

Machine Learning 2

by Xinzhe Lyu

Submission date: 04-May-2023 06:05PM (UTC+0100)

Submission ID: 204760295

File name: 40633_Xinzhe_Lyu_Machine_Learning_2_498640_1970751654.docx (678.65K)

Word count: 2762

Character count: 13112



Machine Learning 2

Student name: Xinzhe Lyu

Student code: 2214776

Programme: MSc Software Engineering and AI

Moudle: Application of Machine Learning

Moudle code: ACCA7025

Lecturer: Gordon Dickers

Abstract

As mentioned in the last Assignment, the classification training of cifar-10 training set using random forest method and mlp method did not achieve a good effect. In this paper, after learning the concept of cnn and referring to papers and online information, we decided to use the Resnet-18 algorithm in the convolutional neural network to train the cifar-10 data set. We use the preprocessing technology of images, and continuously adjust the training times and the learning rate to obtain a better result.

Introduction

In this paper, the ResNet18 algorithm is used to train the Cifar-10 dataset, and the accuracy of the trained model is considerable. In the Related works section, the Cifar-10 dataset is briefly introduced, and the applied technologies CNN, ResNet, and the residual link module among them are introduced in principle. In the Result section, the implementation process of the training model is summarized, including the introduction of the software environment, the ResNet18 algorithm model built by pytorch, and the training results. In the Discussion section, the hyperparameter Settings of the training process and the summary of the results are presented. In the Conclusion section, the advantages and disadvantages of the algorithm are analyzed, and the important problems and solutions in the experimental process are summarized.

Related works

Cifar-10 dataset

The CIFAR10 dataset has a total of 60,000 samples, each sample is a 32*32 pixel RGB image (color image), and each RGB image must be divided into three channels (R channel, G channel, B channel). These 60, 000 samples are divided into 50, 000 training samples and 10, 000 test samples.

Label	Description
0	Airplane
1	Automobile

2	Bird
3	Cat
4	Deer
5	Dog
6	Frog
7	Horse
8	Ship
9	Truck

Table 1. Cifar-10 dataset Label and Description

ResNet

The ResNet network is a reference to the VGG19 network, modified from its base, with residual units added through a short-circuit mechanism, as shown in Figure2. The main changes are that ResNet directly uses convolutions with stride=2 for downsampling, and replaces the fully connected layers with a global average pooling layer. An important design principle of ResNet is: when the feature map size is halved, the number of feature maps is doubled, which maintains the complexity of the network layers. As can be seen in Figure 1, ResNet adds a short-circuit mechanism between each two layers, which constitutes residual learning, compared to the vanilla network, where the dashed line indicates the change in the number of feature maps. Figure 1 shows a 34-layer ResNet, which can also build deeper networks as shown in Figure 1. As can be seen in Table, residual learning is performed between two layers for both 18-layer and 34-layer ResNet. When the network is deeper, residual learning is performed between the three layers, where the convolutional kernels are 1x1, 3x3 and 1x1, respectively. It is also 1/4 of the number of output feature maps.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 1 ResNet structure diagram



ResNet18

The 18 layers of ResNet18 represent 18 layers with weights, including convolutional layers and fully connected layers, excluding pooling layers and BN layers.

