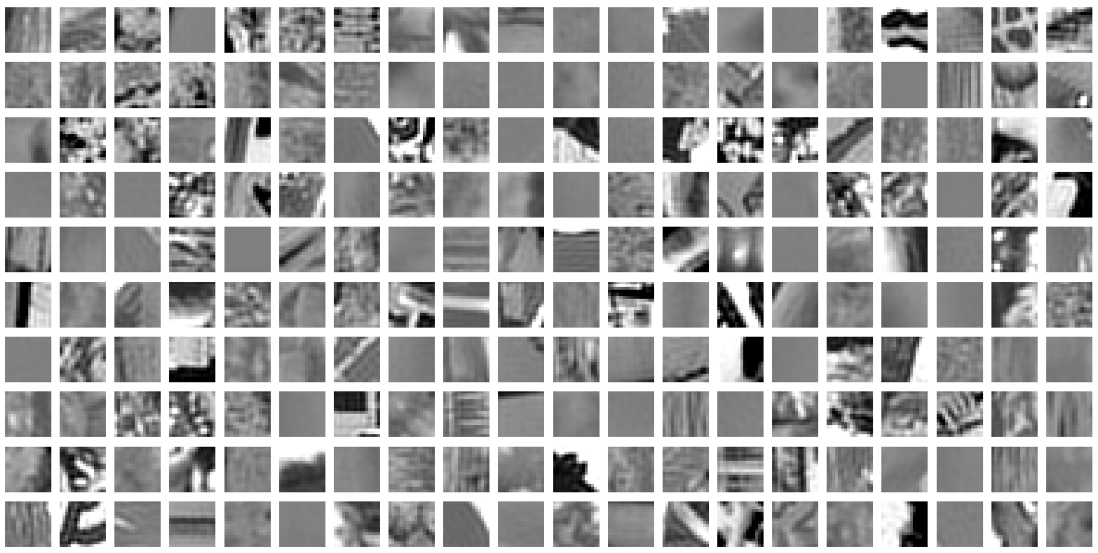


Figure 2 Pre processed 16x16 gray scale patches

As it can be observed, the gray scale patches are closely related with the displayed patches in Figure 1. The pre-processing took the color of the RGB patches and made the 'features' of the patches more defined. Now the shapes in patches are more visible which makes the data more suitable to get its features extracted with an autoencoder.

b) and c)

The initialization of weights and biases are performed as desired in the manual. The important points in the implementation of autoencoder is the additionals meaning that, when the autoencoder model is implemented with SGD and momentum optimizers combined, with 'tied-weights' the autoencoder the quality of learned features significantly increases. The aeCostGrad method peforms forward pass, calculates cost and performs backpropagation. Note that tied weights are used and the backpropogation algorith is impleemnted according to following mathematical equations. N is the mini-batch size.

$$\delta_{out} = (a_3 - X) \odot a_3(1 - a_3)\ , a_3\ is\ the\ output\ activation$$

$$\delta_{sparsity,j} = \beta(-\frac{\rho}{\widehat{\rho_J}} + \frac{1-\rho}{1-\widehat{\rho_J}})$$

$$\delta_{hid} = \left(W\delta_{out} + \delta_{sparsity}\right) \odot a_2(1 - a_2), a_2 is\ the\ hidden\ activation$$

$$\frac{\partial \mathrm{J}}{\partial \mathrm{W}} = \frac{\delta_{hid}X^T}{N}, encoder\ side$$

$$\frac{\partial \mathrm{J}}{\partial \mathrm{W}} = \frac{a_2{\delta_{out}}^T}{N}, decoder\ side$$

$$dW = \frac{\delta_{hid}X^T + a_2{\delta_{out}}^T}{N} + \lambda \mathrm{W}$$

$$db_{enc} = \frac{1}{N}\sum_i \delta_{hid,i}$$

$$db_{dec} = \frac{1}{N}\sum_i \delta_{out,i}$$

The model is trained with 0.9 momentum value, weight decay paramter of 5e-4, 0.075 learning rate, 32 min,-batch size over 200 epochs. The quality of extracted features for different β, ρ values are more understanble by visualizing the hidden weights hence, a snippet of code is implented to observe weights of the network which corresponds to part c).

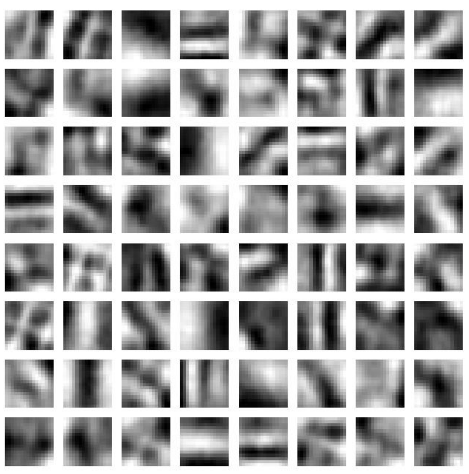
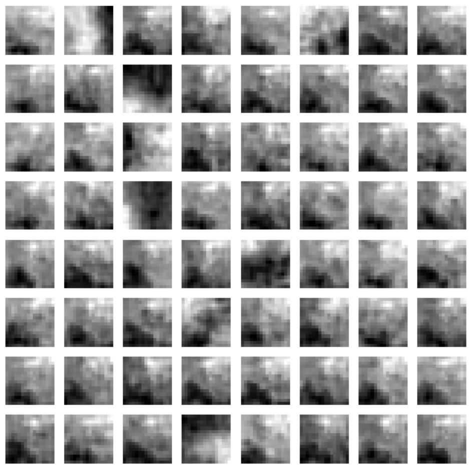
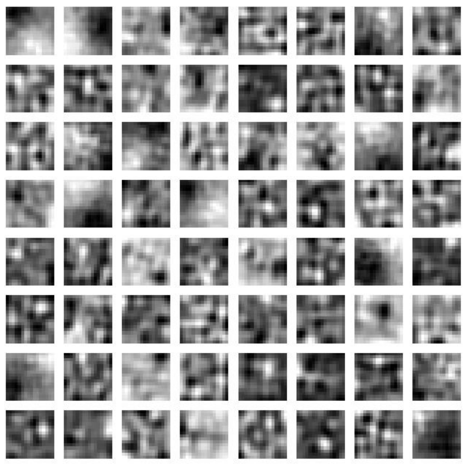




Figure 3 Hidden units when (β, ρ)=(2,2,0.026), (10,0.026), (0.01,0.026) respectively

