Natural Language Processing - IMDB Movie Review							
	Description	Hyperparameters	Number of Epochs	Training Loss	Training Accuracy(%)	Test Accuracy(%)	Comments
Part 1a	(Default) Word Embedding Layer + Fully Connected Layer + BatchNorm + Relu + Dropout + Fully Connected Layer	ADAM optimizer LR=0.001, BatchSize=200, VocabularySize=8000, HiddenUnits=500	6	0.146129922	94.352	86.732	Baseline
	(Default) Word Embedding Layer + Fully Connected Layer + BatchNorm + Relu + Dropout + Fully Connected Layer	ADAM optimizer LR=0.001, BatchSize=200, VocabularySize=8000, HiddenUnits=10	6	0.456190443	79.828	80.936	Choose smaller HiddenUnits to make the neural network tighter, hence prone to underfit. Another interesting observatioin is that the traning accuracy is smaller than test accuracy.
	Word Embedding Layer + Fully Connected Layer + BatchNorm + Relu + Fully Connected Layer	ADAM optimizer LR=0.001, BatchSize=100, VocabularySize=8000, HiddenUnits=2000	6	0.091045912	96.676	87.232	Choose smaller BatchSize to avoid running out GPU memory. Choose larger HiddenUnits to make the neural network wider, so that the model are prone to overfit. It's hard to see the current configurations make the model overfit, but increasing the number of hidden units indeed improves the model performance.
	(Default) Word Embedding Layer + Fully Connected Layer + BatchNorm + Relu + Dropout + Fully Connected Layer	SGD optimizer LR=0.001, Momentum=0.9, Nesterov=True, BatchSize=200, VocabularySize=8000, HiddenUnits=500	6	0.55105129	72.776	74.724	The Nesterov's method has O(1/k^2) convergence rate for convex problems, so I thought it would be faster than Adam. But suprisingly, with the same number of epochs, SGD performs worse than the Adam. And the time for each epoch for the SGD(Nesterov) is about the same as that of the Adam, approximately 16.6s.
Part 1b	(Default) Fully Connected Layer + BatchNorm + Relu + Dropout + Fully Connected Layer	ADAM optimizer LR=0.001, BatchSize=200, VocabularySize=100000, HiddenUnits=500	8	0.289472647	87.976	85.028	Baseline
	(Default) Fully Connected Layer + BatchNorm + Relu + Dropout + Fully Connected Layer	ADAM optimizer LR=0.001, BatchSize=200, VocabularySize=100000, HiddenUnits=300	8	0.296491095	87.7	87.968	This result is actually interesting. Using narrower net, meaning HiddenUnits=300, will slightly harm the train accuracy, but indeed improve the traning accuracy. So maybe HiddenUnits=500 begins to overfit.
	Fully Connected Layer + BatchNorm + Relu + Fully Connected Layer + BatchNorm + Relu + Fully Connected Layer	ADAM optimizer LR=0.001, BatchSize=100, VocabularySize=100000, HiddenUnits=2000	8	0.194250891	91.792	86.412	This result is also interesting comparing to the first and the second results. Firstly, widder and deeper nets intuitively are more likely to overfit. But it seems that the deepth can compensate for the overfitting, that is makes the net not that overfiited provided that test accuracy is ~86%.
	(Default) Fully Connected Layer + BatchNorm + Relu + Dropout + Fully Connected Layer	SGD optimizer LR=0.001, Momentum=0.9, Nesterov=True, BatchSize=200, VocabularySize=100000, HiddenUnits=500	8	0.387589687	83.144	84.2	Again, SGD performs worse than the Adam. And the running time per epoch for the SGD(Nesterov) is slightly(0.02 seconds) faster that of the Adam.
Part 2a	(Default) Embedding + LSTMcell + Batch Normalization + Dropout + MaxPooling1d + Fully Connected Layer	ADAM optimizer LR=0.001, BatchSize=200, VocabularySize=8000, HiddenUnits=500	20	0.087196272	97.048	seq_len accuracy 50 76.184 100 82.088 150 85.636 200 86.884 250 87.308 300 87.812 350 87.888 400 87.924 450 88.084	Trained with sequence_length = 100. The same for the following custom tests. But for testing purpose, the model is tested on various sequence length.
