**COMP5623 Coursework on Image Classification and Visualizations with Convolutional Neural Networks – ImageNet10**

|  |  |
| --- | --- |
| Name | **Kunhao Liang** |
| Student username & ID | **mm20kl / 201469850** |

**QUESTION I [55 marks]**

**1.1 Single-batch training [16 marks]**

|  |
| --- |
| 1.1.1. Display graph 1.1.1 (training & validation loss over training epochs) and briefly explain what is happening and why. [4 marks] |

图表, 折线图

描述已自动生成

The training loss does not change during the whole training process, and the validation loss fluctuates at a high value. This indicates that the model is underfitting. In other words, it is too simple to learn the complicated input features.

|  |
| --- |
| 1.1.2 Display graph 1.1.2 (training & validation loss over training epochs, with modified architecture) and explain how and why it shows that the model is overfitting the training batch. [8 marks] |

图表, 折线图

描述已自动生成

As it can be seen from the graph, the training loss gradually decreases over epochs, while the validation loss increases dramatically after the fifth epoch. It indicates that the model can perform well on the training set, but it has poor generalization to other data. Therefore, the model is overfitting.

|  |
| --- |
| 1.1.3 Fill in table 1.1.3 (your adjusted architecture after single-batch training), adding rows and columns as necessary. [4 marks] |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input channels | Output channels | Layer type | Kernel size | Padding | Stride |
| 3\*128\*128 | 32\*128\*128 | convolutional layer | 3 x 3 | 1 | 1 |
| 32\*128\*128 | 32\*64\*64 | pooling layer | 2 x 2 | / | 2 |
| 32\*64\*64 | 64\*64\*64 | convolutional layer | 3 x 3 | 1 | 1 |
| 64\*64\*64 | 64\*32\*32 | pooling layer | 2 x 2 | / | 2 |
| 64\*32\*32 | 128\*32\*32 | convolutional layer | 3 x 3 | 1 | 1 |
| 128\*32\*32 | 128\*16\*16 | Pooling layer | 2 x 2 | / | 2 |
| 128\*16\*16 | 256\*8\*8 | Convolutional layer | 3 x 3 | 1 | 1 |
| 256\*8\*8 | 256\*8\*8 | Pooling layer | 2 x 2 | / | 2 |
| 256\*8\*8 | 256 | Fully-connected layer | / | / | / |
| 256 | 64 | Fully-connected layer | / | / | / |
| 64 | 10 | Fully-connected layer | / | / | / |

**1.2 Fine-tuning on full dataset [18 marks]**

|  |
| --- |
| 1.2.1 Display graph 1.2.1 and indicate what the optimal number of training epochs is and why. [4 marks] |



From my perspective, the optimal number of training epochs is between 20-30. As it can be seen, after training 30 epochs, the training loss still keep decreasing, but the validation loss increases remarkably with test accuracy fluctuated at 70% until the end of training. This shows that the model is becoming more overfitting. Hence, in order to obtain a relatively good generalization of model, we should stop training at about 30 epochs.

|  |
| --- |
| 1.2.2 Describe in detail your fine-tuning process on the complete dataset, including any adjustments you made to the network or training process to increase prediction accuracy. Explain why these adjustments increased accuracy. [10 marks] |

Adjustment:

* Regularization(Batch Normalization, Dropout)
* Using data augmentation

Batch normalization is added to normalize the convolved features outputted by convolutional layers to take on a similar range of values before being sent to the activation function. It has the effect of regularization, which helps prevent overfitting and improve the generalization of the model. Therefore, the model can have a higher accuracy on the test set. Dropout is also used in the experiment. It randomly eliminates a specific number of neurons of network during training, making a layer with different structures in each training. Therefore, it can improve the robustness of model, further increasing model accuracy. In my experiment, these two regularization methods are both added in the order shown below:

Conv -> Batch Normalization -> Activation Function(ReLU) -> Dropout -> Pooling layer

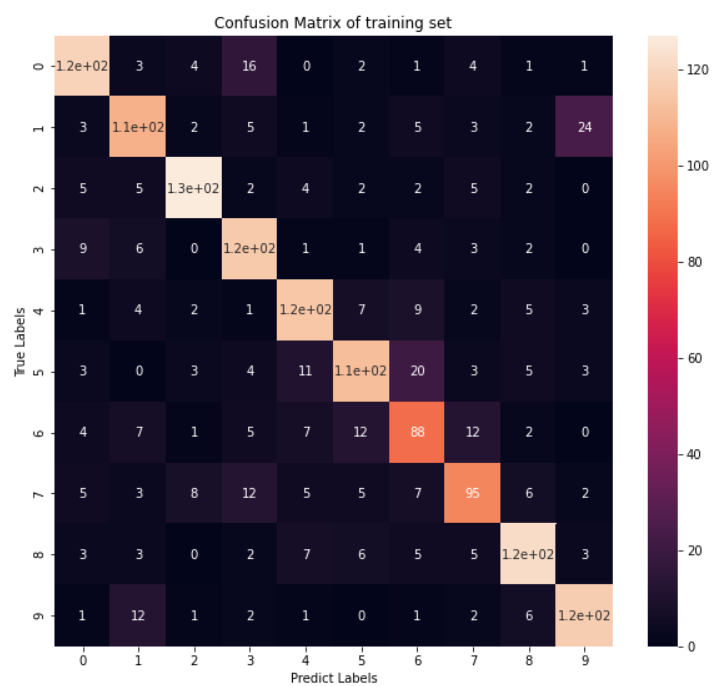
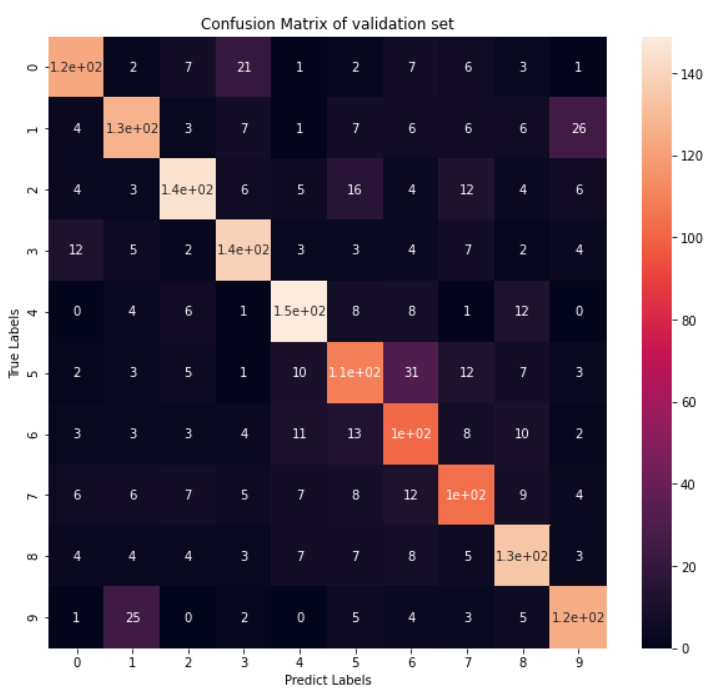
Linear -> Dropout -> Batch Normalization -> ReLU

