Classification and recognization of RGB images using Pytorch

Kai He[1] and **Jin Dai**[2] and **Ruiyue Wang**[3] and **Mengyue Duan**[4]

Abstract

In today's society, machine learning is already everywhere in our life. Classification and recognition of images are often realized by machine learning. This project is base on the classification and recognization of 32*32*3 RGB images in CIFAR10 dataset. There are 10 class in CIFAR10 dataset, During the project, we use neural network, Convolutional Neural Networks (CNN) implementation process, and corresponding Stochastic Gradient Descent algorithm to realize the classification. We alse change the learning-rate(stepsize) and MOMENTUM of SGD to optimize the accuracy of classification.

1. Instruction

The implementation process of this project is to first select a batch in CIFAR as the training set, with a total of 10,000 pictures. Then convolution neural network is established and trained. SGD is used to optimize the model, and the better parameters of neural network are selected to complete the training of classifier. Furthermore, we randomly selected 5 pictures from the 10000 pictures in the test set to test the effect of the classifier and output them. Finally, all the pictures of all the test sets are used for testing, and the classification accuracy of the classifier for 10 class is obtained, and the comprehensive accuracy is calculated. After the first test, in order to optimize the classifier to improve the classification accuracy, we changed SGD's learning rate and MOMENTUM parameters and increased the number of training.

2. CIFAR10 dataset

CIFAR-10 data set consists of 10 types of 32*32*3 color pictures, with a total of 60000 pictures, each type containing 6000 pictures. Among them, 50,000 pictures were used as training set and 10,000 pictures were used as test set.

. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

CIFAR-10 data set is divided into 5 training batches and 1 testing batch, each batch contains 10000 pictures. The pictures in the test set batch are composed of 1,000 pictures randomly selected from each category, and the training set batch contains the remaining 50,000 pictures in random order. However, some training sets batch may contain more pictures of one category than other categories. The training set batch contains 5,000 pictures from each category, totaling 50,000 training pictures.

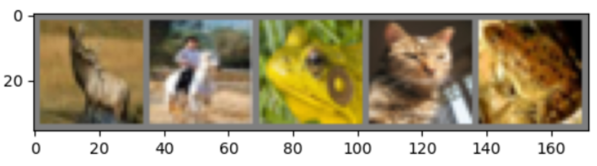


Figure 1. outputsample

3. Convolution Neural Networks

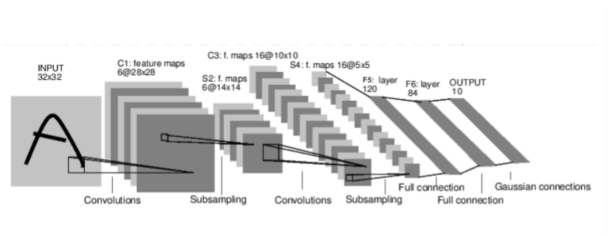


Figure 2. An example of convolution neural network with 32x32x3 picture

CNN is a method which is most commonly used in analysing visual imagery. Convolution network are the neural network that use convolution instead of general matrix multiplication in each layer. When giving a picture, a CNN will do the following steps: 1.Use a wanted method, such as PCA, moving-mean to process the image and use the wanted data as input layer of the CNN. 2.Do convolution in each layer. 3.Use Relu as activation function between each convolution layer. 4.Pooling. 5.Fully connected.

In image processing, a convolution step is to take the inner product of the input data and the filter. For example, for a picture, using a 3x3 filter, the convolution of the first 3x3 matrix of source and the filter is:

$$\begin{bmatrix}0&0&0\\0&1&1\\0&1&2\end{bmatrix}\times\begin{bmatrix}4&0&0\\0&0&0\\0&0&-4\end{bmatrix}=0\times4+0\times0+0\times0+0\times0+1\times0+1\times0+0\times0+1\times0+2\times-4=-8$$

Figure 3. convolution of 3x3 matrix and filter

The result of the whole layer after convolution is called 'features'. Then an activation function is needed to connect between every layer in CNN. In convolution neural network ReLU function is usually chosen as the activation function instead of sigmoid. The ReLU function has the advantage of fast convergence and simple to calculate gradient. Pooling is a method of reducing dimension. it means taking the maximum or average value of an area. For example, for a 4x4 matrix which is divided into four parts, the result of applying pooling is:

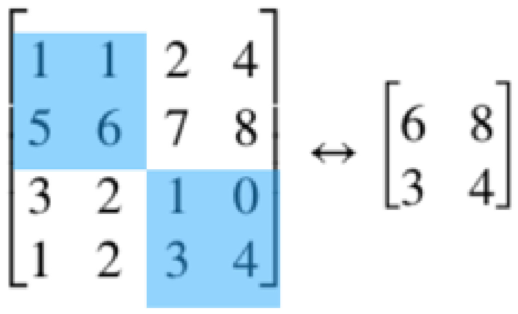
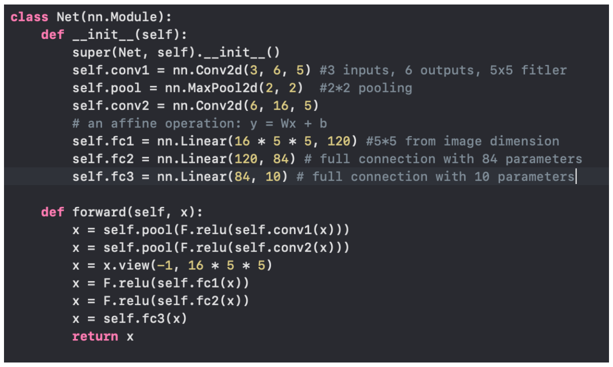


