CIFAR10 Classification Project

GR5242 Final Project

Team Member:

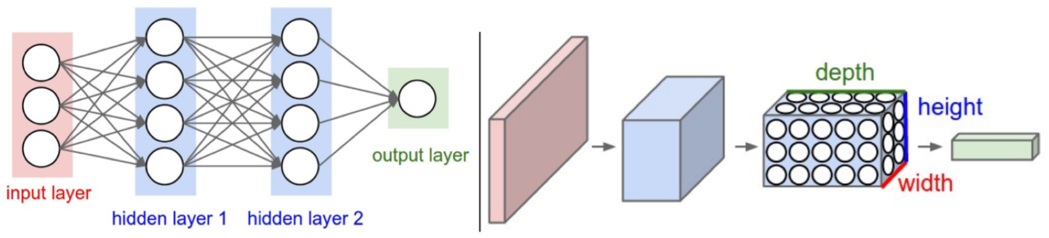
Hao Li (hl3109)
Xiaoyi Li (xl2694)
Yang Shi (ys3047)
Yunfan Li (yl3838)

1. Project introduction

The CIFAR10 is an important image classification dataset. For this project we can apply kinds of machine learning algorithms and deep learning algorithms in order to get a better performance. There are some goals for this project: First, we want to build up the classification model. We apply different Convolutional Neural Networks(CNN) methods and set up the baseline model. Then we try to tune different kinds of parameters and get an improvement model which has a higher accuracy.

2. Model Introduction - Convolutional Neural Networks

Convolutional Neural Networks(CNN) are similar to ordinary Neural Networks. As for image classification problem, the CNN model takes the raw image pixels as inputs and expresses class scores at the end. What makes CNN model different from the ordinary Neural Networks is that CNN architectures have the explicit assumption that those inputs are images that can significantly reduce the amount of parameters in the network and make the whole process more efficient.



Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels).

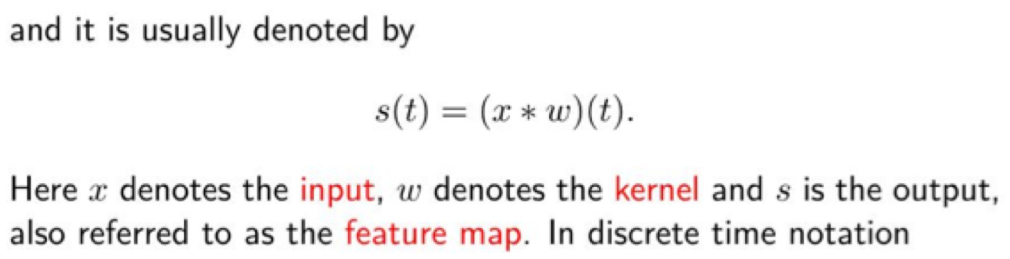
(Source: CS231n Convolutional Neural Networks for Visual Recognition)

As shown in the above graph, Regular Neural Nets have much more parameters than that in a CNN. For example, a CIFAR-10 image has the size of 32x32x3, so that a single fully-connected neuron in a first hidden layer of a regular Neural Network would have 32*32*3 = 3072 weights. If we want several neurons on the layer, then this number would significantly increase. While the layers of CNN have neurons arranged in 3D: width, height, depth. Unlike the regular Neural Network, the neurons in a layer of CNN will only be connected to a small region of the previous layer. CNN model achieves this property through the convolution operations on neural networks.

Those equations below show the convolution operation.

The convolution in a simplified form is defined by

$$s(t) = \int_{\mathbb{D}} x(a)w(t-a)da$$


and it is usually denoted by

$$s(t) = (x * w)(t).$$

Here x denotes the input, w denotes the kernel and s is the output, also referred to as the feature map. In discrete time notation

(source: ECBME 4040 slide)

Those parameters of convolutional layers consist of a set of learnable filters. Every filter is spatially small (for example 5x5x3), but extends through the full depth of the input volume. During the forward pass, it convolves each filter across the width and height of the input volume and compute dot products between the entries of the filter and the input at any position. For each filter, a 2D activation map that gives the responses of that filter at every spatial position will be produced. These activation maps stack together become the output of this conv layer. [1]

Due to convolution operations on neural networks, CNNs have three properties: sparse connectivity, parameter sharing and equivariant representation, which significantly improve machine learning. [2]