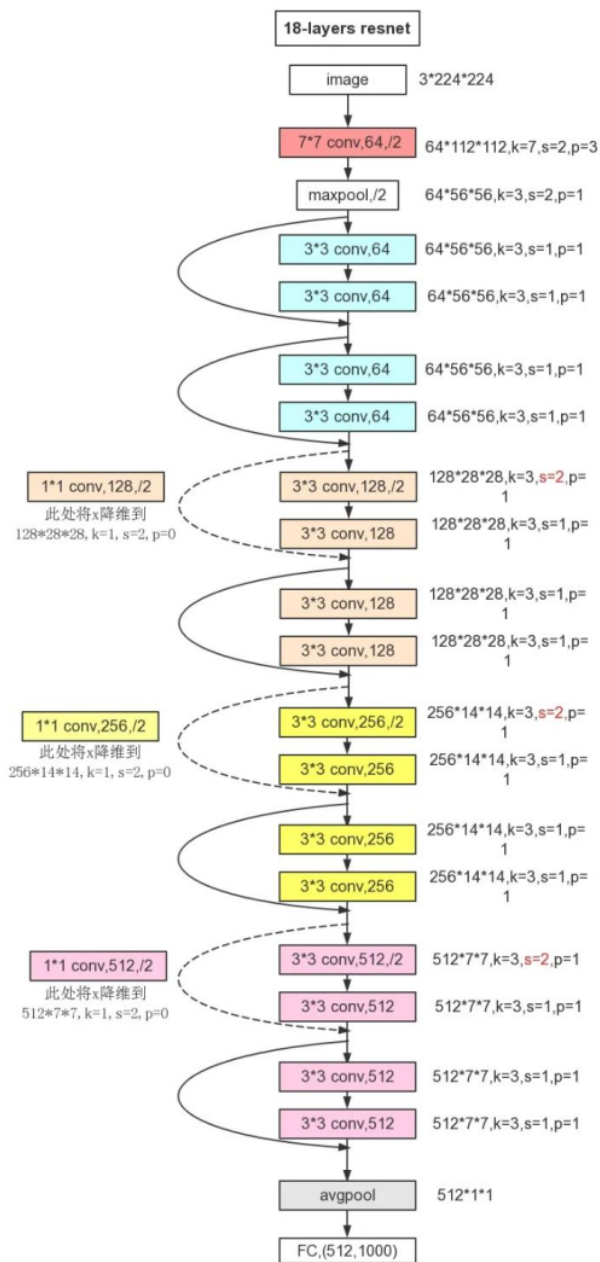


Figure 2 Res₃ structure diagram

layer1

ResNet18, which uses BasicBlock. layer1, which is characterized by no downsampling, and the stride = 1 of the convolutional layer, which does not downsample. The

downsample layer is also not passed through when the shortcut connection is made.

layer2, layer3, layer4

The structure diagram of layer2, layer3, and layer4 is as follows, each layer contains 2 basicblocks, but the first convolutional layer of the first BasicBlock has stride = 2 and will perform downsampling. When making a shortcut connection, it will go through the downsample layer for downsampling and dimensionality reduction.

Residual link shortcut connection

The degeneracy problem of deep networks at least shows that deep networks are not easy to train. But let us consider the fact that now you have a shallow network and you want to build a deep network by stacking new layers on top of each other. One extreme scenario is that these additional layers learn nothing and simply copy the features of the shallow network, such that the new layer is an identity map. In this case, the deep network should perform at least as well as the shallow network, and there should be no degradation phenomena. Well, you have to admit that it must be the current training methods that make it difficult for deep networks to find good parameters.

Dr. Ho proposed residual learning to solve the degradation problem. For a stacking structure (stacked of several layers), the learned feature given an input of x is denoted as $H(x)$. Now we want it to learn the residual $F(x) = H(x) - x$, so that the original learned feature is $F(x) + x$. The reason for this is that residual learning is easier than learning the original features directly. When the residual is 0, the accumulation layer just does the identity mapping and at least the network performance does not degrade, in fact, the residual will not be 0, which will also allow the accumulation layer to learn new features based on the input features and thus have better performance. The structure of the residual learning is shown in Figure 3. This is similar to a "short circuit" in an electrical circuit, so it's a shortcut connection.

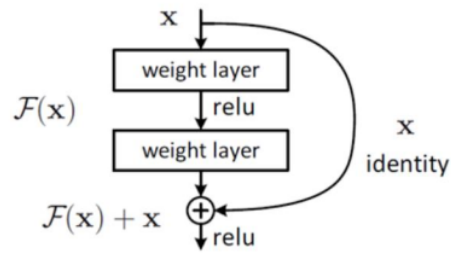


Figure 3 Schematic diagram of the residual link

Results

Software environment for this experiment: Python 3.10 Pytorch1.10

Hardware: NVIDIA GeForce RTX 3080

In this experiment, the hyperparameter tuning Settings are carried out by controlling variables, and finally a best model relative to this experiment is trained.

EPOCH

The best results during training, when adjusting epochs: (10,15,20), are shown in bold in the table.

