



MAMe classification

—based on simple CNN

Lihong Ji

Xiao Fei

Data Preprocessing

The original dataset:

	Image file	Medium	Museum	Museum-based i	Subset	Width	Height	Product si	Aspect ratio
1	436018. jpg	Oil on canvas	Metropolitan Museum of Art	29. 100. 60	train	3144	3840	12072960	0. 8187
2	11779. jpg	Oil on canvas	Metropolitan Museum of Art	1982. 373	train	1707	2136	3646152	0. 7992
3	19022. jpg	Oil on canvas	Metropolitan Museum of Art	2006. 418	train	2845	3811	10842295	0. 7465

The processed dataset:

	img_name	class
1	437446. jpg	0
2	11253. jpg	0
3	10848. jpg	0

Data Preprocessing

The classes of processed dataset are defined as the dictionary at right side.

The original dataset is divided into 3 csv files.

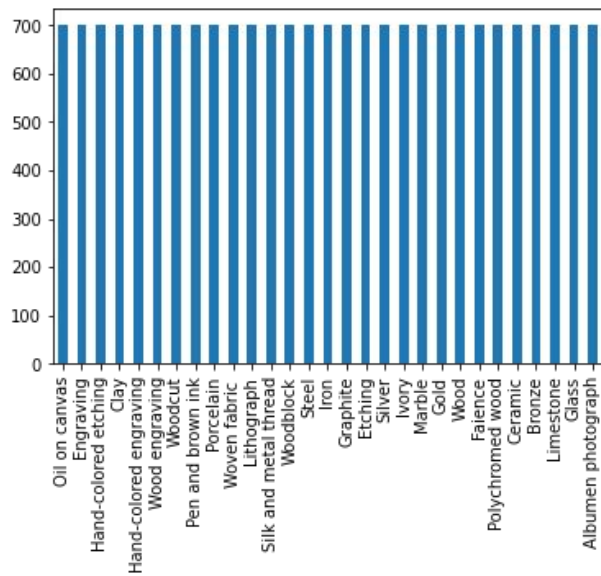
As

1	img_name	class	
2	437446. jpg	0	
3	11253. jpg	0	
4	10848. jpg	0	

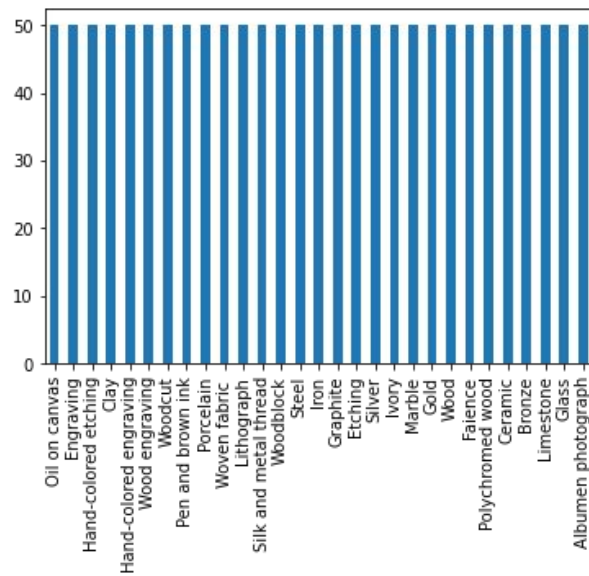
```
{0: 'Oil on canvas',
1: 'Graphite',
2: 'Glass',
3: 'Limestone',
4: 'Bronze',
5: 'Ceramic',
6: 'Polychromed wood',
7: 'Faience',
8: 'Wood',
9: 'Gold',
10: 'Marble',
11: 'Ivory',
12: 'Silver',
13: 'Etching',
14: 'Iron',
15: 'Engraving',
16: 'Steel',
17: 'Woodblock',
18: 'Silk and metal thread',
19: 'Lithograph',
20: 'Woven fabric ',
21: 'Porcelain',
22: 'Pen and brown ink',
23: 'Woodcut',
24: 'Wood engraving',
25: 'Hand-colored engraving',
26: 'Clay',
27: 'Hand-colored etching',
28: 'Albumen photograph'}
```

