Report of Advanced Experiments(Machine learning practical Coursework 4)

21/3/2017

Introduction

The current object recognition method always use the machine learning method. Deep neural networks with a large number of parameters are very powerful machine learning systems. Especially the performance of Convolutional Neural Networks is better than the normal one. The capacity of Convolutional Neural Networks can be controlled by varying their depth and breadth, and they also make strong and mostly correct assumptions about the nature of images (namely, stationarity of statistics and locality of pixel dependencies)(Krizhevsky et al, 2012). Thus, compared to standard feedforward neural networks with similarly-sized layers, CNNs have much fewer connections and parameters and so they are easier to train. In my coursework 3, I have tried to explore how different the performance of neural networks which use different kinds of model and technologies to recognize the image. But actually the performance of normal neural networks is not good enough. The best accuracy on validation dataset of CIFAR-100 is only 0.26. To learn about thousands of objects from millions of images, we need a model with a large learning capacity. Hence in the following task, I will use Convolutional Neural Networks. And I will investigate following research question:

1. Could the Convolutional Neural Networks improve the performance against the last coursework's neural networks?

2. How different kernel size of Convolutional layer and feature maps of Convolutional layers influence the effect of train neural networks?

At the meantime, overfitting is a serious problem in such networks (Srivastava et al, 2014). Large networks are also slow to use, making it difficult to deal with overfitting by combining the predictions of many different large neural nets at test time. The size of our network made overfitting a significant problem. And In order to prevent overfitting, I will explore several technologies which may help model to address overfitting. So there is a research question I need to investigate:

3. How different method which address overfitting influence the training of Convolutional Neural Networks?

As for data which I used, I train the model on CIFAR-100 mainly. And I implement the baseline model on CIFAR-10.

They are labelled subsets of the much larger 80 million tiny images with 6000 images per class for an overall dataset size of 60000. Each image has three (RGB) colour channels and pixel dimension 32×32, corresponding to a total dimension per input image of 3×32×32=3072. For each colour channel the input values have been normalised to the range [0, 1]. CIFAR-100 has images of identical dimensions to CIFAR-10 but rather than 10 classes they are instead split across 100 fine-grained classes. Both CIFAR-10 and CIFAR-100 have standard splited into a training set of 40,000 images, a validation set of 10,000 images and a test set of 10,000 images. References.

In order to investigate the research question above, I will implement a baseline which will use CNN and compare it with model which I inferred in coursework 3 at first. Secondly, I will change the size of kernel of Convolutional layer and the feature maps of it. I use them to investigate the different performance between baseline and themselves. Then I will investigate the method to address overfitting. I will implement several technologies to address overfitting and observe the result.

The Baseline and the Architecture

In the coursework 3, I implemented different activation function(ReLU, sigmoid, tanh), different network architecture(depths and widths), different regularisation methods(L1 penalty, L2 penalty and dropout) and different learning rate schedules by using TensorFlow. And I choose one(with 2 fully connected layers of neural network and 500 units inside and each layer using ReLU as activation function and using constant learning rate scheduler of momentum adptive learning method with the stat learning rate of 0.001 and momentum of 0.9 and without regularization) to compare with the experiments which I will implement following. And the statistics are shown in below Table 1:

	Final acc(valid)	Final error(valid)
Baseline	0.26	5.12

Table 1

From the table above, we could find that the accuracy of the baseline is not good enough, because the model of baseline is not enough complex and it has trend to overfit. So there will be a comparison with the following experiments which will add convolutional layers and change the size of kernel and feature maps. Also will add some technologies to address overfitting.