EPOCH=10

EPOCH	1	2	3	4	5	6	7	8	9	10
Accuracy	51.44%	64.56%	66.21%	78.00%	77.99%	76.45%	81.52%	80.88%	78.41%	82.73%

Table 2 Accuracy per epoch when the maximum epoch is 10

EPOCH=15

EPOCH	1	2	3	4	5	6	7	8	9	10
Accuracy	51.44%	64.56%	66.21%	78.00%	77.99%	76.45%	81.52%	80.88%	78.41%	82.73%

11	12	13	14	15
83.24%	85.88%	80.92%	86.20%	84.71%

Table 3 Accuracy per epoch when the maximum epoch is 15

EPOCH=20

EPOCH	1	2	3	4	5	6	7	8	9	10
Accuracy	52.90%	61.13%	71.68%	73.77%	75.74%	78.48%	80.83%	80.90%	81.84%	80.88%

11	12	13	14	15	16	17	18	19	20
82.86%	85.49%	85.85%	84.29%	85.84%	85.28%	86.87%	85.51%	86.48%	86.36%

Table 4 Accuracy per epoch when the maximum epoch is 20

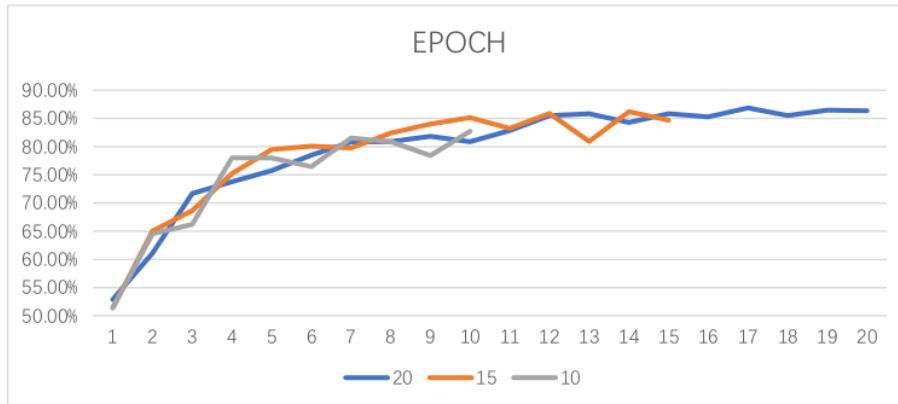


Figure 4 Accuracy per epoch of modified epochs

Batch Size

The best results during training, when adjusting the Batch Size: (128,256,512), are shown in bold in the table.

Batch Size=128

EPOCH	1	2	3	4	5	6	7	8	9	10
Accuracy	53.05%	70.98%	74.44%	78.23%	78.66%	80.52%	83.88%	84.51%	81.40%	85.54%

11	12	13	14	15	16	17	18	19	20
85.96%	83.91%	87.66%	87.94%	85.63%	87.59%	88.13%	87.73%	87.87%	88.53%

Table 5 Accuracy per epoch when batch size is 128

Running time : 513.417S

Batch Size=256

EPOCH	1	2	3	4	5	6	7	8	9	10
Accuracy	52.52%	64.06%	68.51%	73.43%	78.71%	79.63%	73.03%	81.73%	84.01%	84.92%

11	12	13	14	15	16	17	18	19	20
84.35%	86.90%	84.97%	82.25%	85.28%	86.21%	86.26%	87.11%	86.35%	89.16%

Table 6 Accuracy per epoch when batch size is 256

Running time : 488.717S

Batch Size=512

EPOCH	1	2	3	4	5	6	7	8	9	10
Accuracy	51.55%	57.14%	69.23%	70.86%	72.15%	79.31%	78.57%	82.87%	79.62%	82.05%

11	12	13	14	15	16	17	18	19	20
81.21%	83.84%	83.17%	84.28%	84.76%	83.44%	87.08%	85.08%	87.14%	85.17%

