CIFAR-10 Classification using CNN

Abtin Mahyar

1. Data

The process of aggregating and making the data ready for fitting into the models consists of following steps:

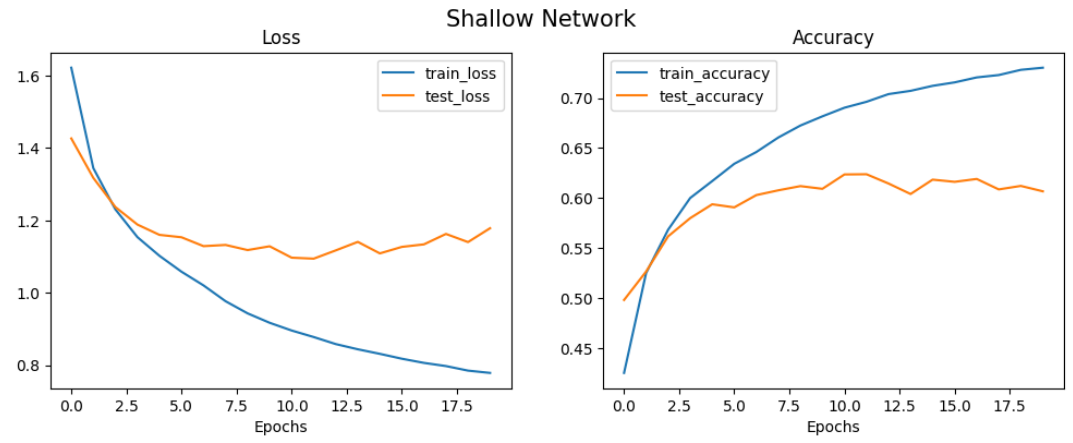
1. First of all, since this dataset is one of the built-in datasets that Torchvision library provides, I downloaded the data using this library. The data was available in two formats of train and test sets. There were 50,000 records in the training set, and 10,000 records in test set. There were 10 different classes as target labels for the records. The size of each input image is 32 * 32 pixels.
2. No especial transform function was performed on the data, since there was no need to augmentation because of size of the dataset, and the quality of dataset. Only one transformation was used in order to convert raw inputs to Torch tensor for ease of computation in future works.
3. 10,000 instances of training data were used as validation set for training phase.
4. Train, validation and test datasets were passed into their own data loader in order to make the data into batches of size 32. For training phase, I used shuffling for each epoch and test set was used for calculating performance of each model using different methods such as F1-score, recall, precision and confusion matrix.

2. Experiments

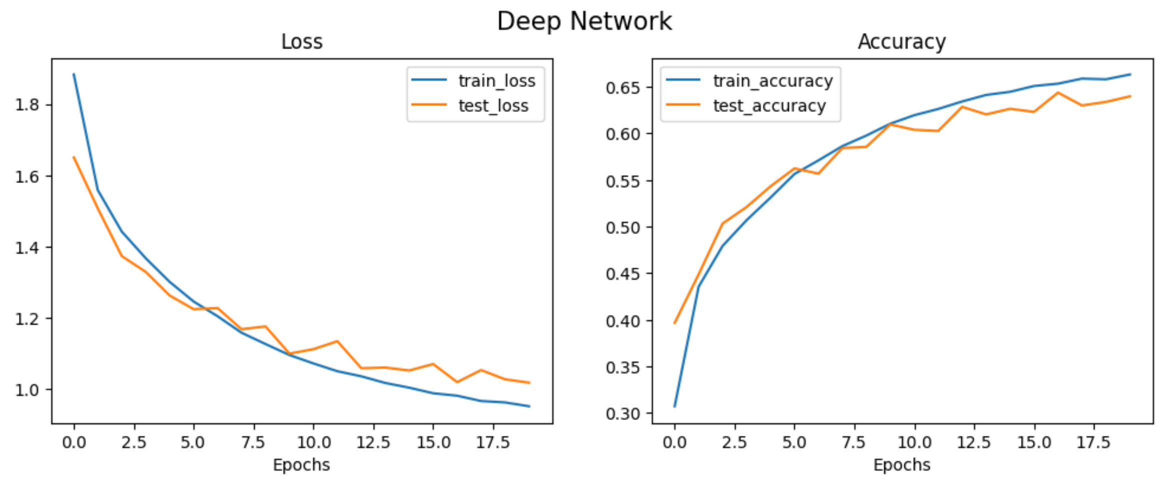
Lots of experiments was performed on the dataset using different architectures of convolution neural networks which are going to discuss in the following section. I used the cross entropy as the loss function and Adam optimizer with learning rate set to 0.001 for the defined models. In each hidden layers in these models there are 10 or 50 neurons (number of kernels) based on the definition of the model and their performance. All of the models were trained for 10 epochs.

