Project 3 – Textual Entailment of SNLI

Name: Eshwar S R
Email ID: eshwarsr@iisc.ac.in

Description: The goal of the project is to classify the given 2 sentences into entailment,contradiction or neutral.

Dataset: SNLI
Tasks:
1. Train a TF IDF based Logistic regression model
2. Train a RNN based classifier

TF-IDF based Logistic Regression classifier

Preprocessing:

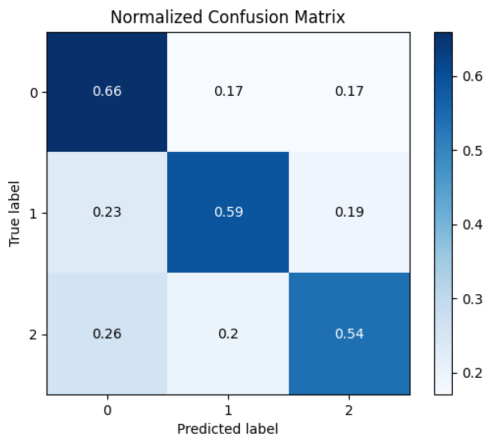
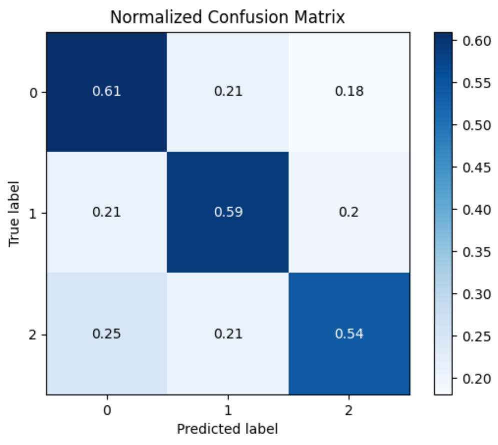
Preprocessed the data by removing the punctuations, stop words and normalizing the tokens based on their lemma using the python package spacy.

Experiment 1:

As part of the first experiment, I tried to implement a TF-IDF based Logistic Regression without any preprocessing. Ran the model with l2 penalty.

The accuracies of train and validation sets are **0.596** and **0.582** respectively.

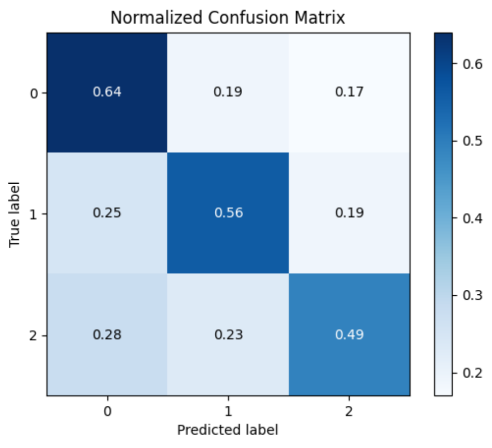
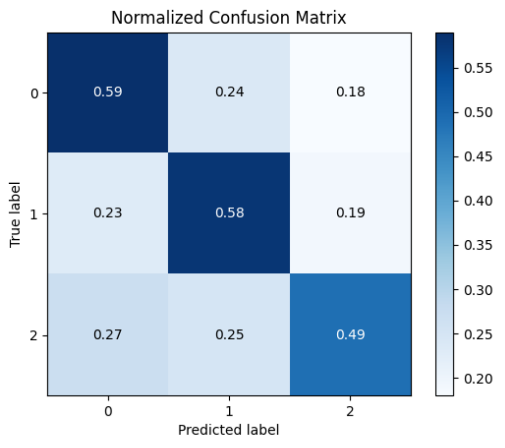
Below are the graphs for the same.

Train Stats	Validation Stats
	

Experiment 2:

Next I tried the same after preprocessing, and with l2 penalty.
The accuracies of train and validation sets are **0.564** and **0.551** respectively.

Below are the graphs for the same.

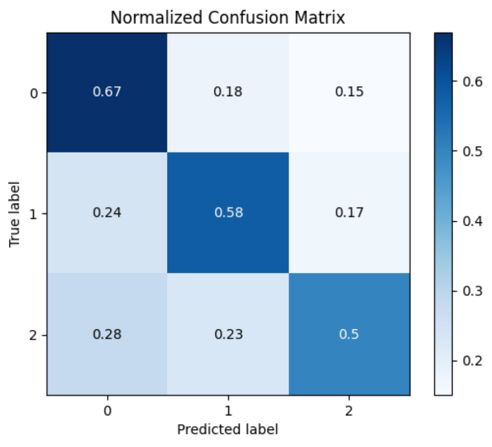
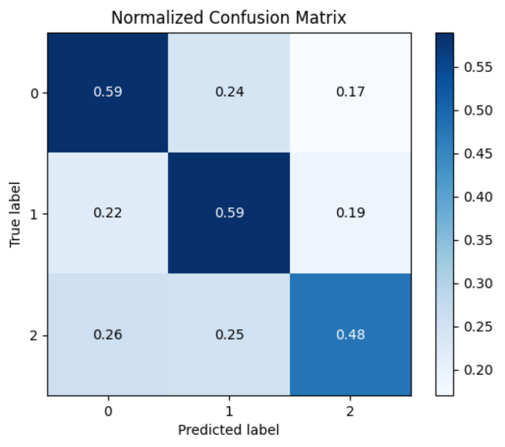
Train Stats	Validation Stats
	

Interestingly, unprocessed data is giving better results than the processed ones. Hence, wanted to try out different regularizations on both processed and un processed data.

Experiment 3:

Next I tried the same after preprocessing, i.e eliminating stop words and punctuations, and also lemmatizing the tokens. Ran the model with no regularization.
The accuracies of train and validation sets are **0.583** and **0.553** respectively.

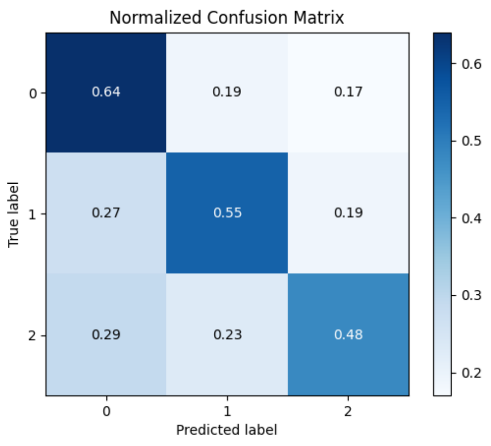
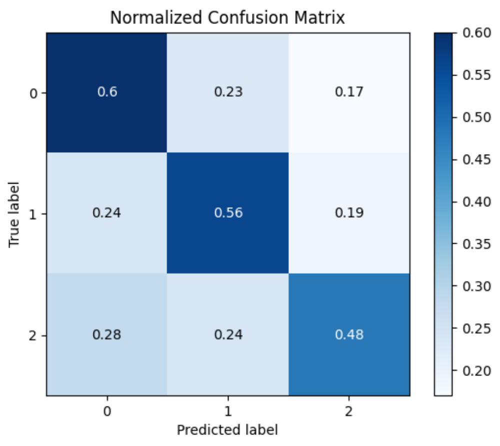
Below are the graphs for the same.

Train Stats	Validation Stats
	

Experiment 4:

Next I tried the same after preprocessing, and with l1 penalty.
The accuracies of train and validation sets are **0.556** and **0.547** respectively.

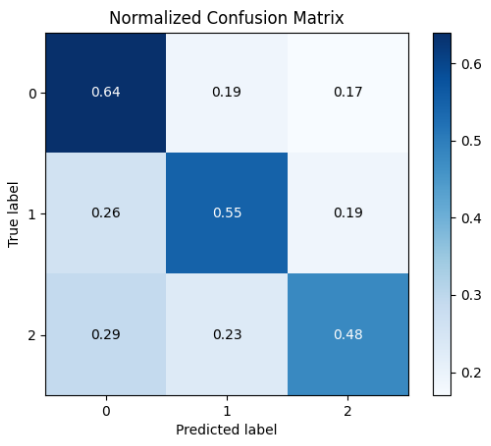
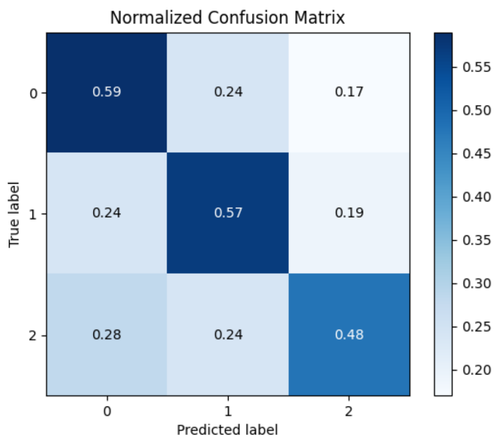
Below are the graphs for the same.

Train Stats	Validation Stats
	

Experiment 5:

Next I tried the same after preprocessing, and with elasticnet penalty with lr ratio=0.5.
The accuracies of train and validation sets are **0.558** and **0.548** respectively.

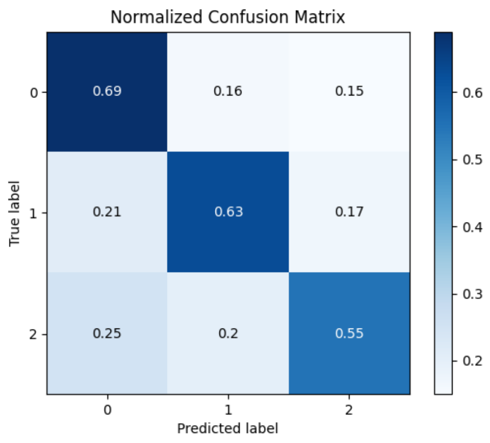
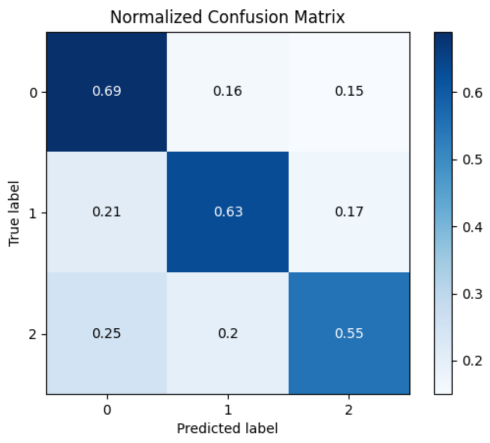
Below are the graphs for the same.

Train Stats	Validation Stats
	

Experiment 6:

Next I tried on un preprocessing data with no penalty.
The accuracies of train and validation sets are **0.622** and **0.579** respectively.

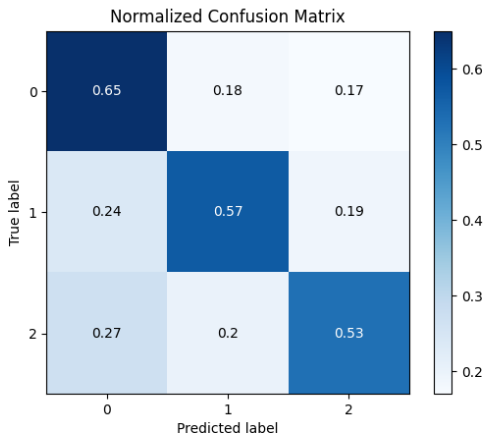
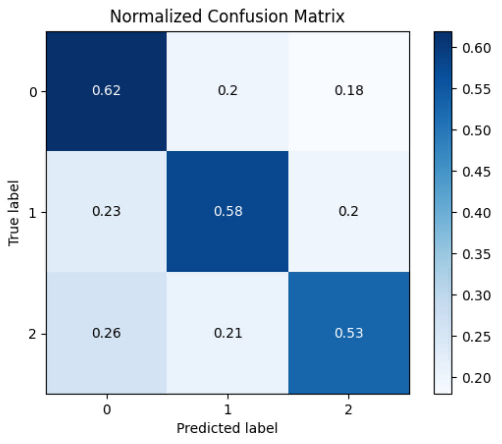
Below are the graphs for the same.