Table 6 Accuracy per epoch when batch size is 512

Running time : 481.006S

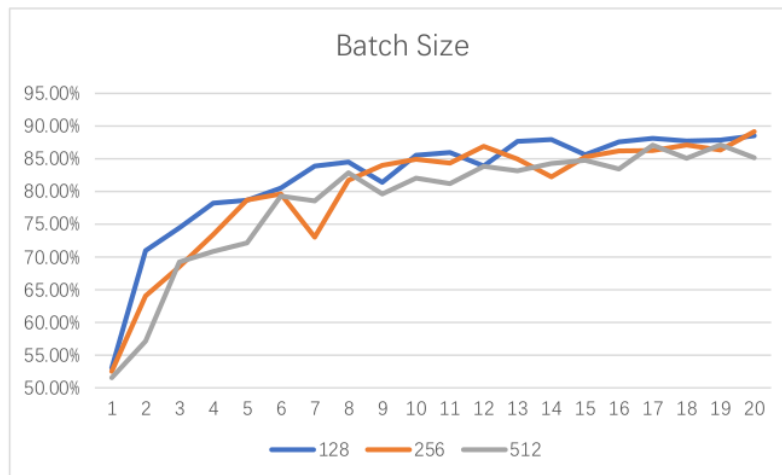


Figure 5 Accuracy per epoch of modified batch size

LR (learn rate)

When adjusting LR: (0.1,0.01,0.001), the best result during training, shown in bold in the table.

LR=0.1

EPOCH	1	2	3	4	5	6	7	8	9	10
Accuracy	25.00%	31.28%	40.32%	44.37%	47.18%	51.99%	55.07%	59.19%	60.84%	54.62%

11	12	13	14	15	16	17	18	19	20
69.32%	70.83%	69.00%	70.13%	68.74%	75.57%	75.85%	75.97%	80.09%	71.38%

Table 7 Accuracy per epoch when learn rate is 0.1

LR=0.01

EPOCH	1	2	3	4	5	6	7	8	9	10
Accuracy	52.90%	61.13%	71.68%	73.77%	75.74%	78.48%	80.83%	80.90%	81.84%	80.88%

11	12	13	14	15	16	17	18	19	20
82.86%	85.49%	85.85%	84.29%	85.84%	85.28%	86.87%	85.51%	86.48%	86.36%

Table 8 Accuracy per epoch when learn rate is 0.01

LR=0.001

EPOCH	1	2	3	4	5	6	7	8	9	10
Accuracy	37.99%	46.50%	50.98%	55.03%	58.34%	57.68%	63.20%	61.25%	68.42%	67.61%

11	12	13	14	15	16	17	18	19	20
69.80%	70.46%	71.44%	72.76%	73.90%	75.65%	75.82%	76.19%	76.29%	78.18%

Table 9 Accuracy per epoch when learn rate is 0.001

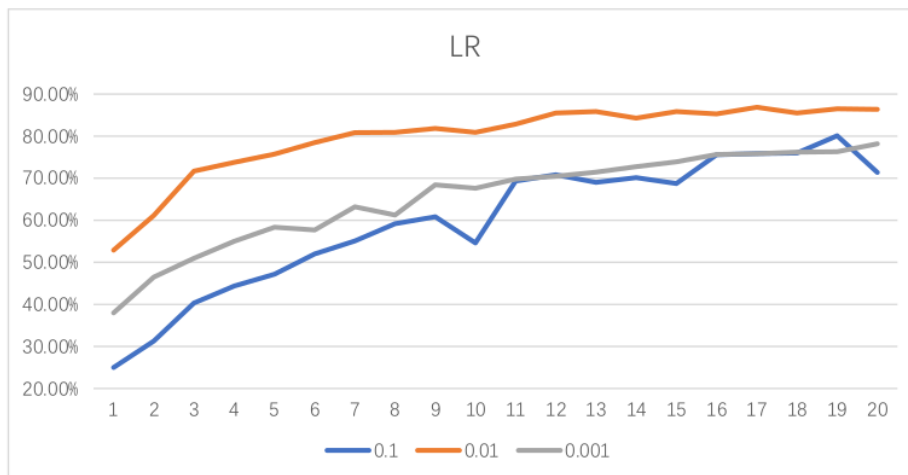


Figure 6 Accuracy per epoch of modified learn rate

Best Model

Training results for EPOCH=30, Batch Size=256, LR=0.01, the best result during training, shown in bold in the table.