Class distribution

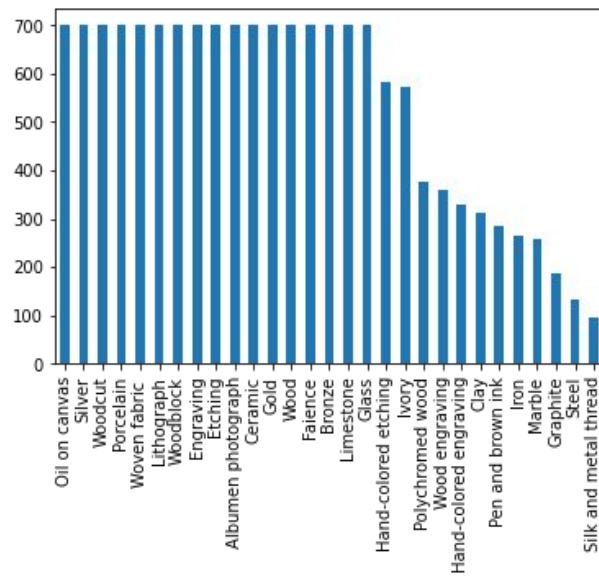
Train



Val



Test





Dataloader based on Pytorch/Tensorflow

Three operations are involved:

1. Rescale all the pixels into the range from 0 to 1.
2. Transform sequence order labels to one hot coding type.
3. Shuffle the images during loading

Some hyperparameters are set up as:

- Learning_rate = $1e-5$,
- Loss_function=Categorical Crossentropy



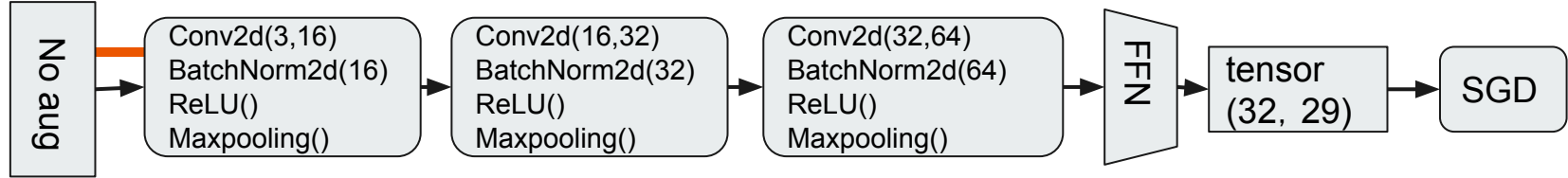
Target parts of the network

Since only simple CNN architecture is concerned, there are 4 aspects:

1. Layer depth (3/5/7)
2. Data augmentation (Yes/No)
3. Optimizer (SGD/Adam)
4. Kernel_size (3/5/7)

1. Optimizer&Data augmentation

3 layers:



Conv2d kernel_size= 3, stride=1, padding=0.

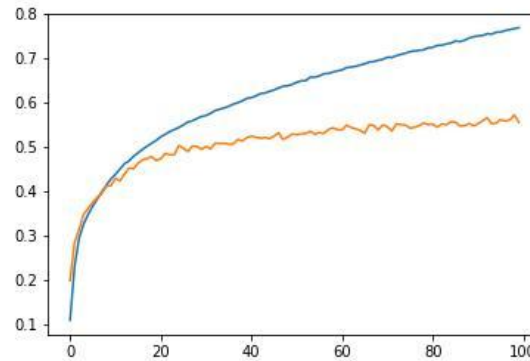
Maxpooling stride=2, stride =2

When :

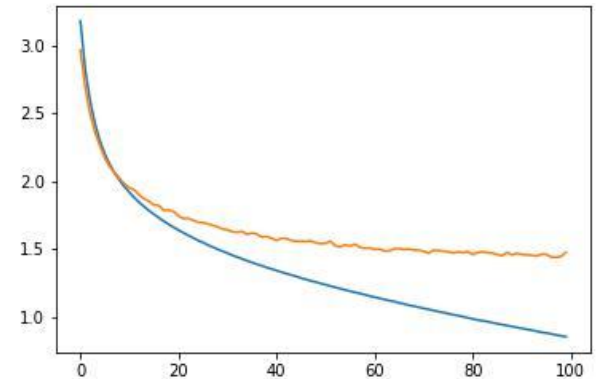
optimizer is **SGD**,

No data augmentation,

The acc on test dataset is **54.83%**

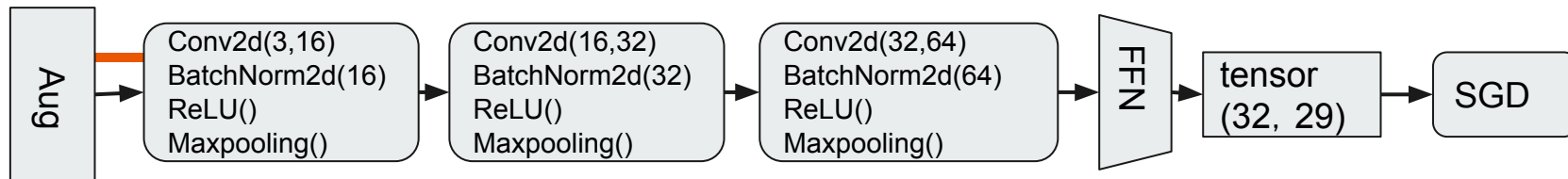


Accuracy



Loss

3 layers:



Conv2d kernel_size= 3, stride=1, padding=0.

