BS6207 Assignement4

CNN Construction

Given this assignment, first I need to build my own CNN model. I did some research of CNN architectures from many papers. And I calculate the output size of each layer targeting our images (128 * 128) based the this function.

- Input: $(N, C_{in}, H_{in}, W_{in})$
- Output: $(N, C_{out}, H_{out}, W_{out})$ where

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor$$

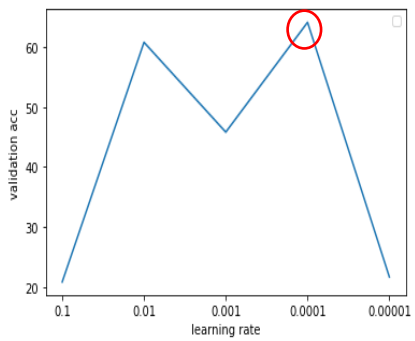
Then I finished my CNN model. The structure of it is shown in the following table.

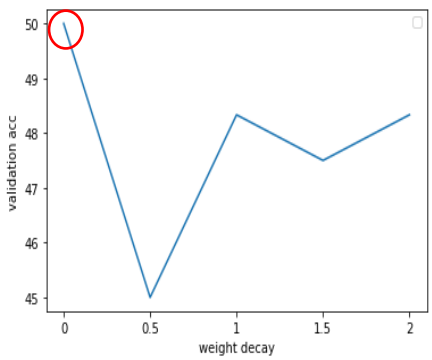
L#	Type	Size	Parameter	Input	output
1	Convolution	16 filters (3 *3)	Stride = 2, padding = 1	3 * 128 * 128	16 * 64 * 64
2	Convolution	32 filters (3 *3)	Stride = 2, padding = 1	16 * 64 * 64	32 * 32 * 32
3	Max pool	2 * 2		32 * 32 * 32	32 * 16 * 16
4	Convolution	64 filters (3 *3)	Stride = 2, padding = 1	32 * 16 * 16	64 * 8 * 8
5	Max pool	2 * 2		64 * 8 * 8	64 * 4 * 4
6	Dropout				
7	Flatten	64 * 4 *4		64 * 4 * 4	1024
8	Dense	(1024, 512)		1024	512
9	Dense	(512, 256)		512	256
10	Dense	(256, 4)		256	4
11	Log SoftMax	(4,1)	dim = 1	4	1

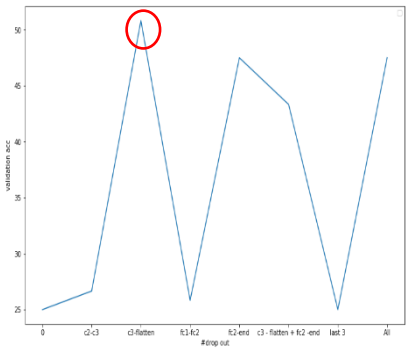
Parameter tuning

I tuned three parameters to improve the performance of CNN model. The evaluation value is validation accuracy. The training size here is 400, 100 from each label.

- Learning rate: The best learning rate is 0.001 among (0.1, 0.001, 0.0001, 0.00001)
- Weight decay: The best Weight decay is 0 among (0, 0.5, 1, 1.5, 2)
- Drop out: I first put the Drop out between each layer. Then I kelp the best two and also all drop out in the last try.
  The best within these combinations is the one between last convolution layer(conv3) and Flatten layer.



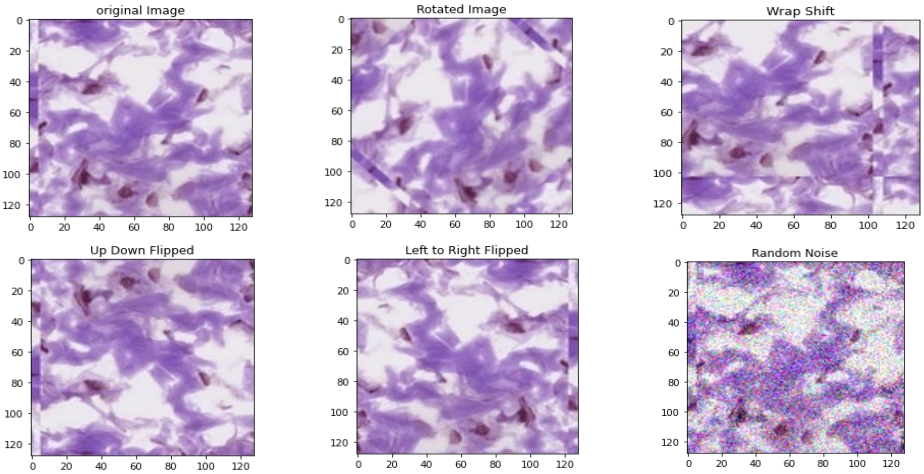




Training size verse test accuracy

- Image augmentation

To deal with classification imbalance, I used rotate, wrap, up to down and left to right reversing and adding noise (5 methods) to do image augmentation. Thus, the smallest date set could be augmented from around 400 to 2000.



- Without image augmentation (small step)

First, I tried the small step of increase of training size without image augmentation. The training size is from 40 to 1200, 40 increasing in each iteration, 10 images from each label. From the result in the left, there is no obvious trend of the increase of training size test accuracy. One of the reasons might be the small size of training size.

- With image augmentation (big step)

Then I applied image augmentation and tried with larger step and dataset. The training size now is from 40 to 8000, 400 increasing in each iteration, 100 images from each label. From the result in the right, in the beginning, the trend is quite normal, and the test accuracy increase from 65 up to about 95. Then there is a sudden drop at the training size around 4500, which is weird. And I could not find the reasonable explanation of it.