COMP5623 Coursework on Image Classification with Convolutional Neural Networks

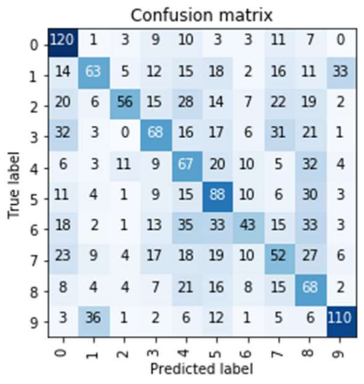
Fanhui Meng (sc19fm) 201373470

1. Present the results of the network architecture experiments and comment on these results, presenting confusion matrices.

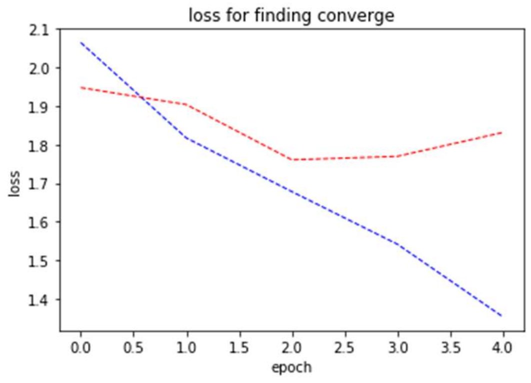
1. 2 layers:

The graph below on the right hand side is to show that I have train the network until they converge. The red line is the valid loss, blue line is the train loss.

The left hand side is the confusion matrix.


Confusion matrix

True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	120	1	3	9	10	3	3	11	7	0
1	14	63	5	12	15	18	2	16	11	33
2	20	6	56	15	28	14	7	22	19	2
3	32	3	0	68	16	17	6	31	21	1
4	6	3	11	9	67	20	10	5	32	4
5	11	4	1	9	15	88	10	6	30	3
6	18	2	1	13	35	33	43	15	33	3
7	23	9	4	17	18	19	10	52	27	6
8	8	4	4	7	21	16	8	15	68	2
9	3	36	1	2	6	12	1	5	6	110



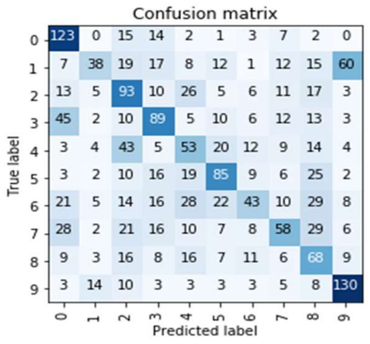
The accuracy as follow:

```
Accuracy of the network on the test images: 40 %
```

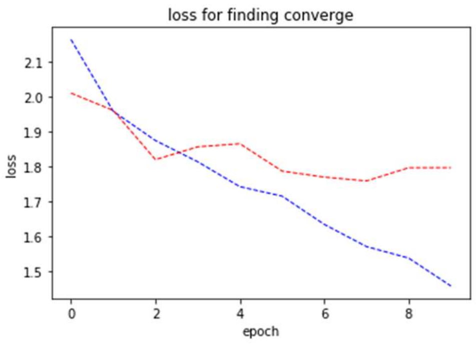
2. 3 layers:

The graph on the right hand side shows the same meaning as above experiment.

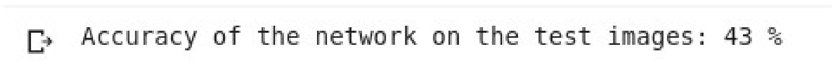
Left is the confusion matrix.


Confusion matrix

True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	123	0	15	14	2	1	3	7	2	0
1	7	38	19	17	8	12	1	12	15	60
2	13	5	93	10	26	5	6	11	17	3
3	45	2	10	89	5	10	6	12	13	3
4	3	4	43	5	53	20	12	9	14	4
5	3	2	10	16	19	85	9	6	25	2
6	21	5	14	16	28	22	43	10	29	8
7	28	2	21	16	10	7	8	58	29	6
8	9	3	16	8	16	7	11	6	68	9
9	3	14	10	3	3	3	3	5	8	130



The accuracy as follow:

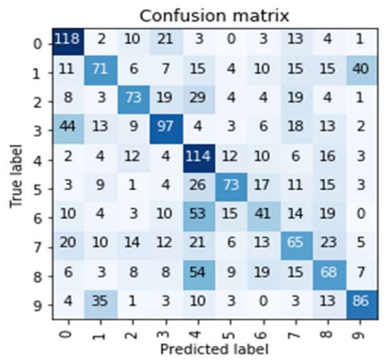


```
Accuracy of the network on the test images: 43 %
```