EPOCH	1	2	3	4	5	6	7	8	9	10
-------	---	---	---	---	---	---	---	---	---	----

Accurac	57.06	68.64	68.42	76.51	77.66	77.51	76.56	82.03	81.93	82.13
y	%	%	%	%	%	%	%	%	%	%

11	12	13	14	15	16	17	18	19	20
84.15%	84.93%	85.16%	84.41%	86.57%	85.34%	86.38%	84.27%	86.84%	87.81%

21	22	23	24	25	26	27	28	29	30
85.84%	87.54%	84.47%	88.33%	88.60%	89.78%	87.39%	88.89%	89.06%	89.77%

Table 10 Accuracy per epoch of best model

Running time : 742.690S

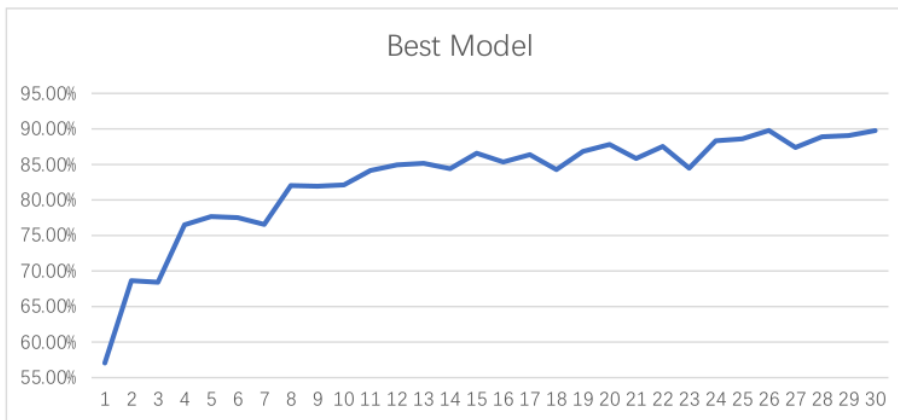


Figure 6 Accuracy per epoch of best model

Discussion

By adjusting the parameters of continuous training, the final conclusions are as follows: In the case of adjusting the EPOCH, the larger the EPOCH, the more rounds of traversing the data set, the better the effect of the model will be. However, as the number of times continues to increase to the critical value, the final loss rate and accuracy of the model will oscillate and converge, which is not that the larger the EPOCH, the better in a certain range. Exceeding the critical range may cause overfitting results to occur. In the case of only adjusting BATCH_SIZE, BATCH_SIZE is the number of batch processing, which is generally raised to the

integer power of 2, and the upper limit of the number size depends on the graphics card memory of the training environment. It can be seen that in the case of 128,256,512, there is no obvious effect on the accuracy of the model, but the training time of the model is affected. The larger BATCH_SIZE, the shorter the time, because more images will be processed at a time, reducing the number of iterations. In the case of adjusting the learning rate, the learning rate of 0.01 is the best, followed by the learning rate of 0.001, and the last is the learning rate of 0.1. It can be seen that the relatively good learning rate requires repeated experiments in the early fitting process, but after the iteration of the training amount in the later stage, the accuracy difference is about 5%.

Conclusion

ResNet18 Pros:

1. Shallow models, shorter training time, and lower computational resource requirements

For some simple image tasks, ResNet18 is sufficient without the need for overly complex models

3. Less parameters, less prone to overfitting

ResNet18 Disadvantages:

1. The model is shallow, the extracted features may not be rich enough, and the performance may not be as good as ResNet50:

For some complex image tasks, ResNet18 may not achieve good performance;

With fewer parameters, you may not fit the data well and your accuracy may be reduced.

Summary of hyperparameter Settings

There are many parameters that need to be adjusted in the neural network, and how to adjust the parameters correctly is very important. The parameters that need to be adjusted are probably as follows:

Number of layers in a neural network.

Number of neurons per layer

How do you initialize the weights and biases?

Which one to choose for the loss function?

Which regularization is to be chosen? L1, L2

How much regularization parameter lambda is appropriate to choose?

How to choose the activation function

Whether to use a dropout or not

How large is the training set?

Choose the size of the mini-batch

What is the correct learning rate?

Which gradient descent algorithm should be chosen?

When to stop Epoch training?

There are more than a dozen parameters that need to be tuned, and following a certain parameter sequence can reduce the amount of work to some extent. Training a neural network takes a long time, so start with the simplest network results.

Start with the simplest network architecture: a neural network with two layers: an input layer and an output layer.

The number of training and validation sets should be small to speed up training, and the other parameters should be fixed arbitrarily.