By using regularization methods, the maximum prediction accuracy increased by 5%(from 62% to 67%).

Then I proposed data augmentation by using ‘RandomHorizontalFlip’ in the torchvision.transform function to flip image horizontally with given probability, and it resulted in a 2%-3% increase in test accuracy. Finally, after training enough epochs, the model can reach a 70% average test accuracy.

|  |
| --- |
| 1.2.3 Display two confusion matrices 1.2.3 (one each for complete validation set and complete training set) for your final trained model and interpret what is shown. [4 marks] |

The confusion matrix for validation set(left) and training set(right) are presented below:



Similar patterns can be seen from these two confusion matrixes. The model is not good at distinguishing objects of label 1 from label 9, as 35 images whose true label is 1 are misclassified as 9. A similar situation also can be seen between label 5 and label 6.

**1.3 Evaluation and code [21 marks]**

|  |
| --- |
| 1.3.1 Please include *[my\_student\_username]\_test\_preds.csv* with your final submission.[8 marks]  1.3.2 Please submit all relevant code you wrote for Question I in Python file *[my\_student\_username]\_q1.py*. No need to include the config or ImageNet10 files. [13 marks] |

*No response needed here.*

**QUESTION II [45 marks]**

**2.1 Preparing the pre-trained network [20 marks]**

|  |
| --- |
| 2.1.1 Read through the provided template code for the AlexNet model *alexnet.py*. What exactly is being loaded in line 59? [2 marks] |

|  |
| --- |
| 2.1.2 Write the code in *explore.py* after line 50 to read in the image specified in the variable args.image\_path and pass it through a single forward pass of the pre-trained AlexNet model. [5 marks]  2.1.3 Fill in function extract\_filter() after line 84 extracting the filters from a given layer of the pre-trained AlexNet. [4 marks]  2.1.4 Fill in function extract\_feature\_maps() after line 105 extracting the feature maps from the convolutional layers of the pre-trained AlexNet. [6 marks]  Please submit all your Question II code in a Python file *[my\_student\_username]\_explore.py*. |

*No response needed here.*

|  |
| --- |
| 2.1.5 Describe in words, not code, how you ensure that your filters and feature maps are pairs; that the feature maps you extract correspond to the given filter. [3 marks] |

In my code, I plotted one filter at each channel, and plotted features maps of all channels. Then, the number of feature maps is the same as the number of filters, and they are shown with the same order. Therefore, each feature map and filter can be paired easily.

**2.2 Visualizations [25 marks]**

|  |
| --- |
| 2.2.1 For three input images of different classes, show three pairs of filters and corresponding feature maps, each from a different layer in AlexNet. Indicate which layers you chose. For each pair, briefly explain what the filter is doing (for example: horizontal edge detection*)* which should be confirmed by the corresponding feature map. [15 marks] |

Image #1, class: \_\_\_Fish\_\_\_\_\_\_\_

|  |  |  |  |
| --- | --- | --- | --- |
|  | Filter | Feature map | Brief explanation |
| Early layer |  |  | Sharpen the image |
| Intermediate layer |  |  | Detect the features at the center of image |
| Deep layer |  |  | Detecting the whole image, and the shape of fish is shown |

Image #2, class: \_\_\_Bird\_\_\_\_\_\_\_\_

|  |  |  |  |
| --- | --- | --- | --- |
|  | Filter | Feature map | Brief explanation |
| Early layer |  |  | Edge detection |
| Intermediate layer |  |  |  |
| Deep layer |  |  | Detect the features at the right bottom corner |

Image #3, class: \_\_\_Cat\_\_\_\_\_\_\_

|  |  |  |  |
| --- | --- | --- | --- |
|  | Filter | Feature map | Brief explanation |
| Early layer |  |  | Edge detection |
| Intermediate layer |  |  | Detecting the features at the center of image |
| Deep layer |  |  | Detecting the whole image, and the shape of cat is shown |

|  |
| --- |
| 2.2.2 Comment on how the filters and feature maps change with depth into the network. [5 marks] |

|  |
| --- |
| Marks reserved for overall quality of report. [5 marks] |

*No response needed here.*