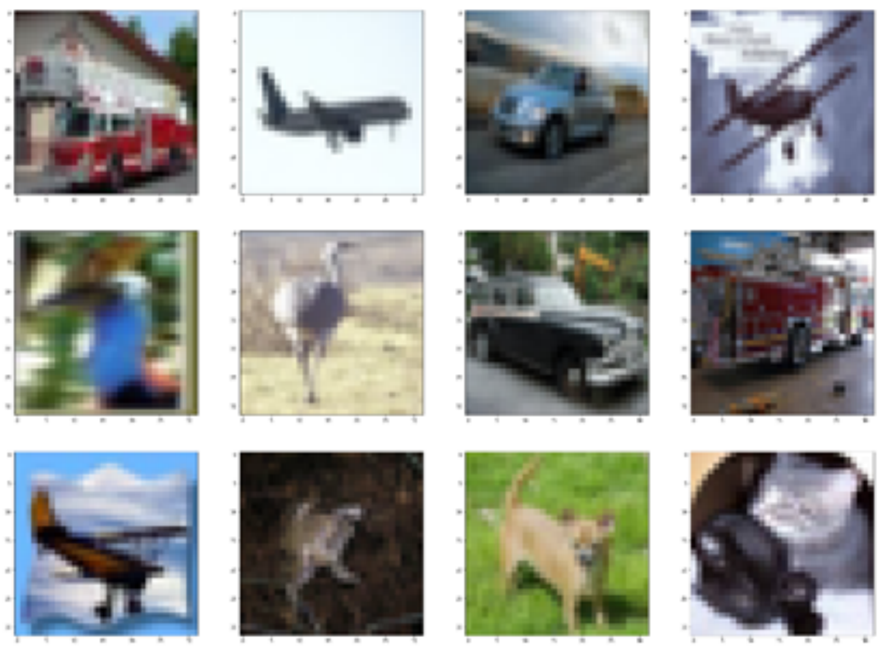
3 Data Introduction

3.1 Data Description

We use the CIFAR-10 dataset that consists of 60000 32*32 images in our project. Such dataset has 10 classes—airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. We use 40000 images as training set,10000 images as validation set and 10000 images as test images.

For each picture, we reshape it into a (32,32,3) numpy array. The first two dimensions correspond to a (x, y) coordinate of a pixel. And the last dimension corresponds to the channels of picture-Red, Green and Blue. We set the datatype of those images to float32.

Here are some random images in the training set:



3.2 Data preprocessing

Step 1 Data augmentation:
We expand the size of training set by scaling, rotating, horizontal flipping, vertical flipping, width shifting and height shifting. The neural network is disclosed to a huge variety of variations. After processing these steps, it makes our model perform better both on training set and test set.

Step 2 Normalization:
Then we do the normalization on all training set and test set. Normalization transforms the range of the values of pixel. It is a significant step in data preprocessing because it makes each pixel has a similar distribution and range. We can calculate the mean and standard deviation for each pixel. Then we minus the mean first and divide it by the standard deviation next for each image.

Step 3 One hot encoding:
Due to the fact that variables of y are categorical variables, we transform the label of y into one hot encoding on both training set and test set

Here are some examples:

Step 4 Whitening
Time-domain processing leads to the coming out of “whitening”. Due to the uniform power spectrum, when we try to eliminate the correlation from the signal, it becomes more similar to “white” noise. Whitening is very efficient in data preprocessing part. It will improve the accuracy of model significantly.
There are two things we need to do with whitening. First, it will make features less correlate with each other. Second, all features will have the same variance. After we apply ZCA(whitening) to images, we find whitened data closer to original data comparing with PCA.

4. Algorithm and Model Building

4.1 Algorithm

Here we adapt a traditional CNN model, which totally has 6 convolution layers, pooling layers and flatten layer. We select eLu as activation function after each convolution layer since it is more robust than ReLu and can achieve almost all goals of ReLu. For pooling layers, max pooling is the best and most general choice because it keeps majority of original features, such as the edge of specific shapes, and significantly reduces variance and calculation complexity.

After the baseline trail, we try dropout, batch normalization and weight decay to improve the original model. For dropout, the rate usually is 0.2 in the input layer and up to 0.5 in hidden layers, which not only reduces the time needed, also increases the accuracy. We try several parameter combinations and finally find 0.2, 0.3, 0.4 can provide satisfied results. Batch normalization improves the performance and stability of the model from several ways, which enables higher range of learning rates and regularizes the whole model. Batch normalization helps address the problem that too-high learning rate may result in the gradients that explode or vanish, as well as get stuck in poor local minima in traditional deep networks. When training with batch normalization, a training example is seen in conjunction with other examples in the mini-batch, and the training network no longer producing deterministic values for a given training example.[3] Weight is also known as l2 regularization, which adds a tuning parameter to a model to induce smoothness in order to prevent overfitting. The regularization term is defined as the sum of the squares of all the feature weights and weights close to zero have little effect on model complexity, while outlier weights can have a huge impact.[4]