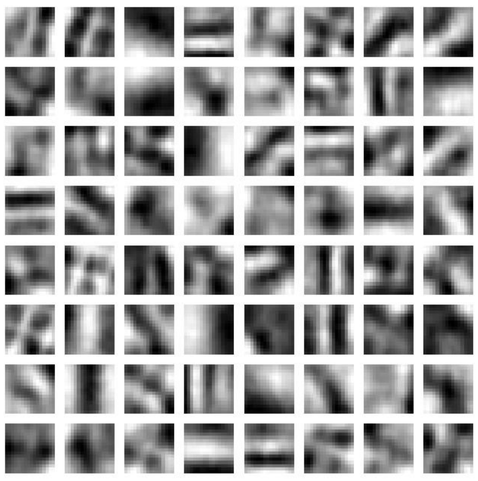
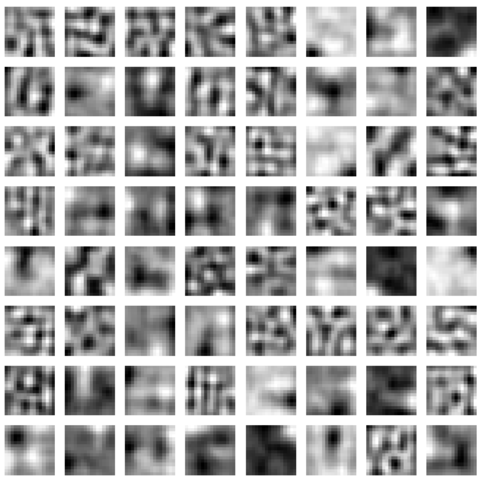
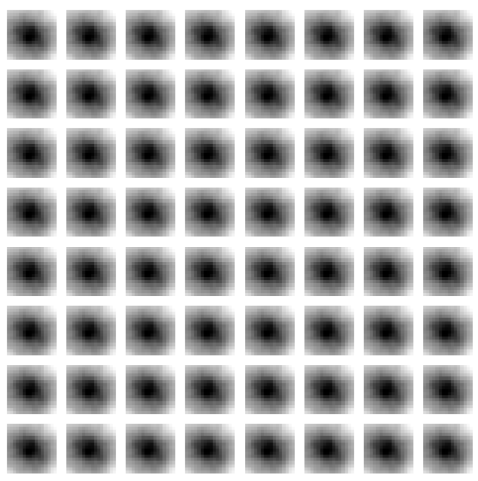




Figure 4 Hidden units when (β, ρ)=(2,2.026), (2.2,0.9), (2.2,0.001) respectively

After justfying the figures provided above with the definitions of ρ and β in the manual, it can be concluded that ρ sets the target activity of neurons ahigher value of ρ makes the learned features less sparse whereas a lower value will make sparse, distinct representation of features however a value which is relatively too low will prohibit the learning of any feature. β controls how tightly the network must adhere to that target activity level, smaller value relaxes the sparsity requirement whereas with a high value the model is contrained to find sparser solutions. As it can be observed from the figures, for low and high extremes of β, ρ values quality of features learned by the model decreases therefore the (β, ρ) is set to (2,2.026) as the moderate point according to observations. As a result of this discussion, it is clear that in this autoencoder model, the hidden units represent the features of the natural images.

d) It is important to note that the code is alternated such that if part variable is assigned to 3 it gives output to previous part when it is entered as 4 it gives the output for part d. In this part, 3 diffrent low, medium, large with (β,ρ) fixed to (2,2.026) values are selected for decay paramater and Lhid within the specified range and the autoencoder is trained with 9 different combinations. The selected values are given in the following figure.

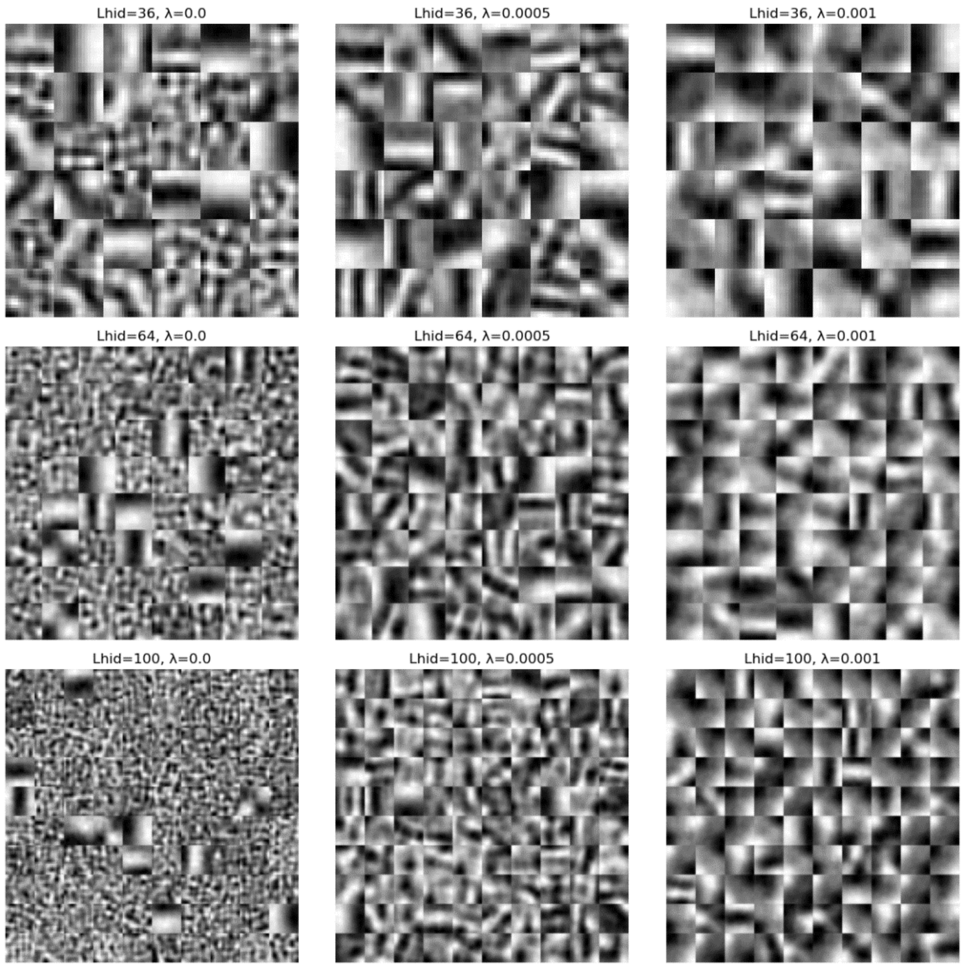


Figure 5 Learned features for different Lhid and Lambda

Theoritically, Tyhkonov regularization is responsible for penalizing the weights which shares the similar idea with the standart l2 norm regularization. The value of lambda determines the significance of the penalization. On the other hand, the amount of hidden units is closely related with the model complexity which demosntrates the ability to lean complex features. However increasing the model compelxity is not always preferable due to overfitting and computational efficiency. As the outputs are observed it can be seen that the experimental observations hold with the theoritical background. For instance, observe the row where Lhid=64, at the minimum λ value the weights are highly noisy with high variance as lamda increases to highest point, the weights are penalized excesively and the model tends to underfit with low variance meaning that features become smoother and unable capture the complexity of features. When the clomn where λ=0.0005 is observed it can be seen that as number of hidden units increases the hidden units capture more complex features of natural images. It is again important to note that as the number of hidden units increase the model tends to overfit. However in this case, the model still seems to capture distinct features when Lhid=100 meaning that in this experiment number of hidden unit can be chosen as 100 since the the weight decay compansates the overfitting and it can be obsered that with λ=0.0005 and Lhid=100 the model achieves to learn distinct and complex fetaures of natural images.