Maxpooling stride=2, stride =2

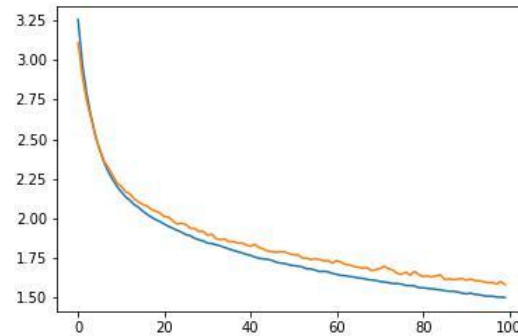
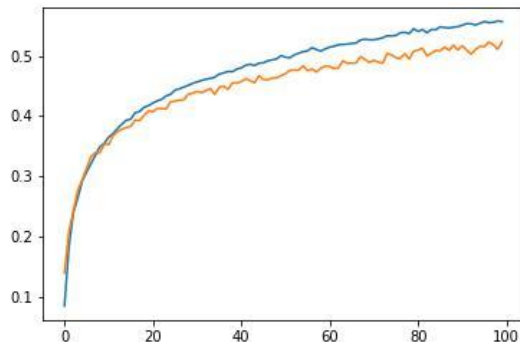
When :

optimizer is **SGD**,

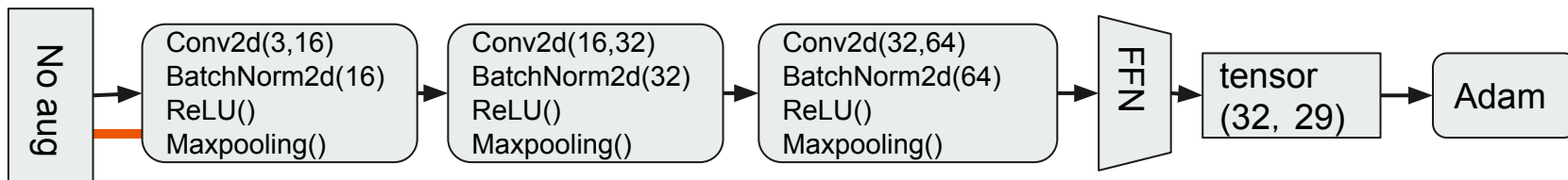
Yes data augmentation,

The acc on test dataset is

51.48%(↓ ≈3%)

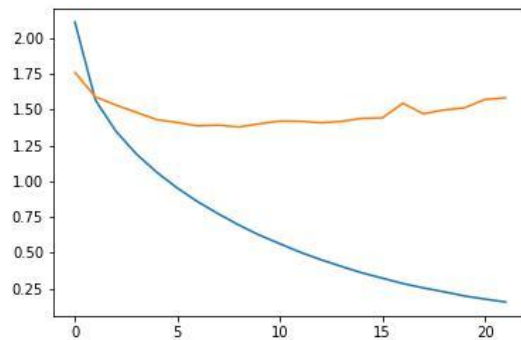
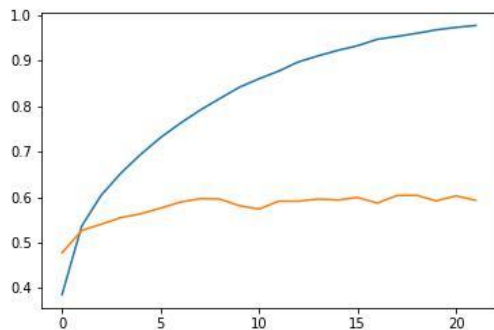


3 layers:



Conv2d kernel_size= 3, stride=1, padding=0.

Maxpooling stride=2, stride =2



(Much faster reaching convergence)

When :