4.2 Model Description

4.2.1 Baseline model

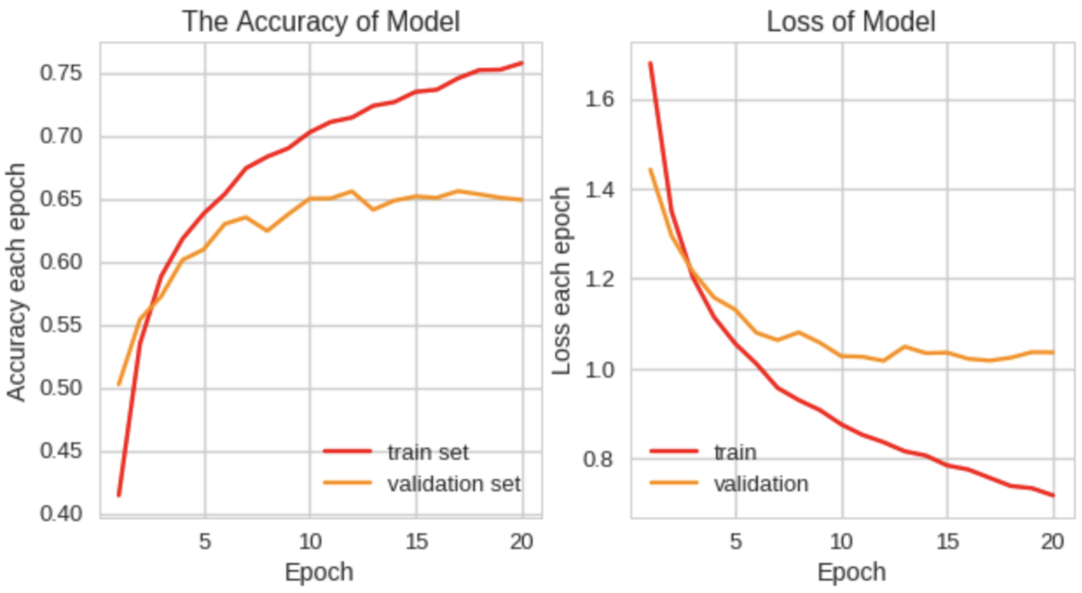
First, we build a basic CNN model with one layer to fit the data. We try to use Keras to build the model and run it to find the accuracy of such a model, and treat it as a baseline model.

Before building the model, we try to normalize the data, which can make each pixel has a similar distribution and range, so the result will become better.

The baseline model has following structure.

- Conv2D with ReLu: Kernel size: 3 x 3 filter numbers: 32
- MaxPooling2D: pool size 2 x 2
- Fully Connected

The output will be 10 probabilities which show the group the example should be in.
With batch size equals to 500 and epoch 20, the accuracy of the baseline model is about 64.65%, which is not a satisfy result for us. So, we consider improving the model.



Baseline Model

4.2.2 Model Improvement 1

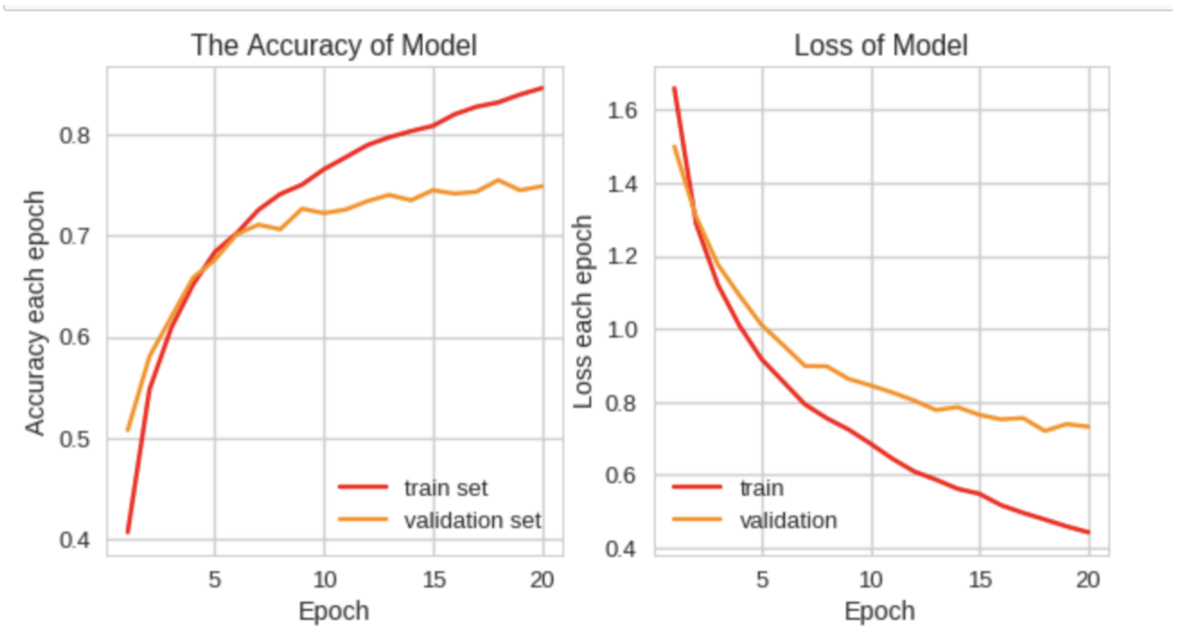
To make the model much better, we try to add some more layers and do some transformation of data.