Train Stats	**Validation Stats**
	

Experiment 7:

Next I tried on un preprocessing data with l1 penalty.
The accuracies of train and validation sets are **0.585** and **0.576** respectively.

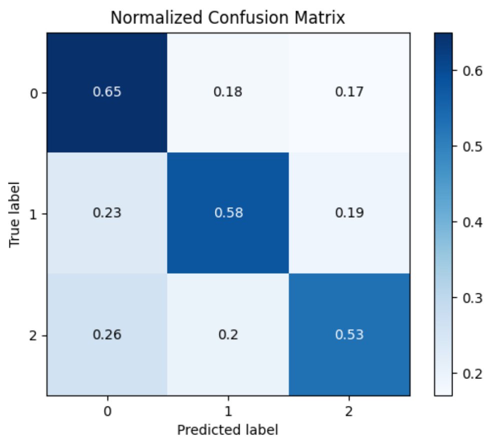
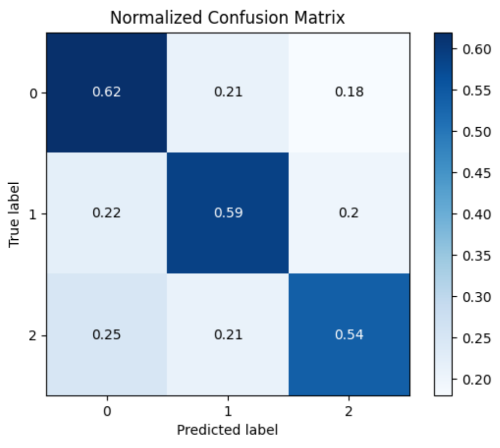
Below are the graphs for the same.

Train Stats	**Validation Stats**
	

Experiment 8:

Next I tried on un preprocessing data with elasticnet penalty with l1_ration=0.5.
The accuracies of train and validation sets are **0.588** and **0.579** respectively.

Below are the graphs for the same.

Train Stats	Validation Stats
	

Conclusion:

Below are the results of all the 8 logistic regression models.

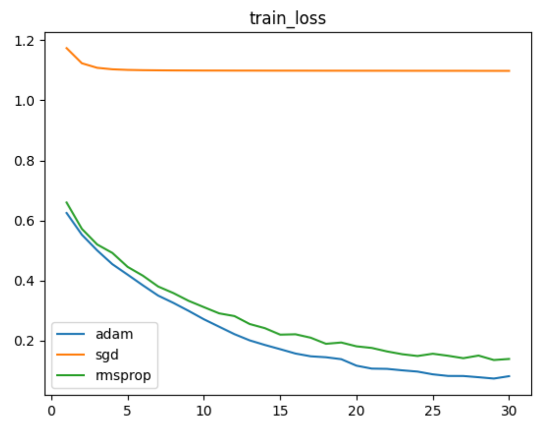
Model Name	Train Accuracy	Validation Accuracy
LR_unprocessed_none	0.622	0.579
LR_unprocessed_l2	**0.596**	**0.582**
LR_unprocessed_l1	0.585	0.576
LR_unprocessed_elasticnet	0.588	0.579
LR_processed_none	0.583	0.553
LR_processed_l2	0.564	0.551
LR_processed_l1	0.556	0.547
LR_processed_elasticnet	0.558	0.548

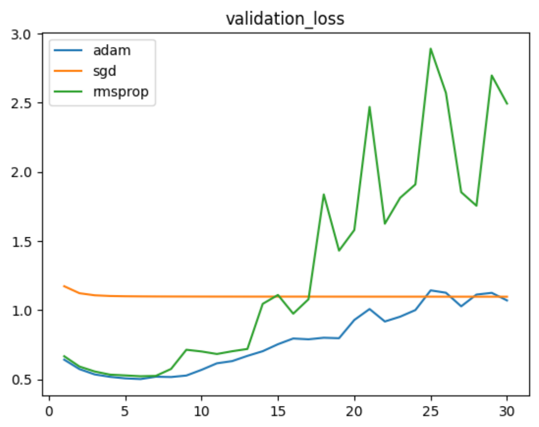
Of all the models, interestingly unprocessed l2 regularized TF IDF based Logistic regression models seems to be comparable. Both the accuracies are nearby and clearly shows that the model capacity needs to be increased. So, in the next part, I try and explore the RNN based methods.

Recurrent Neural Network classifier

The usual procedure to solve this kind of problems is to first embed the words into a fixed length vector and then feed the sequence of vectors to a RNN model to have a representation for the sequence of words. Once we have the representation, we use FC layers to classify it.
In all my below experiments I used **glove** embeddings.

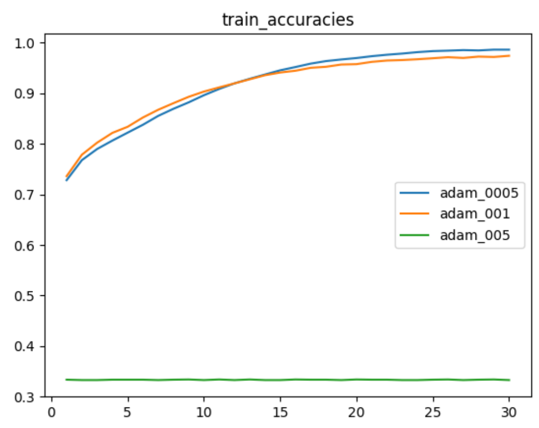
Experiment 1:
For the first experiment, I took a fairly complex model with 1 Bi directional RNN layer followed by 3 FC layers. And experimented with 3 optimizers, namely venilla SGD, Adam and RMS prop. Below are the graphs for the same.
Note: In all the graphs, x axis represent the number of epochs and y axis represent the metric mentioned in title of the graph.

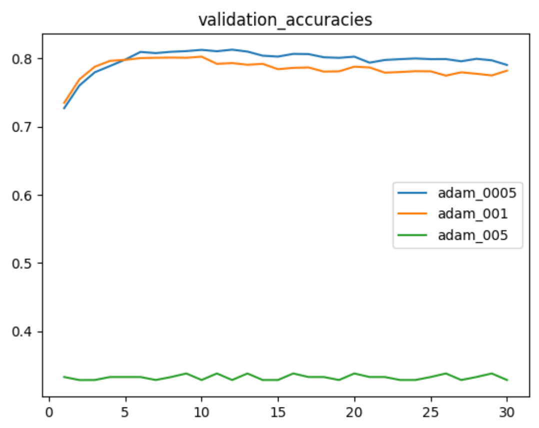




As the graph suggests, Adam optimizer is performing better than the other 2. (The model start to overfit after 8 epochs).

Experiment 2:
Adam optimizer seems working well for this problem. So as part of the next experiment, I played around with different learning rates.

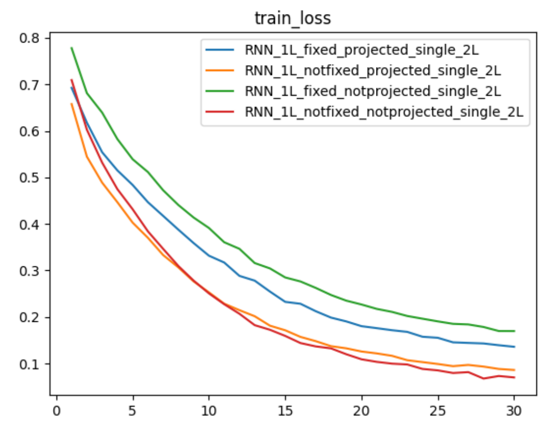
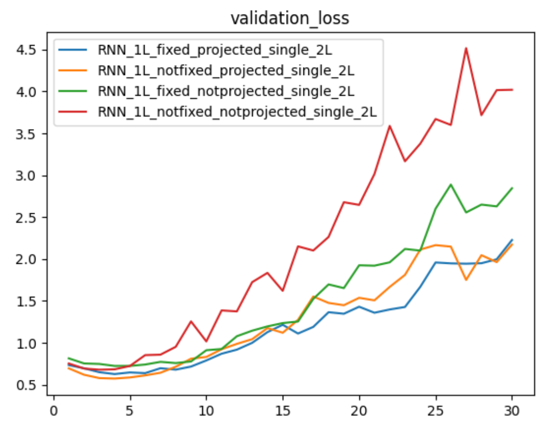
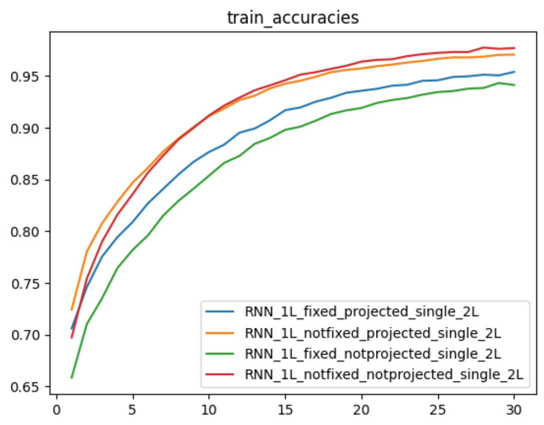
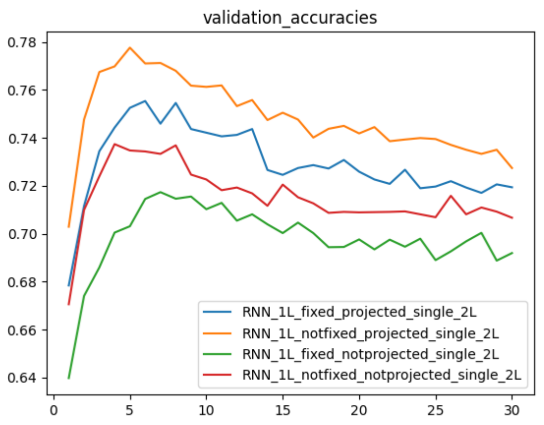
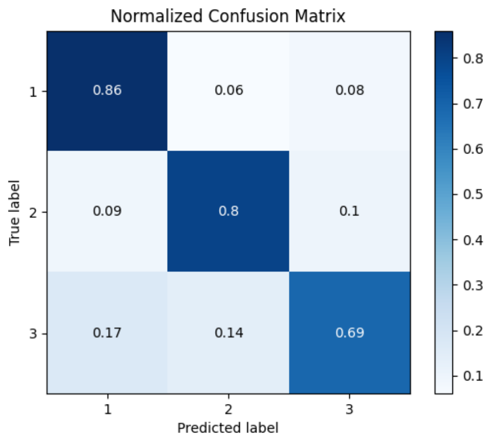
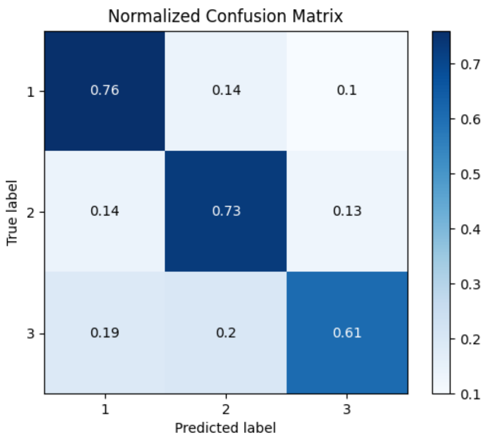




As the plot suggests, learning rate of 0.0005 seems to work well. Hence, fixed the lr to 0.0005

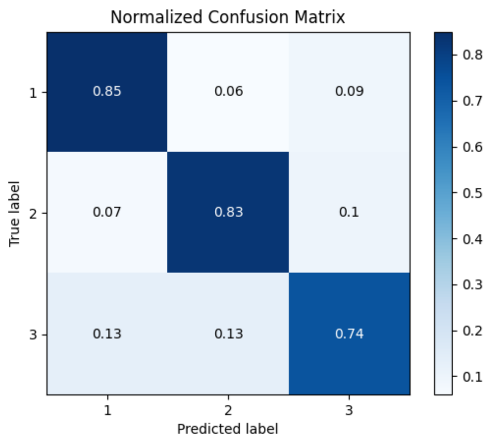
Experiment 3:

Now that the the optimizer and learning rate are fixed, I started playing around with the model architecture. I tried with 4 different models of uni directional LSTMs with combinations of fixing the embeddings and projecting the embeddings.
By projecting, I mean I had a FC layer right after embedding layer to increase the dimensions of embeddings before feeding it to the LSTMs.
Below are the graphs for the same.