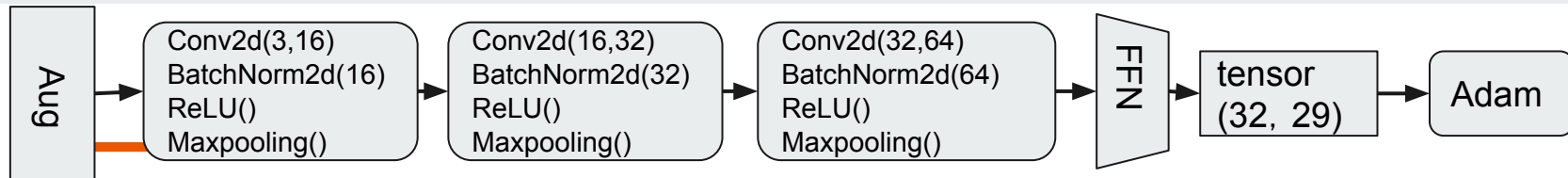
optimizer is **Adam**,

No data augmentation,

The acc on test dataset is

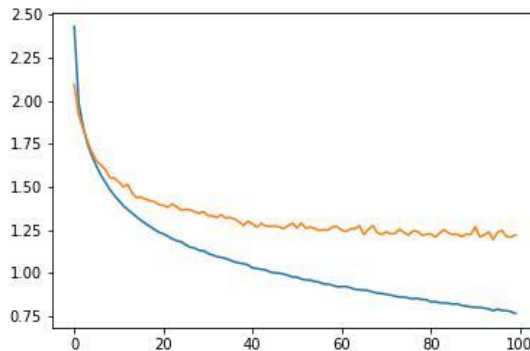
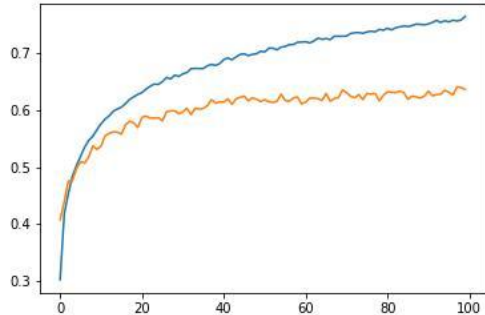
58.17%( $\approx 3\%$)

3 layers:



Conv2d kernel_size= 3, stride=1, padding=0.

Maxpooling stride=2, stride =2



When :

optimizer is **Adam**,

Yes data augmentation,

The acc on test dataset is

64.08%%(↑ ≈ 9%)

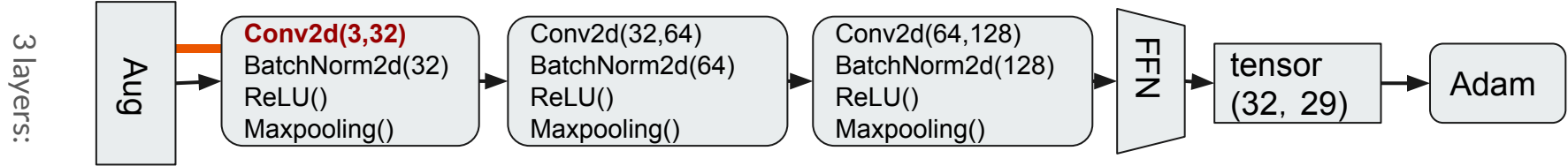


According to the results,

The model with 3 layers (starting with 16 channels), data augmentation and adam as the optimizer reaches the highest accuracy 64.08% on the test dataset.

Hence, the next try is to increase the number of channels of Conv2d while maintaining the rest set up.

2. Convolutional channel



Conv2d kernel_size= 3, stride=1, padding=0.

Maxpooling stride=2, stride =2

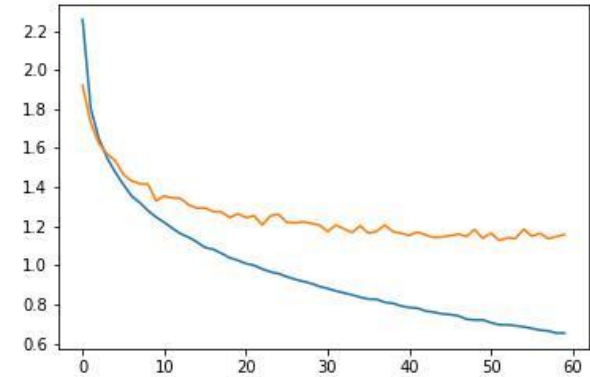
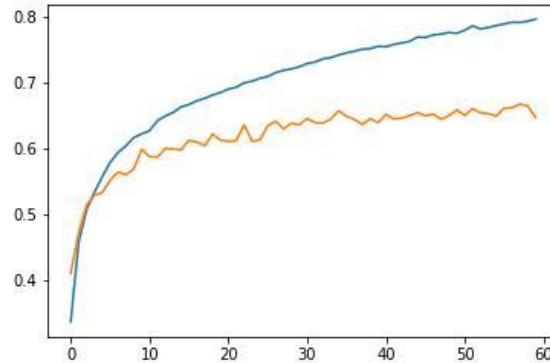
When :

optimizer is **Adam**,

With data augmentation,

The acc on test dataset is

66.32%( $\approx 2.3\%$)





Depending on these results, we decided to try bigger size kernel, because recently one paper released that taking advantage of 31×31 kernel could enhance the performance of downstream tasks, several even better than transformer-based architecture. [1]

[1]Ding X, Zhang X, Han J, et al. Scaling up your kernels to 31×31 : Revisiting large kernel design in cnns[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022: 11963-11975.