Convolutional Neural Networks

METHODS

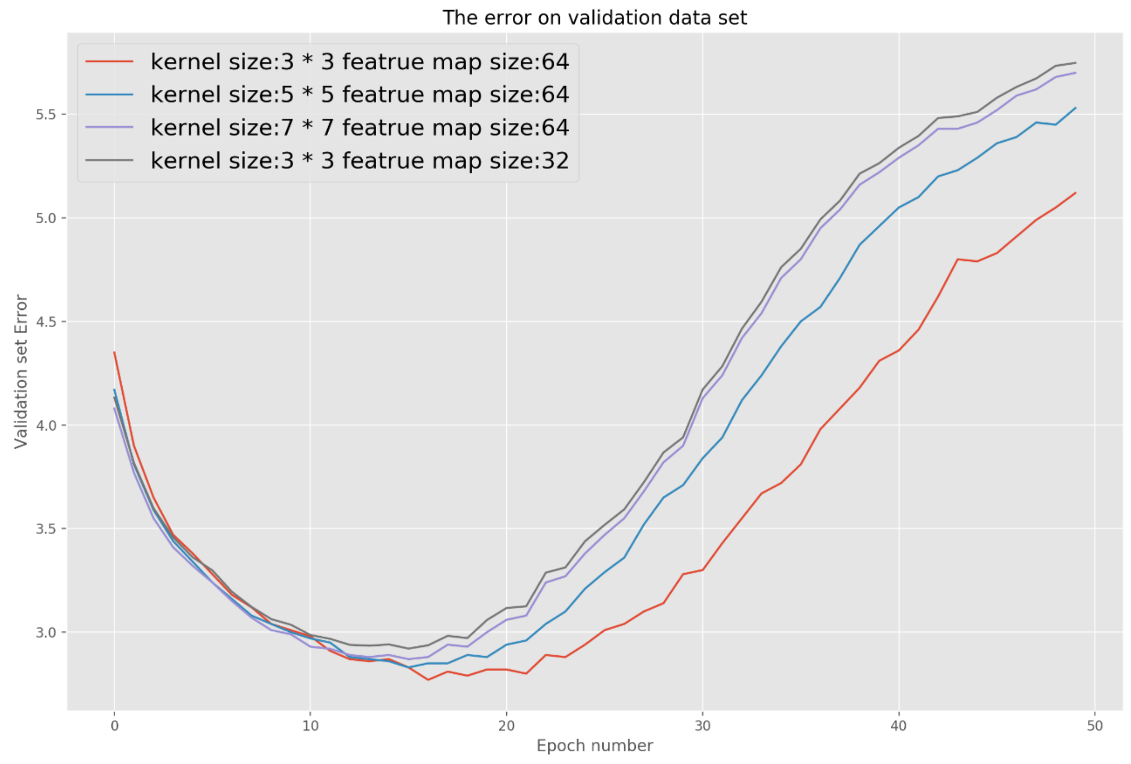
Because Convolutional networks have recently enjoyed a great success in large-scale image and video recognition (Simonyan et al, 2014), we use the convolutional neural network to investigate how better it performs than the normal one. To measure the improvement brought by changing setting of CNN, all our convolutional layer configurations are designed using the different kernel size and feature maps size, inspired by Simonyan et al. (2014). It is because that there are many hyperparameters we need to considerate and it is hard to design experiments to all these hyperparameters. So I just focus on the kernel size and feature map size.

In order to investigate the research question: Could the Convolutional Neural Networks improve the performance against the last coursework's neural networks? And how different kernel size of Convolutional layer and different feature maps of Convolutional layers influence the effect of train neural networks? I implement Convolutional Neural Networks by add convolutional layers and vary the kernel size and feature maps size. He(2016) uses the kernel size of 3×3 and the number of feature maps are {16, 32, 64} in his deep network; Krizhevsky(2009) sets the kernel size to 5×5 and the number of feature maps to 64. So I decided to vary the kernel size from 3 * 3 to 7 * 7 and size of feature map {32, 64}. In other word, I design 5 experiment:{3*3*64; 5*5*64; 7*7*64; 3*3*32}. And the second implement will become the new baseline of following experiment.

According to He et al' s(2016) work and tutorial on tensorflow, I design the architecture of CNN like following. There is the first convolutional layer with the different kernel size and then it is followed by a max pooling layer which is used to reduce the dimension by half. And then there is a normalization after pooling layer. After this convolutional layer, I add another same convolutional layer, max pooling layer, and a normalization. After these two convolutional layer, I add two fully connected layer which is implement in coursework 3. Then I define the error function to softmax cross entropy with logits(using tf.nn.softmax_entropy_with_logits) and calculate the accuracy by reduce mean function(tf.reduce_mean) and use the Momentum as the learning method which set the learning rate to 0.001 and momentum to 0.9 (tf.train.MomentumOptimizer).

I implement above by using tensorflow. I achieve the define a _variable_with_weght_decay and _variable_on_cpu function to define the kernel and calculate the biases. I calculate the conv by tf.nn.conv2d and tf.nn.bias_add function and using tf.nn.relu to activate conv. I use tf.nn.lrn to normalize the output of convolution layer(set alpha = 0.001 / 9.0 and beta = 0.75) and use tf.nn.max_pooling to make a pooling layer. And I use the fully_connected_layer function which I implement in coursework 3 to build fully connected layer.