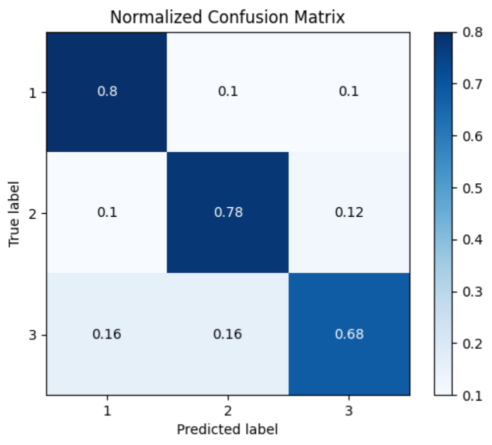
Train Stats	**Validation Stats**
	
	
	

RNN_1L_fixed_notprojected_single_2L

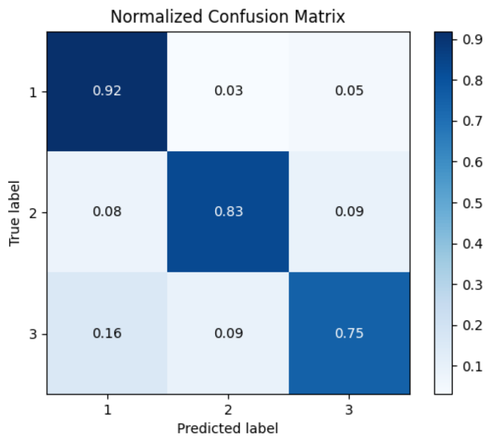
RNN_1L_fixed_notprojected_single_2L



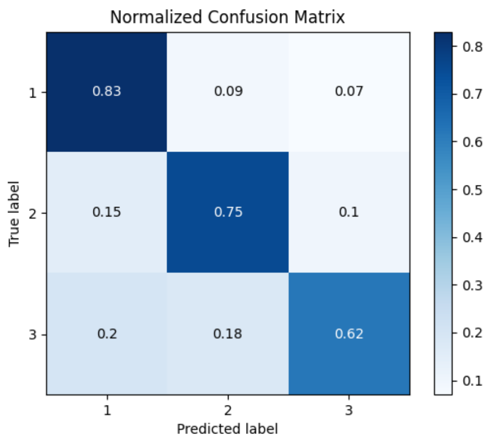
RNN_1L_fixed_projected_single_2L



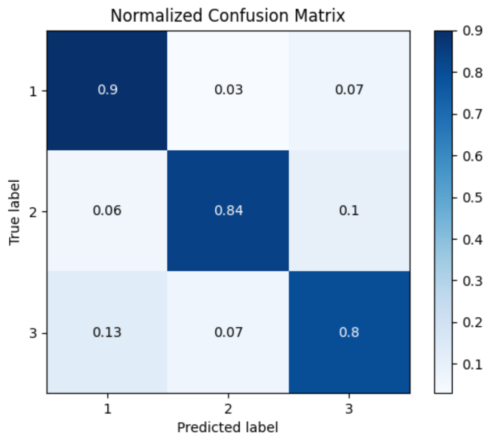
RNN_1L_fixed_projected_single_2L



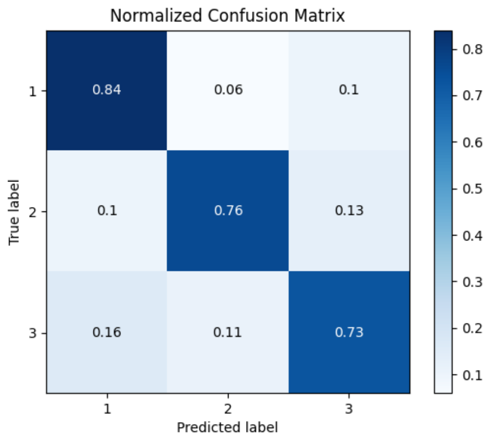
RNN_1L_notfixed_notprojected_single_2L



RNN_1L_notfixed_notprojected_single_2L



RNN_1L_notfixed_projected_single_2L

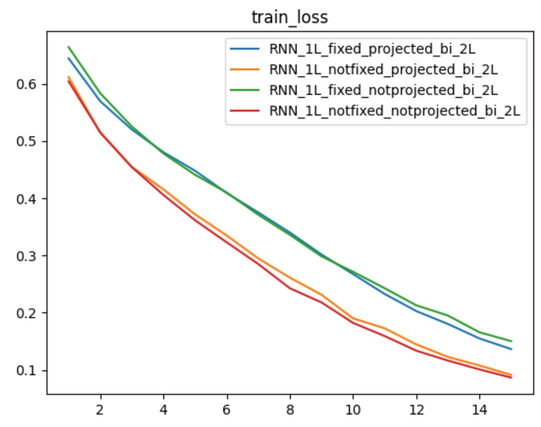
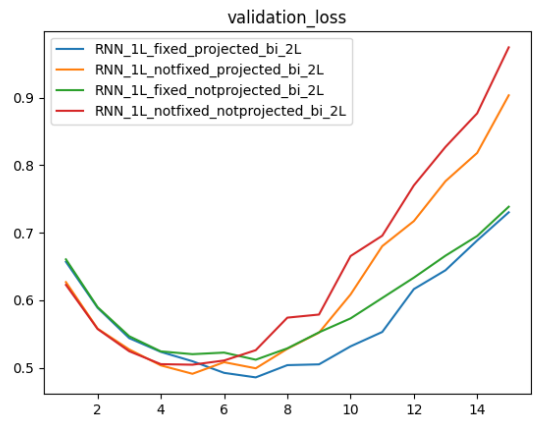
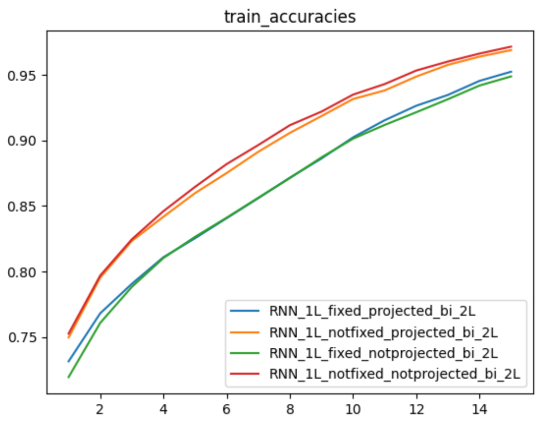
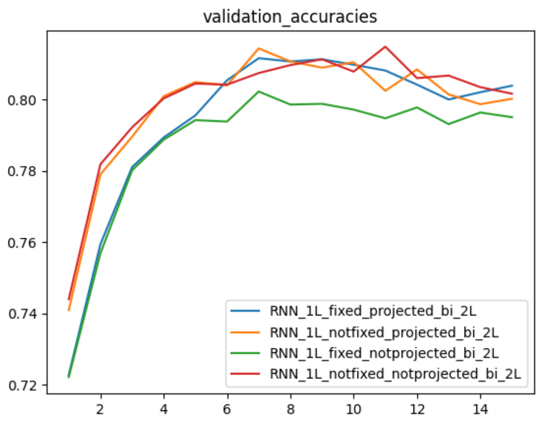
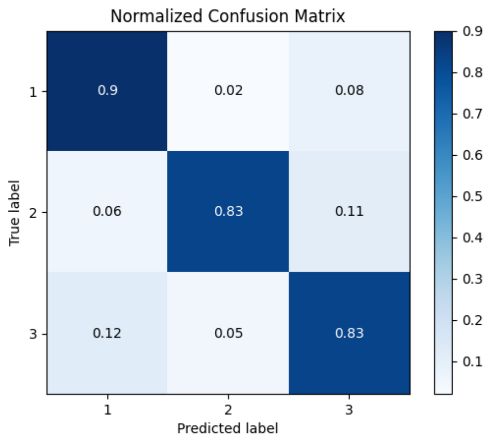
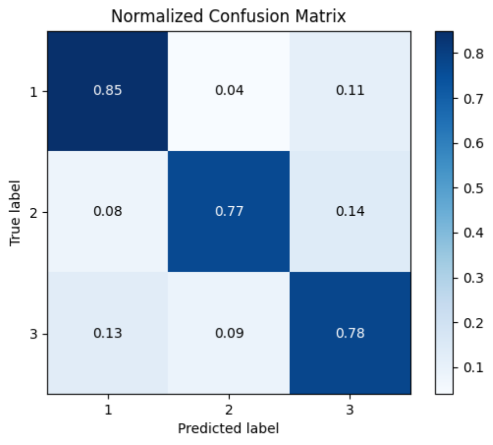


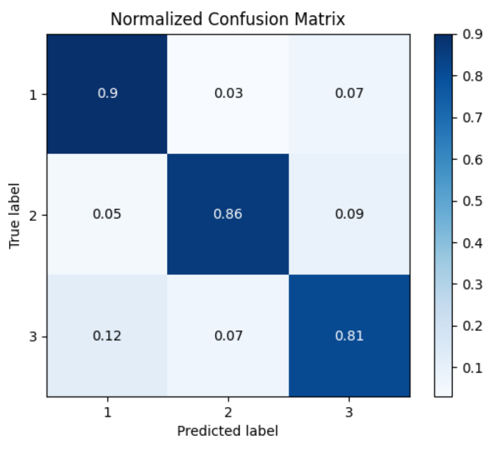
RNN_1L_notfixed_projected_single_2L

As the plots suggests, all the models are overfitting after epoch 5. I had been saving the model every epoch and the confusion matrices shown above are for epoch 5 of respective models.