2. Adjust the learning steps and

If α is too large, this may result in going higher and skipping the local nadir, but if α is too small, the learning rate is too slow. The regularization parameter λ is not set and can start from 0.001, 0.01, 0.1, 1, 10. If the loss starts to increase, stop and make a smaller fine-tuning. How we found that $\alpha = 0.1$ worked best was by trying $\alpha = 0.05$ and 0.5.

If the accuracy is not very good, then $\alpha = 100$. If you find that the accuracy not only increases, but even decreases, it may be that $\alpha=10$ is a bit too large, and then train again with $\alpha=1$ and find a significant improvement. This means that $\alpha = 1$ is more appropriate than 10, and if you try the same method again and again, you will get a good α value.

Choose an appropriate number of epochs for training.



If, as the number of epochs increases, the accuracy changes very little for a certain amount of time (say, 5 to 10 epochs), we can stop the Epoch. You can start by setting the number of epochs higher and see where there is little change in accuracy and set the number accordingly.

4. Adjust the regularization parameter lambda

Experimental lambda, from 1.0, 10, 100... Look for the right one. Note that the size of lambda is proportional to the number of samples. For example, the number of samples and lambda are 1000 and 0.4, respectively.

Adjusting the mini-batch size

Minibatch selection is too small: computational resources are not fully utilized. Too big: Weights and biases are slow to update, the mini-batch size is relatively independent of other parameters, and once a suitable one is found, it does not need to be changed.

6. How to choose a gradient descent algorithm

SGD is fine for beginners,

7. How to choose the excitation function

There is no particularly good way to choose the excitation function and only experimental comparisons can be relied on. The commonly used activation functions are:

8. How do you set the weights and biases?

The simplest way: let W and b follow $N(0, 1 / \sqrt{n_{in}})$, n_{in} : the number of input neurons, setting W and b can speed up the learning rate, and in very few neural networks, W and b can even affect the final training accuracy.

9. Which regularization to choose?

$L1$ and $L2$ are terms added at the end of the cost function. Both $L1$ and $L2$ aim to reduce the weight of the effect of connectivity on cost, but there are some differences between $L1$ and $L2$. $L1$ decreases the weight by a constant amount and $L2$ decreases the weight by a fixed percentage; If the weight itself is large, $L1$ decreases much less than $L2$, and conversely if the weight itself is small, $L1$ decreases more. $L1$ tends to

focus on a small number of important connections.

Note that for L1, the partial derivative of the cost with respect to W , when $W=0$, is meaningless because it is not distinguishable at $W=0$. In this case, the un-regulazied expression is used, $\text{sgn}(0) = 0$. The purpose of regularization is to reduce the weight, and when the weight = 0, there is no need to reduce it.

10. Whether or not to use a dropout

Unlike L1 and L2 regularization, dropout does not add a term to the cost, but changes the outcome of the neural network itself.

11. How big is the training set?

In short, the larger the training set, the better, and increasing the training set also reduces overfitting to some extent.

Machine Learning 2

GRADEMARK REPORT

FINAL GRADE

42/100

GENERAL COMMENTS

Instructor

The abstract should really stand alone.

Don't be afraid to include summary results in your abstract.

You do some good work here. Well done BUT YOU ARE USING THE WRONG DATASET - USE THE FACIAL EXPRESSION DATASET AS SPECIFIED IN CLASS!! IT SITS NETXT TO THE ASSIGNMENT ON MOODLE!!!

You provide a review of work related to relevant CNNs - good. BUT NO REFS specified!! YOU MUST INCLUDE REFS!

Your experiments are well designed in general and you provide lots of results - consider graphs or placing some details in an appendix.

Later your report becomes less structured and turns into a list of notes - relevant and useful but this needs restructuring - did you run out of time?

Mark reflects the errors/omissions highlighted above.

PAGE 1

PAGE 2



Comment 1

not 48x48??

PAGE 3



Comment 2

include ref to original figure



Comment 3

cite original in figure title



Comment 4

Use early stopping with a validation set.

GRADING FORM: SMU GENERIC_GJ

XINZHE LYU

0

RESEARCH-INFORMED LITERATURE

KNOWLEDGE AND UNDERSTANDING

ANALYSIS

PRACTICAL APPLICATION AND DEP.

SKILLS FOR PROFESSIONAL PRAC.