RESULTS AND DISCUSSION



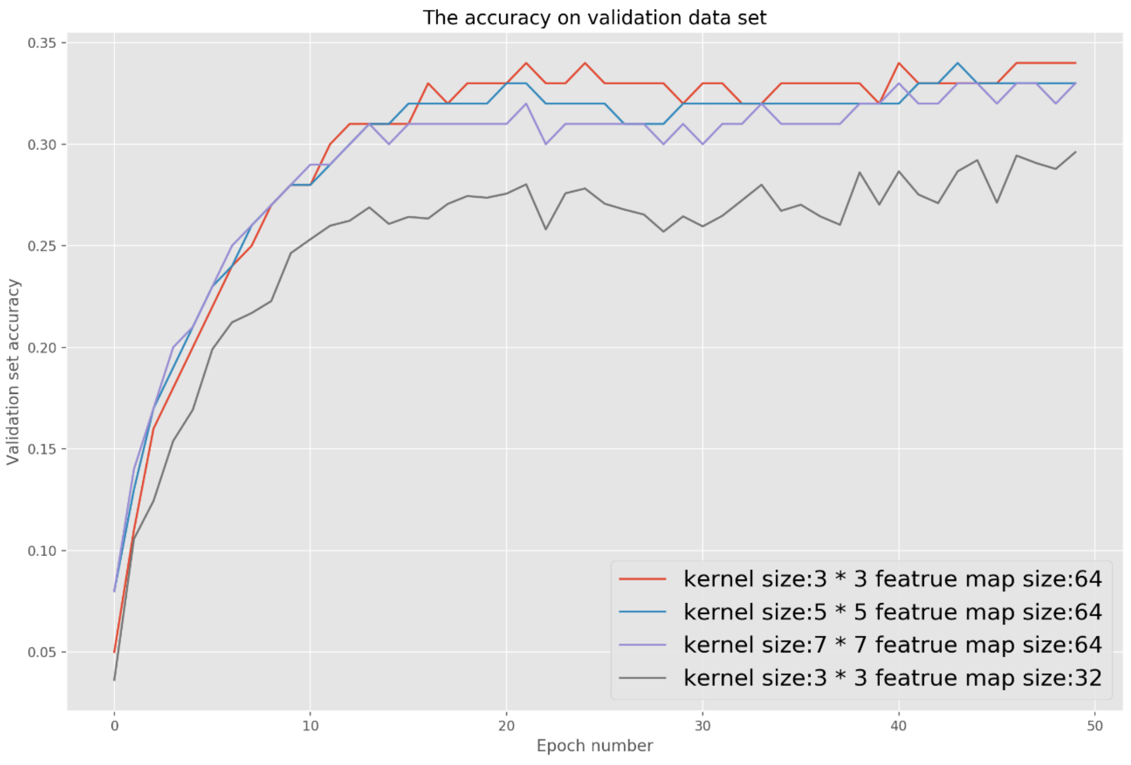
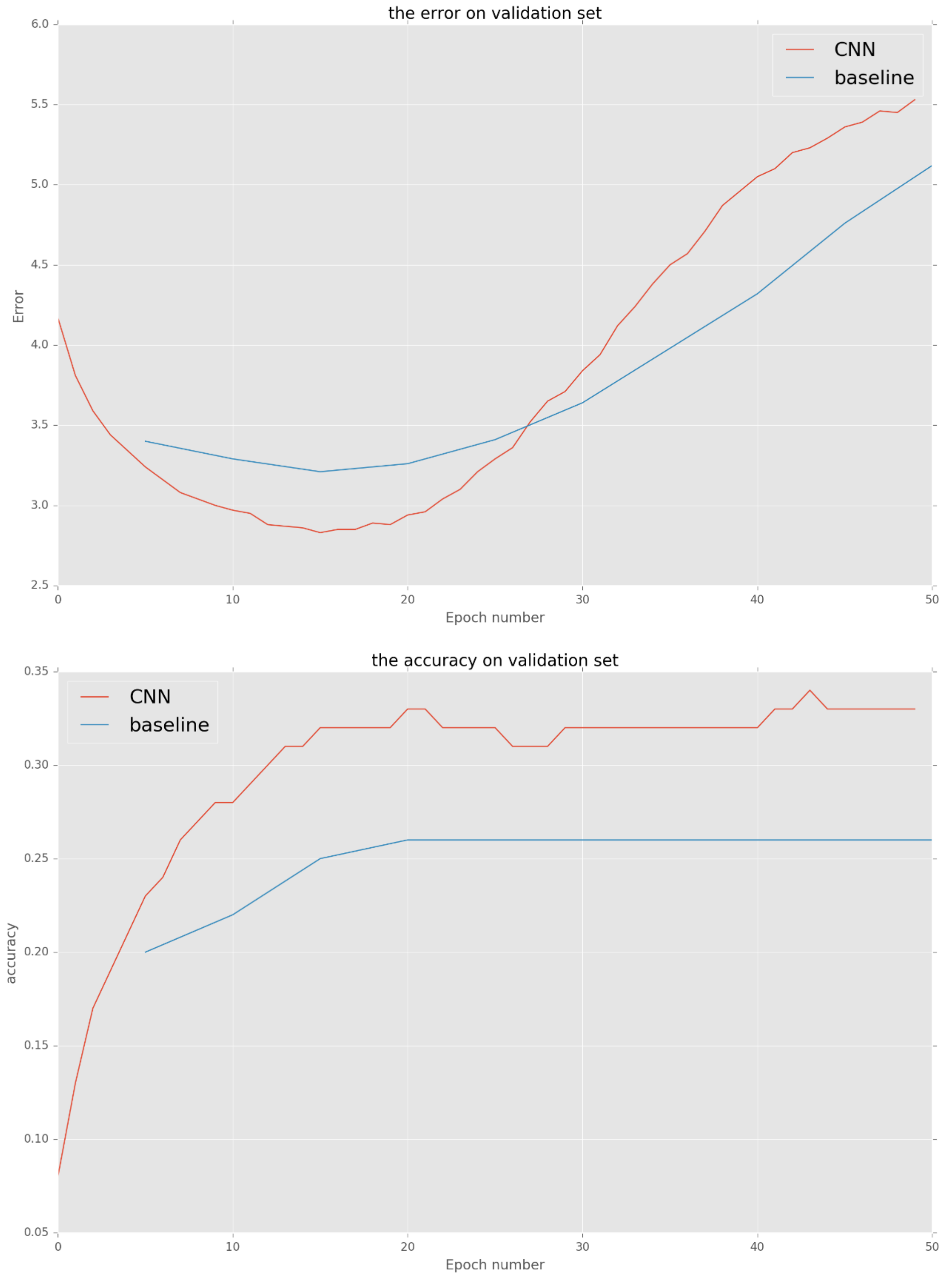


Figure 1



the error on validation set
Error
Epoch number
CNN
baseline
the accuracy on validation set
accuracy
Epoch number
CNN
baseline

Figure 2

No	Kernel and feature map size	Final valid Acc.	Final valid Err.
1	3 * 3; 64	0.34	5.12
2	5 * 5; 64	0.33	5.53
3	7 * 7; 64	0.33	5.70
4	3 * 3; 32	0.296	5.75

Table 2

1. From the Table 1 and Table 2 we could easily find that the final validation accuracy is obviously increasing from 0.26 to 0.34 when we use convolutional neural networks with 3*3*64. But the final validation error is almost the same. That means that even if we use the convolutional layer, there is still trend of overfitting.