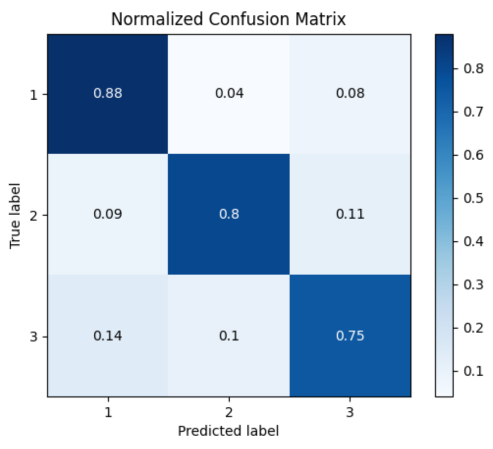
Experiment 4:

As part of next experiment, I wanted to try out the bi directional LSTMs to see if any improvements.
Below are the graphs for the same.

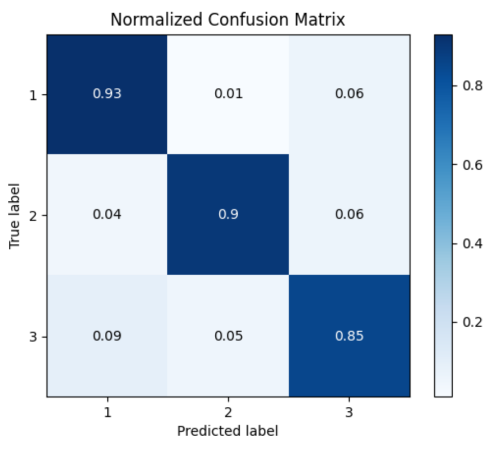
Train Stats	**Validation Stats**
	
	
 *RNN_1L_fixed_notprojected_bi_2L*	 *RNN_1L_fixed_notprojected_bi_2L*



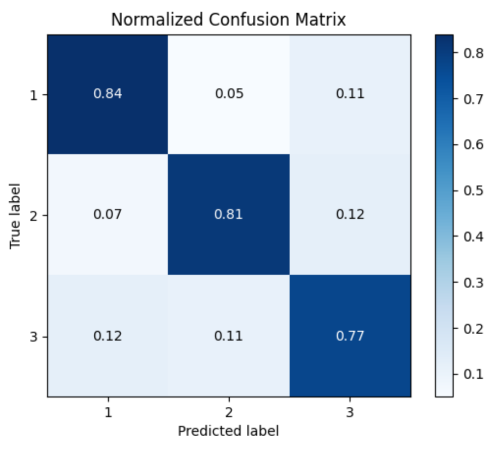
RNN_1L_fixed_projected_bi_2L



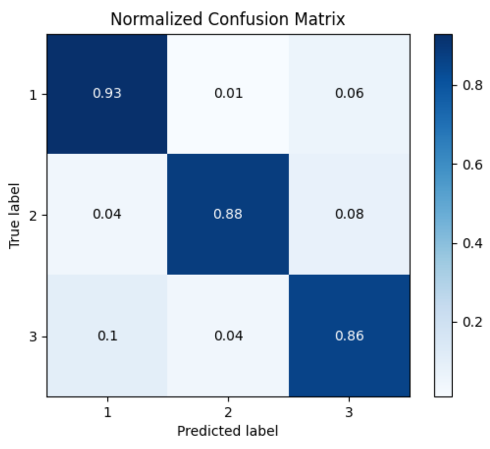
RNN_1L_fixed_projected_bi_2L



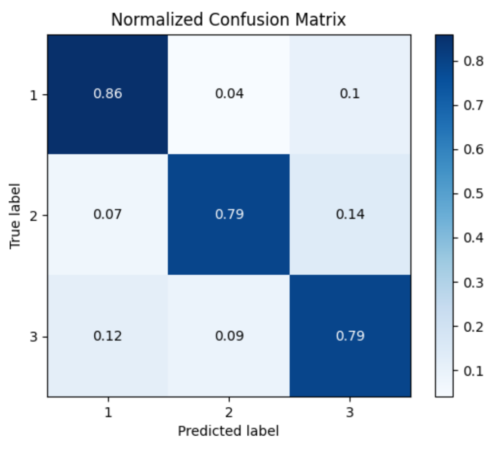
RNN_1L_notfixed_notprojected_bi_2L



RNN_1L_notfixed_notprojected_bi_2L



RNN_1L_notfixed_projected_bi_2L

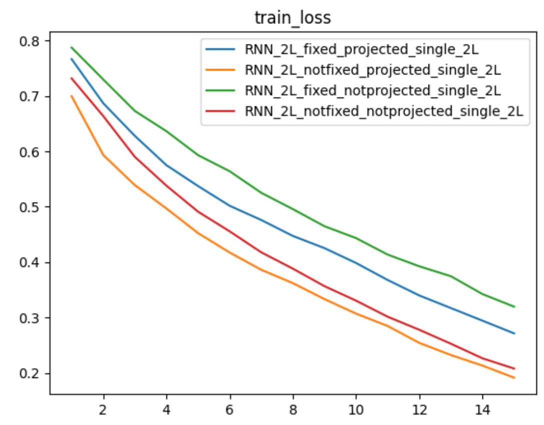
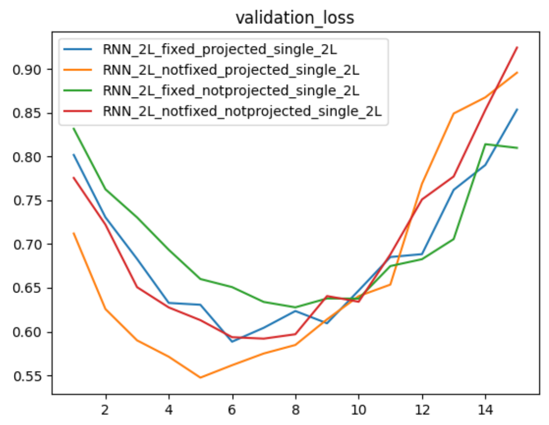
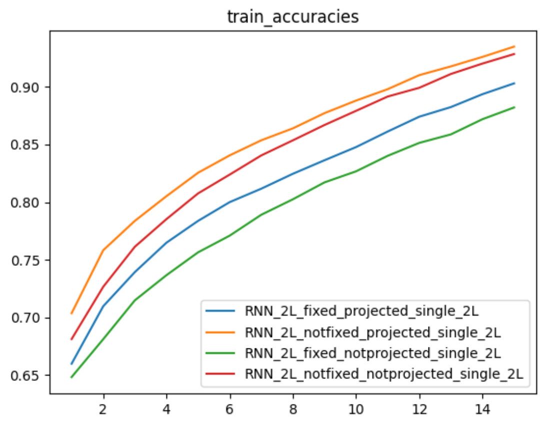
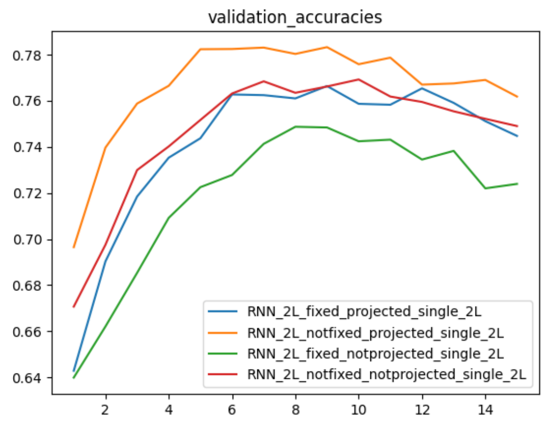
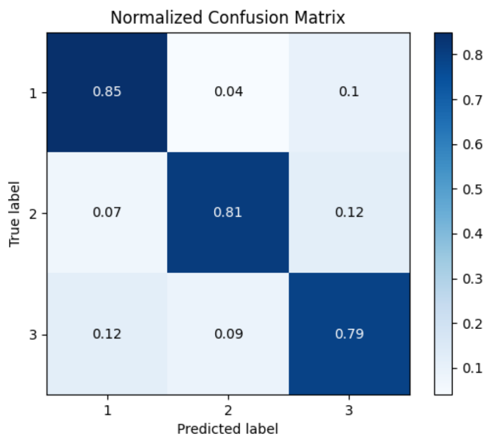
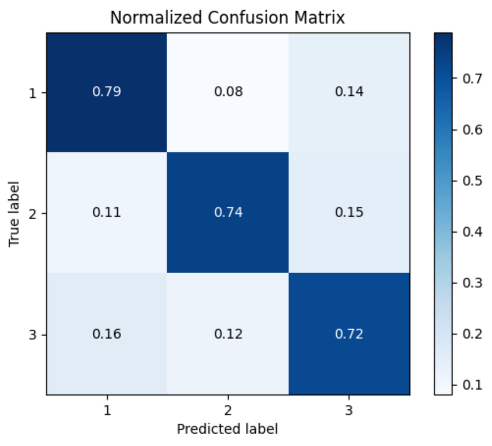


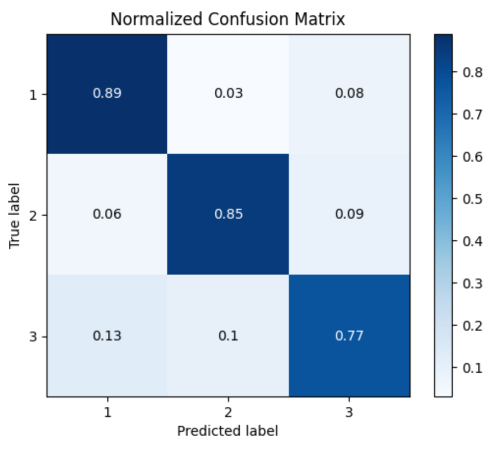
RNN_1L_notfixed_projected_bi_2L

As the plots suggest the models are ovefitting after epoch 7. The plots are of epoch 7.

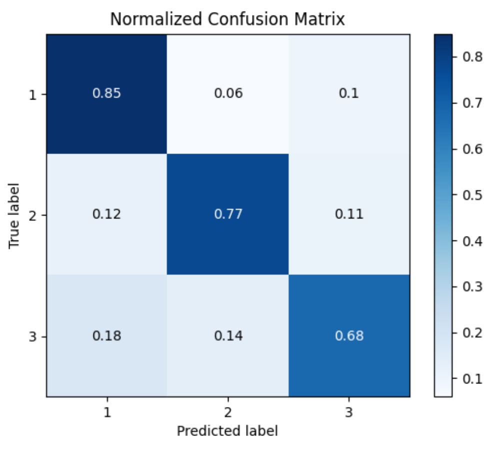
Experiment 5:

As the improvements were not to a great extent, I wanted to try out with 2 layer uni directional LSTM followed by 2 FC layers.
Below are the graphs for the same.