Q2) The objective of this question is to obtain reasonable fourth word guesses for a given triagram. This goal is planned to be achieved by training a neural network with a sturcture that recieves 3 inputs, ‘P’ hidden units that process the inputs and an output that generates probobality for all vocabluary instances with soft-max function which is benefitial to observe which Word is guessed by the neural network as the continuing Word of the trigram. One of the important concept is the embedding matrix which is referred as R in this question. Embedding matrix is planned have a dimension of 250Xd where D denotes the length of the embedding vector. The rows of embedding matrix denotes an embedding vector for all members of the 250 word vocabulary. Befor testing the implemented model on the test data it is trained with the training data and validated on the validaiton data. According to the lowest validation lost with a certain combination of D and P values, the network parameters (weights, biases) are loaded to a best_params array and the fourth Word is guessed with the test set with these parameters. Therefore, in total the model is trained and validated on three different set of (D,P) as (32,256), (16,128), (8,64) and it is tested on the test set with the weights and biases of the lowest validation loss output given a certain combination of D,P out of the three combinations. It is important to note that since the network performance is satisfactory the momentum, learning rate, batch size and such parameters given in the question are remained unchanged.

a) The data is read through the ‘h5py’ and samples are observed with simple print comments to obtain a better understanding of the data structure. It is clear to see that the generated arrays after reaidng the data contains the index out of 250 vocab size. After understanding the data structue the model is implemented by taking the following equations in consideration.

Cross-Entropy Loss

$$L = -\frac{1}{m}\sum_{i=1}^{m} \log(p(y_i|x_i))$$

Forward Propagation**:**

$$z_{out} = \sigma(xW_h + b_h)W_{out} + b_{out}$$

$$o = softmax(z_{out})$$

Backward Propagation**:**

The backward propagation of neural network is mathematically modelled as following.

$$\delta_{output} = \frac{o - y_{one\_hot}}{m}$$

$$\frac{\partial \mathrm{L}}{\partial b_{output}} = \sum_{i=1}^{m} \delta_{output(i)}$$

$$\frac{\partial \mathrm{L}}{\partial W_{output}} = \mathrm{h}^T \delta_{output}$$

$$\delta_{hidden} = \left(\delta_{output} \cdot W_{output}{}^T\right) . \sigma'(z_{hidden})$$

$$\delta_{hidden} = \left(\delta_{output} \cdot W_{output}{}^T\right) . \mathrm{h} . (1 - \mathrm{h})$$

$$\frac{\partial \mathrm{L}}{\partial b_{hidden}} = \sum_{i=1}^{m} \delta_{hidden(i)}$$

$$\frac{\partial \mathrm{L}}{\partial W_{hidden}} = \mathrm{e}^{T} \delta_{hidden}$$

$$\frac{\partial \mathrm{L}}{\partial e_j} = W_{hidden,j}{}^{T} \delta_{hidden}\ where\ j\ represents\ a\ word\ in\ trigram$$

Parameter Update with Momentum**:**

The parameters are updated according to the momentum update with velocities which can be mathematicall described as following where θ represents parameters.

$$v^{(t)} = \mu v^{(t-1)} - \eta \nabla_{\theta} (L(\theta^{(t-1)})$$
$$\theta^{(t)} = \theta^{(t-1)} + v^{(t)}$$

As the final steps of the implementation of part a, a training loop is constructed for different (D,P) values. This training loop extracts the weights and baises as best_params after obtaining the lowest validation loss for a certain combination of (D,P). Afterwards the model is tested with best_params and corresponding (D,P) set by calling the evaluate_model method. Below the figures that demonstrate validation and training as a function of epoch number are provided.

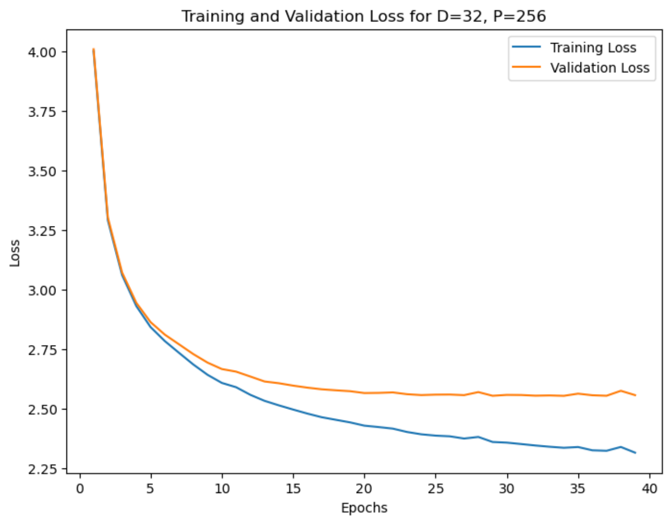


Figure 6 Training and validation loss as a function of epoch number for (D,P)=(32,256)

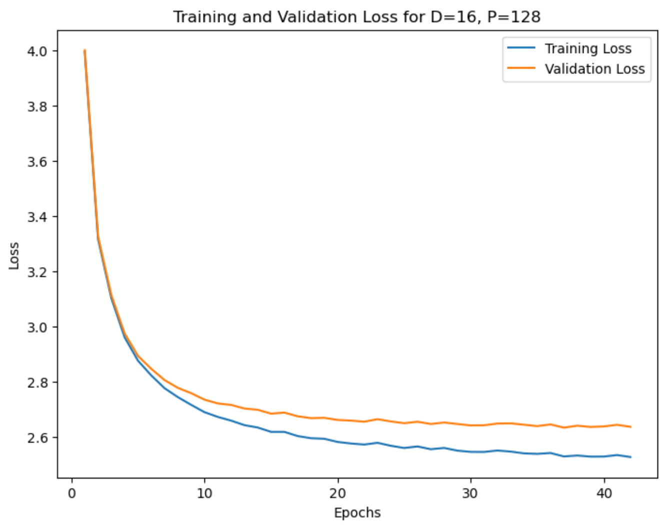


Figure 7 Training and validation loss as a function of epoch number for (D,P)=(16,128)

