Report for assignment 2

Nima Taheri, Mahdieh Sajedi Pour

11/17/2022

1) Overview and Problem Statement

In exercise 3, we worked with CIFAR-10 which consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. Our goal is to implement a convolutional neural network to solve this imagery classification problem and try to find the optimum model.

2) Data Preparation

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, we 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 test data were used as validation set for training phase.
4. Train and validation datasets were passed into their own data loader.

3) Methodology

A confusion matrix is used to show the differences in each model performance.

A) 1st model

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 29, 29, 32)	1568
max_pooling2d (MaxPooling2D)	(None, 14, 14, 32)	0
conv2d_1 (Conv2D)	(None, 11, 11, 32)	16416
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 32)	0
flatten (Flatten)	(None, 800)	0
dense (Dense)	(None, 256)	205056
dense_1 (Dense)	(None, 10)	2570

Total params: 225,610
Trainable params: 225,610
Non-trainable params: 0

B) 2nd model

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 29, 29, 32)	1568
max_pooling2d_2 (MaxPooling 2D)	(None, 14, 14, 32)	0
dropout (Dropout)	(None, 14, 14, 32)	0
conv2d_3 (Conv2D)	(None, 11, 11, 32)	16416
max_pooling2d_3 (MaxPooling 2D)	(None, 5, 5, 32)	0
dropout_1 (Dropout)	(None, 5, 5, 32)	0
flatten_1 (Flatten)	(None, 800)	0
dense_2 (Dense)	(None, 256)	205056
dense_3 (Dense)	(None, 10)	2570

Total params: 225,610
Trainable params: 225,610
Non-trainable params: 0

C) 3rd model

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 30, 30, 64)	1792
conv2d_5 (Conv2D)	(None, 28, 28, 64)	36928
max_pooling2d_4 (MaxPooling 2D)	(None, 14, 14, 64)	0
dropout_2 (Dropout)	(None, 14, 14, 64)	0
conv2d_6 (Conv2D)	(None, 12, 12, 128)	73856
conv2d_7 (Conv2D)	(None, 10, 10, 128)	147584
max_pooling2d_5 (MaxPooling 2D)	(None, 5, 5, 128)	0
dropout_3 (Dropout)	(None, 5, 5, 128)	0
flatten_2 (Flatten)	(None, 3200)	0
dense_4 (Dense)	(None, 1024)	3277824
dense_5 (Dense)	(None, 1024)	1049600
dense_6 (Dense)	(None, 10)	10250

Total params: 4,597,834
Trainable params: 4,597,834
Non-trainable params: 0

D) 4th model

Layer (type)	Output Shape	Param #
conv2d_8 (Conv2D)	(None, 32, 32, 32)	896
conv2d_9 (Conv2D)	(None, 32, 32, 32)	9248
max_pooling2d_6 (MaxPooling 2D)	(None, 16, 16, 32)	0
dropout_4 (Dropout)	(None, 16, 16, 32)	0
conv2d_10 (Conv2D)	(None, 16, 16, 64)	18496
conv2d_11 (Conv2D)	(None, 16, 16, 64)	36928
max_pooling2d_7 (MaxPooling 2D)	(None, 8, 8, 64)	0
dropout_5 (Dropout)	(None, 8, 8, 64)	0
conv2d_12 (Conv2D)	(None, 8, 8, 128)	73856
conv2d_13 (Conv2D)	(None, 8, 8, 128)	147584
max_pooling2d_8 (MaxPooling 2D)	(None, 4, 4, 128)	0
dropout_6 (Dropout)	(None, 4, 4, 128)	0
flatten_3 (Flatten)	(None, 2048)	0
dense_7 (Dense)	(None, 128)	262272
dropout_7 (Dropout)	(None, 128)	0
dense_8 (Dense)	(None, 10)	1290

