Report for EQ2425 Analysis and Search of Visual Data

EQ2425, Project 3

Mengyu Liang
lmengyu@kth.se

Tingting Li
tinl@kth.se

Jiacheng Xu
jiacxu@kth.se

October 13, 2024

Summary

In this report, we build and train a three-layer convolutional neural network for recognition of image. Then we modify several relevant parameters of the network structure and the training. Based on the performance, we finally choose the preferred configurations: number of filters [64,128,256], filter size [7*7,5*5,3*3], Leaky Relu, with Dropout, with batch normalization layer, but without extra fully connected layer. We also chose some hyperparameters based on the experiment: higher learning rates have a high recall in the early training stage and Data shuffling can slightly increase the final recall rate.

1 Introduction

Convolutional Neural Networks (CNNs) have become a cornerstone in the field of deep learning, particularly in image recognition and classification tasks. The architecture of CNNs, which consists of multiple layers, enables them to automatically learn hierarchical features from data. However, the performance of a CNN model heavily relies on careful tuning of both the network structure and the training settings.

This report aims to investigate the impact of various architectural and training parameters on the performance of a three-layer CNN model. Starting from a predefined configuration, we will systematically adjust the network's structural parameters and the training configurations to explore how these factors influence the model's effectiveness.

2 Problem Description

This project involves building and training a three-layer convolutional neural network (CNN) to solve a classification task. The initial network will be constructed following a predefined configuration, after which the architecture and training process will be systematically modified to evaluate the impact of different parameters on performance.

The primary objective is to determine the optimal settings for the network by experimenting with various configurations and comparing their performance. The parameters to be investigated are grouped into two categories: network structure and training settings.

Network Structure Parameters:

1. **Number of layers vs. Number of filters**: Evaluate the effect of altering the depth of the network and the number of convolutional filters in each layer.
2. **Filter size**: Investigate how changing the convolutional kernel size impacts model performance.
3. **ReLU vs. Leaky ReLU**: Compare the use of the traditional ReLU activation function with the Leaky ReLU variant to observe differences in gradient propagation and model training.
4. **Dropout vs. without Dropout**: Test the impact of dropout on reducing overfitting by randomly omitting neurons during training.
5. **Batch Normalization Layer**: Analyze how adding batch normalization influences the convergence speed and model generalization.

Training Settings:

1. **Batch size**: Test how different batch sizes affect training stability and model performance.
2. **Learning rate**: Adjust the learning rate to find the optimal balance between fast convergence and stable training.
3. **Data shuffling**: Compare the impact of shuffling the training data between epochs on the model's ability to generalize.

The outcome of these experiments will help identify a set of parameters that yields the best performance for the task at hand, enabling a deeper understanding of how different architectural and training decisions affect CNNs.

3 Results

3.1 Network Structure

3.1.1 Number of layers vs. Number of filters

Increasing the number of layers from three to five led to a notable improvement in the recall rate, as deeper networks can capture more complex features. However, adding too many layers beyond five resulted in diminishing returns, with overfitting becoming more apparent. However, increasing the number of filters per layer can't lower recall rate.

3.1.2 Filter size

The experiments on filter size showed that larger filters allowed the network to capture broader spatial features, which improved performance in early layers. However, too large filters reduced the model's ability to capture fine details. A filter size of 7x7, 5x5, 3x3 provided a good balance between performance and computational efficiency.

3.1.3 ReLu vs. Leaky ReLu

The comparison between ReLU and Leaky ReLU activation functions indicated that while ReLU performed well in most cases, the Leaky ReLU variant provided more stable training and faster convergence, especially in deeper networks. Leaky ReLU helped mitigate the "dying ReLU" problem by allowing a small gradient for negative values, which prevented neurons from becoming inactive.

3.1.4 Dropout vs. without Dropout

Applying dropout during training significantly reduced overfitting, especially when training with a larger number of layers. The network with dropout achieved better generalization on the test set, with a slight trade-off in terms of slower convergence during training. Without dropout, the model quickly fit the training data but performed worse on the validation set, indicating overfitting.

3.1.5 Batch Normalization Layer

Incorporating batch normalization layers led to a noticeable improvement in training speed and convergence. Models with batch normalization required fewer epochs to reach similar accuracy compared to models without it. Moreover, batch normalization helped stabilize the learning process, reducing sensitivity to initialization and improving generalization.

3.2 Training Settings

3.2.1 Batch size

The experiments on batch size demonstrated that smaller batch sizes (e.g., 64) led to noisier updates during training but improved generalization on the test set. Larger batch sizes (e.g., 256) provided smoother training curves. A batch size of 64 to 256 provided the best trade-off between convergence stability and test set performance.