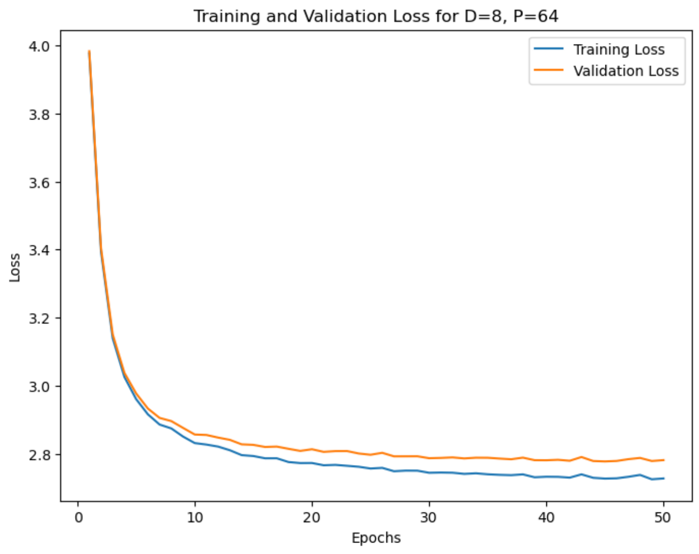
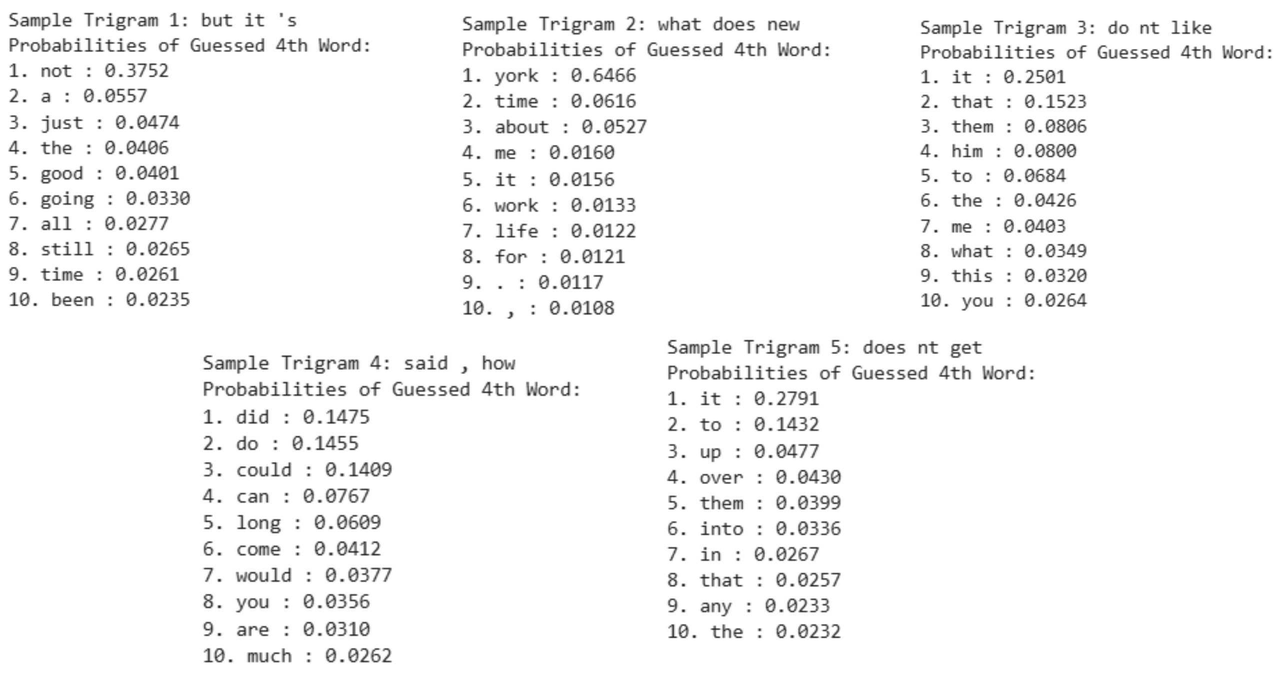


Figure 8 Training and validation loss as a function of epoch number for (D,P)=(8,64)

Intiutively, as D increases, richer representation of words can be obtained meaning that more nuanced relationships between words are obtained. On the other hand P is directly related with the model complexity and as P increases the model can learn more complex functions and relations. However note that high D,P pairs may overfit the model and low D,P pair may underfit the model. If training loss is observed the model with higher D,P values tend to memorize data hence resulting in the lowest training loss whereas as D,P decreases model becomes more biased, obtaining higher training loss. For the validation loss in this question, early stopping algorithm is applied to the neural network model hence the training is stopped when lowest validation loss is achieved. In other words even tough the 32,256 pair is expected to overfit, with early stopping overfitting is prevented, and due to ability to capture more complex relationships and richer reprenstation of words the model performs the lowest validation error in 32,256 pair. In part b the model is tested with (D,P)=(32,256)

b) In this part model is tested with (D,P)=(32,256) to guess the continuing Word of the test triagram. In the following figures 5 different triagrams are proivded and the top ten probabilites for the fourth Word is provided.



```
Sample Trigram 1: but it 's
Probabilities of Guessed 4th Word:
1. not : 0.3752
2. a : 0.0557
3. just : 0.0474
4. the : 0.0406
5. good : 0.0401
6. going : 0.0330
7. all : 0.0277
8. still : 0.0265
9. time : 0.0261
10. been : 0.0235
```

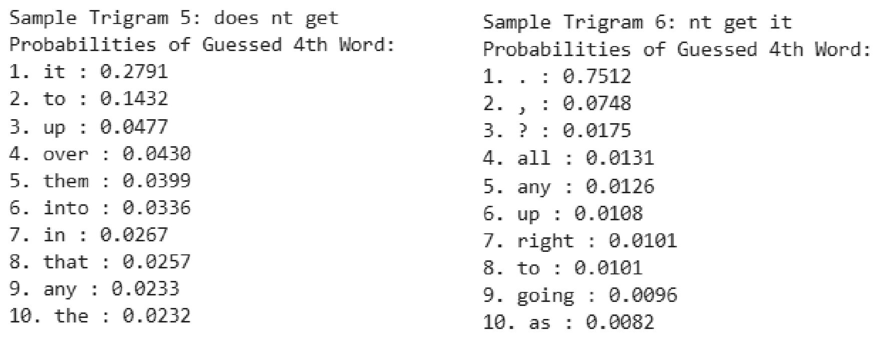
```
Sample Trigram 2: what does new
Probabilities of Guessed 4th Word:
1. york : 0.6466
2. time : 0.0616
3. about : 0.0527
4. me : 0.0160
5. it : 0.0156
6. work : 0.0133
7. life : 0.0122
8. for : 0.0121
9. . : 0.0117
10. , : 0.0108
```

```
Sample Trigram 3: do nt like
Probabilities of Guessed 4th Word:
1. it : 0.2501
2. that : 0.1523
3. them : 0.0806
4. him : 0.0800
5. to : 0.0684
6. the : 0.0426
7. me : 0.0403
8. what : 0.0349
9. this : 0.0320
10. you : 0.0264
```

```
Sample Trigram 4: said , how
Probabilities of Guessed 4th Word:
1. did : 0.1475
2. do : 0.1455
3. could : 0.1409
4. can : 0.0767
5. long : 0.0609
6. come : 0.0412
7. would : 0.0377
8. you : 0.0356
9. are : 0.0310
10. much : 0.0262
```

```
Sample Trigram 5: does nt get
Probabilities of Guessed 4th Word:
1. it : 0.2791
2. to : 0.1432
3. up : 0.0477
4. over : 0.0430
5. them : 0.0399
6. into : 0.0336
7. in : 0.0267
8. that : 0.0257
9. any : 0.0233
10. the : 0.0232
```

Figure 8 Demonstration of sample triagrams



```
Sample Trigram 5: does nt get
Probabilities of Guessed 4th Word:
1. it : 0.2791
2. to : 0.1432
3. up : 0.0477
4. over : 0.0430
5. them : 0.0399
6. into : 0.0336
7. in : 0.0267
8. that : 0.0257
9. any : 0.0233
10. the : 0.0232
```

```
Sample Trigram 6: nt get it
Probabilities of Guessed 4th Word:
1. . : 0.7512
2. , : 0.0748
3. ? : 0.0175
4. all : 0.0131
5. any : 0.0126
6. up : 0.0108
7. right : 0.0101
8. to : 0.0101
9. going : 0.0096
10. as : 0.0082
```

Figure 9 Demosntration of the guess for sample triagram 5 in sample triagram 6(true label)

As it can be observed from Figure 8 the guessed words makes sense in the context of vocabulary and also the model achieves to predict the fourth Word by taking grammer, sentence structure and punctuation in consideration. In addition, sample triagram 5 has an index of 331 and in Figure 9 the triagrams with indexes 331 and 332 are demonstarted which actually indicates that the prediction of sample triagram 5 is already in sample triagram 6 in the test set(true label).