Total params: 550,570
Trainable params: 550,570
Non-trainable params: 0

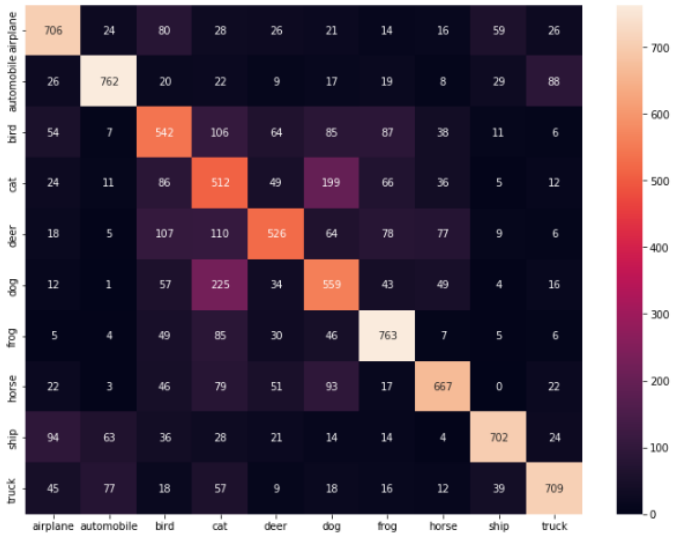
E) 5th model

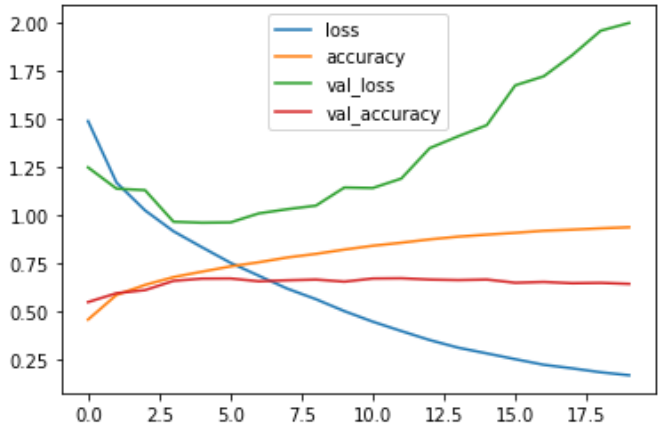
Layer (type)	Output Shape	Param #
conv2d_14 (Conv2D)	(None, 32, 32, 32)	896
batch_normalization (BatchNormalization)	(None, 32, 32, 32)	128
conv2d_15 (Conv2D)	(None, 32, 32, 32)	9248
batch_normalization_1 (BatchNormalization)	(None, 32, 32, 32)	128
max_pooling2d_9 (MaxPooling2D)	(None, 16, 16, 32)	0
dropout_8 (Dropout)	(None, 16, 16, 32)	0
conv2d_16 (Conv2D)	(None, 16, 16, 64)	18496
batch_normalization_2 (BatchNormalization)	(None, 16, 16, 64)	256
conv2d_17 (Conv2D)	(None, 16, 16, 64)	36928
batch_normalization_3 (BatchNormalization)	(None, 16, 16, 64)	256
max_pooling2d_10 (MaxPooling2D)	(None, 8, 8, 64)	0
dropout_9 (Dropout)	(None, 8, 8, 64)	0
conv2d_18 (Conv2D)	(None, 8, 8, 128)	73856
batch_normalization_4 (BatchNormalization)	(None, 8, 8, 128)	512
conv2d_19 (Conv2D)	(None, 8, 8, 128)	147584
batch_normalization_5 (BatchNormalization)	(None, 8, 8, 128)	512
max_pooling2d_11 (MaxPooling2D)	(None, 4, 4, 128)	0
dropout_10 (Dropout)	(None, 4, 4, 128)	0
flatten_4 (Flatten)	(None, 2048)	0
dense_9 (Dense)	(None, 128)	262272
batch_normalization_6 (BatchNormalization)	(None, 128)	512
dropout_11 (Dropout)	(None, 128)	0
dense_10 (Dense)	(None, 10)	1290