3. 4 layers:

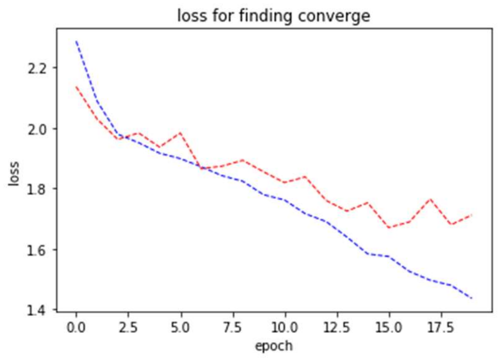
The graph on the right hand side shows the same meaning as above experiment.

Left is the confusion matrix.

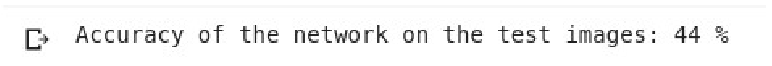


Confusion matrix

True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	118	2	10	21	3	0	3	13	4	1
1	11	71	6	7	15	4	10	15	15	40
2	8	3	73	19	29	4	4	19	4	1
3	44	13	9	97	4	3	6	18	13	2
4	2	4	12	4	114	12	10	6	16	3
5	3	9	1	4	26	73	17	11	15	3
6	10	4	3	10	53	15	41	14	19	0
7	20	10	14	12	21	6	13	65	23	5
8	6	3	8	8	54	9	19	15	68	7
9	4	35	1	3	10	3	0	3	13	86



The accuracy as follow:

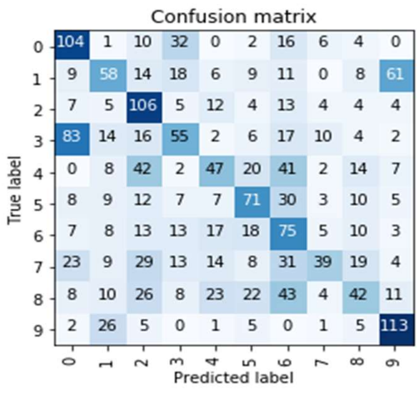


```
Accuracy of the network on the test images: 44 %
```

4. 5 layers:

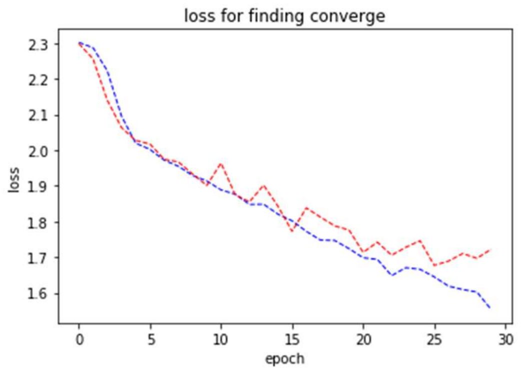
The graph on the right hand side shows the same meaning as above experiment.

Left is the confusion matrix.

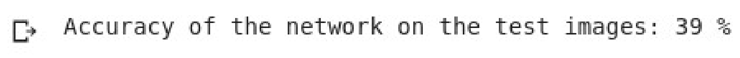


Confusion matrix

True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	104	1	10	32	0	2	16	6	4	0
1	9	58	14	18	6	9	11	0	8	61
2	7	5	106	5	12	4	13	4	4	4
3	83	14	16	55	2	6	17	10	4	2
4	0	8	42	2	47	20	41	2	14	7
5	8	9	12	7	7	71	30	3	10	5
6	7	8	13	13	17	18	75	5	10	3
7	23	9	29	13	14	8	31	39	19	4
8	8	10	26	8	23	22	43	4	42	11
9	2	26	5	0	1	5	0	1	5	113



The accuracy as follow:

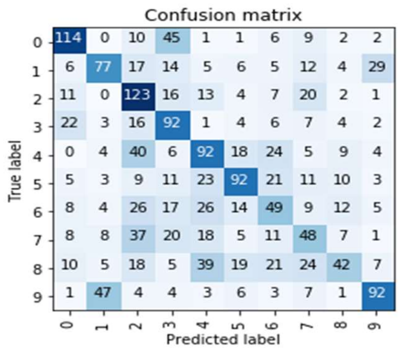


```
Accuracy of the network on the test images: 39 %
```

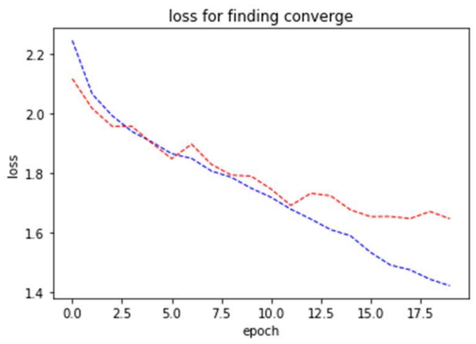
5. 4 layers model (Change the kernel size as 5 X 5)

The graph on the right hand side shows the same meaning as above experiment.