Q3) RNN, LSTM, GRU models are considered to be remarkable in time series data processing. In this question the goal is to implement all of the models with a follow up MLP for human activity classification. After the experimentation the performance of the models is discussed in the context of test accuracy and computational power.

a)In this part of the question a RNN with a follow-up MLP structure is implemented to perform huamn activity classification. Inıtially when the code is umplemented by strictly obeying to instructions given in manual, during the training exploding gradient problem occurs since 150 time steps are processed sequentially. Hence some modifications are done in addition to instructions in manual. To observe better outputs a patience counter is implemented for early stopping, data is normalized before the training, ReLU activation in hidden layer of MLP is used, after obtaining the gradients the gradients are clipped with a 1.0 gradient clipping value and as the final change the learning rate is set to 0.001 to perform more ‘catious’ parameter update. Another important point is that for the classification of the processed time series, a single hidden layer neural network is used since a hidden layer is sufficient to decribe any compelx relationship according to universal approximation theorem. The hidden layer size is set to be 128 with trial and error by testing 2 other values. The size of 10 performed relatively poorly compared to 128 hidden units most probably due to underfitting whereas 256 hidden units also output a lower test accuracy compared to 128 most probably due to overfitting. The reason that the model is tested with 3 different hidden MLP size is because it tkaes a lot of time to train this model. The mathematical background which is used in the implementation is accordingly. The validation, training loss over epochs confusion matrices and test accuracy can be observed in the following figures.

RNN Forward Pass:

$$h_t = \tanh(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$

MLP Forward Pass:

$$a = ReLU\left(W_{hy}h_{150} + b_y\right)$$

$$z = W_{out}a + b_{out}$$

$$\hat{y} = softmax(z)$$

Categorical Cross-entropy loss:

$$L = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{C} y_{ij}\log(\hat{y}_{ij})$$

Output Layer Gradients:

$$\delta_{out} = \frac{\hat{y} - y}{m}$$

$$\frac{\partial \mathrm{L}}{\partial W_{out}} = \mathrm{a}^T \delta_{out}$$

$$\frac{\partial \mathrm{L}}{\partial b_{output}} = \sum_{i=1}^{m} \delta_{output(i)}$$

MLP Hidden Layer Gradients:

$$\delta_{Why} = \delta_{out} {W_{out}}^T ReLU'(a)$$

$$\frac{\partial \mathrm{L}}{\partial W_{hy}} = {h_{150}}^T \delta_{Why}$$

$$\frac{\partial \mathrm{L}}{\partial b_y} = \sum_{i=1}^{m} \delta_{Why,i}$$

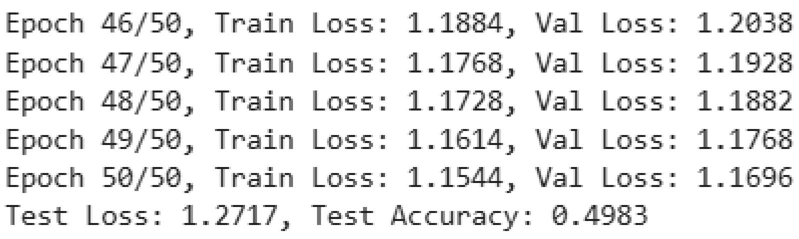
BPTT for a time step t:

$$d_{tanh}{}^{t} = \delta_h \odot \tanh'(h_t)$$

$$\frac{\partial L}{\partial W_{xh}} = \frac{\partial L}{\partial W_{xh}} + x_t{}^{T} d_{tanh}{}^{t}$$

$$\frac{\partial L}{\partial W_{hh}} = \frac{\partial L}{\partial W_{hh}} + x_{t-1}{}^{T} d_{tanh}{}^{t}$$

$$\delta_h = W_{hh}{}^{T} d_{tanh}{}^{t}$$



```
Epoch 46/50, Train Loss: 1.1884, Val Loss: 1.2038
Epoch 47/50, Train Loss: 1.1768, Val Loss: 1.1928
Epoch 48/50, Train Loss: 1.1728, Val Loss: 1.1882
Epoch 49/50, Train Loss: 1.1614, Val Loss: 1.1768
Epoch 50/50, Train Loss: 1.1544, Val Loss: 1.1696
Test Loss: 1.2717, Test Accuracy: 0.4983
```