Total params: 552,874
Trainable params: 551,722
Non-trainable params: 1,152

4) Result

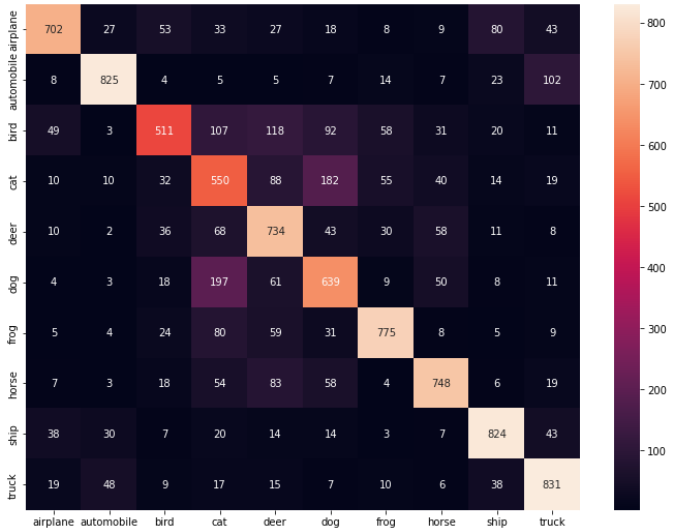
A) 1st model

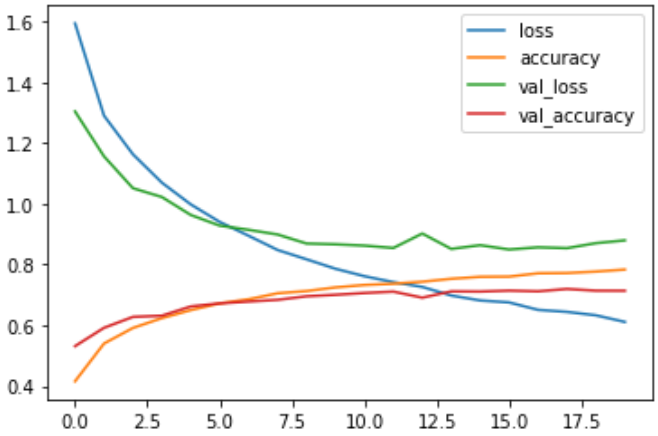




As we can see this was a really simple model. By observing the plot above we can see that we were able to get 88% training accuracy and 65% test accuracy. In addition, we can see that validation loss is increasing too much and that means our model is performing good on the training data set but fails to generalize on the unseen data (overfitting). Confusion matrix showed us model is performing awful on predicting birds, cats, deer and dogs.

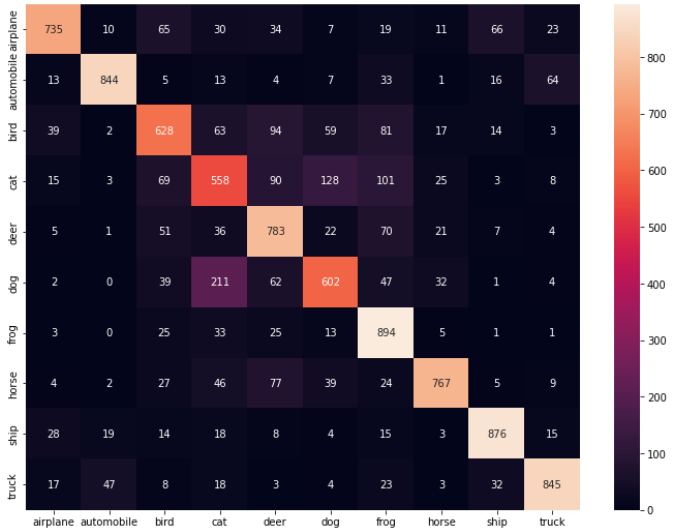
B) 2nd model

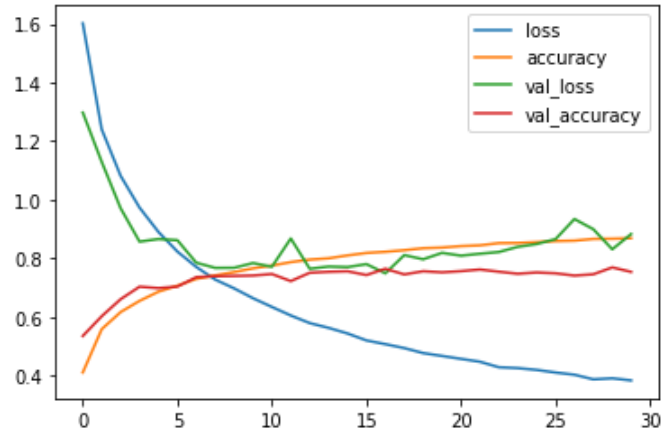




Now we can see some improvements in validation accuracy. Validation loss is decreased significantly by using dropout. From confusion matrix we can understand that this model is doing better in predicting deer and dogs.

