The Comparison of Different Machine Learning Platform and Models for CIFAR 100

Wen-Chuan Chang

DATS 6203 Machine Learning II

The George Washington University

12/04/2018

Introduction

Pattern recognition can be defined as the classification of data. This project adopts three methods to do pattern recognition of CIFAR100. CIFAR-100 consists of 60000 32x32 color images in 100 classes, with 600 images in each class. In these 600 images, there are 500 images in the training set and 100 images in the testing set. In addition, the 100 classes were grouped into 20 subclasses. One of methods to approach the CIFAR100 adopts Convolutional Neural Network (CNN) programmed using Pytorch.

Learning network and algorithm

The code for the computation of Convolutional Neural Network (CNN) is programmed using Pytorch, which deploys dynamic computational graphs. The dynamic computational graph defines the order of computations that are required to be performed. The Pytorch uses Object Oriented Programming feature to set up the framework for deep learning.

CNN typically consists of convolutional layers, pooling layers, fully connected layers and normalization layers. Convolutional layers utilize kernels (weights) to identify elemental features in the input with Shared Weights. The method of Shared Weight is considered as local connected neural nets to extract information from the input imagines. To sum up, a convolution layer then takes a set of feature maps as an input, and produces another set of feature maps as an output. Then,

the pooling layers are keys to ensure that the subsequent layers of the CNN are able to pick up larger-scale detail than just edges and curves. However, in this project, the pooling layer connected after the second convolution layer greatly reduce the number of output resulting in insufficient input information.

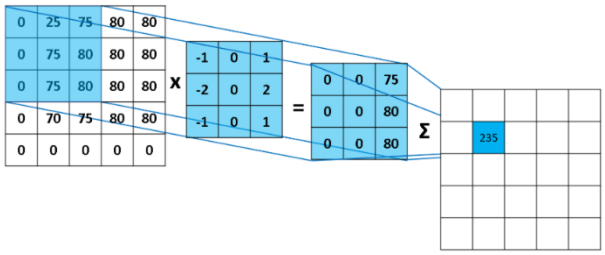

Figure 1: : a step in the Convolution Process

Initially, CNN with two convolutional layers connected with pooling layer and two fully connected layer with hidden size=120 and 100 separately is applied into the datasets CIFAR100. In the condition of ReLU as transfer function, Cross-entropy loss function as performance index, SGD as optimization function, the model reach 16% accuracy and the training model takes 191s. Subsequently, to improve the performance at accuracy, the changes of kernel size, stride, and pooling layer has investigated in this report.

```
class SimpleNet(nn.Module):

    def __init__(self, num_classes=100):
        super(SimpleNet, self).__init__()
        self.conv1 = nn.Conv2d(in_channels=3, out_channels=6, kernel_size=5)
        self.relu = nn.ReLU()
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)

        self.conv2 = nn.Conv2d(in_channels=6, out_channels=16, kernel_size=5)
        self.fc1 = nn.Linear(16* 5* 5, 120)# flatern
        self.fc2 = nn.Linear(120, num_classes)

    def forward(self, x):
        out = self.pool(self.relu(self.conv1(x)))
        out = self.pool(self.relu(self.conv2(out)))

        out=out.view(-1,16*5*5)

        out = self.relu(self.fc1(out))
        out = self.fc2(out)
        return out
```

Figure 2: CNN Model by Pytorch

Experimental setup

The initial model, CV_1, is introduced with two convolution layers and two fully connected layers. The accuracy of CV_1 is only 16%. The first parameter to adjust the initial model is increasing the number of kernels. 5 times kennels is adopted as CV-2 model. The number of kennels in the first convolution layer increase from 6 to 30, and 30 to 80 in the second convolution layer. As a result, the accuracy increases from 16% to 22%, and the training time increases from 191s to 242s. Compared to MLP, CNN costs more training time in writing number into computer's memory.

The second parameter to adjust model is changing kernel size. Kernel size of 3, 5, 8 has been used in this project. The models of CV_2, CV_3, CV_4 adopt different Kernel sizes. However, there is no linear correlation between different kernel size, and kernel size =5 has the best performance at accuracy among the models of CV_2, CV_3, CV_4.