Figure 10 Demosntration of test accuracy with RNN architecture

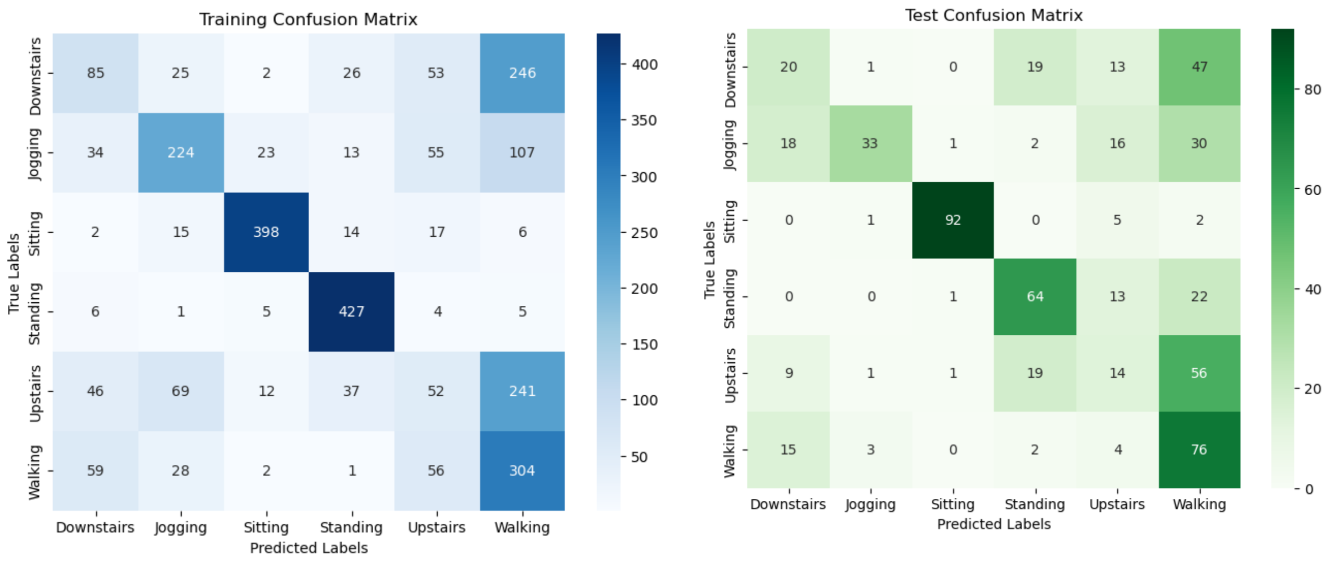


Figure 11 Confusion matrices for training and test

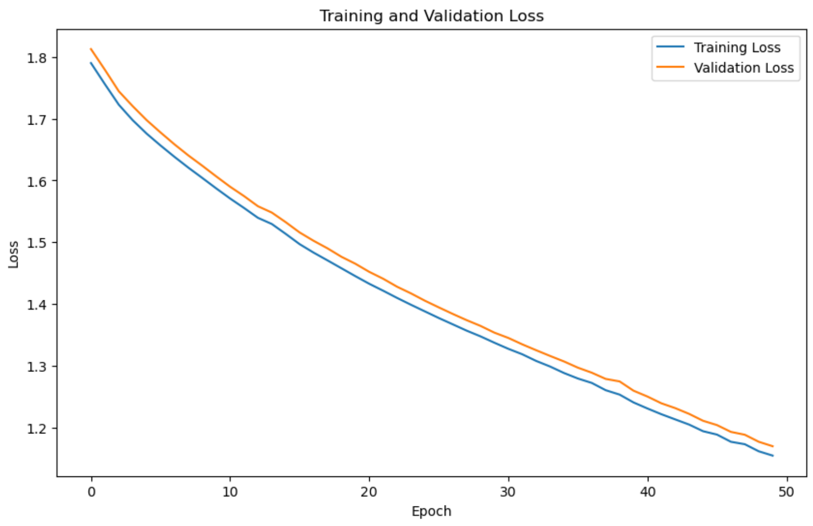


Figure 12 Training and validation loss as a function of number of epochs

After the experiment with RNN it can be concluded that after the precatiouns described previosly the implementation Works ‘reasonable’, it outputs a test accuracy and validation loss greater than training. However as expected, the RNN processes 150 time steps sequentially and there is an exploding gradient problem, eventough the gradients are clipped and this problem is solved the gradients does not reflect the ‘true’ value for gradients. As a result of this implementation of RNN from scratch, a test accuracy of %49.83 is obtained which was expected. In the following parts it is expected to obtain a higher test accuracy due to advantages of LSTM and GRU over bare RNN in the context of sequential data processing.

b)In this part of the question an LSTM structure is impelemted with a following MLP structure to classify predicitons. LSTM’s are a type of RNN which are effective in learning long term dependicies in sequential data. They use memory cells, input,otuput, forget Gates, states to selectively retain or discard information making them worthy in time-series processing. The structure of LSTM and state equations (forward propogation) can be inspected below. The backpropogation of an LSTM follows a similar logic with RNN which utilizes BPTT provided in part a. However in LSTM before the BPTT the gradients w.r.t. gate parameters should be calculated.

In this implementation, the additional changes are remained unchanged from previous part except the learning rate. With experimentation, the most optimal learning rate for this part in terms of providing stable updates and test accuracy is observed to be 0.01.

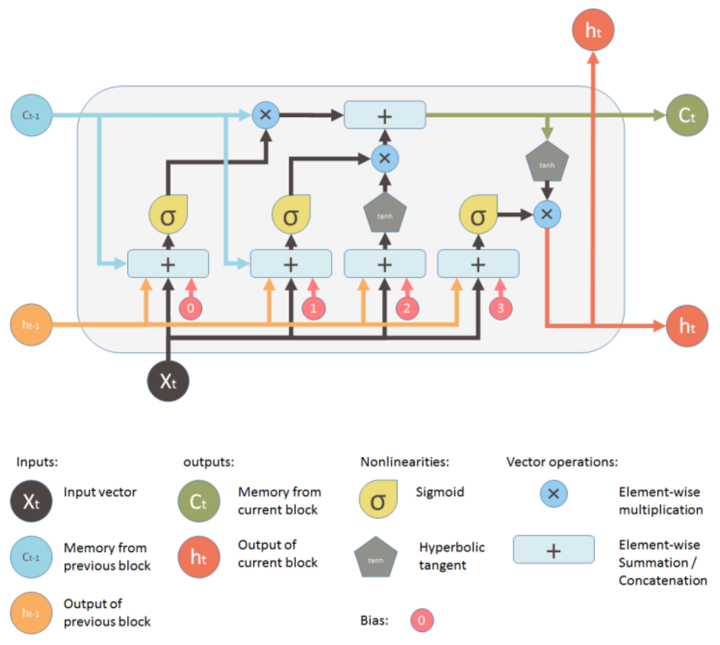


Figure 13 LSTM structure

LSTM Forward Propagation Equations:

$$f_t = \sigma\left(W_f[h_{t-1}, x_t] + b_f\right)$$

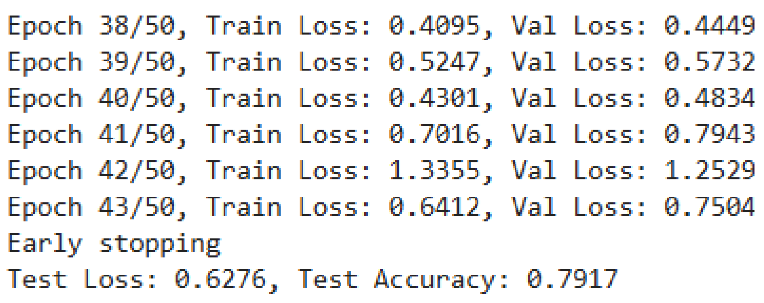
$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$

$$\tilde{C} = tanh(W_c[h_{t-1}, x_t] + b_c)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}$$