3. Network Depth

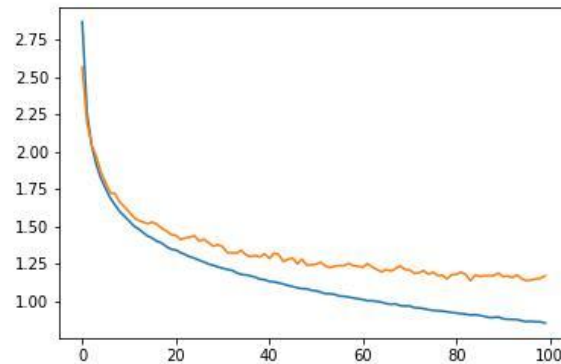
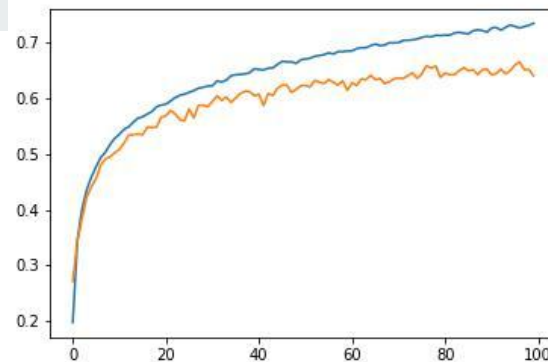
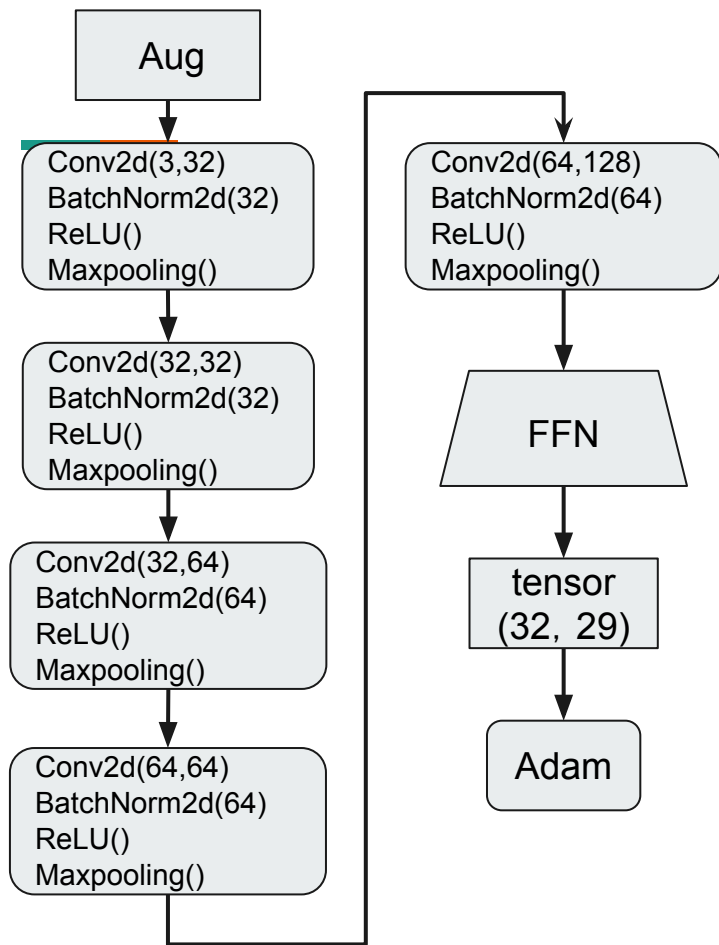
Candidate options for depth are:

3 layers ($3*32 \rightarrow 32*64 \rightarrow 64*128$)

5 layers ($3*32 \rightarrow 32*32 \rightarrow 32*64 \rightarrow 64*64 \rightarrow 64*128$)

7 layers ($3*16 \rightarrow 16*16 \rightarrow 16*32 \rightarrow 32*32 \rightarrow 32*64 \rightarrow 64*64 \rightarrow 64*128$)