1. Neural Network Depth Effect
   For experimenting this section, I made two different neural networks. First one is the shallow one which consist of 2 blocks. First one is the convolution block which has 2 convolution layers with 10 kernels with size of 3 * 3 with ReLU activation function, after these two, there is a max pooling layer with kernel size of 2 in order to reduce the number of parameters and computation and extract most notable features in the data. Second block is the classifier which has a flatten layer in order to convert data from 2d space to on 1d following with a linear layer with 10 neurons in order to perform the classification task. The changes in loss and accuracy of the model on both train and test sets are illustrated in the following plot. As it can be seen from the plot, model has started to overfit to the training data and the difference between losses of training and validation sets are high. Moreover, model can not perform on the validation set therefore it can be understood that

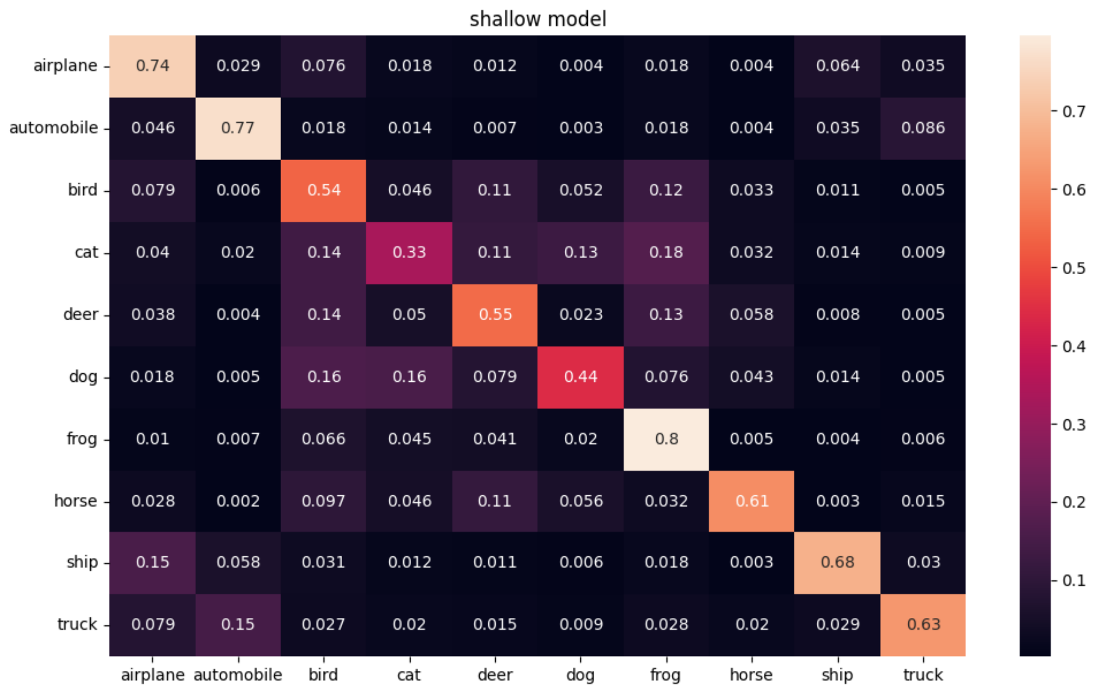
the model can not extract the most important features for doing the classification task and it is not generalized for the mentioned dataset therefore some regularization techniques and changing in architecture is needed.