		CV_1	CV_2	CV_3	CV_4
	batch size	200	200	200	100
	epoch	20	20	20	40
Conv1	nn.Conv2d(in_channels=3, out_channels=6, kernel_size=5)	v			
	nn.Conv2d(in_channels=3, out_channels=30, kernel_size=3)			v	
	nn.Conv2d(in_channels=3, out_channels=30, kernel_size=5)		v		
	nn.Conv2d(in_channels=3, out_channels=30, kernel_size=8)				v
pool1	nn.MaxPool2d(kernel_size=2, stride=2)	v	v	v	v
Conv2	nn.Conv2d(in_channels=6, out_channels=16, kernel_size=5)	v			
	nn.Conv2d(in_channels=30, out_channels=80, kernel_size=3)			v	
	nn.Conv2d(in_channels=30, out_channels=80, kernel_size=5)		v		
	nn.Conv2d(in_channels=30, out_channels=80, kernel_size=8)				v
pool2	nn.MaxPool2d(kernel_size=2, stride=2)	v	v	v	v
ip	nn.Linear(**, 120)	v	v	v	v
	nn.Linear(**, 100)	v	v	v	v
	Accuracy	16%	22%	20%	17%
	training time(s)	191	242	230	229

Moreover, the model of CV_5 used larger stride. Because larger stride will reduce the number of inputs into the next layer. The number of inputs has down to 80*5*5 after first convolution and pooling layer. Then, increasing stride is not suitable to use in the second convolution layer. Besides, to keep the inputs at the minimum number of inputs , 80*1*1, at the fully connected layer, the pooling at the second convolution layer should not also included. As expected, the performance at accuracy drop to 12% due to a smaller number of outputs from feature maps.

From the experiment of model of CV_5, it implies that the original model has potential risk of insufficient number of inputs from feature maps. Therefore, the model of removing second pooling layer has been proposed to increase the number of outputs from feature maps. In the model of CV_6, without second pooling layer, the number of inputs into first fully connected layer increased to 80*10*10. And, the accuracy of CV_6 has also increased to 28%.

		CV_2	CV_5	CV_6
	batch size	200	200	200
	epoch	20	20	20
Conv1	nn.Conv2d(in_channels=3, out_channels=30, kernel_size=5)	v		v
	nn.Conv2d(in_channels=3, out_channels=30, kernel_size=5,stride=3)		v	
pool1	nn.MaxPool2d(kernel_size=2, stride=2)	v	v	v
Conv2	nn.Conv2d(in_channels=30, out_channels=80, kernel_size=5)	v		v
	nn.Conv2d(in_channels=30, out_channels=80, kernel_size=5,stride=1)		v	
pool2	nn.MaxPool2d(kernel_size=2, stride=2)	v		
ip	nn.Linear(**, 120)	v	v	v
	nn.Linear(**, 100)	v	v	v
	Accuracy	22%	12%	28%
	training time(s)	242	186	246

Finally, CV_7 combing the advantage of CV_6 and increasing neurons at fully connected layers reaches the best performance at accuracy of 37%. Because the more number of weights has been used, the iteration and epoch times should also add to the model CV_7.

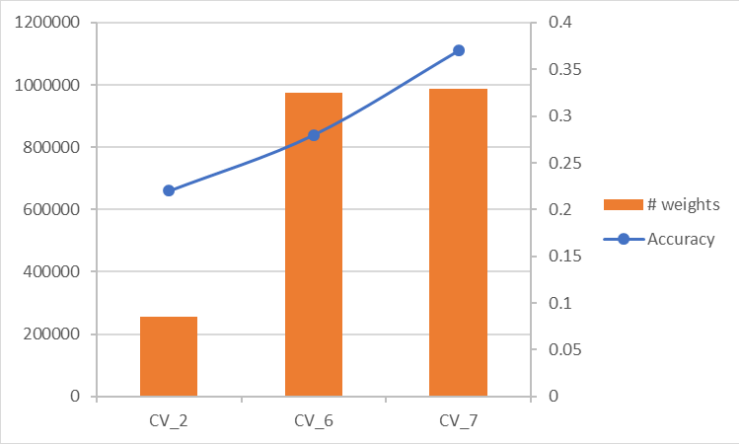

Figure 3: the accuracy and the number of weights of CV_2, CV_6,CV_7

		CV_2	CV_6	CV_7
	batch size	200	200	100
	epoch	20	20	40
Conv1	nn.Conv2d(in_channels=3, out_channels=30, kernel_size=5)	v	v	v
pool1	nn.MaxPool2d(kernel_size=2, stride=2)	v	v	v
Conv2	nn.Conv2d(in_channels=30, out_channels=80, kernel_size=5)	v	v	v
pool2	nn.MaxPool2d(kernel_size=2, stride=2)	v		
ip	nn.Linear(**, 120)	v	v	v
	nn.Linear(**, 110)			v
	nn.Linear(**, 100)	v	v	v
	Accuracy	22%	28%	37%
	training time(s)	242	246	500

Result

CNNs are usually applied to image data. Every image is a matrix of pixel values, and the CNN learns the features from the input images. Typically, they emerge repeatedly from the data to gain prominence. If the number of inputs extracting the information from the original picture is insufficient, the training model leads to lower performance at accuracy. To increase the number of inputs from feature maps, removing second pooling layer is proposed in this project.