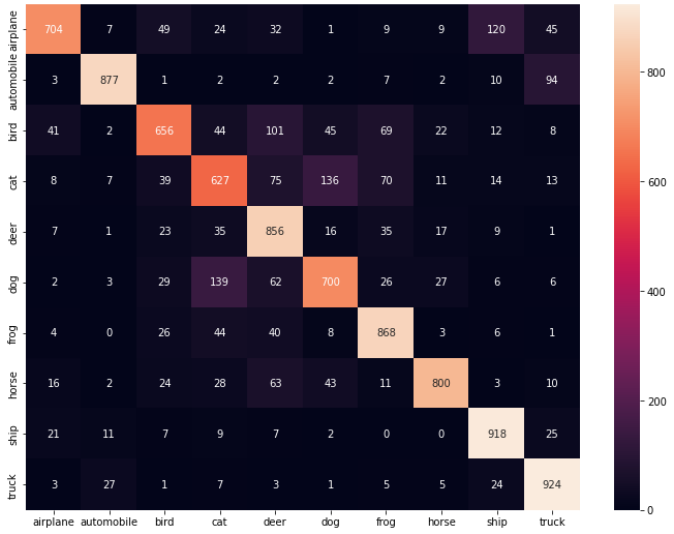
C) 3rd model

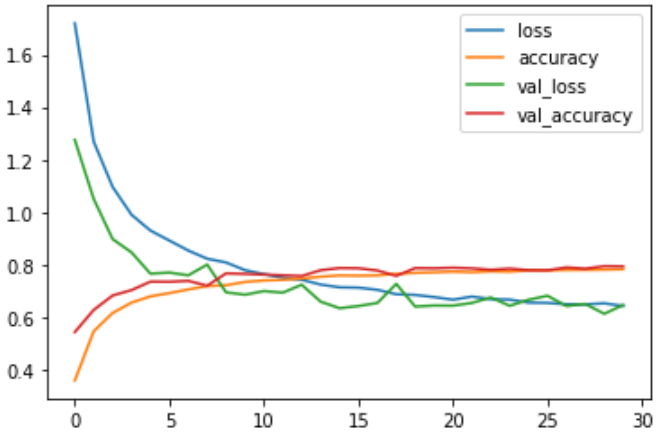




Accuracy is a bit better than previous model but, validation loss has been increased and that is not what we wanted. As we can see in confusion matrix, we are still having problem classifying birds and cats.

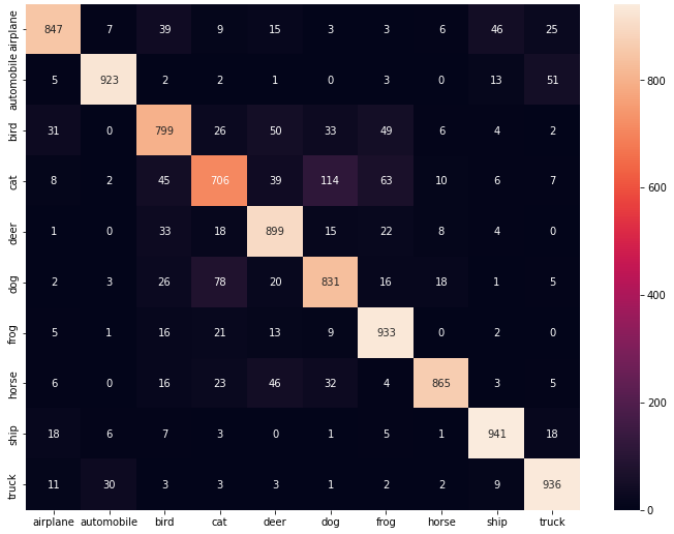
D) 4th model

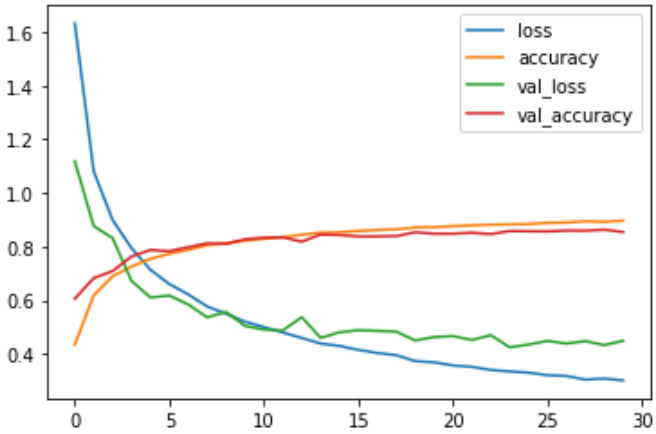




Validation loss is not increasing. Now the accuracy boosted to 79% from 76% which is nice. We can see we used different initializer, which I learned from machine learning mastery. In confusion matrix we can see that model is getting better to understand birds and cats. There is a little problem and that is the model predicting cats as dogs and vice versa for about 200 out of 2000.

E) 5th model





As you can see, we used batch normalization here to observe what will happen to the model. It works just as same as we normalize the input data where we divided the x_train/255. What we are doing is we want to arrange all the features in same scale so that model converges easily. Whenever we pass the CNN through a batch normalization layer, we are normalizing the weights so that our model will be stable and we can train model for a longer period. We compute mean and variance for each mini batch not the whole data in batch normalization.
We can see a significant increase in validation accuracy 86.8% and significant increase in validation loss (decreased around 0.2). If you look at left to right diameter in confusion Matrix, you see it is brighter than before which means the model is predicting better and the worst prediction is for cats and the best prediction is for ships.

Overall, using batch normalization, dropout, and appropriate number of layers and filters we could gain 87% accuracy. Batch normalization was the best thing to do. With using it the optimum model was found.