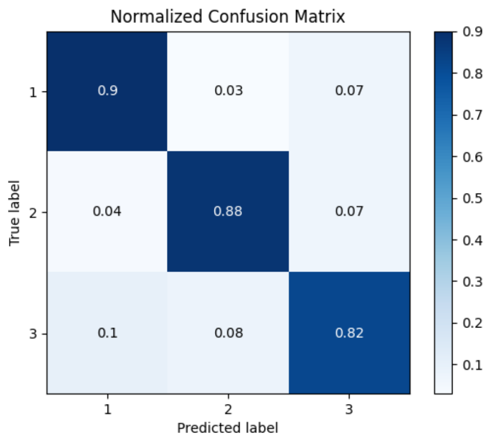
Train Stats	**Validation Stats**
	
	
 *RNN_2L_fixed_notprojected_single_2L*	 *RNN_2L_fixed_notprojected_single_2L*



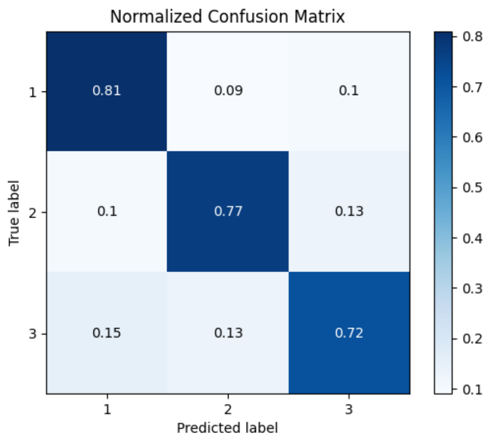
RNN_2L_fixed_projected_single_2L



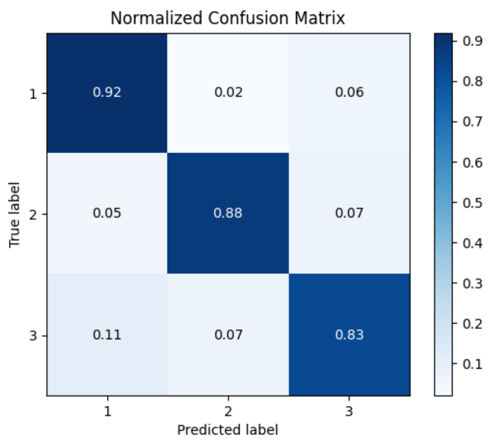
RNN_2L_fixed_projected_single_2L



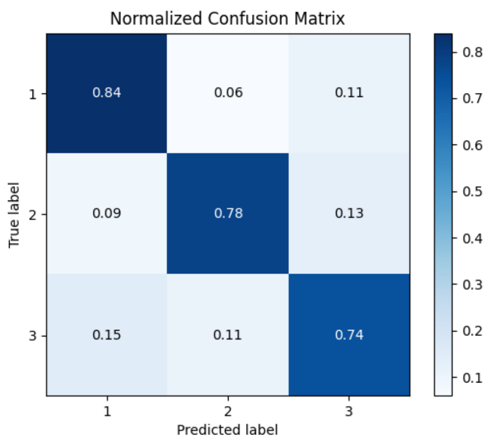
RNN_2L_notfixed_notprojected_single_2L



RNN_2L_notfixed_notprojected_single_2L



RNN_2L_notfixed_projected_single_2L

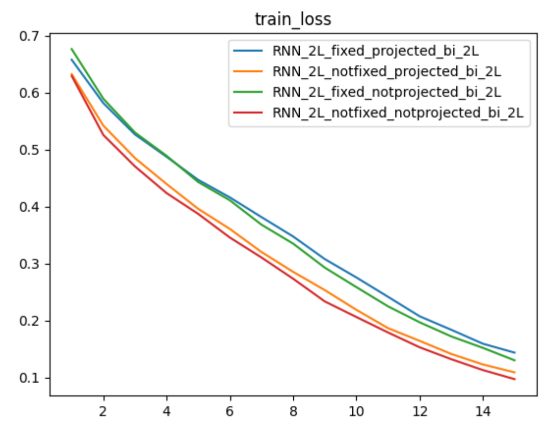
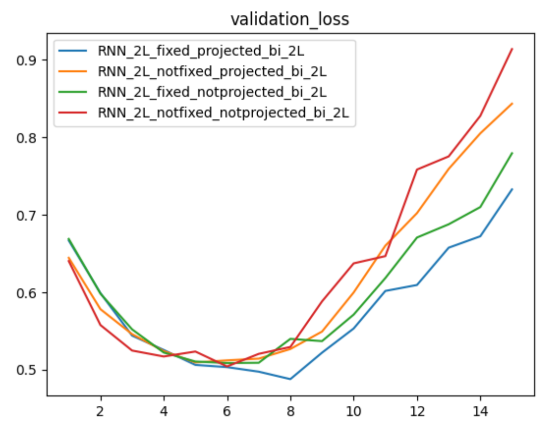
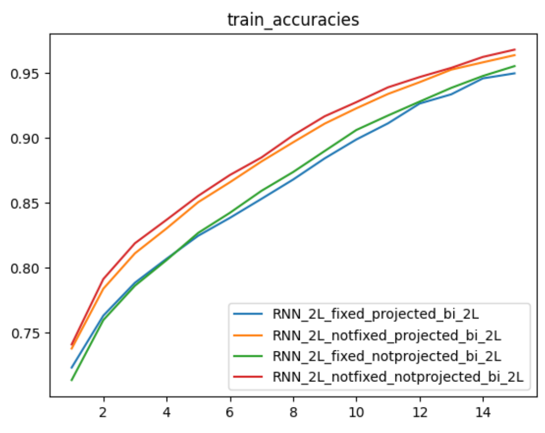
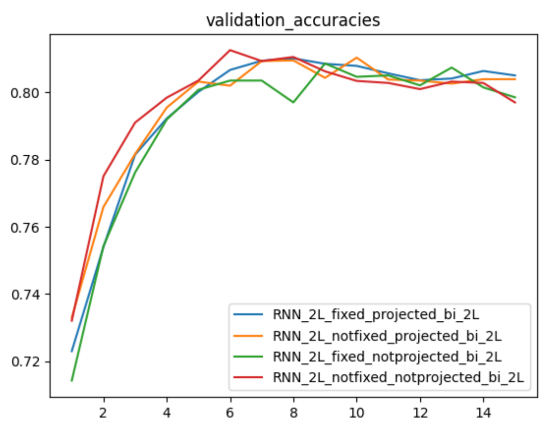
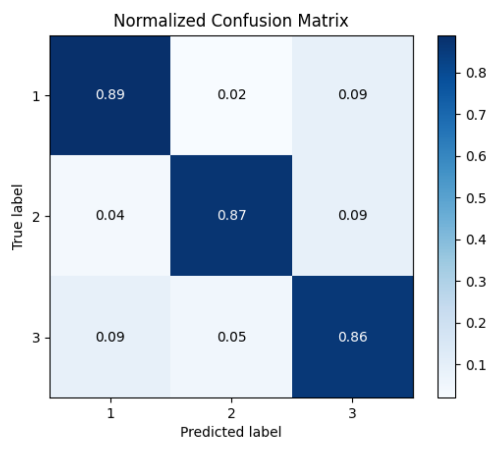
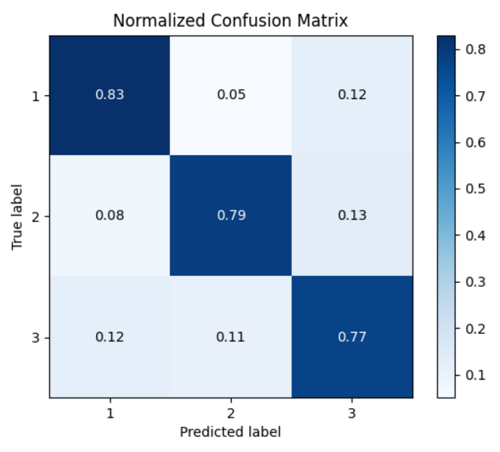


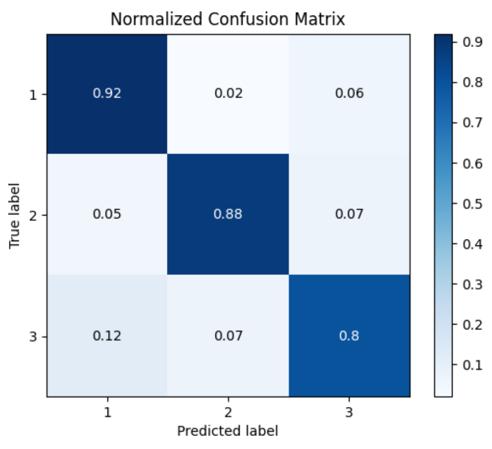
RNN_2L_notfixed_projected_single_2L

As the plots suggest the models at epoch 9 are giving good accuracies. (The plots are of epoch 9).

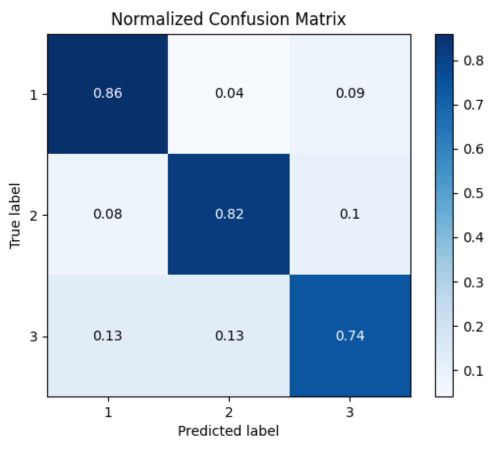
Experiment 6:

Next, I wanted to try out with 2 layer bi directional LSTM followed by 2 FC layers. Below are the graphs for the same.

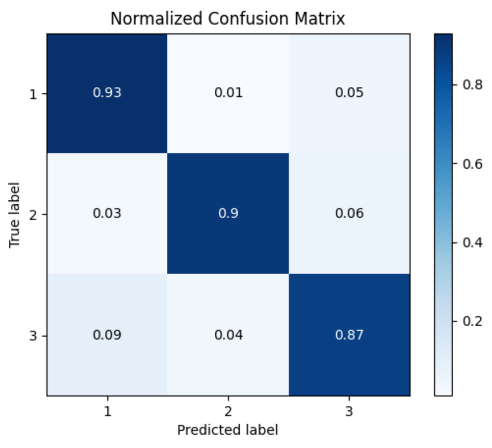
Train Stats	**Validation Stats**
	
	
	