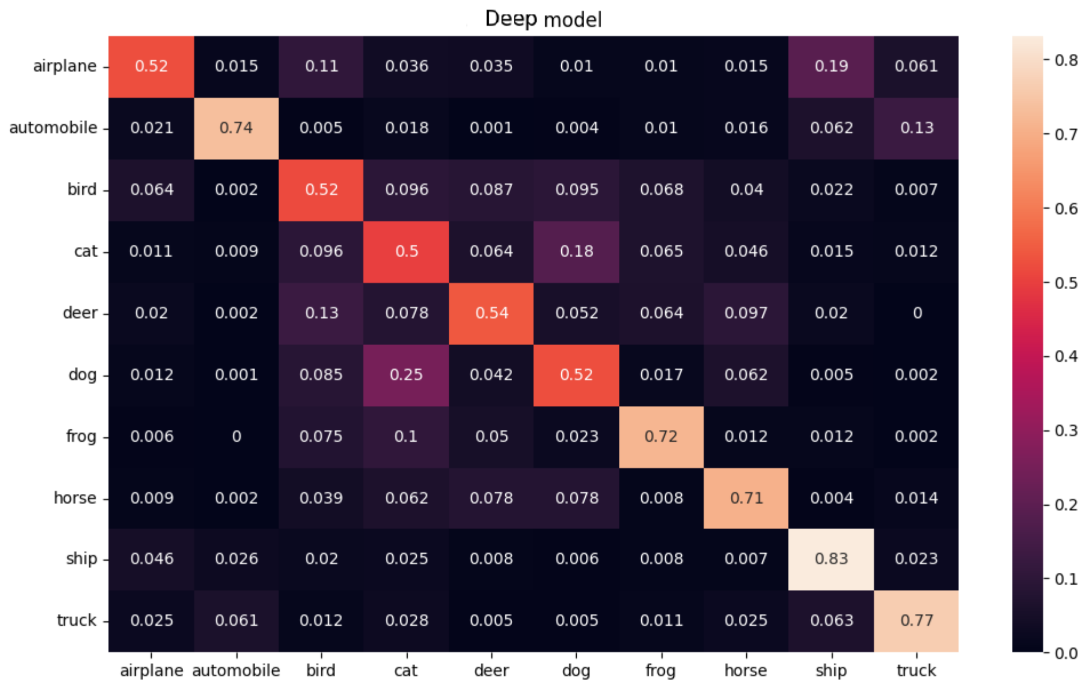


Second model is the deeper one, which consist of 4 blocks. First 3 blocks are convolution blocks which all of them have 2 convolution layers with 10 kernels with size of 3 * 3 with ReLU activation function, after these two, there is a max pooling layer with kernel size of 2 in order to reduce the number of parameters and computation and extract most notable features in the data. Last block is the classifier which has a flatten layer in order to convert data from 2d space to on 1d following with a linear layer with 10 neurons in order to perform the classification task. The changes in loss and accuracy of the model on both train and test sets are illustrated in the following plot. As it can be seen from the plot, since the architecture is more complex, network is able to extract more important features from the inputs, as a result, the fluctuation in accuracy decreased and the slope of decreasing loss increased in compare to shallow model. Also, performance of the model based on different metrics such as f1 score, precision, and recall has increased in compare to the previous model; therefore, we can conclude that with increase in number of hidden layers in this particular task, accuracy and performance have boosted up.