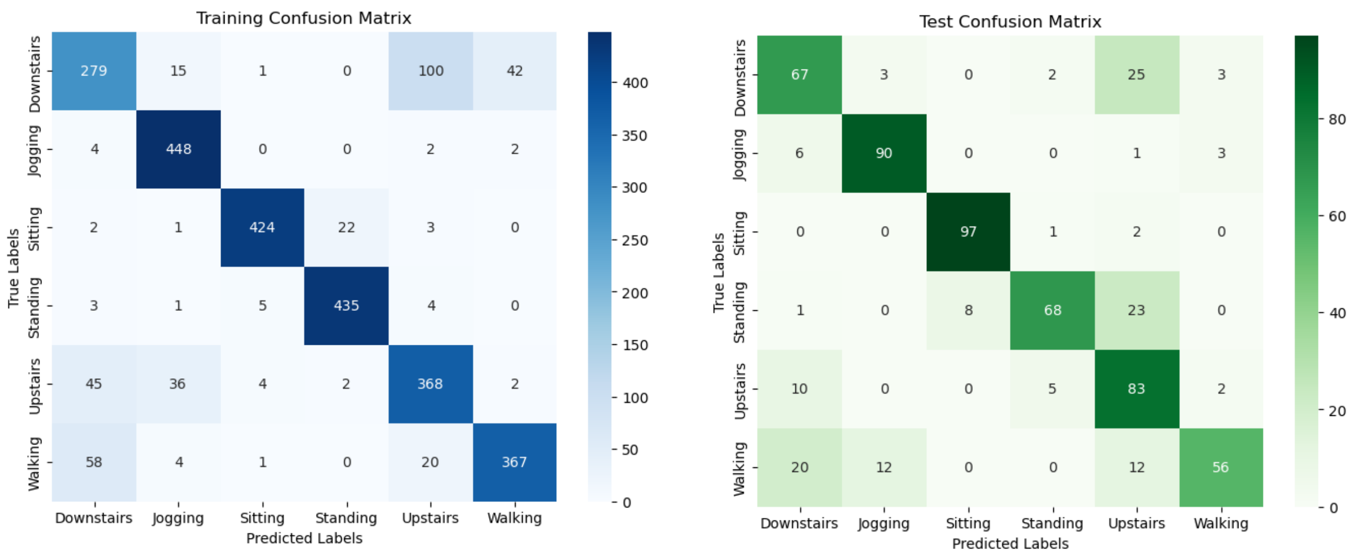
$$o_t = \sigma(W_0[h_{t-1}, x_t] + b_0)$$

$$h_t = o_t * \tanh(C_t)$$



```
Epoch 38/50, Train Loss: 0.4095, Val Loss: 0.4449
Epoch 39/50, Train Loss: 0.5247, Val Loss: 0.5732
Epoch 40/50, Train Loss: 0.4301, Val Loss: 0.4834
Epoch 41/50, Train Loss: 0.7016, Val Loss: 0.7943
Epoch 42/50, Train Loss: 1.3355, Val Loss: 1.2529
Epoch 43/50, Train Loss: 0.6412, Val Loss: 0.7504
Early stopping
Test Loss: 0.6276, Test Accuracy: 0.7917
```

Figure 14 Demonstration of test accuracy with LSTM architecture



	Downstairs	Jogging	Sitting	Standing	Upstairs	Walking
Downstairs	279	15	1	0	100	42
Jogging	4	448	0	0	2	2
Sitting	2	1	424	22	3	0
Standing	3	1	5	435	4	0
Upstairs	45	36	4	2	368	2
Walking	58	4	1	0	20	367

	Downstairs	Jogging	Sitting	Standing	Upstairs	Walking
Downstairs	67	3	0	2	25	3
Jogging	6	90	0	0	1	3
Sitting	0	0	97	1	2	0
Standing	1	0	8	68	23	0
Upstairs	10	0	0	5	83	2
Walking	20	12	0	0	12	56

Figure 15 Training and test confusion matrix of the model with LSTM architecture

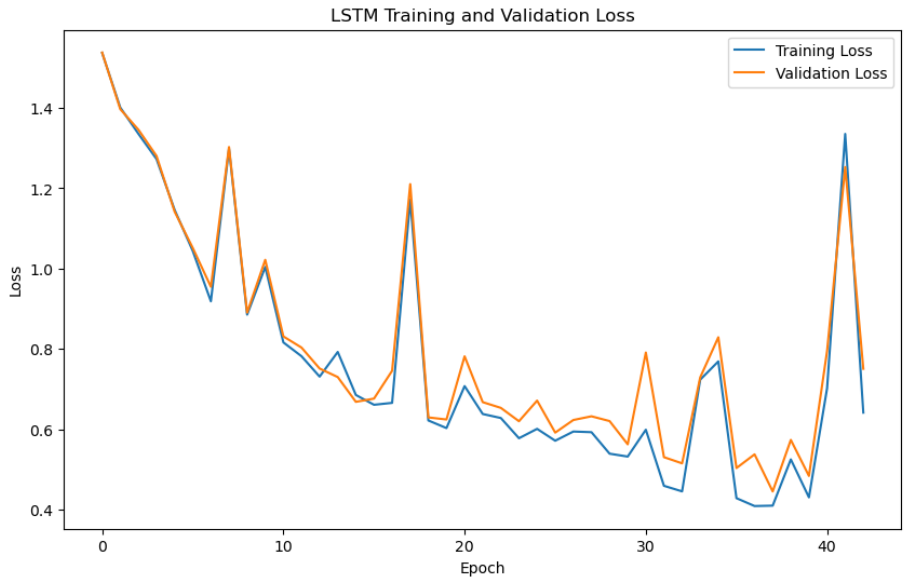


Figure 16 LSTM training and validation loss as a function of number of epochs

As it is indicated previously, LSTM’s are able to retain or and discard information with the help of their gated structure hence they are successful in processing long-term sequential data. Compared to the model with RNN architecture, the test accuracy performance increased significantly with a test accuracy of %79.17. Also from the confusion matrices, it is observable that model is successful in prediciting the true class for a given input compared to the confusion matrices of RNN architecture. In contrast to the improved test performance it is important to note that training process is longer in LSTM architecture hence it is less computational efficient compared to the previous model.

c)As the final step of the question an GRU architecture is used with the previosly used MLP architecture. GRU's also have gated structure similar to LSTM structure with less gates and less equations. Due to this gated structure it is expected to have similar performance with LSTM with a higher computational efficiency. The implementation of back propagation is similar to the back propagation of LSTM however different gradients w.r.t. gate parameters are ofcourse different since the forward propagation equations are different. The structure of a GRU and forward propagation equations are as following. In this part the hyperparameters the hyperparamters remained unchanged from the previous part except the learning rate which is now set to 0.1 for a better accuracy performance according to observations with other learning rate values.

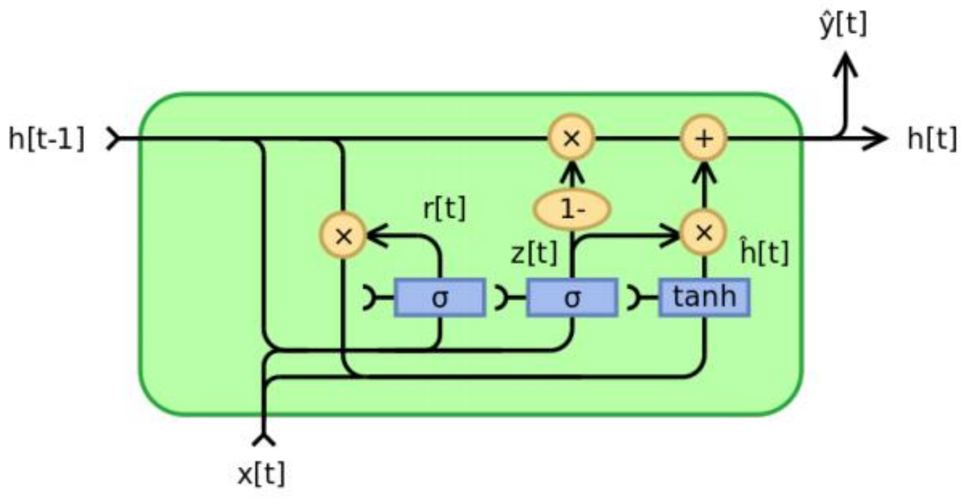


Figure 17 GRU structure

GRU Forward Propagation Equations:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$$