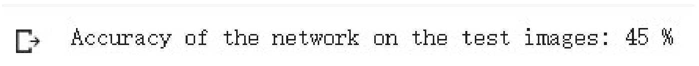
Left is the confusion matrix.



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	114	0	10	45	1	1	6	9	2	2
1	6	77	17	14	5	6	5	12	4	29
2	11	0	123	16	13	4	7	20	2	1
3	22	3	16	92	1	4	6	7	4	2
4	0	4	40	6	92	18	24	5	9	4
5	5	3	9	11	23	92	21	11	10	3
6	8	4	26	17	26	14	49	9	12	5
7	8	8	37	20	18	5	11	48	7	1
8	10	5	18	5	39	19	21	24	42	7
9	1	47	4	4	3	6	3	7	1	92



The accuracy as follow:



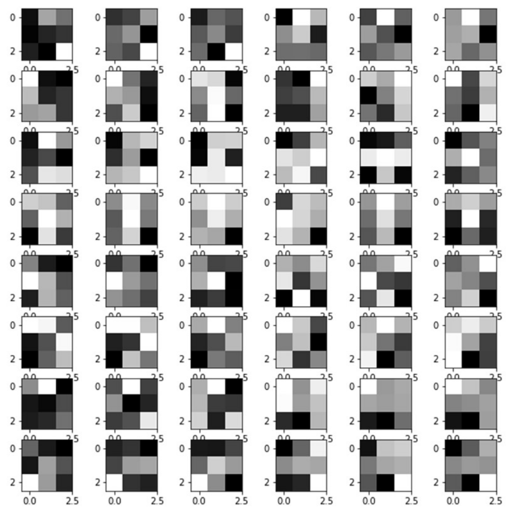
```
Accuracy of the network on the test images: 45 %
```

Comment for these result:

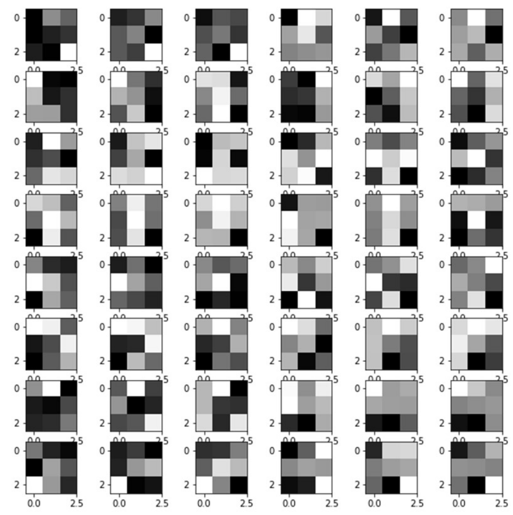
- For the architectures with less layers, they are more likely to converge when training. For example, the architecture with 2 layers converged only after 4 training epoches. As for 5 layers architecture, it converge after nearly 30 training epoches.
- Compare with different architectures' accuracy, I found that more layers, more accurate is incorrect. For example, my best performance architecture is 4 layers rather than 5 layers.
- I add up all the labels and predicated labels tensor respectively, then put them into the confusion matrix function. Then use another function to make the matrix more straightforward. As the accuracy goes up, more and more predicated labels are presented in the diagonal line of the matrix.
- As for the same layer architecture with different kernel size. I found that larger kernel size have better performance.

2. Present and discuss the results of the filter visualizations at the three checkpoints.

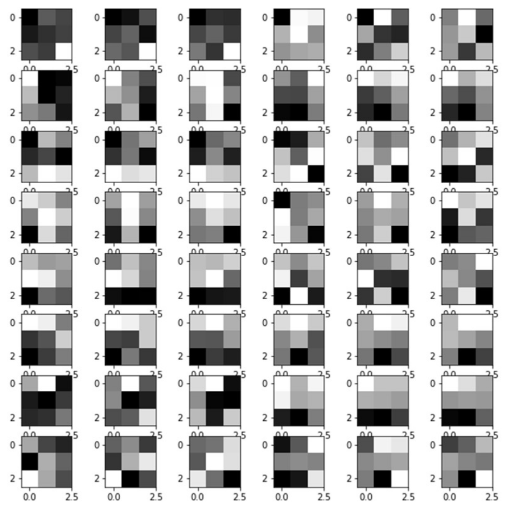
Before training:



Halfway during training



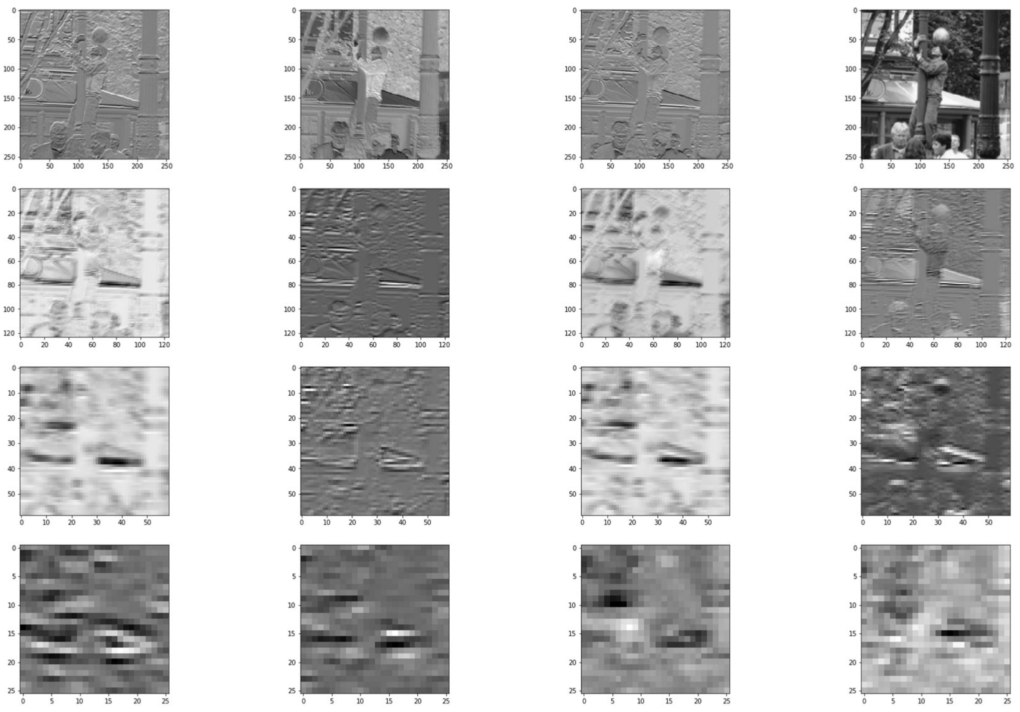
After training:



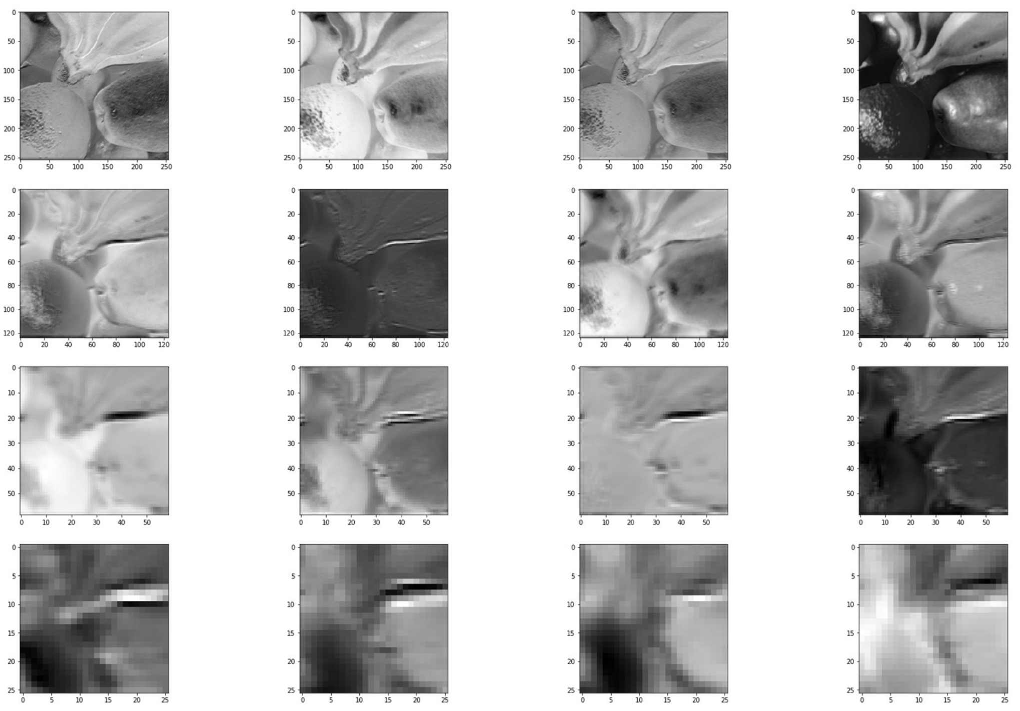
The filters in the half way of training has slight difference when compare with the initial filters. Almost half of the filters have changed. As for the filters after training, almost each one of them have changed. Training is also for adjusting these filters, so that apply the filter into the image can extract more accurate feature.