We realize that we can also add dropout layers to the model which can not only decrease the time the model will spend, but also raise the accuracy.

The second model has following structure.

- Conv2D with ReLu: Kernel size: 3 x 3 filter numbers: 32
- Normalization
- Conv2D with ReLu: Kernel size: 3 x 3 filter numbers: 32
- Normalization
- Dropout: 0.3
- MaxPooling2D: pool size 2 x 2
- Conv2D with ReLu: Kernel size: 3 x 3 filter numbers: 64
- Normalization
- Conv2D with ReLu: Kernel size: 3 x 3 filter numbers: 64
- Normalization
- Dropout: 0.3
- MaxPooling2D: pool size 2 x 2
- Fully Connected
- Dense with Softmax: 10 units

After adding more layers, the result turns to be better which is about 74.27% with batch size equals to 500 and epoch equals to 20.



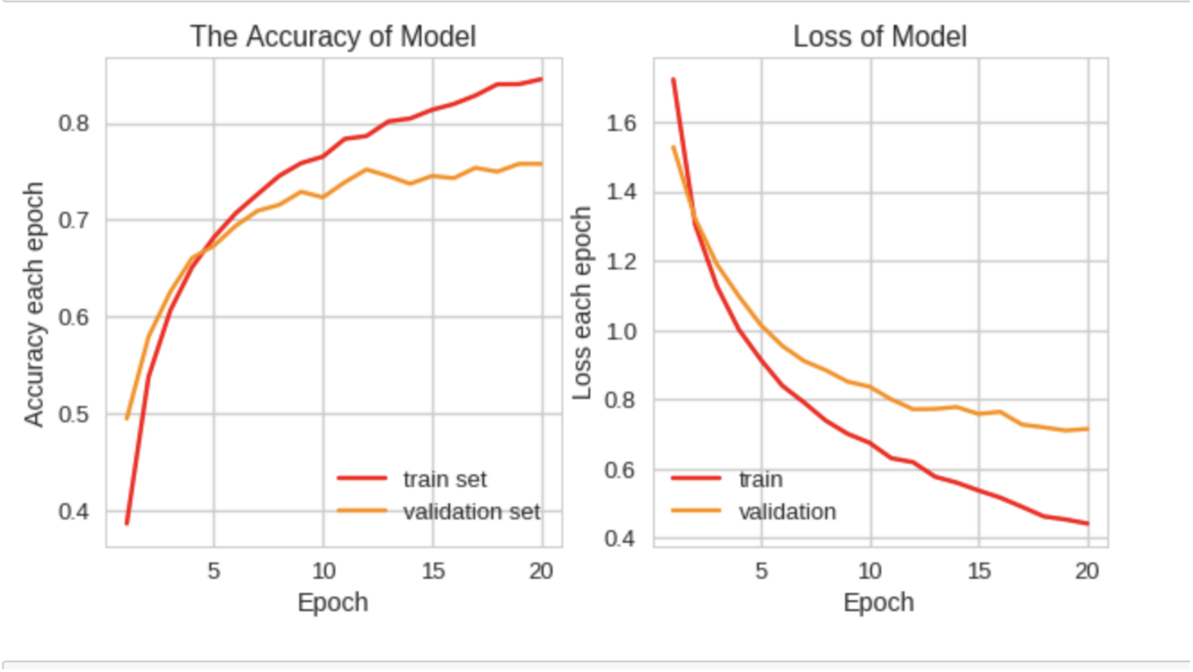
Four Layers Model

4.2.3 Model Improvement 2

We wonder that is there any other methods we can use to keep raising the accuracy? Then we search on the internet and find that doing image augmentation is a widely used method to preprocess the data which can improve the result. So, we try to do image augmentation before we start the CNN model.

Before we run the CNN part, we do some image augmentations like whitening, rotation, flipping and shifting height and width. After doing such augmentations, we can get more data to train and the result can be much more reliable.

The structure of the model is the same as above, which is a four-layer CNN model. Finally we get the result that the accuracy after doing such transformation to original data will be about 75.37% with batch size equals to 500 and epoch equals to 20.



Four Layers + Augmentation

As the size of training set raises, the accuracy of the model will go up. Such changing maybe happens for the reason that we have a larger training set, we can train more, the model can be better due to more history data.