Figure 4. pooling

In this project, the pictures in the library are all 32x32x3 in size. The three dimensions of the picture are divided into three different 32x32x1 matrixes, and then perform convolution separately. There are three inputs for each picture and 6 outputs (6 different classes) in this project. The filter size is chosen as 5x5. The convolution neural network can be constructed by library of pytorch as follow:

Fully connected layers has the same principle as traditional multi-layer perceptron neural network. It connects every neuron in one layer to every neuron in another layer.

For the 32x32 picture, 2 convolution layers and 2 pooling layers are used before fully connected layer. In fully connected layer 1, the input is a 16x5x5 source and we set the number of weight as 120. The number of parameters of second and third fully connected layer are set as 84 and 10. Note that when the total number of parameters in fully connected layer increase, the performance of the model will increase. But a too large number of parameters will reduce the computational efficiency of the model.



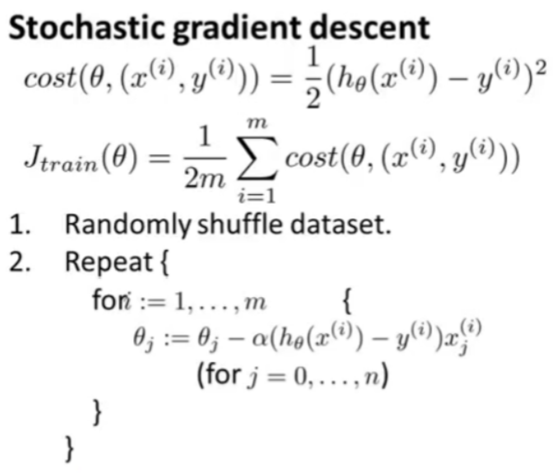
```python
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5) #3 inputs, 6 outputs, 5x5 fitler
        self.pool = nn.MaxPool2d(2, 2)  #2*2 pooling
        self.conv2 = nn.Conv2d(6, 16, 5)
        # an affine operation: y = Wx + b
        self.fc1 = nn.Linear(16 * 5 * 5, 120) #5*5 from image dimension
        self.fc2 = nn.Linear(120, 84) # full connection with 84 parameters
        self.fc3 = nn.Linear(84, 10) # full connection with 10 parameters

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 16 * 5 * 5)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

Figure 5. Achieve CNN using pytorch library

Ian Goodfellow and Yoshua Bengio and Aaron Courville (2016). Deep Learning. MIT Press. p. 326.

4. Stochastic gradient descent

As the batch gradient descent(normal gradient descent) method needs all training samples when updating each parameter, the training process will become abnormally slow with the increase of the number of samples. Stochastic Gradient Descent (SGD) is proposed to solve the problem of batch gradient descent. Random gradient descent is updated iteratively through each sample. If the sample size is large (e.g., hundreds of thousands), theta may have been iterated to the optimal solution using only tens of thousands or thousands of samples. Compared with the above batch gradient descent, one iteration requires more than 100,000 training samples, one iteration cannot be optimal, and 10 iterations require traversing the training samples. However, one of the problems associated with SGD is that the noise is more than BGD, which makes SGD not go toward the overall optimization direction every iteration.



Stochastic gradient descent

$$cost(\theta,(x^{(i)},y^{(i)}))=\frac{1}{2}(h_\theta(x^{(i)})-y^{(i)})^2$$

$$J_{train}(\theta)=\frac{1}{2m}\sum_{i=1}^{m}cost(\theta,(x^{(i)},y^{(i)}))$$