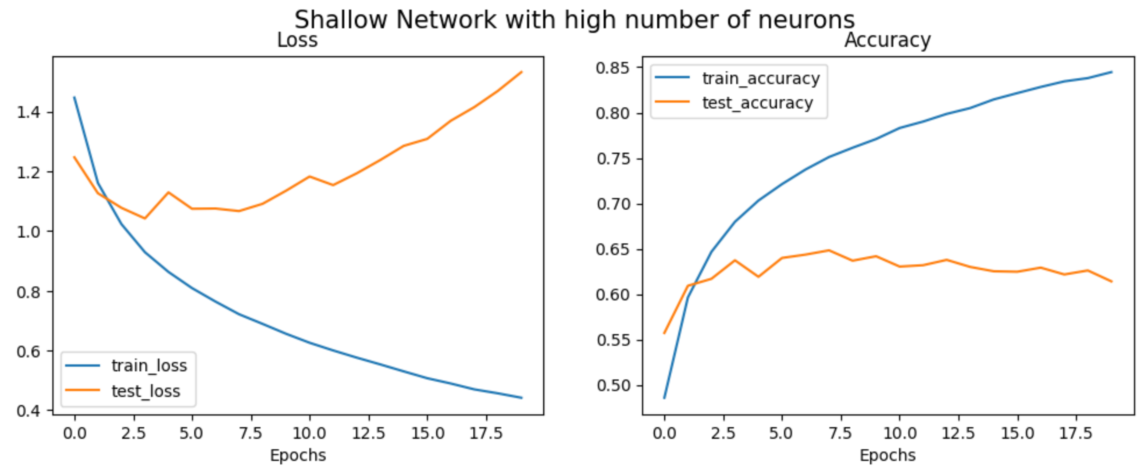
Following heatmaps demonstrates performance of these models on test set. As it can be seen from these plots, the overall values on diagonal indices in deep model are higher than these values in shallow model, so deep model has the better performance than shallow one in this particular task.

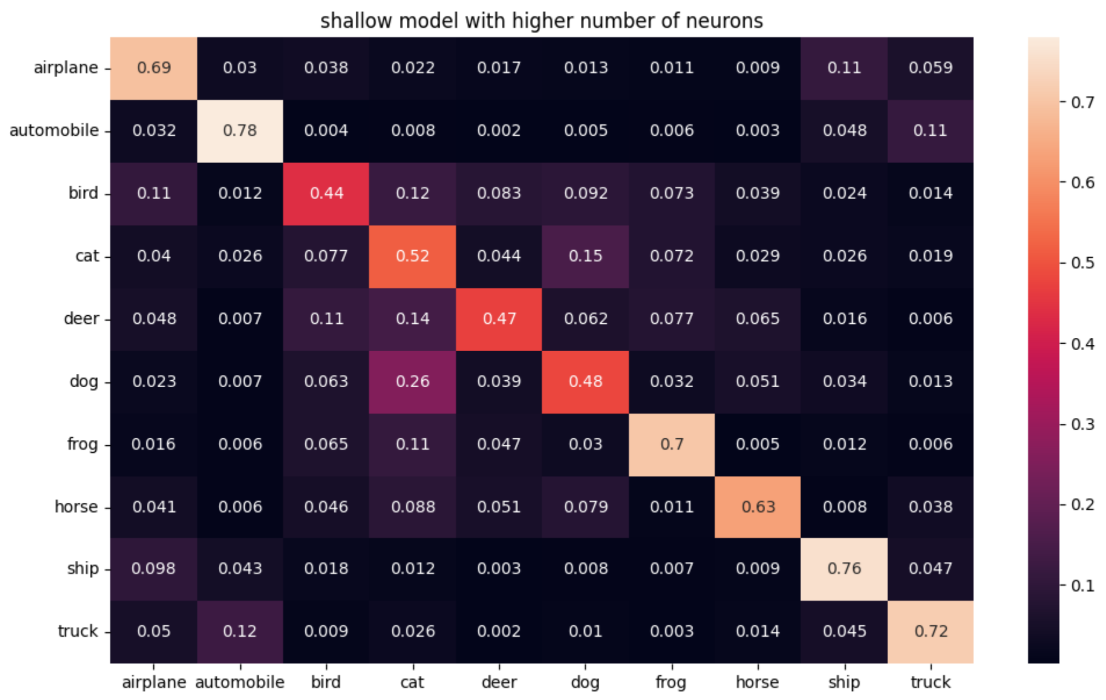




2. <u>Effect of number of neurons in hidden layers</u>

   For this experiment I made a neural network same as the shallow one but in each hidden layer there are 50 neurons (kernels) instead of 10 neurons which the results are shown in the following plots. As it can be seen, model is highly overfitted to the training data because of high number of neurons in each hidden layer and made it sensitive to outliers and prevent the model form being generalized for this task. Also, there is not much difference in performance on validation set in compare to shallow model; therefore, in this particular task, increasing number of neurons in hidden layers when the model is shallow, will not help the model to increase its performance.