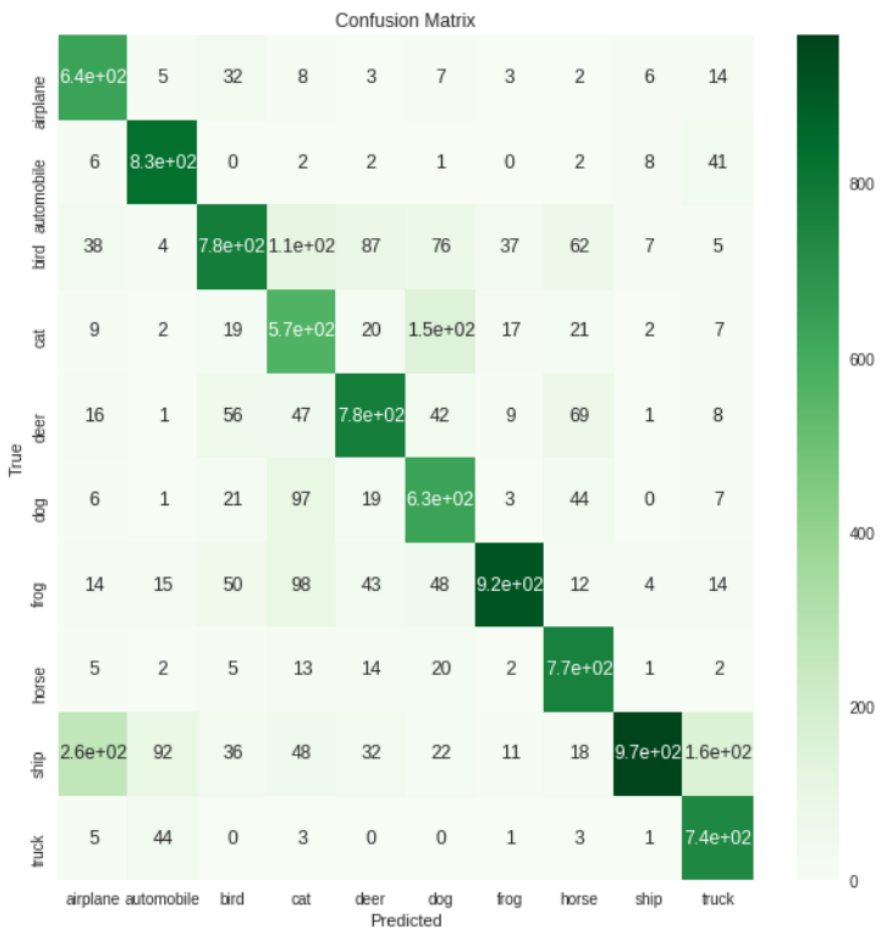
Image Augmentation is a preprocessing method to original data. We still want to find whether there is any other improvement we can do to our model besides just adding more layers, for the reason that only adding more layers will not only spend much more time to get a result, but also cause the decay of the accuracy of the model.

4.2.4 Model Improvement 3

Besides adding dropout layers, we also add more layers to make the model better. In addition, we change the activation function from 'relu' to 'elu' to find if there is any improvements to the model.
Then we get such a model.

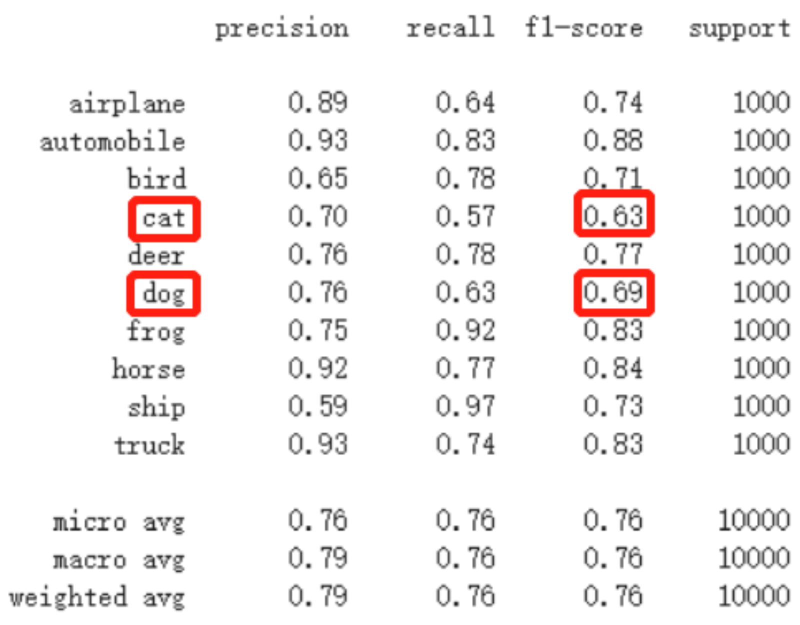
- Conv2D with eLu: Kernel size: 3 x 3 filter numbers: 32
- Normalization
- Conv2D with eLu: Kernel size: 3 x 3 filter numbers: 32
- Normalization
- Dropout: 0.2
- MaxPooling2D: pool size 2 x 2
- Conv2D with eLu: Kernel size: 3 x 3 filter numbers: 64
- Normalization
- Conv2D with eLu: Kernel size: 3 x 3 filter numbers: 64
- Normalization
- Dropout: 0.3
- MaxPooling2D: pool size 2 x 2
- Conv2D with eLu: Kernel size: 3 x 3 filter numbers: 128
- Normalization
- Conv2D with eLu: Kernel size: 3 x 3 filter numbers: 128
- Normalization
- Dropout: 0.4
- MaxPooling2D: pool size 2 x 2
- Fully Connected
- Dense with Softmax: 10 units