3.2.2 Learning rate

The learning rate played a crucial role in determining the training dynamics. A higher learning rate (e.g., 0.01) resulted in faster initial convergence but often caused instability or overshooting during training. Conversely, a lower learning rate (e.g., 0.0001) led to slower convergence but ensured stable, gradual improvement. An optimal learning rate of 0.001 provided a good balance between speed and accuracy.

3.2.3 Data shuffleing

Data shuffling during training had a positive impact on model performance, particularly in preventing the model from becoming biased towards certain patterns in the data order. Shuffling the data in each epoch led to better generalization and more consistent results across different runs. Without shuffling, the model occasionally struggled to generalize well, especially with larger datasets.

3.3 Tabular Result

We control variables to get the optimal Net setting. Compared to the baseline, every net has one change, they are: Net_A1: more filters on convolutional layers, Net_A2: extra fully connection, Net_B: larger filter size, Net_C: use LeakyRelu instead of Relu, Net_D: add Drop out layer, Net_E: add Batch normalization.

Worth mentioning, that we have an early stop technique that uses a validation data set. Within 20 epochs, if there's no increase in the accuracy of the validation data set, we then stop training. With the early stop, the recall rate is not as high as the full 300 epochs, but it prevents overfitting and reduces a lot of training time.

	Baseline	4.A.1	4.A.2	4.B	4.C	4.D	4.E
Batch size	64	64	64	64	64	64	64
Data shuffling	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
Conv2d CL 1	24 5x5	**64** 5x5	24 5x5	24 **7x7**	24 5x5	24 5x5	24 5x5
Conv2d CL 2	48 3x3	**128** 3x3	48 3x3	48 **5x5**	48 3x3	48 3x3	48 3x3
Conv2d CL 3	96 3x3	**256** 3x3	96 3x3	96 3x3	96 3x3	96 3x3	96 3x3
FC NL 1	512 Relu	512 Relu	512 Relu	512 Relu	**512 LeakyRelu**	512 Relu	512 Relu
FC NL 2	X	X	**128 ReLU**	X	X	X	X
FC NL 3	10 Softmax	10 Softmax	10 Softmax	10 Softmax	10 Softmax	10 Softmax	10 Softmax
Dropout	FALSE	FALSE	FALSE	FALSE	FALSE	**TRUE**	False
Batch Normalization	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	**TRUE**
Learning rate	0.001	0.001	0.001	0.001	0.001	0.001	0.001
Average Top-1 Recall(%)	62.05	67.47	19.4	65.49	63.35	65.85	67.73

Table 1: part 4 of the model comparison table.

Then, we apply the optimal parameter we get from part 4, as the baseline of part 5. In part 5, we compare the effect of Batch size, Learning rate, and Data shuffling. We find Data shuffling and bigger learning rate can increase recall rate.

	Baseline	5.A	5.B	5.C
Batch size	64	**256**	64	64
Data shuffling	FALSE	FALSE	FALSE	**TRUE**
Conv2d CL 1	64 5x5	64 5x5	64 5x5	64 5x5
Conv2d CL 2	128 3x3	128 3x3	128 3x3	128 3x3
Conv2d CL 3	256 3x3	256 3x3	256 3x3	256 3x3
FC NL 1	512 LeakyRelu	512 LeakyRelu	512 LeakyRelu	512 LeakyRelu
FC NL 2	X	X	X	X
FC NL 3	10 Softmax	10 Softmax	10 Softmax	10 Softmax
Dropout	TRUE	TRUE	TRUE	TRUE
Batch Normalization Layer	TRUE	TRUE	TRUE	TRUE
Learning rate	0.1	0.001	**0.1**	0.001
Average Top-1 Recall Rate (%)	69.7	68.91	74.1	70.8
Time (s)	2725 (125 epoch)	37525 (238 epoch)	X	X

Table 2: part 5 of the model comparison table.

4 Conclusions

Based on our experimental results, the optimal configuration for image classification using the CIFAR-10 dataset is achieved with model Net_E, which attains the highest average Top-1 recall rate of 67.73%. This model utilizes convolutional layers with 24, 48, and 96 filters, ReLU activation functions, and fully connected layers without the inclusion of dropout or batch normalization. Future work could explore further fine-tuning of these parameters and investigate the impact of additional regularization techniques to enhance the model's robustness.

Appendix

Who Did What

- Jiacheng wrote the summary and part of the result.
- Tingting wrote the introduction, problem description, and part of querying.
- Mengyu offered emotional support and wrote the rest of the report.