2. Also from the Table 1 and Table 2 we could easily find that when we increase the size of kernel or decrease the size of feature map, the final validation accuracy will decrease and final validation error will increase. It means that the performance of final result is worse and the overfitting is aggravated when we increase the size of kernel or decrease the size of feature map. From Figure 1, we also could find the trend of what I am saying above. The error is increasing apparently as the accuracy rate has been not changing much.

3. From the Figure 2 we could find that although using convolutional layers could improve the performance of training, there is still overfitting and looks even more serious.

Addressing Overfitting

METHODS

Our neural network architecture has a large of parameters. Although the 100 classes of CIFAR-100 make each training example impose of constraint on the mapping from image to label, this turns out to be insufficient to learn so many parameters without considerable overfitting. Below, we describe the three primary ways in which we combat overfitting. We can collect larger datasets, learn more powerful models, and use better techniques for preventing overfitting.

Dropout

Dropout is a technique for addressing this problem. The key idea is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much. During training, dropout samples from an exponential number of different “thinned” networks. At test time, it is easy to approximate the effect of averaging the predictions of all these thinned networks by simply using a single unthinned network that has smaller weights. This significantly reduces overfitting and gives major improvements over other regularization methods. We show that dropout improves the performance of neural networks on supervised learning tasks in vision, speech recognition, document classification and computational biology, obtaining state-of-the-art results on many benchmark data sets (Srivastava, 2014). And I use dropout in fully connected layers.

The choice of which units to drop is random. In the simplest case, each unit is retained with a fixed probability p independent of other units, where p can be chosen using a validation set or can simply be set at 0.5, which seems to be close to optimal for a wide range of networks and tasks. For the input units, however, the optimal probability of retention is usually closer to 1 than to 0.5. So I choose to drop 0.8 of total data set.

I implement dropout by using tf.nn.dropout(nonlinearity(tf.matmul(inputs, weights) + biases), keep_prob) in fully_connected_layer function.

L1 and L2 Regularisation

One method for trying to reduce overfitting is therefore to try to decrease the flexibility of the model. We can do this by simply reducing the number of free parameters in the model (e.g. by using a shallower model with fewer layers or layers with smaller dimensionality). A common method for doing this is to add an additional term to the objective function being minimised during training which penalises some measure of the complexity of a model as a function of the model parameters. So there is two regularization way L1 and L2 regularisation. The first, L1 regularization, uses a penalty term which encourages the sum of the absolute values of the parameters to be small. The second, L2 regularization, encourages the sum of the squares of the parameters to be small. It has frequently been observed that L1 regularization in many models causes many parameters to equal zero, so that the parameter vector is sparse(Ng, A.Y., 2004). And according to the coursework 2, I set the L1 parameters to 1e-5, and L2 parameters to 0.0001.

I implement the L2 regularisation in fully_connected_layer function by everytime when updating the weight add a weight decay using tf.multiply(tf.nn.l2_loss(weights), wd, name='weight_loss') function.

Data augmentation

The easiest and most common method to reduce overfitting on image data is to artificially enlarge the dataset using label-preserving transformation(Krizhevsky et al, 2012). We employ three distinct forms of data augmentation, all of which allow transformed images to be produced from the original images with very little computation, so the transformed images do not need to be stored on disk. We explore three data augmentation strategies. The first strategy is to use no augmentation. The second strategy is to use flip augmentation, mirroring images randomly about the yaxis producing two samples from each image. The third strategy is using rotation. The forth strategy which inspired by Simonyan(2014), termed C+F augmentation, combines cropping and flipping. For CNN-based representations, the image is downsized so that the smallest dimension is equal to 36 * 36 pixels. Then randomly crops are extracted 31 * 31 from the four corners and the centre of the image. Note that the crops are sampled from the whole image, rather than its 31 * 31 centre, as done by. These crops are then randomly flipped about the y-axis, producing 10 perturbed samples per input image. The same crops are extracted, but at the original image resolution(Ahmed, 2015). So I design three more experiment which using three kind of data augmentation, dropout and L2 regularision and compare with the experiment 1 model which has 5 * 5 kernel size and 64 feature maps CNN without data augmentation.