Then we get a CNN model with six layers including dropout layers and using max pooling method. Then we get the result that the accuracy of such CNN model is about 76.34% with batch size equals to 300 and epoch equals to 20.



	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	6.4e+02	5	32	8	3	7	3	2	6	14
automobile	6	8.3e+02	0	2	2	1	0	2	8	41
bird	38	4	7.8e+02	1.1e+02	87	76	37	62	7	5
cat	9	2	19	5.7e+02	20	1.5e+02	17	21	2	7
deer	16	1	56	47	7.8e+02	42	9	69	1	8
dog	6	1	21	97	19	6.3e+02	3	44	0	7
frog	14	15	50	98	43	48	9.2e+02	12	4	14
horse	5	2	5	13	14	20	2	7.7e+02	1	2
ship	2.6e+02	92	36	48	32	22	11	18	9.7e+02	1.6e+02
truck	5	44	0	3	0	0	1	3	1	7.4e+02

Heatmap 1

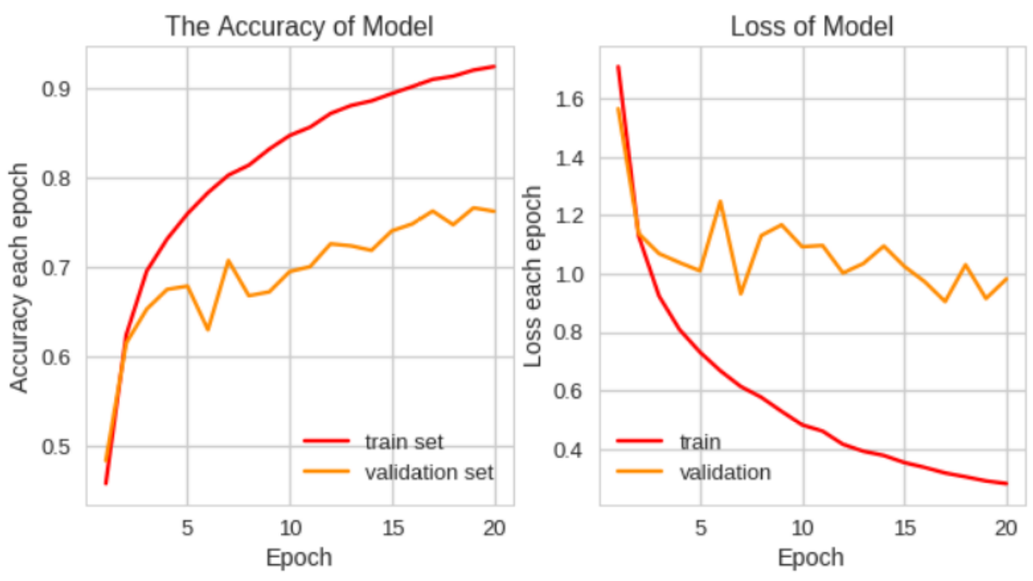


```
              precision    recall  f1-score   support

    airplane       0.89      0.64      0.74      1000
  automobile       0.93      0.83      0.88      1000
        bird       0.65      0.78      0.71      1000
         cat       0.70      0.57      0.63      1000
        deer       0.76      0.78      0.77      1000
         dog       0.76      0.63      0.69      1000
        frog       0.75      0.92      0.83      1000
       horse       0.92      0.77      0.84      1000
        ship       0.59      0.97      0.73      1000
       truck       0.93      0.74      0.83      1000

   micro avg       0.76      0.76      0.76     10000
   macro avg       0.79      0.76      0.76     10000
weighted avg       0.79      0.76      0.76     10000
```

