Report of Deep learning with PyTorch : Object classification

Students : ZHAO Mengzi, ZHOU Juncheng

I - Construction of MyConvolutionalNetwork

At first, we add the second layer :

self.conv2 = nn.Conv2d(18,30,kernel_size=3, stride=1, padding=1)
self.pool2 = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)

Then we need to update the size of output of the last convolution by using the function get_output_size(), we have :

out_size = get_output_size(in_size=32, kernel_size=3, stride=2, padding=1)

We can obtain the size of the first convolution : 16
For the second convolution that we create, we use also the get_output_size() function :

out_size = get_output_size(in_size=16, kernel_size=3, stride=2, padding=1)

We obtain the size of the new convolution : 8
So we update the size of the output :

*self.flattened_size = 30 * 8 * 8*

To use the new layer :

x = F.relu(self.conv2(x))
x = self.pool2(x)

or

x = F.tanh(self.conv2(x))
x = self.pool2(x)

But we tested these two different functions, we found that the relu's performance is better than tanh.

II - Exploring CNN architecture with different parameters

At first, we try different parameters of the neural network structure :

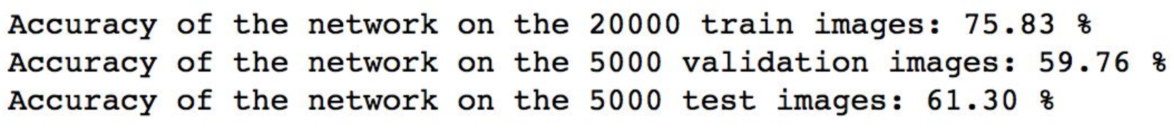
- **kernel size**

1 - kernel size of layer :

We test with 2 layers as below

self.conv1 = nn.Conv2d(3, 18, kernel_size=3, stride=1, padding=1)
self.pool = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)
self.conv2 = nn.Conv2d(18,30,kernel_size=3, stride=1, padding=1)
self.pool2 = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)

We have result :



```
Accuracy of the network on the 20000 train images: 75.83 %
Accuracy of the network on the 5000 validation images: 59.76 %
Accuracy of the network on the 5000 test images: 61.30 %
```

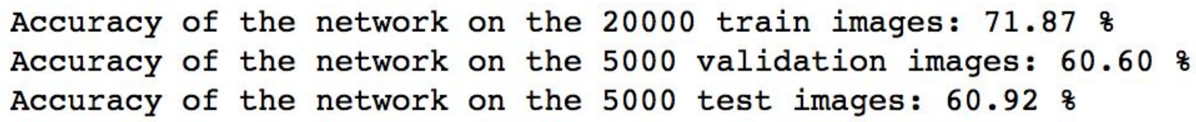
When we set the kernel size of the layers equal to 7 :

```
self.conv1 = nn.Conv2d(3, 18, kernel_size=7, stride=1, padding=1)
self.pool = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)
self.conv2 = nn.Conv2d(18,30,kernel_size=7, stride=1, padding=1)
self.pool2 = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)
```

In this case, according to the function *get_output_size,* we update the size of output :

*self.flattened_size = 30 * 5 * 5*

We have :



```
Accuracy of the network on the 20000 train images: 71.87 %
Accuracy of the network on the 5000 validation images: 60.60 %
Accuracy of the network on the 5000 test images: 60.92 %
```

We can notice that when the kernel size of the layer is larger, the accuracy on the test set becomes worse. We can know when the size of kernel is larger, we can obtain more features, but when the size of kernel is large, the complexity of computation is large, it is not good to increase the depth, and when the kernel size is smaller, it can greatly reduce the computational complexity. For example, the complexity of the convolution kernel whose size = 7 is 49M (M is a constant) and the complexity of the 3 kernels whose size = 3 is 27M. It is better when we have some small size kernels than a large size kernel.

2 - test with kernel size of MaxPool2d :

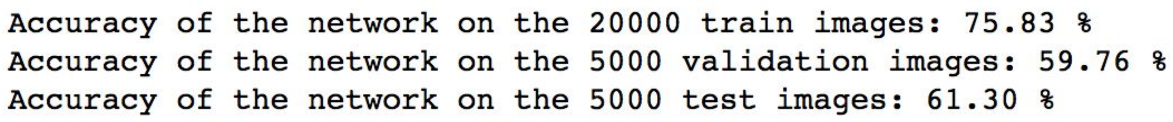
Pooling layer is to compress the input feature map, on the one hand, it makes the feature map smaller and simplifies the computational complexity of the network. On the one hand, feature compression is performed to extract the main features. We use the pooling layer behind the convolution layer by pooling to reduce the eigenvector output by the convolution layer while improving the result (less prone to overfitting)

```
self.conv1 = nn.Conv2d(3, 18, kernel_size=3, stride=1, padding=1)
self.pool = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)
self.conv2 = nn.Conv2d(18,30,kernel_size=3, stride=1, padding=1)
self.pool2 = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)
```

According to the function *get_output_size*, we have :