RNN_2L_fixed_notprojected_bi_2L	*RNN_2L_fixed_notprojected_bi_2L*



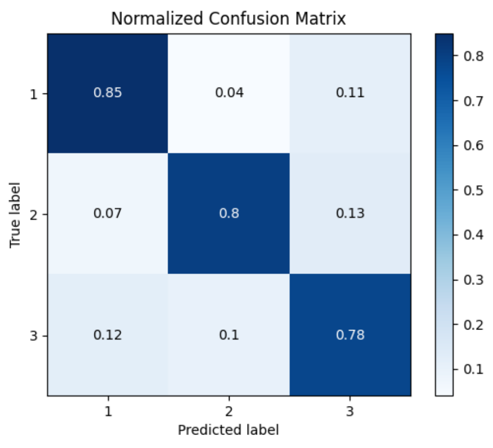
RNN_2L_fixed_projected_bi_2L



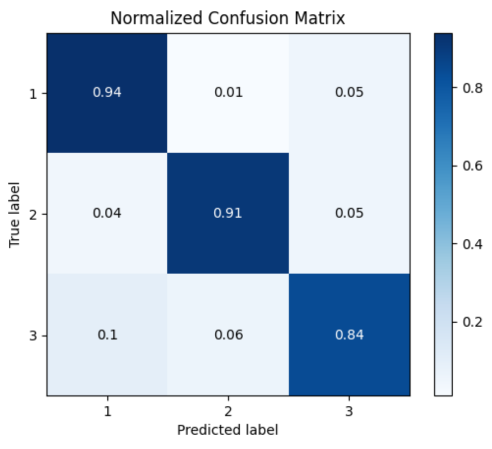
RNN_2L_fixed_projected_bi_2L



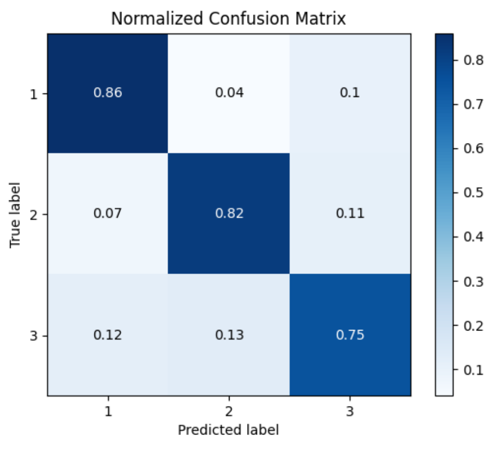
RNN_2L_notfixed_notprojected_bi_2L



RNN_2L_notfixed_notprojected_bi_2L



RNN_2L_notfixed_projected_bi_2L

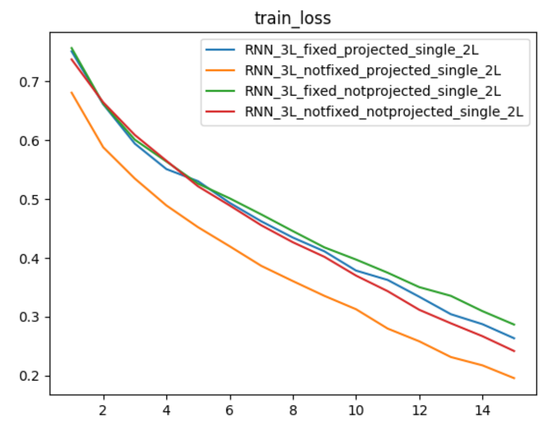
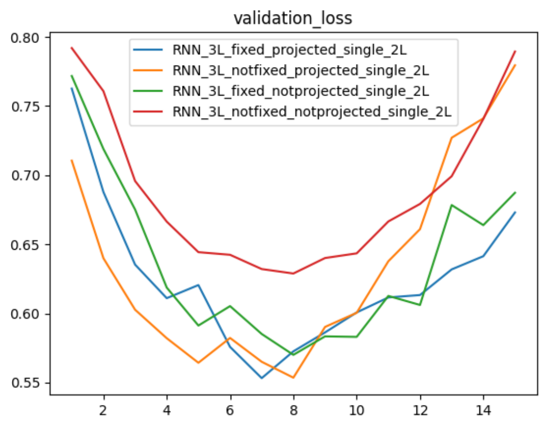
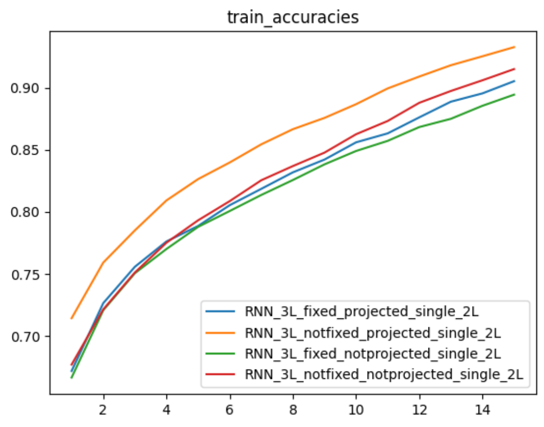
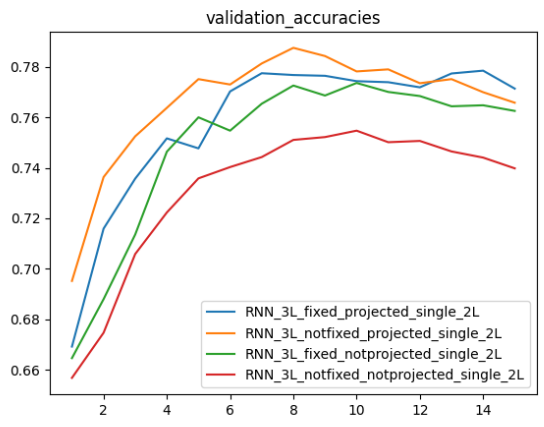
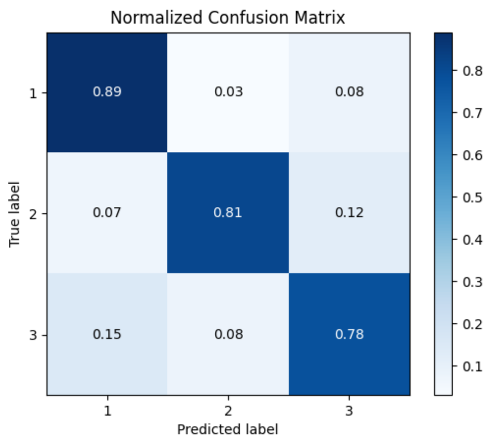
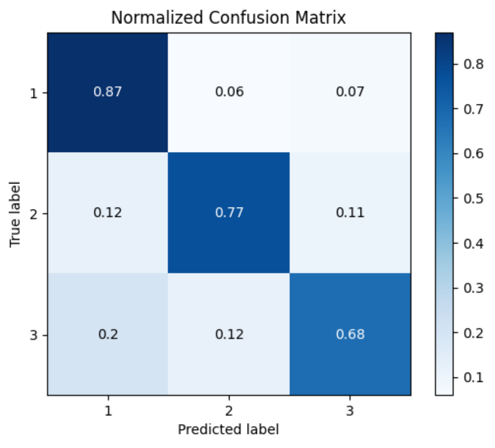


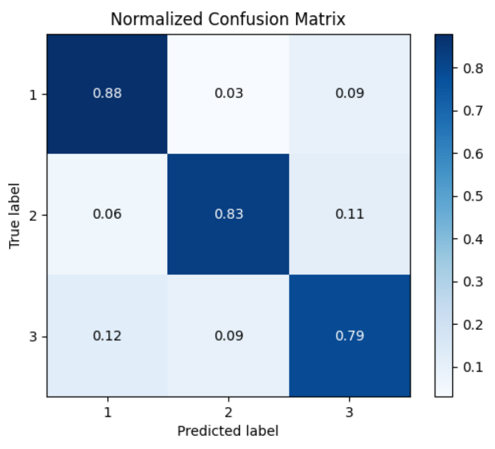
RNN_2L_notfixed_projected_bi_2L

As the plots suggest, the best models are of that of epoch 8. (Plots from epoch 8)

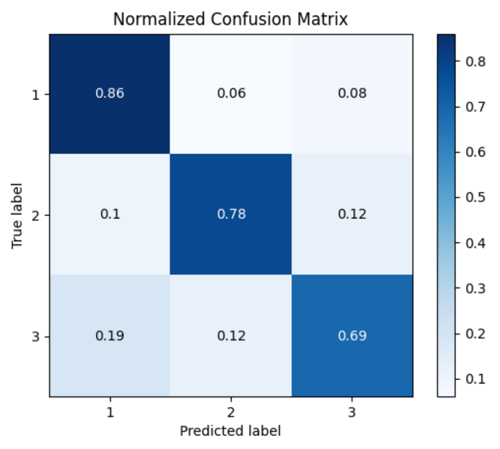
Experiment 7:

As part of next experiment, I wanted to try out with 3 layer uni directional LSTM followed by 2 FC layers.
Below are the graphs for the same.

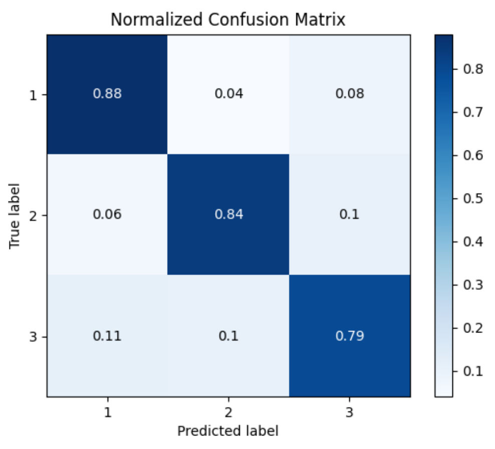
Train Stats	**Validation Stats**
	
	
	