Confusion Matrix 1



6 Layers + Augmentation + eLu

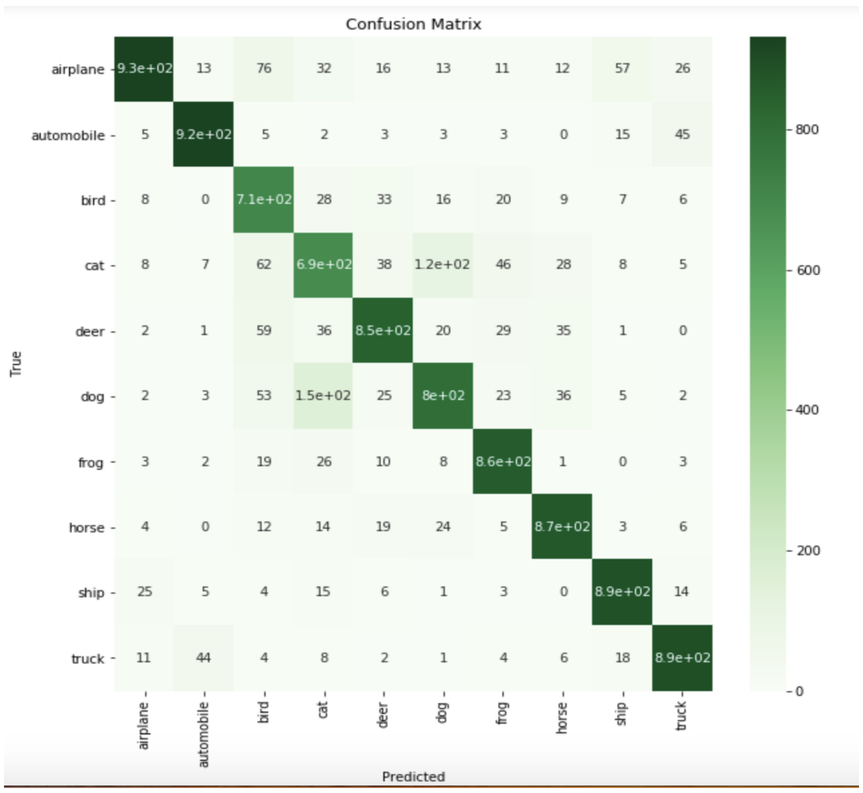
From those heatmaps, we find the fact that the classification of the pictures of dogs and cats are pretty confuse. This phenomenon may be caused by the fact that max pooling keeps the character of edges, which can be an error to those results of classification. So, we try to use average pooling to solve such problem.

4.2.5 Final Model

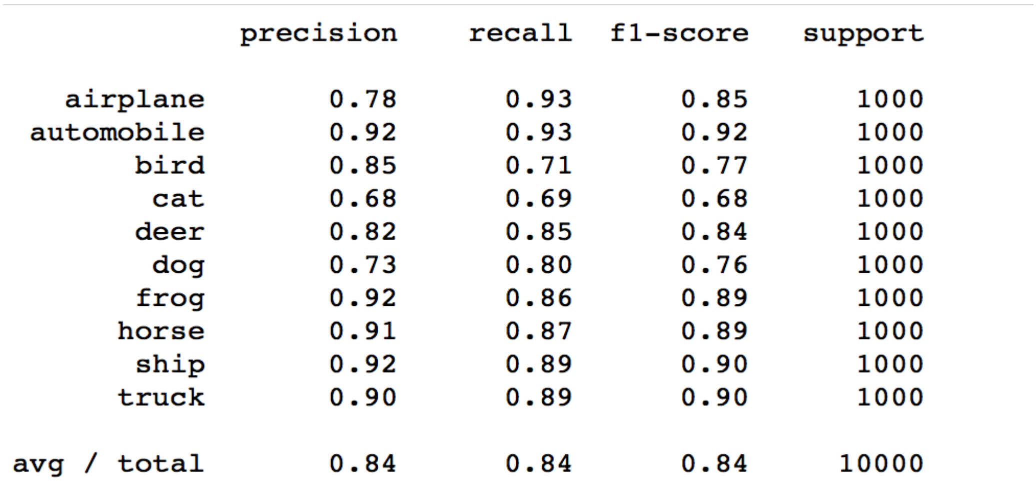
Then we get the structure following.

- Conv2D with eLu: Kernel size: 3 x 3 filter numbers: 32
- Normalization
- Conv2D with eLu: Kernel size: 3 x 3 filter numbers: 32
- Normalization
- Dropout: 0.2
- AvgPooling2D: pool size 2 x 2
- Conv2D with eLu: Kernel size: 3 x 3 filter numbers: 64
- Normalization
- Conv2D with eLu: Kernel size: 3 x 3 filter numbers: 64
- Normalization
- Dropout: 0.3
- AvgPooling2D: pool size 2 x 2
- Conv2D with eLu: Kernel size: 3 x 3 filter numbers: 128
- Normalization
- Conv2D with eLu: Kernel size: 3 x 3 filter numbers: 128
- Normalization
- Dropout: 0.4
- AvgPooling2D: pool size 2 x 2
- Fully Connected
- Dense with Softmax: 10 units

The accuracy of such CNN model is about 84.03% with batch size equals to 128 and epoch equals to 50. The result is pretty better than that of baseline, so these methods do enhance the model to classification.