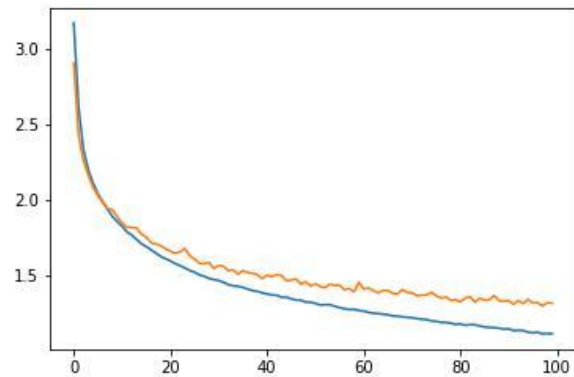
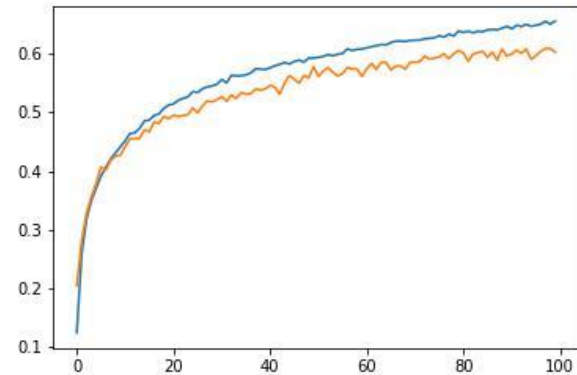
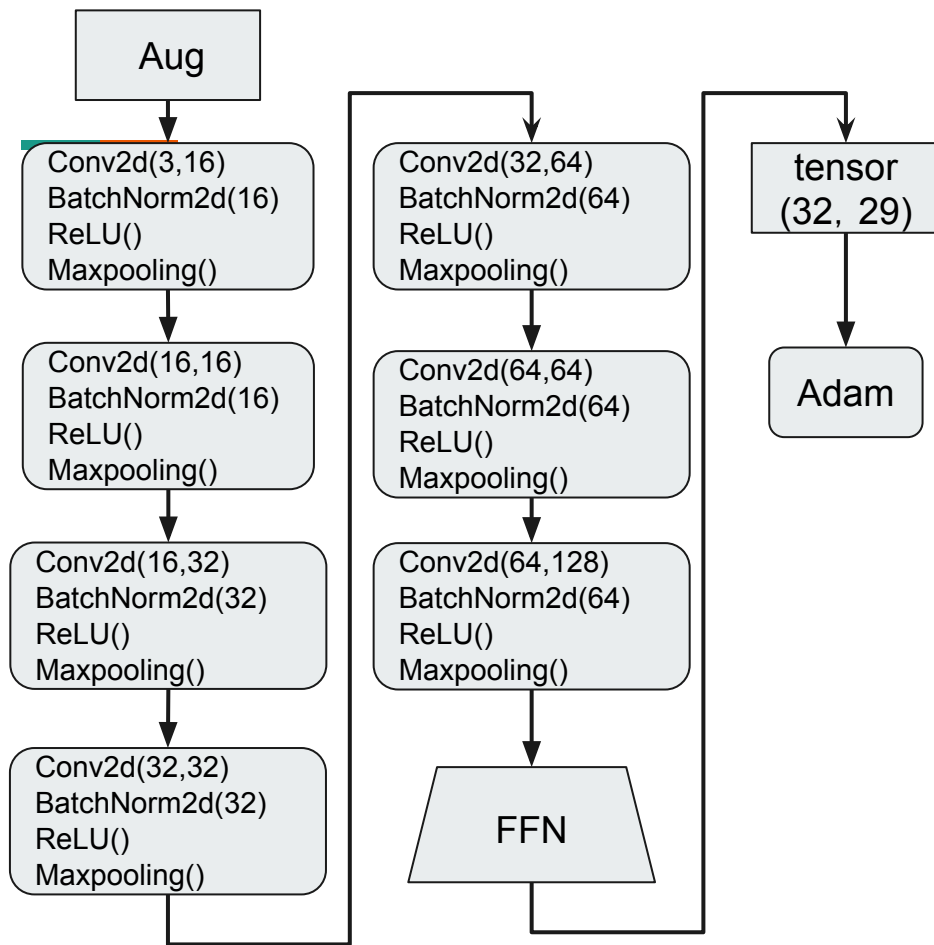
5 layers:



The acc on test dataset is


64.82%(↓ ≈ 1.5%)

7 layers:



The acc on test dataset is

59.93%(↓ ≈ 6.4%)



According to the results from the various depth attempts, we can draw conclusion that to this dataset, adding **more** convolution layers would degenerate the performance of the model (within 100 epochs).