3. Present and comment on some of the feature maps extracted from the different layers of the final trained network for some selected test images.

The first image's feature maps are shown below:



The feature maps of second image from different class:



As the convolution layer goes deeper, the image’s size as well as it’s resolution ratio are becoming lower. This is because the 2 X 2 with 2 stride max pooling layer would led to a two-fold reduction of the image.

The first few layers’ feature maps are more likely to extract the texture or detail of the image. For example, in the first image, the first layer’s feature maps are able to detect the standing man as well as the ball on his head.

As for deeper layers’ feature maps, they show the shapes, contours or something else that has the strongest features. Although the deeper layers’ feature maps present less features, what they present is the most representative features.

In the same layer, different channels also output different feature maps with different features. For example, in the first layer of the first images, one of the feature map highlight the ball, the other highlight the roof, etc.

4. Describe the two adjustments you made to your network and justify why you believe they will improve the network’s ability to generalise on unseen data, as well as many experiment you have used to validate your approach.

The first thing I did is data augmentation. This also include image uniform scale, horizontal flip, change the image brightness, contrast, etc. Everytime when I did data augmentation, the images would change, so it’s pretty like I add some ‘new data’ into the image set. And use these data to train the neural network. Use this method can improve my neural network’s generalization capability and prevent overfitting problem effectively.

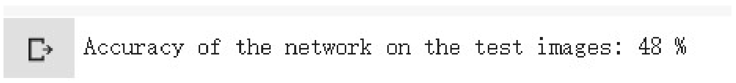
The second thing is to add more fully-connected layers. Increase the depth of the entire neuro network, to get a better performance.

Unfortunately, my method didn’t go well. I have tried many different methods. So far, this is the best method I got.

I have tried to adjust more output for each convolution layer, which means I got more convolution kernels in each layer. I have also tried to change the learning rate. At the very beginning, I gave a larger number to the learning rate to make it quicker to find the optimum. Then, I make it small, so that it can have a better approach to the optimum rather than swing between the best point. I also did adjust the kernel size, the stride of the convolution layer and max pooling layer. But it seems like all of my attempts didn’t work.

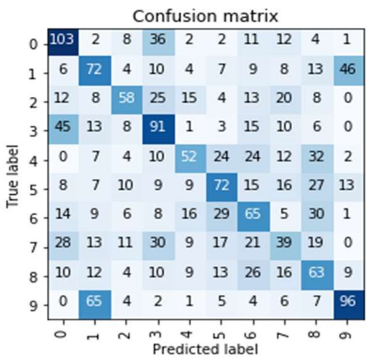
Below is the best performance experiment result that I've got.

The accuracy:

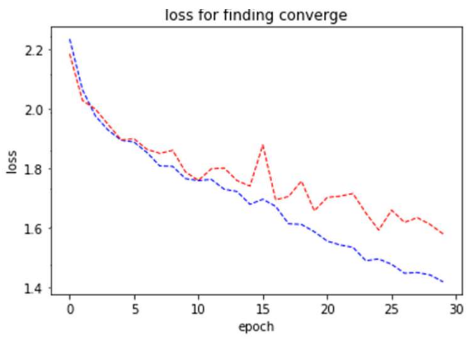


```
Accuracy of the network on the test images: 48 %
```

Confusion matrix and the graph that I have trained the network converge.



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	103	2	8	36	2	2	11	12	4	1
1	6	72	4	10	4	7	9	8	13	46
2	12	8	58	25	15	4	13	20	8	0
3	45	13	8	91	1	3	15	10	6	0
4	0	7	4	10	52	24	24	12	32	2
5	8	7	10	9	9	72	15	16	27	13
6	14	9	6	8	16	29	65	5	30	1
7	28	13	11	30	9	17	21	39	19	0
8	10	12	4	10	9	13	26	16	63	9
9	0	65	4	2	1	5	4	6	7	96


loss for finding converge
loss
epoch