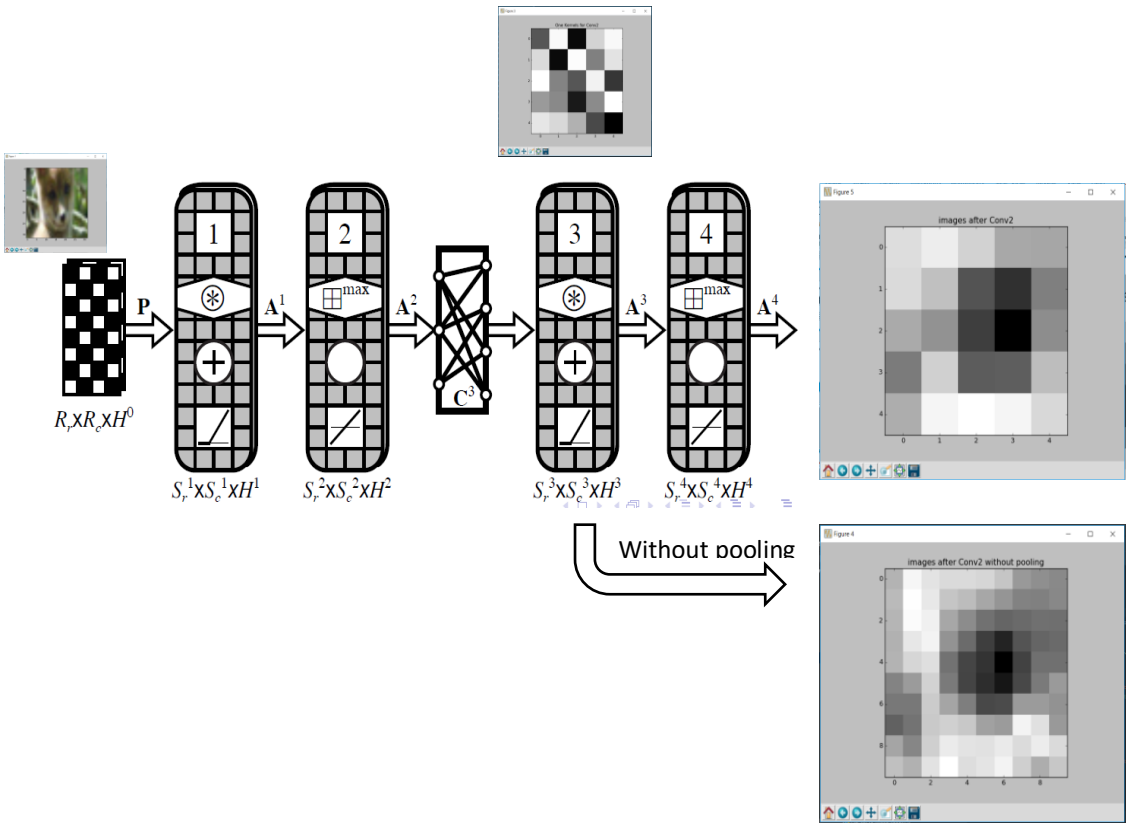

Figure 4: Figure 5: Output after convolution2 with and without pooling

Take an example of test picture of a fox. The output after second convolution layer without pooling has more features about a fox, the input. It captures more three dimensional information about fox's facial features compared the result with pooling layer. To sum up, the CNN can efficiently apply to pattern recognition, but the parameter used in the convolution and pooling layers may resulting in insufficient inputs. Therefore, the parameters, kernel number, kernel size, stride in the convolution and pooling layers, need to be optimized to create a better CNN model.

Conclusions

One of methods to approach the CIFAR100 adopts Convolutional Neural Network (CNN) programmed using Pytorch. The convolution layers in the CNN identify elemental features in the input. Therefore, the number of outputs of convolutions layers should be enough to ensure that the model extracts sufficient information from the input pictures. In the project, the comparison between models of CV_1 and CV_6 confirmed that the initial model is lack of the number of inputs. After increasing the number of neurons and training times, the final model CV_7 reaches the highest accuracy of the CNN model in this project.

References

Cs.toronto.edu. (2018). CIFAR-10 and CIFAR-100 datasets. [online] Available at: https://www.cs.toronto.edu/~kriz/cifar.html [Accessed 4 Dec. 2018].

Pytorch.org. (2018). *torch.nn — PyTorch master documentation*. [online] Available at:

https://pytorch.org/docs/stable/nn.html [Accessed 4 Dec. 2018]