I implement these three new method by using following function:

1. Just using tf.image.random_flip_left_right(img)

2. Just using , tf.image.rot90()

3. Using code following:

    reshape_input_1 = tf.map_fn(lambda img:tf.transpose(img),reshape_inputs)

    result = tf.map_fn(lambda img: tf.image.resize_image_with_crop_or_pad(img, 36, 36), reshape_input_1)

    result_1 = tf.map_fn(lambda img: tf.image.random_flip_left_right(img), result)

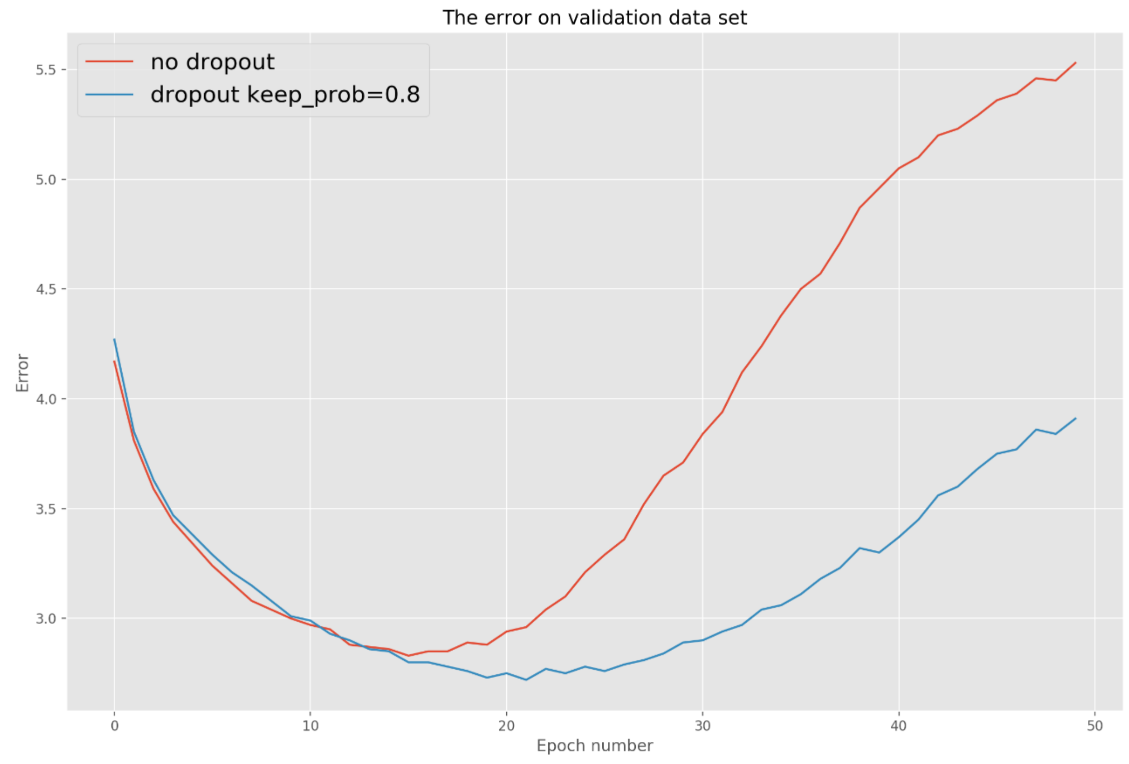
    result_2 = tf.map_fn(lambda img: tf.image.crop_to_bounding_box(img, 0, 0, 32, 32) , result_1)

I design more 6 experiments shown following

No	Kernel and feature map size	Drop out	Regularision	Data arguments
2 (Baseline)	5 * 5; 64	no	no	no
5	5 * 5; 64	yes	no	no
6	5 * 3; 64	yes	L1 with 1e-5 parameter	no
7	5 * 5; 64	yes	L2 with 0.0001 parameter	no
8	5 * 5; 64	yes	L2 with 0.0001 parameter	Method 1
9	5 * 5; 64	yes	L2 with 0.0001 parameter	Method 2
10	5* 5; 64	yes	L2 with 0.0001 parameter	Method 3
11 (On CIFAR-10)	5* 5; 64	yes	L2 with 0.0001 parameter	Method 3