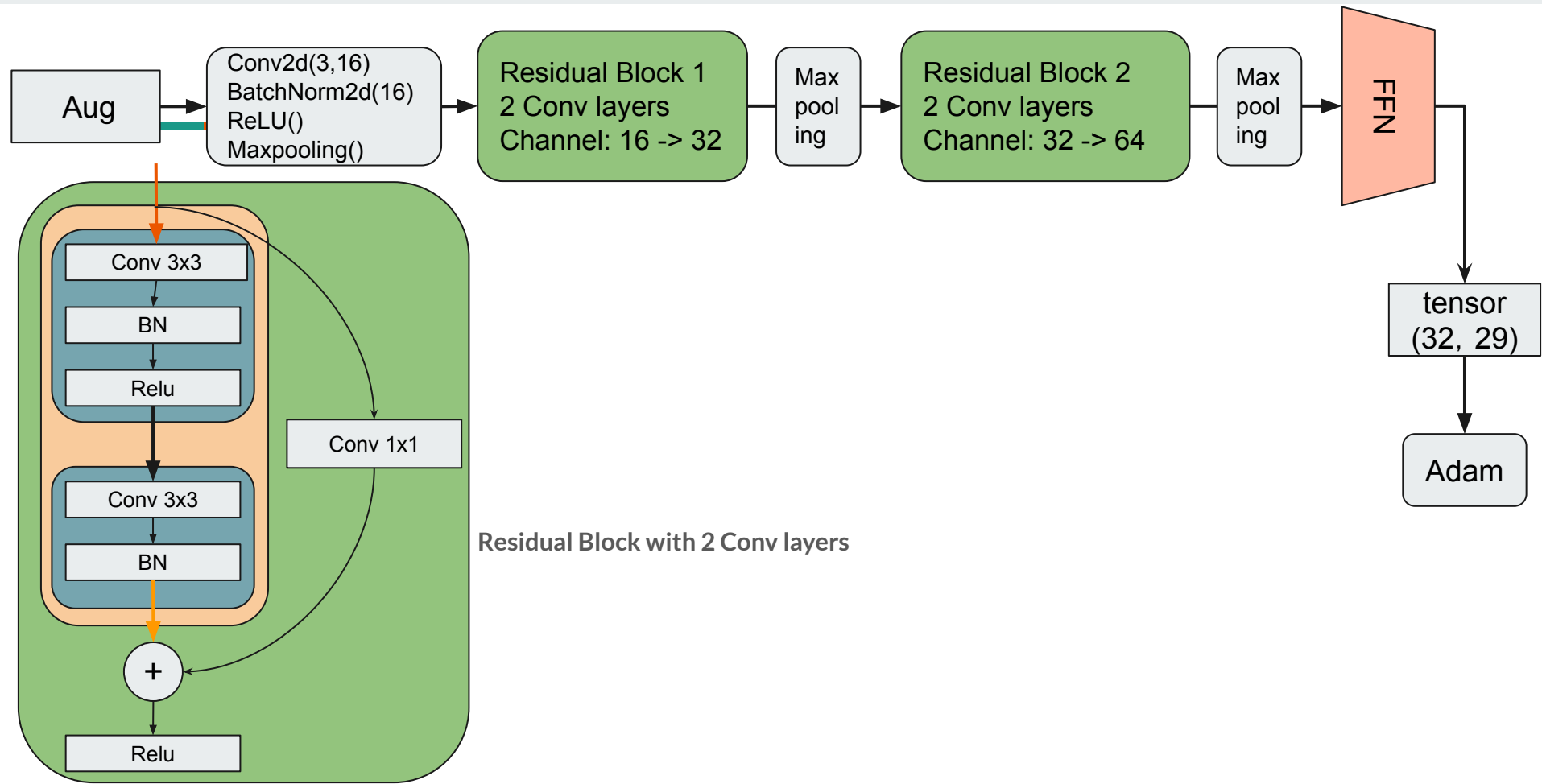
However, only having 3/5/7 depths seems not fair enough to draw this conclusion, then we tried fusing **skip connection** (residual network) for deeper architecture and higher accuracies.