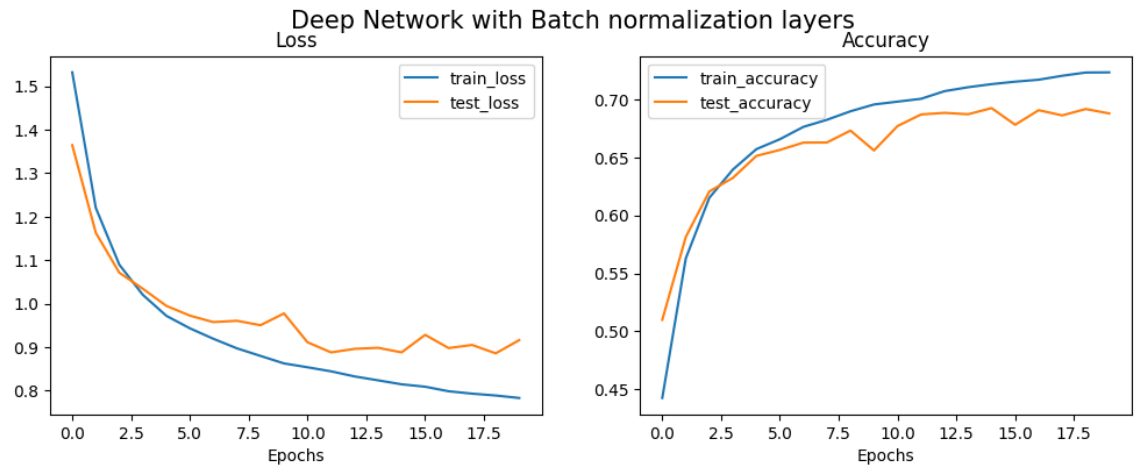


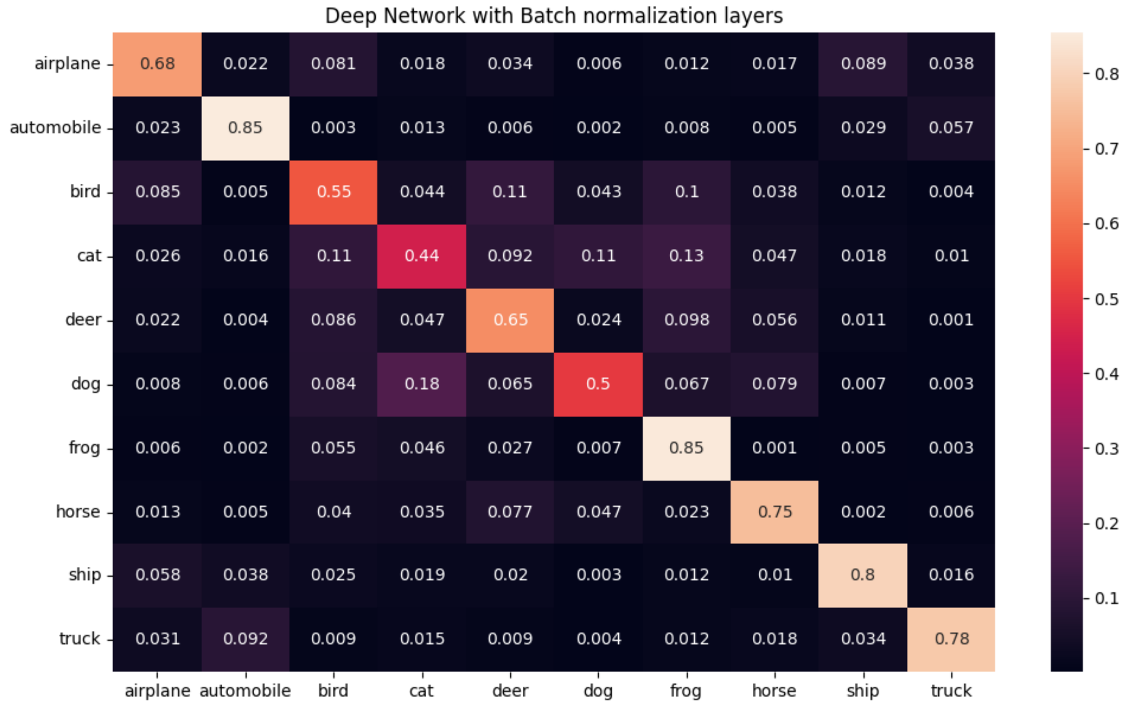


	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	0.69	0.03	0.038	0.022	0.017	0.013	0.011	0.009	0.11	0.059
automobile	0.032	0.78	0.004	0.008	0.002	0.005	0.006	0.003	0.048	0.11
bird	0.11	0.012	0.44	0.12	0.083	0.092	0.073	0.039	0.024	0.014
cat	0.04	0.026	0.077	0.52	0.044	0.15	0.072	0.029	0.026	0.019
deer	0.048	0.007	0.11	0.14	0.47	0.062	0.077	0.065	0.016	0.006
dog	0.023	0.007	0.063	0.26	0.039	0.48	0.032	0.051	0.034	0.013
frog	0.016	0.006	0.065	0.11	0.047	0.03	0.7	0.005	0.012	0.006
horse	0.041	0.006	0.046	0.088	0.051	0.079	0.011	0.63	0.008	0.038
ship	0.098	0.043	0.018	0.012	0.003	0.008	0.007	0.009	0.76	0.047
truck	0.05	0.12	0.009	0.026	0.002	0.01	0.003	0.014	0.045	0.72

3. <u>Effect of using Batch Normalization</u>

For this experiment I made a neural network same as the deep one but add a batch normalization layer after each activation function for convolution layers except for the final layer which was the linear layer that does the classification task. The results are shown in the following plots. As it can be seen, adding batch normalization layer has increased the performance of the model on the validation and test set notably. Also, it adds some regularization to the model; as a result, fluctuations in loss and accuracy have decreased and generalization for this task has increased during epochs.