```
1.  Randomly shuffle dataset.
2.  Repeat {
        for $i := 1,\dots,m$        {
            $\theta_j := \theta_j - \alpha(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$
                (for $j = 0,\dots,n$)
        }
    }
```

Figure 6. SGD Algorithm

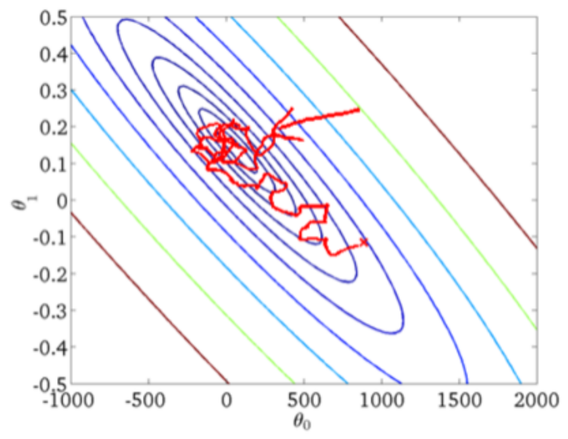


Figure 7. SGD

The derivative of the loss function and the loss function are obtained by the back propagation algorithm of the neural network.

Defect of SGD: Very slow progress along shallow dimension, jitter along steep direction. Going to local minimum or saddle point will result in gradient being 0 and cannot be moved. In fact, saddle point problem is more common in high-dimensional problems.

Advantages of SGD: Fast training speed.

5. Experimental results

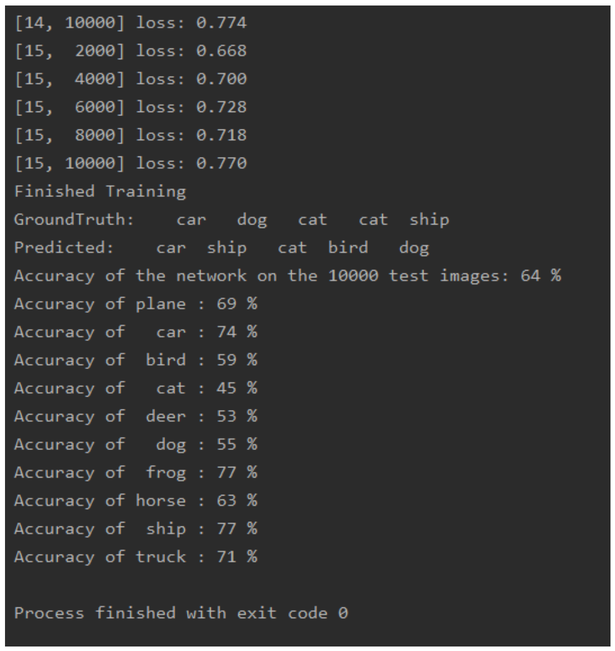
In the experiment, we try to change the training times, learning rate, and MOMENTUM to get the best accuracy of the classification. The learning rate represents the step size of the function at any time during each repeat. When learning rate is smaller, we will have more chance to get the lower cost function. But we will spend more training times to arrive global smaller value.

Momentum update

v = mu * v - learning$_rate * dxx + v = v$

A variable V initialized to 0 and a super parameter mu are introduced here. It is inappropriate to say that this variable (mu) is regarded as momentum in the optimization process (the general value is set to 0.9), but its physical meaning is more consistent with the friction coefficient . This variable effectively suppresses the speed and reduces the kinetic energy of the system, otherwise the particle will never stop at the bottom of the mountain. This parameter is usually set to one of [0.5,0.9,0.95,0.99].

Training tims: With the appropriate learning rate and MOMENTUM set, the The more times you train, the closer it is to global minimum.



```
[14, 10000] loss: 0.774
[15,  2000] loss: 0.668
[15,  4000] loss: 0.700
[15,  6000] loss: 0.728
[15,  8000] loss: 0.718
[15, 10000] loss: 0.770
Finished Training
GroundTruth:    car   dog   cat   cat  ship
Predicted:    car  ship   cat  bird   dog
Accuracy of the network on the 10000 test images: 64 %
Accuracy of plane : 69 %
Accuracy of   car : 74 %
Accuracy of  bird : 59 %
Accuracy of   cat : 45 %
Accuracy of  deer : 53 %
Accuracy of   dog : 55 %
Accuracy of  frog : 77 %
Accuracy of horse : 63 %
Accuracy of  ship : 77 %
Accuracy of truck : 71 %

Process finished with exit code 0
```

Figure 8. Accuracy:0.64 training times:15 lr:0.0005 momentum:0.9

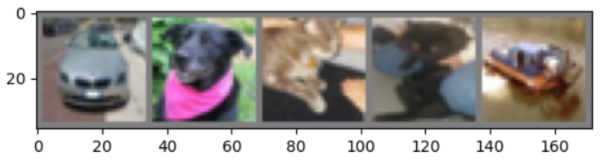


Figure 9. 5picture of test

Other examples:

Accuracy:0.31 training times:2 lr:0.01 momentum:0.9.

Accuracy:0.58 training times:4 lr:0.001 momentum:0.9.

Accuracy:0.60 training times:15 lr:0.001 momentum:0.9.

Accuracy:0.64 training times:15 lr:0.0005 momentum:0.9.

When we choose big learning rate and training a few times, Accuracy of classification is very low. And as we increase the training times and decrease the learning rate, Accuracy of classification is more and more high. Limited by the configuration of computer hardware and other issues (such as without GPU, and cannot work with multiple threads), we can not train hundred or thousand times and set the learning rate very small, like 0.00001, and set momentum 0.99. When we realize that operation, the accuracy of classifer will be far greater than than 0.64.

6. Team member contribution

Kai He : Program design and implementation, integration and modification of reports

Jin Dai : Write convolution neural network report and participate in program design and debugging.

Ruiyue Wang : Project presentation, data optimization of project, and reports modification

Mengyue Duan : PPT production and modification, arrangement of data and pictures, and presentation design.

Acknowledgements

Thanks to Professor Deniz for the support and instruction.

References

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.