Table 3

RESULTS AND DISCUSSION



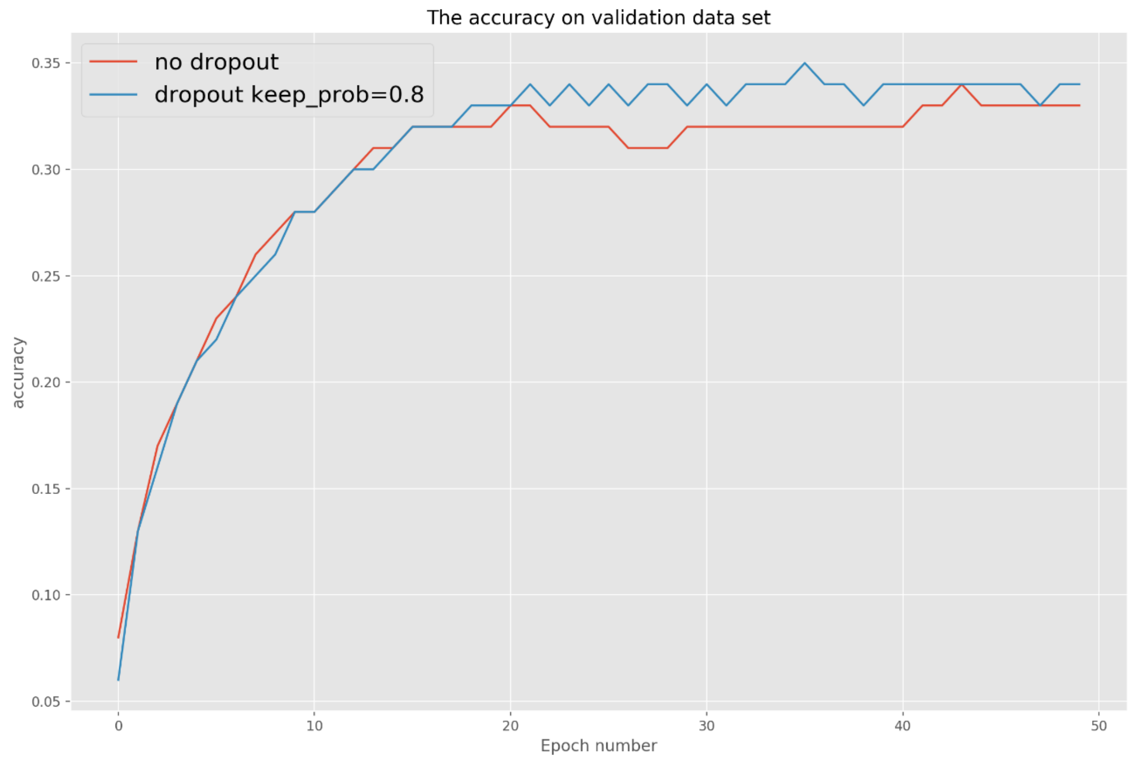


Figure 3

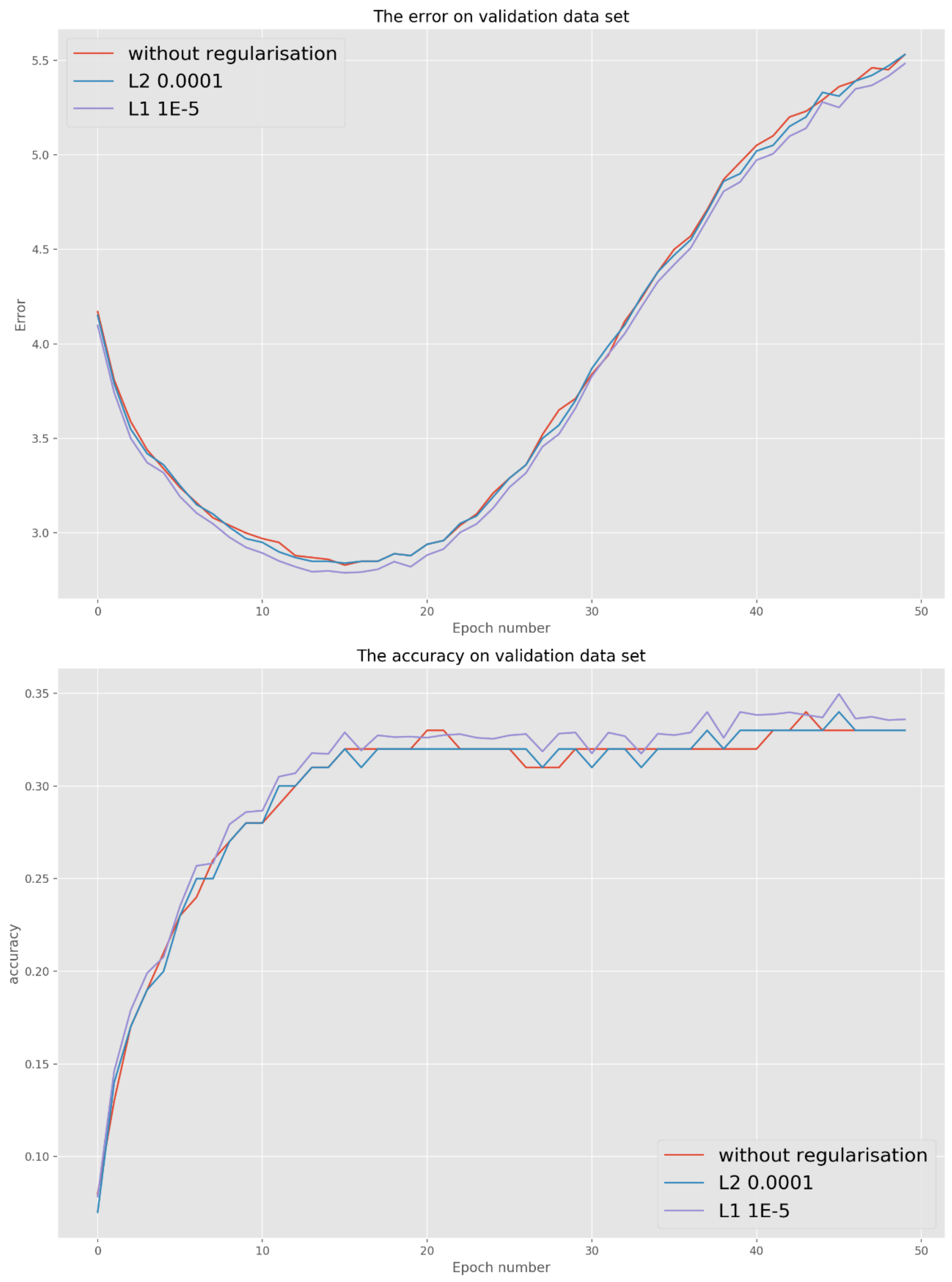

The error on validation data set
without regularisation
L2 0.0001
L1 1E-5
Error
5.5
5.0
4.5
4.0
3.5
3.0
0
10
20
30
40
50
Epoch number
The accuracy on validation data set
accuracy
0.35
0.30
0.25
0.20
0.15
0.10
0
10
20
30
40
50
Epoch number
without regularisation
L2 0.0001
L1 1E-5