*self.flattened_size = 30 * 8 * 8*

We have result :



```
Accuracy of the network on the 20000 train images: 75.83 %
Accuracy of the network on the 5000 validation images: 59.76 %
Accuracy of the network on the 5000 test images: 61.30 %
```

We set the kernel size of MaxPool2d larger, kernel_size = 3 for the MaxPool2d, we have :

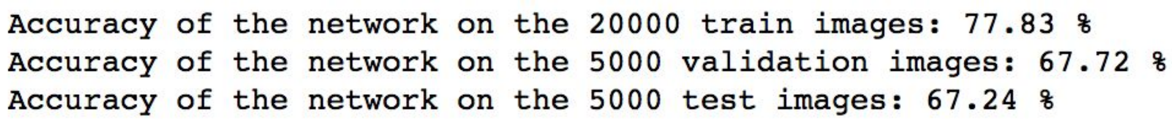
```
self.conv1 = nn.Conv2d(3, 18, kernel_size=3, stride=1, padding=1)
```

```
    self.pool = nn.MaxPool2d(kernel_size=3, stride=2, padding=0)
    self.conv2 = nn.Conv2d(18,30,kernel_size=3, stride=1, padding=1)
    self.pool2 = nn.MaxPool2d(kernel_size=3, stride=2, padding=0)
```

According to the function *get_output_size*, we have :

*self.flattened_size = 30 * 7 * 7*

We have result :

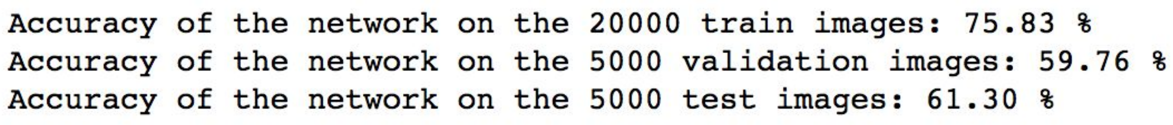


```
Accuracy of the network on the 20000 train images: 77.83 %
Accuracy of the network on the 5000 validation images: 67.72 %
Accuracy of the network on the 5000 test images: 67.24 %
```

We can notice that when the kernel size of MaxPool2d is larger, the accuracy of the network on the test set is better.

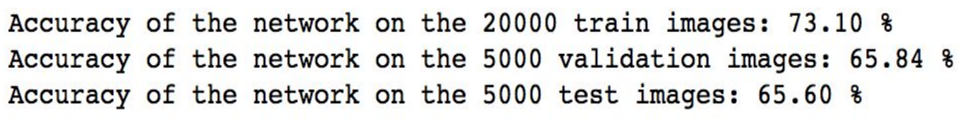
- **number of convolutional layers**

At the beginning, we have just one convolutional layer, we have results as below :



```
Accuracy of the network on the 20000 train images: 75.83 %
Accuracy of the network on the 5000 validation images: 59.76 %
Accuracy of the network on the 5000 test images: 61.30 %
```

Then we add a new layer as we said in the first part of this report, we have :



```
Accuracy of the network on the 20000 train images: 73.10 %
Accuracy of the network on the 5000 validation images: 65.84 %
Accuracy of the network on the 5000 test images: 65.60 %
```

We can notice that when we augment the number of the convolutional layers, the accuracy of the network on the test set is better than before.

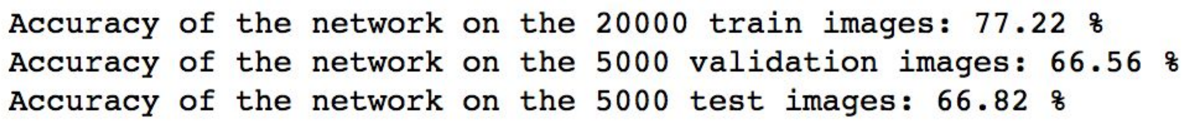
We add the third layer :

```
    self.conv3 = nn.Conv2d(30,40,kernel_size=3, stride=1, padding=1)
    self.pool3 = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)
```

According to the function *get_output_size*, we have :

*self.flattened_size = 40 * 4 * 4*

We have result :



```
Accuracy of the network on the 20000 train images: 77.22 %
Accuracy of the network on the 5000 validation images: 66.56 %
Accuracy of the network on the 5000 test images: 66.82 %
```

We can notice that when the number of layer is more, the result is better. But when there are so many layers, it may cause overfitting.

But we find that when we try to set the value of kernels_size in the pooling bigger, and increase the number of the layers at the same time the result is not good.

- **size of fully connected layers**

At the beginning, we have :

self.fc1 = nn.Linear(self.flattened_size, 64)
self.fc2 = nn.Linear(64, 10)

The result is :

```
Accuracy of the network on the 20000 train images: 75.83 %
Accuracy of the network on the 5000 validation images: 59.76 %
Accuracy of the network on the 5000 test images: 61.30 %
```

self.fc1 = nn.Linear(self.flattened_size, 120)
self.fc2 = nn.Linear(120, 10)

We have the result :

```
Accuracy of the network on the 20000 train images: 79.19 %
Accuracy of the network on the 5000 validation images: 65.54 %
Accuracy of the network on the 5000 test images: 64.46 %
```

We can notice that when the size of fully connected layers is larger, the result is better.