RNN_3L_fixed_notprojected_single_2L	*RNN_3L_fixed_notprojected_single_2L*



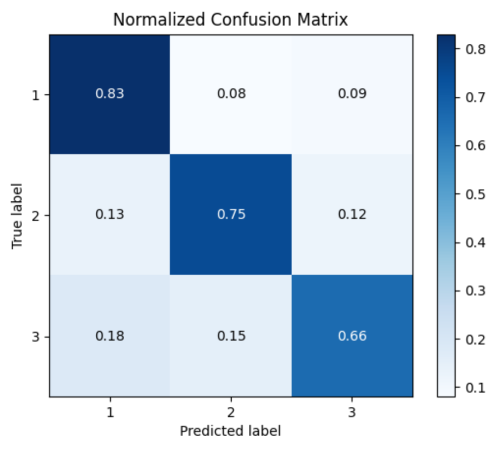
RNN_3L_fixed_projected_single_2L



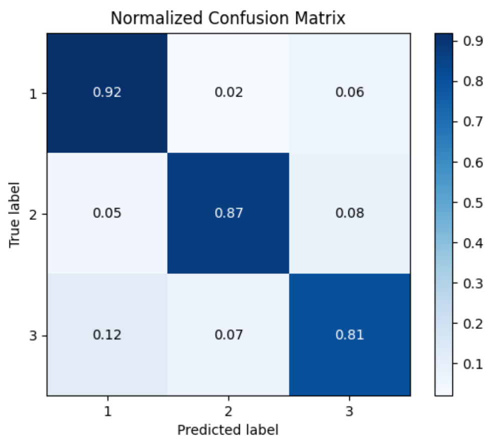
RNN_3L_fixed_projected_single_2L



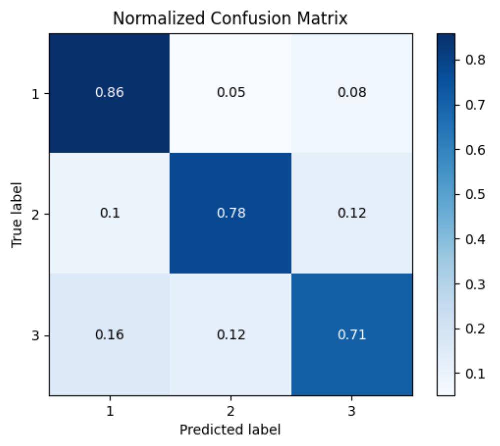
RNN_3L_notfixed_notprojected_single_2L



RNN_3L_notfixed_notprojected_single_2L



RNN_3L_notfixed_projected_single_2L

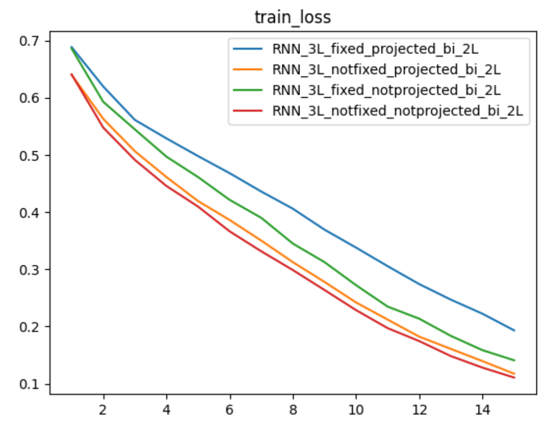
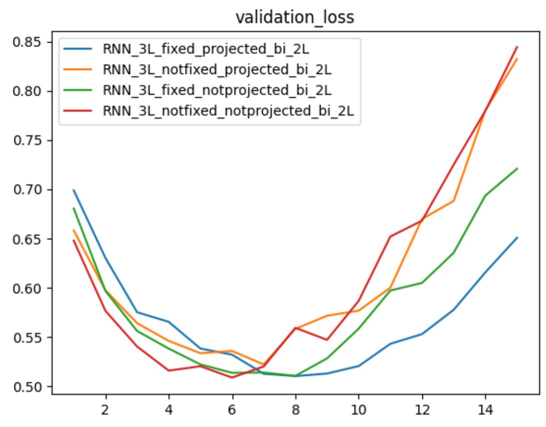
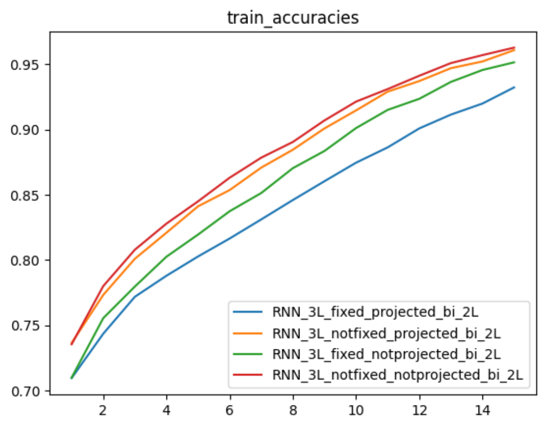
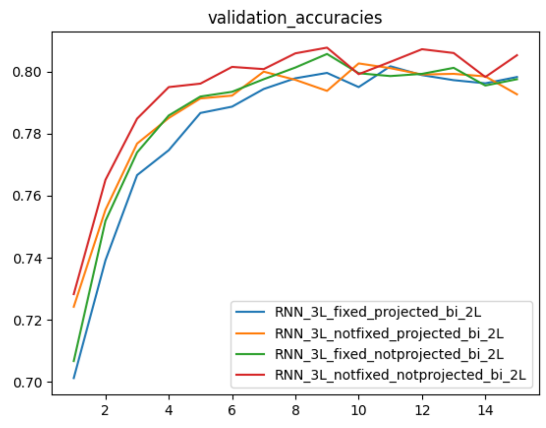
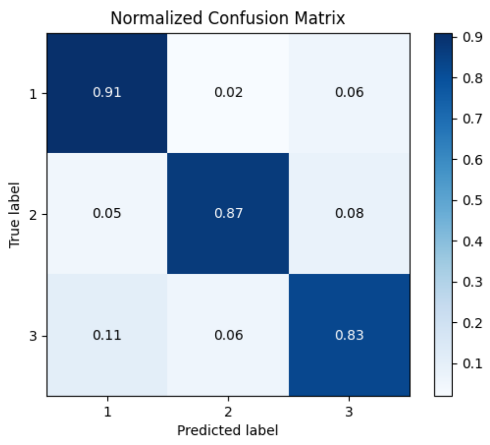
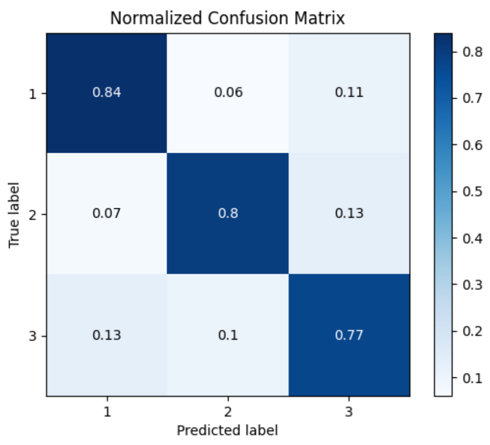


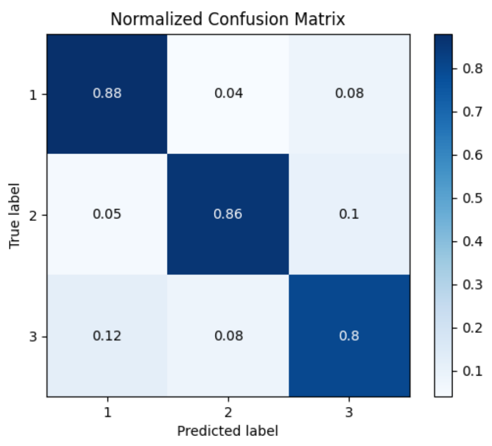
RNN_3L_notfixed_projected_single_2L

As the plots suggest, the best models are of that of epoch 8. (Plots from epoch 8)

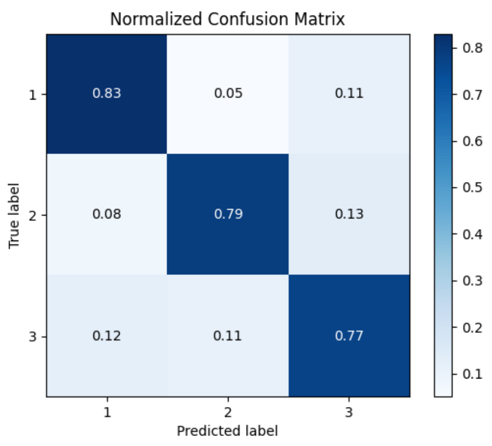
Experiment 8:

Next, I wanted to try out with 3 layer bi directional LSTM followed by 2 FC layers.
Below are the graphs for the same.

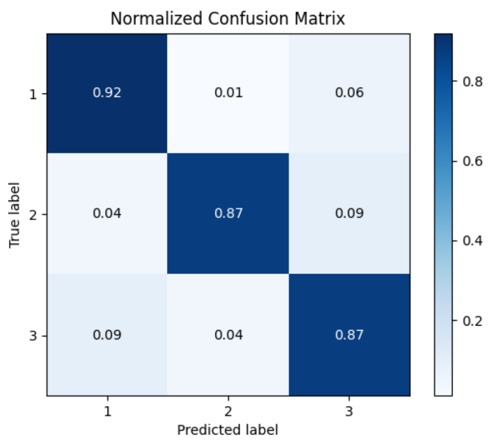
Train Stats	Validation Stats
	
	
	
RNN_3L_fixed_notprojected_bi_2L	*RNN_3L_fixed_notprojected_bi_2L*



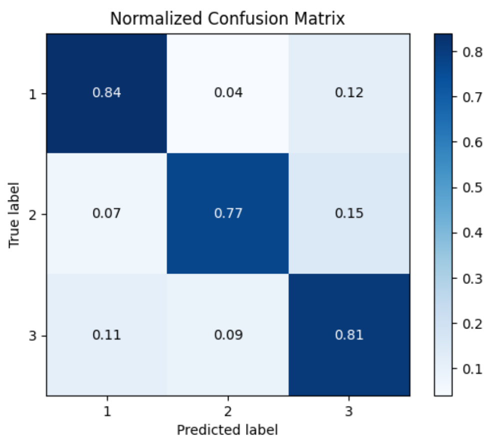
RNN_3L_fixed_projected_bi_2L



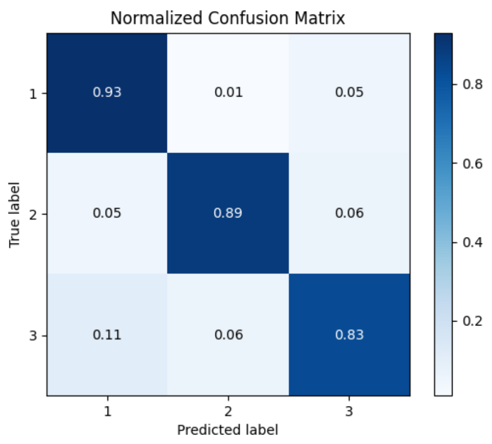
RNN_3L_fixed_projected_bi_2L



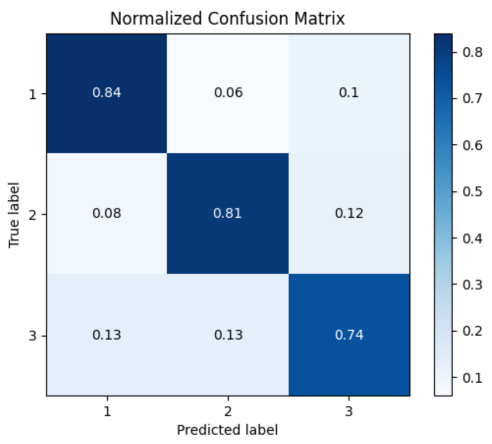
RNN_3L_notfixed_notprojected_bi_2L



RNN_3L_notfixed_notprojected_bi_2L



RNN_3L_notfixed_projected_bi_2L



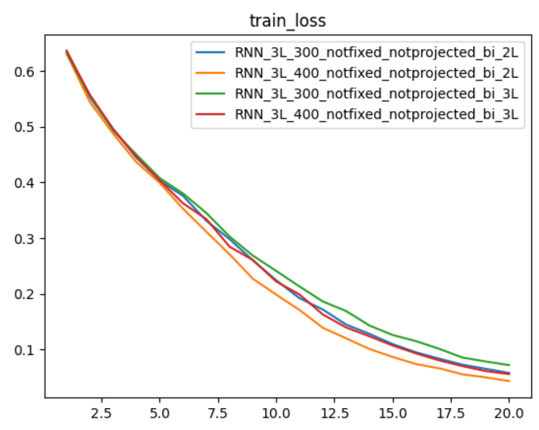
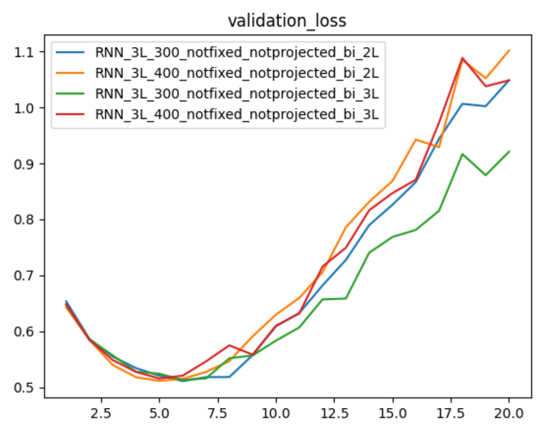
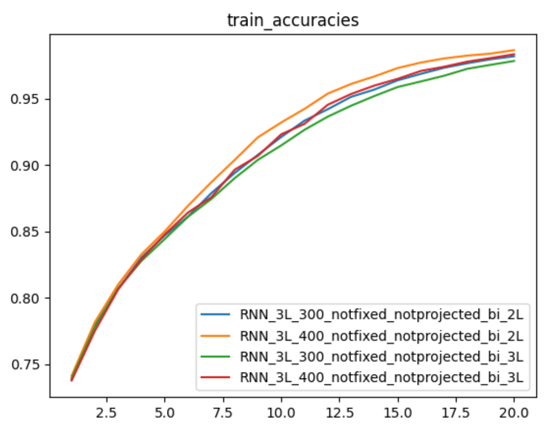
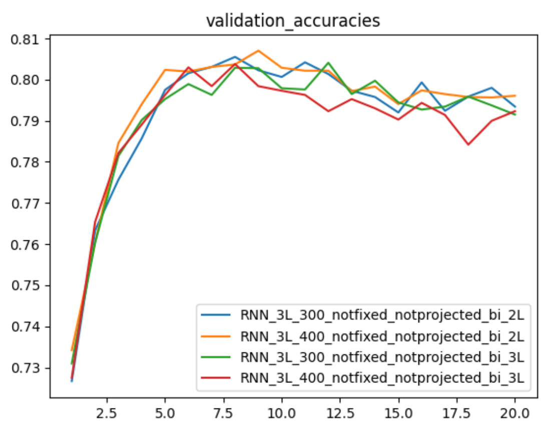
RNN_3L_notfixed_projected_bi_2L