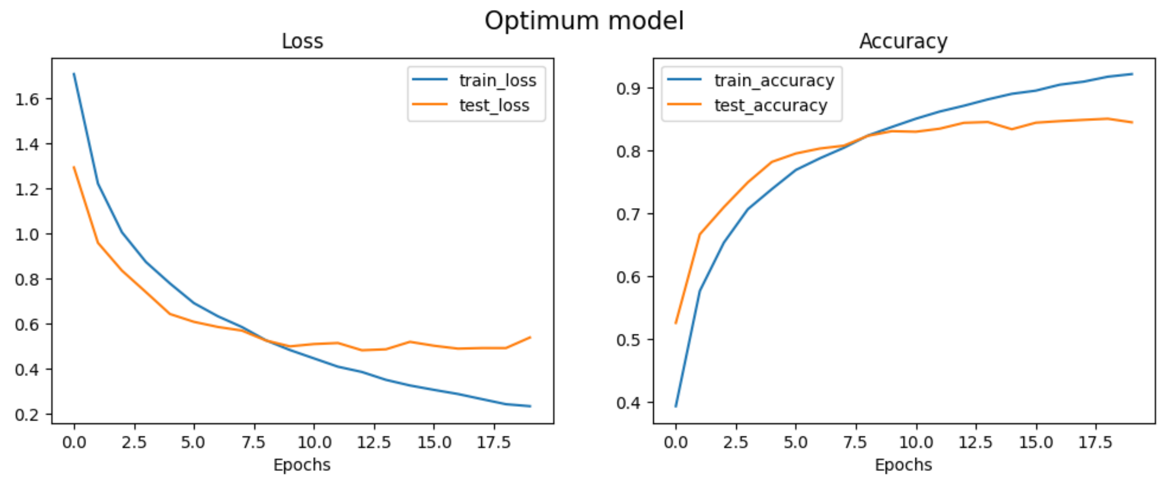


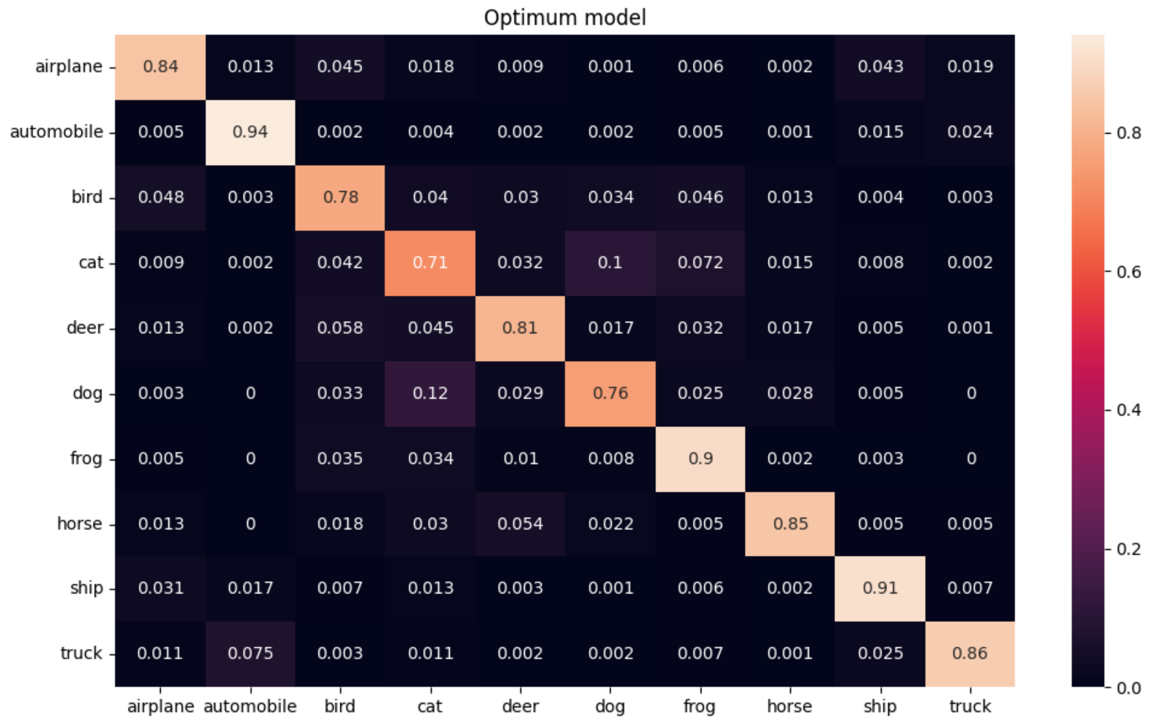
	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	0.68	0.022	0.081	0.018	0.034	0.006	0.012	0.017	0.089	0.038
automobile	0.023	0.85	0.003	0.013	0.006	0.002	0.008	0.005	0.029	0.057
bird	0.085	0.005	0.55	0.044	0.11	0.043	0.1	0.038	0.012	0.004
cat	0.026	0.016	0.11	0.44	0.092	0.11	0.13	0.047	0.018	0.01
deer	0.022	0.004	0.086	0.047	0.65	0.024	0.098	0.056	0.011	0.001
dog	0.008	0.006	0.084	0.18	0.065	0.5	0.067	0.079	0.007	0.003
frog	0.006	0.002	0.055	0.046	0.027	0.007	0.85	0.001	0.005	0.003
horse	0.013	0.005	0.04	0.035	0.077	0.047	0.023	0.75	0.002	0.006
ship	0.058	0.038	0.025	0.019	0.02	0.003	0.012	0.01	0.8	0.016
truck	0.031	0.092	0.009	0.015	0.009	0.004	0.012	0.018	0.034	0.78

4. <u>Optimum model</u>

For this experiment I made a neural network with mix of methods that leads to a good result from the above experiments and my own knowledge. Model consists of 5 blocks. First 4 blocks are convolution blocks which each block has 2 convolution layers, and kernel size of these 2 layers are the same for each block. Kernel size increased in these 4 layers with power of 2 from 32 to 256. There is ReLU activation function following with batch normalization layer after convolution layers. At the end of each block, there are a max pooling layer following with dropout layer in order to prevent overfitting and regularization (batch normalization and dropout) and reduce the number of parameters and computation and extract most notable features in the data (pooling layer). Last layer which consists of 3 linear layers with respectively 512, 1024, and 10 neurons is used for taking the calculated representation from previous blocks and doing the classification. There are also batch normalization and dropout layers in this block in order to prevent overfitting and increase generalization.