	(Default) Embedding + LSTMcell + Batch Normalization + Dropout + MaxPooling1d + Fully Connected Layer	ADAM optimizer LR=0.001, BatchSize=200, VocabularySize=8000, HiddenUnits=100	20	0.229649618	91.104	seq_len accuracy 50 75.808 100 82.04 150 84.8 200 86.412 250 86.892 300 87.088 350 87.276 400 87.268 450 87.432	It seems that decresing the number of hidden units won't significantly harm the test accuracy. It turns out that testing on longer sequence will more likely to give higher test accuray. This is validated by all the experiments (except those with SGD optimizers.)
	(Default) Embedding + LSTMcell + Batch Normalization + Dropout + MaxPooling1d + Fully Connected Layer	ADAM optimizer LR=0.001, BatchSize=100, VocabularySize=10000, HiddenUnits=500	20	0.081477948	97.156	seq_len accuracy 50 75.868 100 82.116 150 85.064 200 86.74 250 87.392 300 87.76 350 87.712 400 87.916 450 88.004	Increasing the vocabulary size by 2000 won't significantly imporve the training accuracy, however, will have slightly negative impacts on the testing accuracy, where is a sign of overfitting.
	(Default) Embedding + LSTMcell + Batch Normalization + Dropout + MaxPooling1d + Fully Connected Layer	SGD optimizer LR=0.001, BatchSize=200, VocabularySize=8000, HiddenUnits=500	20	0.617245913	65.872	seq_len accuracy 50 69.04 100 72.728 150 73.776 200 73.152 250 72.76 300 72.308 350 71.784 400 71.308 450 71.304	SGD with Nesterov momentum performs terribly in both training accuracy and testing accuracy.
Part 2b	(Default) Embedding + LSTMcell + Batch Normalization + Dropout + MaxPooling1d + Fully Connected Layer	ADAM optimizer LR=0.001, BatchSize=200, VocabularySize=10000, HiddenUnits=500	20	0.208934055	91.592	seq_len accuracy 50 79.100 100 84.936 150 87.76 200 89.112 250 89.42 300 89.544 350 89.42 400 89.532 450 89.44	Trained with sequence_length = 100. The same for the following custom tests. But for testing purpose, the model is tested on various sequence length. I failed to achieved 91% test accuracy for the given model structure (from the tutorial) and hyperparameters (from 1a). Maybe increasing the number of epoches can eventually achieve ~91% test accuracy. But compared with 2a), using glove features indeed helps to boost the performance, though not significant gains for some testing sequence length. But in my later experiments, I do find the glove features give the highest testing accuracy so far.
	(Default) Embedding + LSTMcell + Batch Normalization + Dropout + MaxPooling1d + Fully Connected Layer	ADAM optimizer LR=0.001, BatchSize=200, VocabularySize=10000, HiddenUnits=100	20	0.282447548	88.052	seq_len accuracy 50 79.096 100 84.58 150 87.552 200 88.62 250 89.296 300 89.208 350 89.324 400 89.476 450 89.508	The test accuracy is consistently much better than using the trained word-embeddings, which means that transfer learning helps when we take the temporal information into account. On the other hand, reducing the hidden units to 100 won't significantly har the testing accuracy, but indeed lower the train accuracy by ~3%.
	(Default) Embedding + LSTMcell + Batch Normalization + Dropout + MaxPooling1d + Fully Connected Layer	ADAM optimizer LR=0.001, BatchSize=100, VocabularySize=10000, HiddenUnits=2000	20	0.185335722	92.624	seq_len accuracy 50 78.388 100 84.484 150 87.26 200 88.8 250 89.872 300 90.008 350 90.408 400 90.568 450 90.752	Increasing the number of hidden units to 2000 indeed improves the test accuarcy once the test sequence length is larger than 200. Interestingly, its test accuracy is slightly worse than that of 100 hidden units when test sequnce length are 50, 100, 150.
	(Default) Embedding + LSTMcell + Batch Normalization + Dropout + MaxPooling1d + Fully Connected Layer	SGD optimizer LR=0.001, BatchSize=200, VocabularySize=8000, HiddenUnits=500	20	0.477326702	76.992	seq_len accuracy 50 74.908 100 79.588 150 81.804 200 83.168 250 83.576 300 83.784 350 83.972 400 84.152 450 84.14	Even with the SGD optimizer, the glove feature embeddings indeed imporve the training accuracy and testing accuracy, especially for the testing accuracy, nearly 10% improvement.