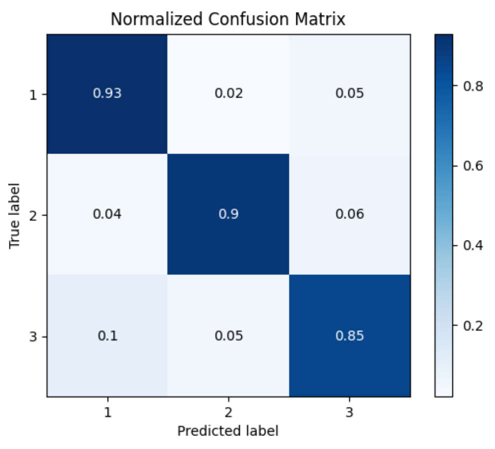
As the plots suggest, the best models are of that of epoch 8. (Plots from epoch 8)

Experiment 9:

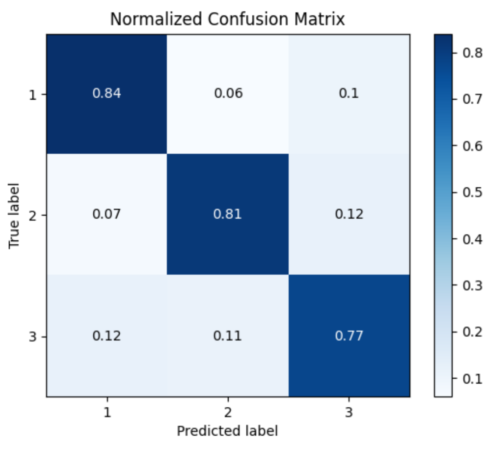
The 3 layer bi directional models with out fixed embeddings and without projections are performing better. Lastly, I wanted to try out with different configurations of FC layers and LSTM hidden states. I tried with a combination of LSTMs with hidden units of 300 or 400 followed by 2 or 3 layers of FC.

Below are the graphs for the same.

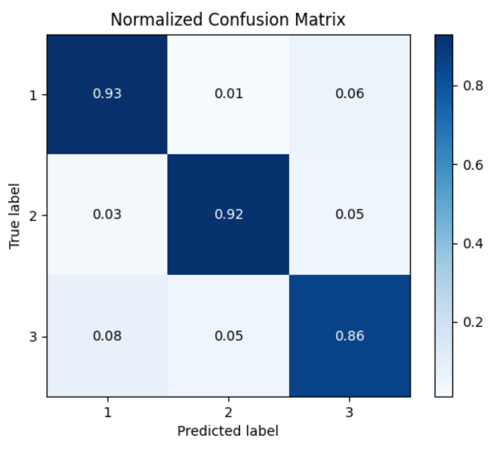
Train Stats	**Validation Stats**
	
	



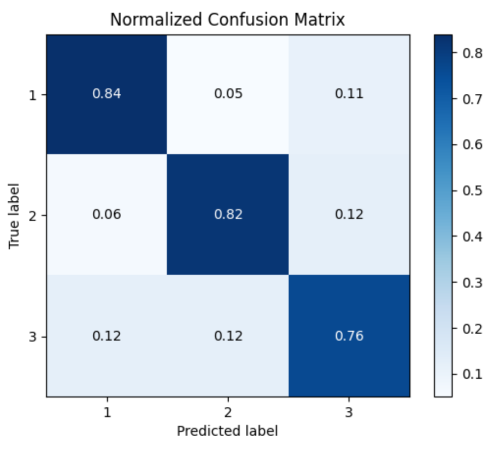
RNN_3L_300_notfixed_notprojected_bi_2L



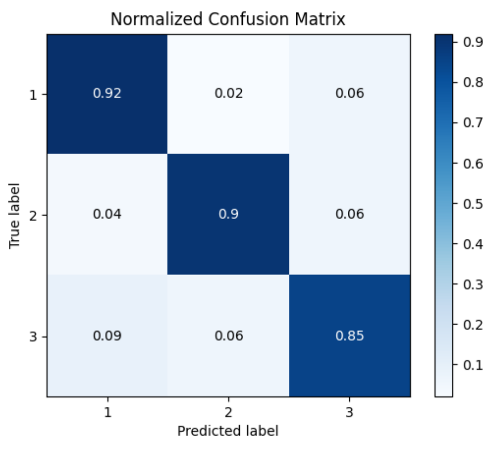
RNN_3L_300_notfixed_notprojected_bi_2L



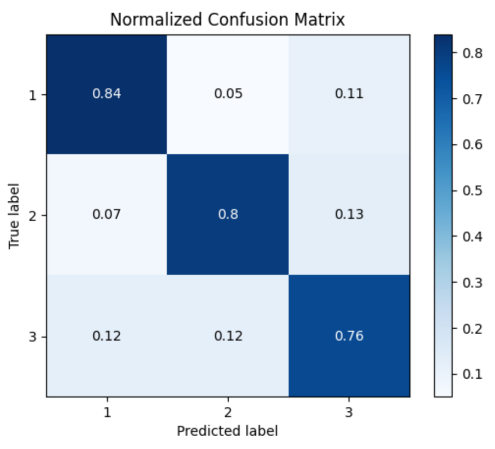
RNN_3L_400_notfixed_notprojected_bi_2L



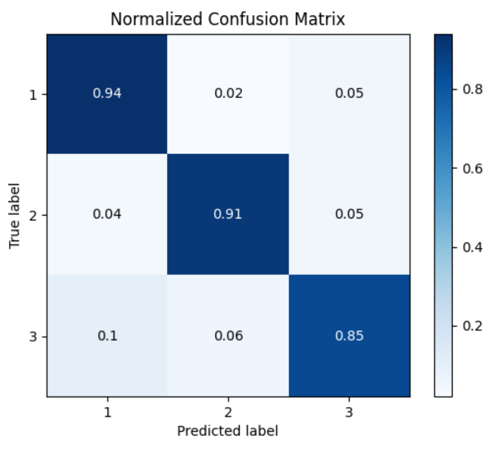
RNN_3L_400_notfixed_notprojected_bi_2L



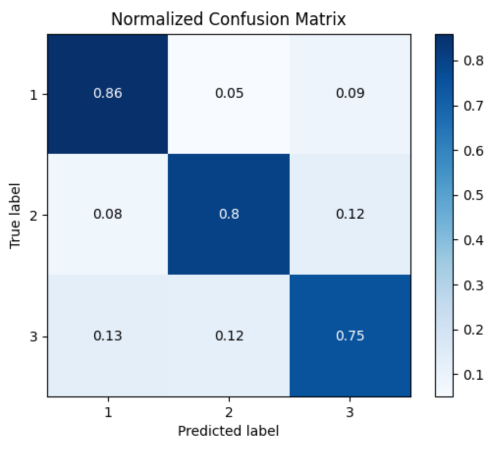
RNN_3L_300_notfixed_notprojected_bi_3L



RNN_3L_300_notfixed_notprojected_bi_3L



RNN_3L_400_notfixed_notprojected_bi_3L

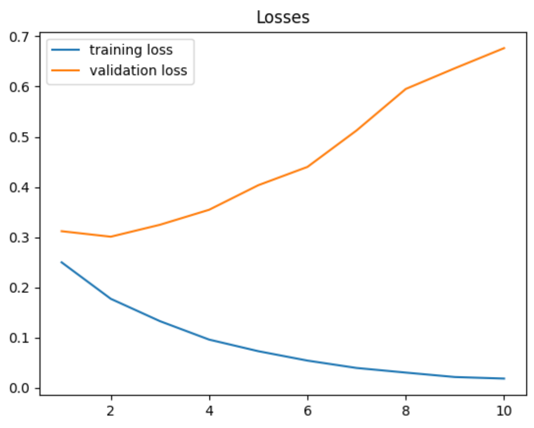


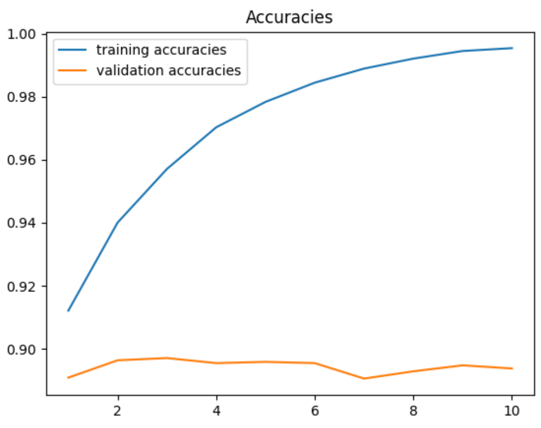
RNN_3L_400_notfixed_notprojected_bi_3L

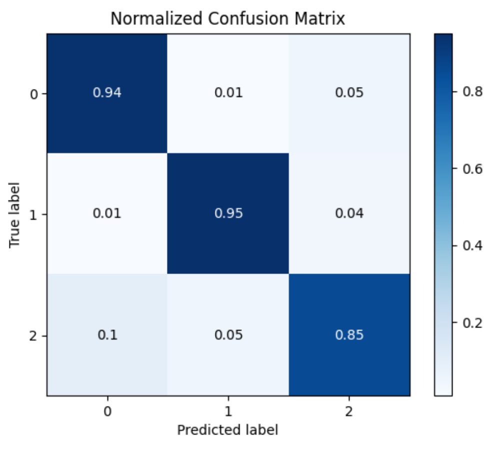
The best model is that of epoch 8. Hence using the model and plots from epoch 8.

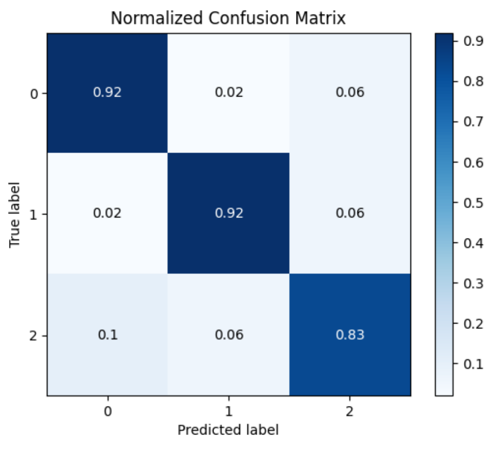
BERT based classifier

As, the new attention models are performing better than the RNN models in many tasks, I wanted to try out fine tuning BERT and see the performace on this task. I used the allennlp module for fine tuning bert using their library functions.









Clearly, BERT based classifier is doing better than the RNN models. Just training the model with one epoch gave train and test accuracies close to 90%. (increase in performance by 10% from RNN models)After that, seems like model is overfitting. Hence taking the model after one model. (Plots are from model after Epoch 1)

Conclusion:

Of all the tried models, BERT based classifier does best. Hence, chosing that model as the final deep model.

Code Details:

Repo: https://github.com/EshwarSR/IISc-DL-Project-3.git

Please refer to the file README.md for more details on the code base.