Results are shown in the following plots. As it can be seen, model is performing good on the data, loss values have decreased sharply and there is not much fluctuation in the values which means the model was regularized pretty well and training phase has stopped before the model tempt to overfitted to the training set. This model has the best performance in the term of accuracy, f1 score, precision, recall in compare to the prior models. This conclusion is also can be understood from the confusion matrix; As it can be seen, most of the values in diagonal indices are high which means that the model can classify the input images from each other to the predefined labels pretty well.

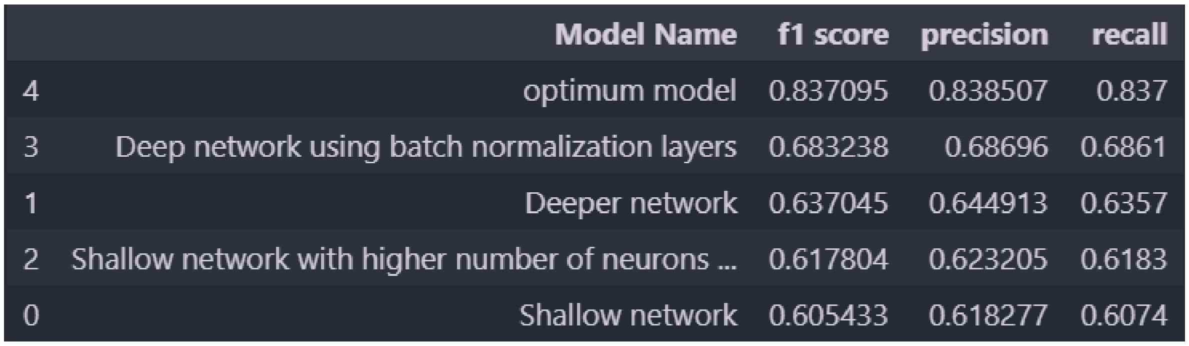




	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	0.84	0.013	0.045	0.018	0.009	0.001	0.006	0.002	0.043	0.019
automobile	0.005	0.94	0.002	0.004	0.002	0.002	0.005	0.001	0.015	0.024
bird	0.048	0.003	0.78	0.04	0.03	0.034	0.046	0.013	0.004	0.003
cat	0.009	0.002	0.042	0.71	0.032	0.1	0.072	0.015	0.008	0.002
deer	0.013	0.002	0.058	0.045	0.81	0.017	0.032	0.017	0.005	0.001
dog	0.003	0	0.033	0.12	0.029	0.76	0.025	0.028	0.005	0
frog	0.005	0	0.035	0.034	0.01	0.008	0.9	0.002	0.003	0
horse	0.013	0	0.018	0.03	0.054	0.022	0.005	0.85	0.005	0.005
ship	0.031	0.017	0.007	0.013	0.003	0.001	0.006	0.002	0.91	0.007
truck	0.011	0.075	0.003	0.011	0.002	0.002	0.007	0.001	0.025	0.86

3. Results

Results of above experiments are aggregated in the following table and sorted based on the f1 score of the model on test data. I used different metrics such as f1-score, precision, recall for comparing the models with each other. Since the task was multi-class classification, and number of different records in each class was equal, I used macro averaging for calculating these scores.



	Model Name	f1 score	precision	recall
4	optimum model	0.837095	0.838507	0.837
3	Deep network using batch normalization layers	0.683238	0.68696	0.6861
1	Deeper network	0.637045	0.644913	0.6357
2	Shallow network with higher number of neurons ...	0.617804	0.623205	0.6183
0	Shallow network	0.605433	0.618277	0.6074

As it can be concluded from the above table, the optimum model has the best performance on the test data compare to other models by far amount of difference. Moreover, increasing number of hidden layers helps the model to increase its performance in compare to increasing number of neurons of the hidden layers. Also, adding batch normalization layers, increase the regularization and generalization of the model and has increased model performance on the test data.