We combine these ways above, we have result :

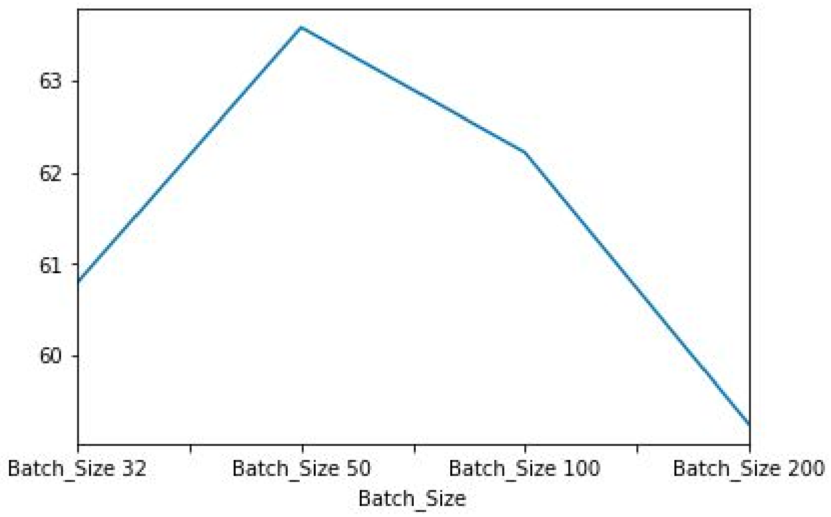
```
Accuracy of the network on the 20000 train images: 71.27 %
Accuracy of the network on the 5000 validation images: 66.34 %
Accuracy of the network on the 5000 test images: 65.60 %
```

- **batch_size**

The batch_size means number of samples that going to be propagated through the network.
We test with 2 variables :

n_epochs= 10
learning_rate= 0.001

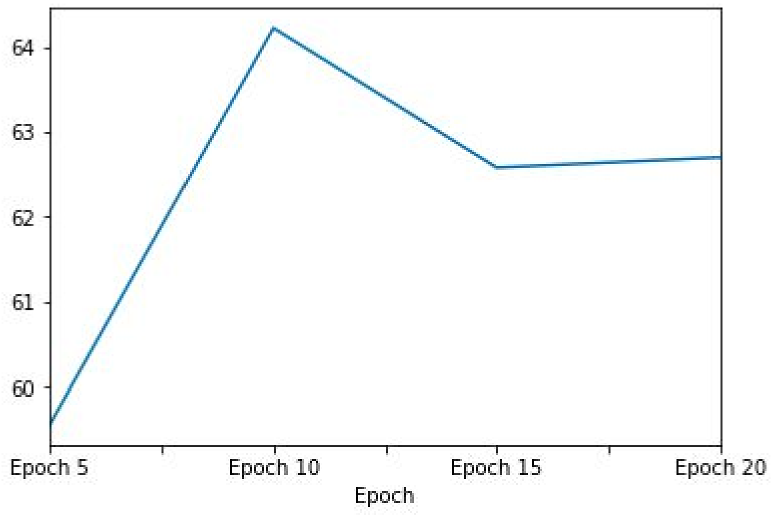
According to the value of test accuracy, we have the graph as below :



According to this graph, we can notice that the value of test accuracy augment from the batch_size = 32 to batch_size = 50, then the test accuracy decreases, so we can know that Batch_Size is not as big as possible.

- number of epochs

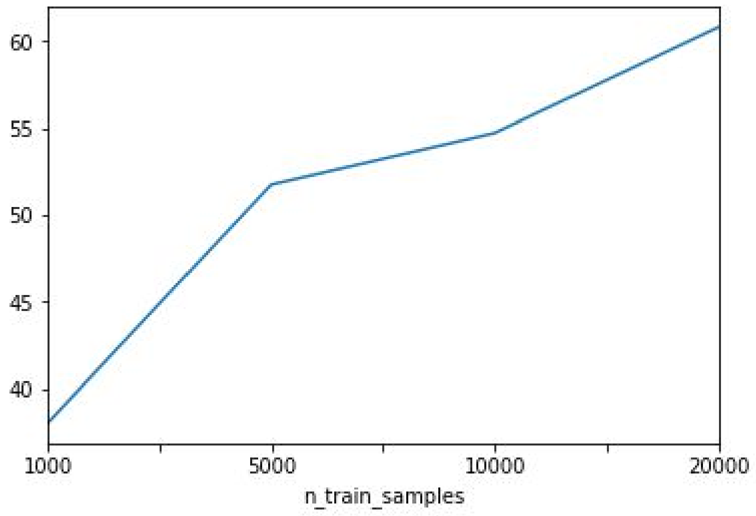
One epoch consists of one full training cycle on the training set.



From Epoch = 5 to Epoch = 10, we can notice that the test accuracy increases until more than 64, and then it decreases. So when Epoc is more than 10, the test accuracy becomes bad.

- size of the training set/validation set

We change also the size of the set, when we change the size. The axe y is the test accuracy.
The result with different size of train sets:


n_train_samples