Figure 4

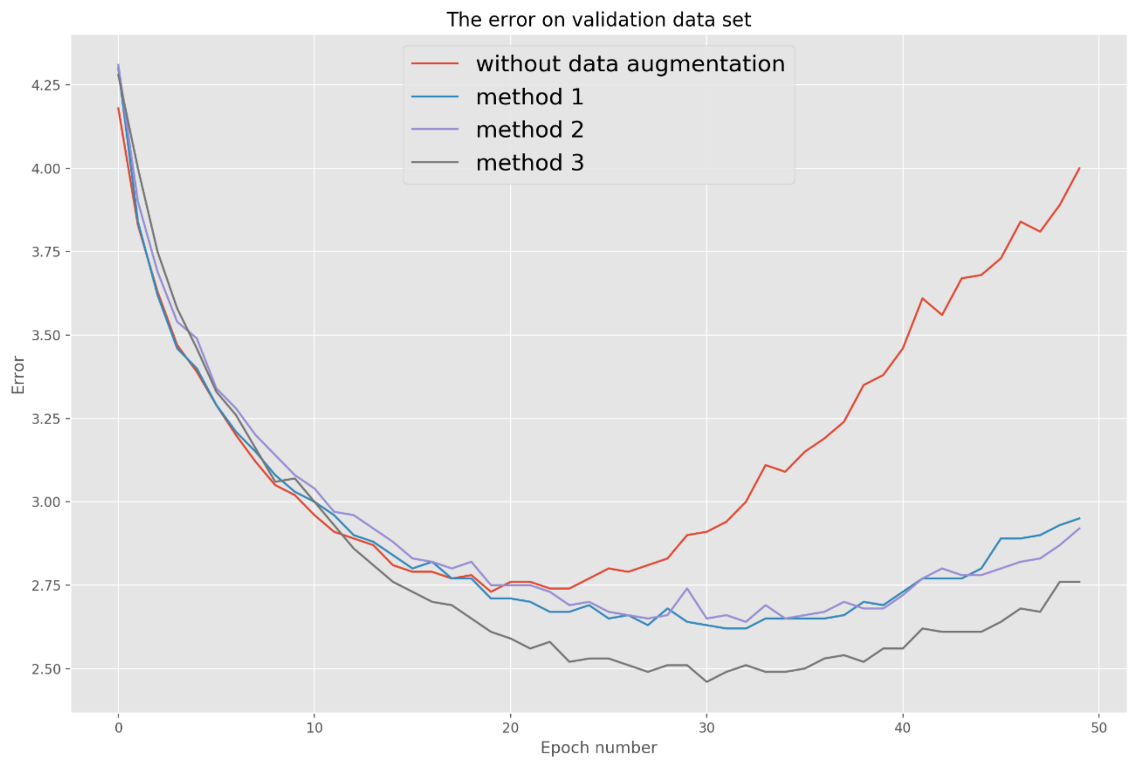
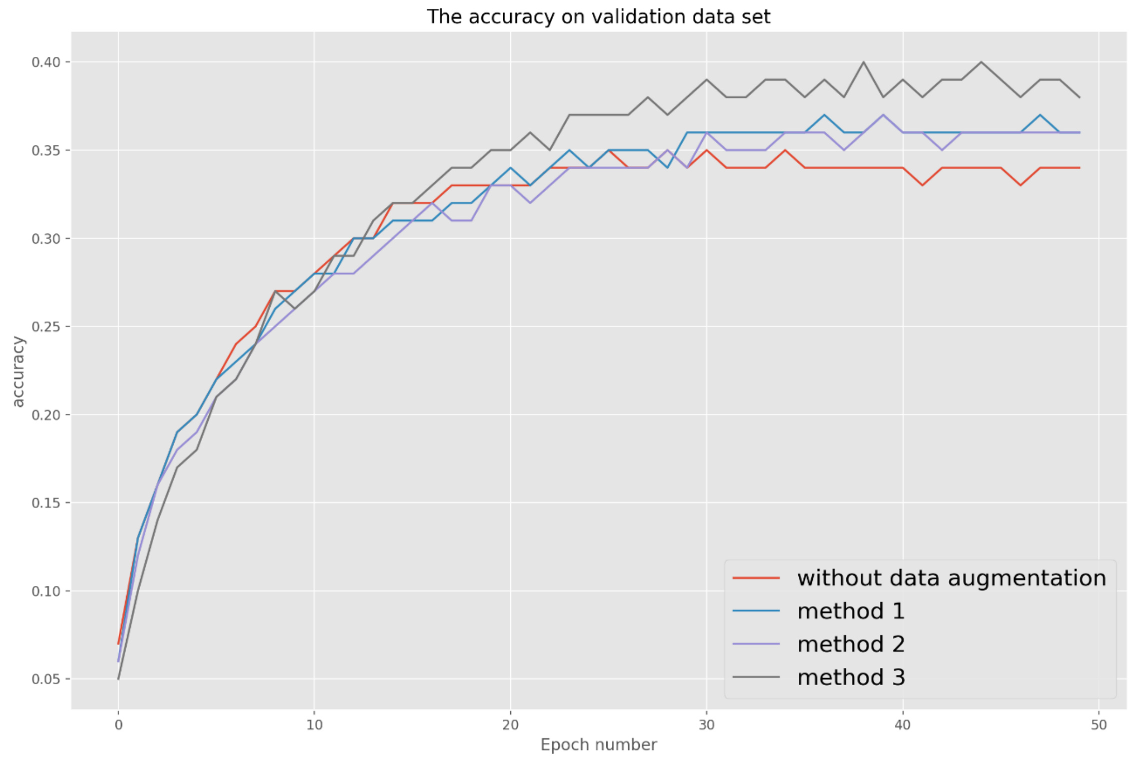