Heatmap 2

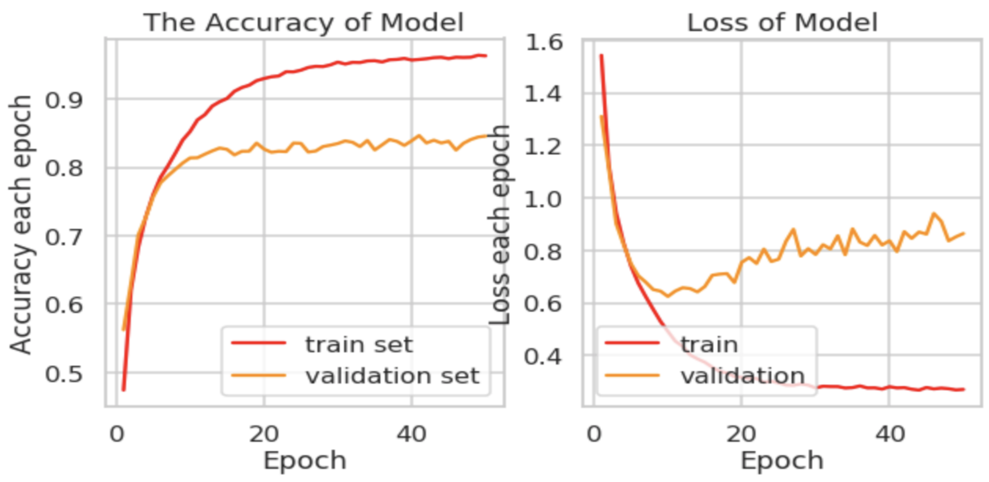


```
             precision    recall  f1-score   support

   airplane       0.78      0.93      0.85      1000
 automobile       0.92      0.93      0.92      1000
       bird       0.85      0.71      0.77      1000
        cat       0.68      0.69      0.68      1000
       deer       0.82      0.85      0.84      1000
        dog       0.73      0.80      0.76      1000
       frog       0.92      0.86      0.89      1000
      horse       0.91      0.87      0.89      1000
       ship       0.92      0.89      0.90      1000
      truck       0.90      0.89      0.90      1000

avg / total       0.84      0.84      0.84     10000
```

Confusion Matrix 2



6 Layers + Augmentation + eLu + Average Pooling

After above steps of consideration and attempts, we finally get a Six-Layer CNN model with the accuracy about 84.03% with batch size equals to 128 and epoch equals to 50. Adding different improving method on baseline model can be a good method to improve the accuracy of classification of the model. Methods for preprocessing like augmentation, normalization and so on can make data much easier to be used for classification. Different layers like pooling or dropout can not only reduce the time we use, but also raise the accuracy.

During running the code, sometimes the accuracy may go down and then raise, it is because the accuracy may just exceed the local minimum point. It may happen due to different learning rates.

Actually, there may be some place to improve in this model and that will be our future objective.

5. Conclusion

The goal for our project is that we want to build a classification model which the accuracy can over 85%. Due to the Convolutional Neural Networks(CNN) is the commonly used image processing method, so at first we try the CNN baseline model which only have one layer, but the accuracy is not good. Then we make a series of improvements such as applying image augmentation, adding more layers, tuning the parameters. Finally we get the final model which has six layers and the accuracy is about 84.03%. If we have more time we will put more attention on distinguishing images of dogs and cats. For the final model we use average pooling instead of max pooling which improves the accuracy of distinguishing images of dogs and cats significantly. But compared with other type of images the accuracy of dogs and cats also should be raised. There are some other possible solutions such as adding more training data of dogs and cats and trying other deep learning methods on them.

Github Link: https://github.com/tassadarsy/CIFAR-10-project

Reference

[1] CS231n Convolutional Neural Networks for Visual Recognition
https://cs231n.github.io/convolutional-networks/#overview

[2] Course Slide: ECBM E4040 Neural Networks and Deep Learning: Convolutional Neural Networks (CNN)

[3] Ioffe S., Szegedy C. (2015). Batch Normalization: Accelearting Deep Network Training by Reducing Internal Covariate Shift. arXiv:1502.03167v3 [cs.LG] 2 Mar 2015. https://arxiv.org/pdf/1502.03167.pdf

[4] Regularization for Simplicity: L2 Regularization.
https://developers.google.com/machine-learning/crash-course/regularization-for-simplicity/l2-regularization