$$\tilde{h}_t = tanh(W_h x_t + U_h h_{t-1} + b_z)$$

$$h_t = z_t * h_{t-1} + (1 - z_t) * \tilde{h}_t$$

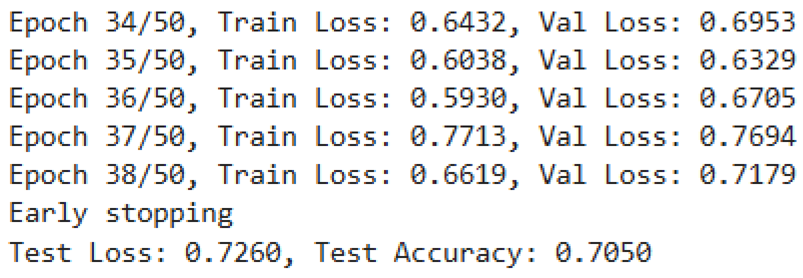

```
Epoch 34/50, Train Loss: 0.6432, Val Loss: 0.6953
Epoch 35/50, Train Loss: 0.6038, Val Loss: 0.6329
Epoch 36/50, Train Loss: 0.5930, Val Loss: 0.6705
Epoch 37/50, Train Loss: 0.7713, Val Loss: 0.7694
Epoch 38/50, Train Loss: 0.6619, Val Loss: 0.7179
Early stopping
Test Loss: 0.7260, Test Accuracy: 0.7050
```

Figure 18 Demonstration of test accuracy with GRU architecture

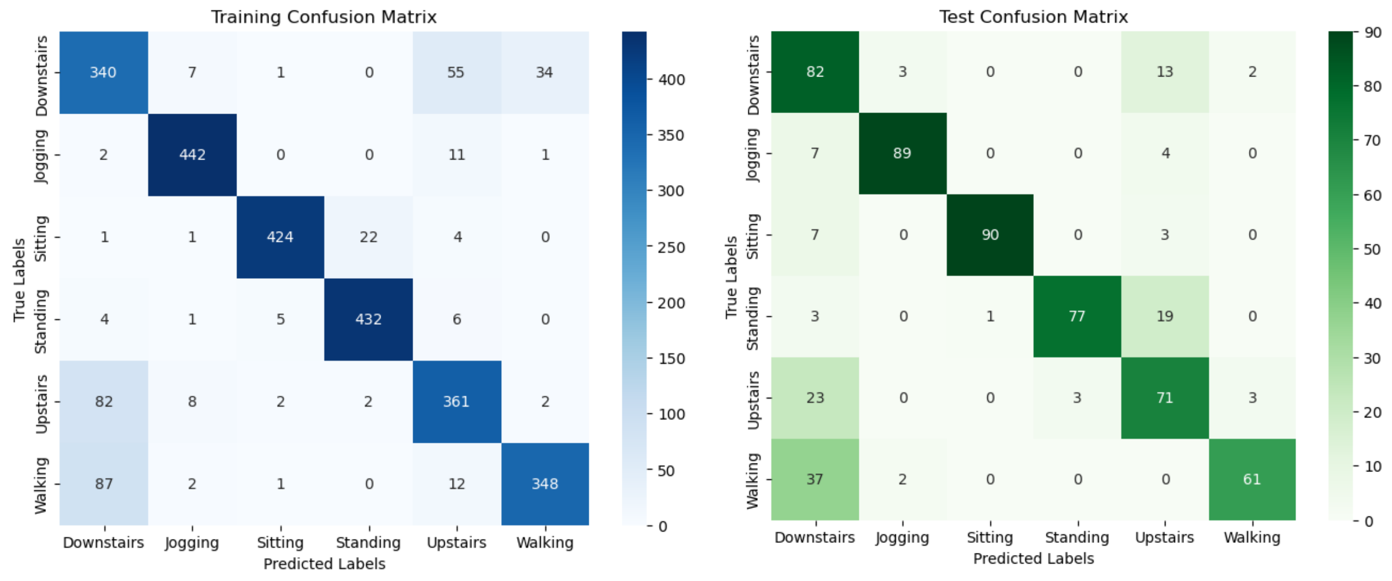


Figure 19 Training and test confusion matrices of the model with GRU architecture

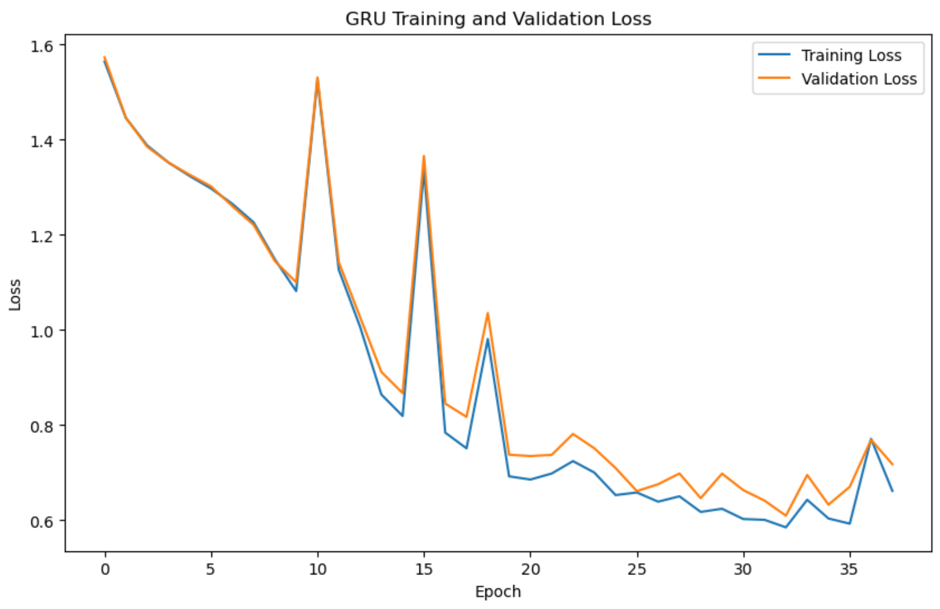


Figure 20 GRU training and validation loss as a function of number of epochs

The test accuracy for GRU is came out to be %70.50. From the confusion matrices the predictions for activity classification is satisfactory. The performance in terms of accuracy is close to the LSTM architecture if the confusion matrices are inspected however GRU slightly has a lower performance in the context of accuracy and both LSTM and GRU architectures perform better than RNN architecture which can be validated with the test accuracy percentages. However, training process of GRU is shorter compared to the LSTM since it has less equations compared to LSTM whereas both the gated RNN structures' training process is longer compared to mere LSTM model. Therefore between the models with RNN, LSTM, GRU structures with test accuracies %49.83, %79.17, %70.50 respectively, it is more desirable to select a moderate structure which is in this case GRU since it has valid accuracy performance and a moderate computational eficiency among the experimented architectures.