The error on validation data set
without data augmentation
method 1
method 2
method 3
Error
4.25
4.00
3.75
3.50
3.25
3.00
2.75
2.50
0
10
20
30
40
50
Epoch number

The accuracy on validation data set
accuracy
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
without data augmentation
method 1
method 2
method 3
0
10
20
30
40
50
Epoch number

Figure 5

No	Type	Final valid Acc.	Final valid Err.
2	5 * 5; 64 with nothing	0.33	5.53
5	Only dropout	0.34	3.91
6	With L1 1E-5	0.34	5.56
7	With L2 0.0001	0.33	5.53
8	With method 1 of data argument	0.36	2.95
9	With method 2	0.36	2.92
10	With method 3	0.38	2.76
11	On CIFAR-10	0.69	0.98

Table 4

Using the Table 3 and Table 4 we could easily compare the different experiments which just vary one aspect, for example we could use experiment 2 and 5 to explore the effect of using dropout and use experiment 2, 6, 7 to explore the different regularization method and use experiment 2, 8, 9, 10 to explore the different data augmentation method.

1. From the figure 3 and tables above we could see that, when using dropout the performance of training is better and the it apparently addressed the overfitting. The final valid Error is decrease obviously from 5.53 to 3.91. And from the curve, there is obviously reduce the overfitting even if there is still slight overfitting.

2. From the figure 4 and tables above we could find that, when using L1, L2 regularization there is just slightly reducing of overfitting(overall no use) against without such regularization. And it looks like the L1 regularisation performs a liitle bit better than L2 whatever on reducing overfitting or on the performance of training.

3. And according to Srivastava(2014) the method of dropout uses less time than other regularization method. (Because of the cluster is changing the performance at different specific time, I was not record the running time of each epoch as previous coursework)

4. Frome the figure 5 and tables above we could easily find that using data augmentation is a better way to reduce overfitting. It performs very well. And it even performs better than dropout on reducing overfitting and increasing accuracy.

5. There are some differences in using different ways, but these differences are not so obvious. But the third method is better than using some of the methods alone.

6. According to experiment 11, we could get a good accuracy and error on CIFAR-10.

Conclusions

At first our results shows that using convolution neural networks could perform better than using normal multiple layer neural networks obviously. That is because convolution neural networks is more complex and the capability of learning is better than the traditional neural networks. But there is still the problem of overfitting.

Secondly from the exploration above, we could find that when we increase the size of kernel or decrease the size of feature map, the final validation accuracy will decrease and final validation error will increase. Bur the improvement is not such obviously. At the meantime the overfitting is aggravated apparently when we increase the size of kernel or decrease the size of feature map. So on CIFAR-100 we always use 3*3 or 5*5 as kernel size and 64 as feature maps size.

Finally in these three kind of method to reducing overfitting, the data augmentation performances the best whatever on reducing overfitting or increasing accuracy. And the mixed method of data augmentation is better than using them individually. And I think there is no necessary using L1 and L2 regularisation. It not only increases the consumption of calculation, but also is not much help on reducing the fit.

In this time of coursework, because of the limited time and computing resources, I did not implement some more powerful architecture of deep neural networks. According to He's(2015) work, deep residual network could generate very good results on both CIFAR-10 and CIFAR-100 datasets. We could also follow their experiments to have a try on this complex neural networks. Also we could explore how to reduce overfitting on residual network.

References

Srivastava N, Hinton G E, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting[J]. Journal of Machine Learning Research, 2014, 15(1): 1929-1958.

Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]//Advances in neural information processing systems. 2012: 1097-1105.

Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.

He, Kaiming, et al. "Identity mappings in deep residual networks." European Conference on Computer Vision. Springer International Publishing, 2016

Ahmed, E., Jones, M. and Marks, T.K., 2015. An improved deep learning architecture for person re-identification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3908-3916).

Ng, A.Y., 2004, July. Feature selection, L 1 vs. L 2 regularization, and rotational invariance. In Proceedings of the twenty-first international conference on Machine learning (p. 78). ACM.