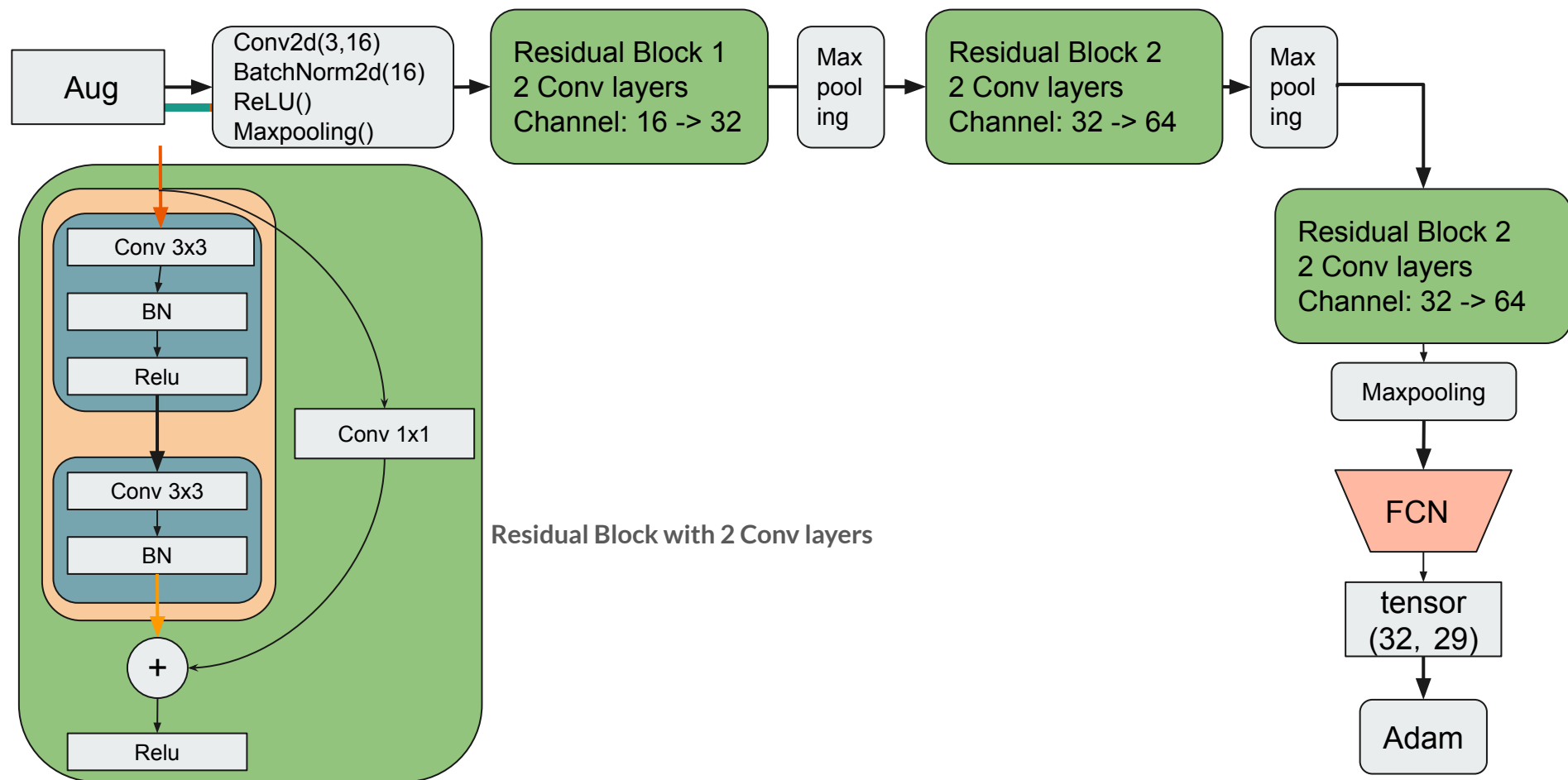


4. Self-designed ResNet

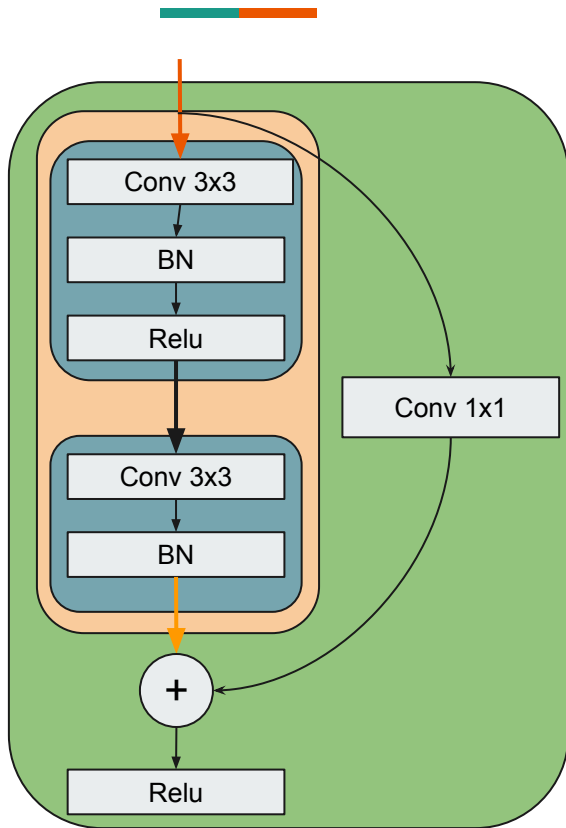
In total, 2 types of architectures are designed: one shallow and one much deeper.

Both are mainly composed of one 3×16 Conv2d layer, several Residual blocks and one fully connected network.

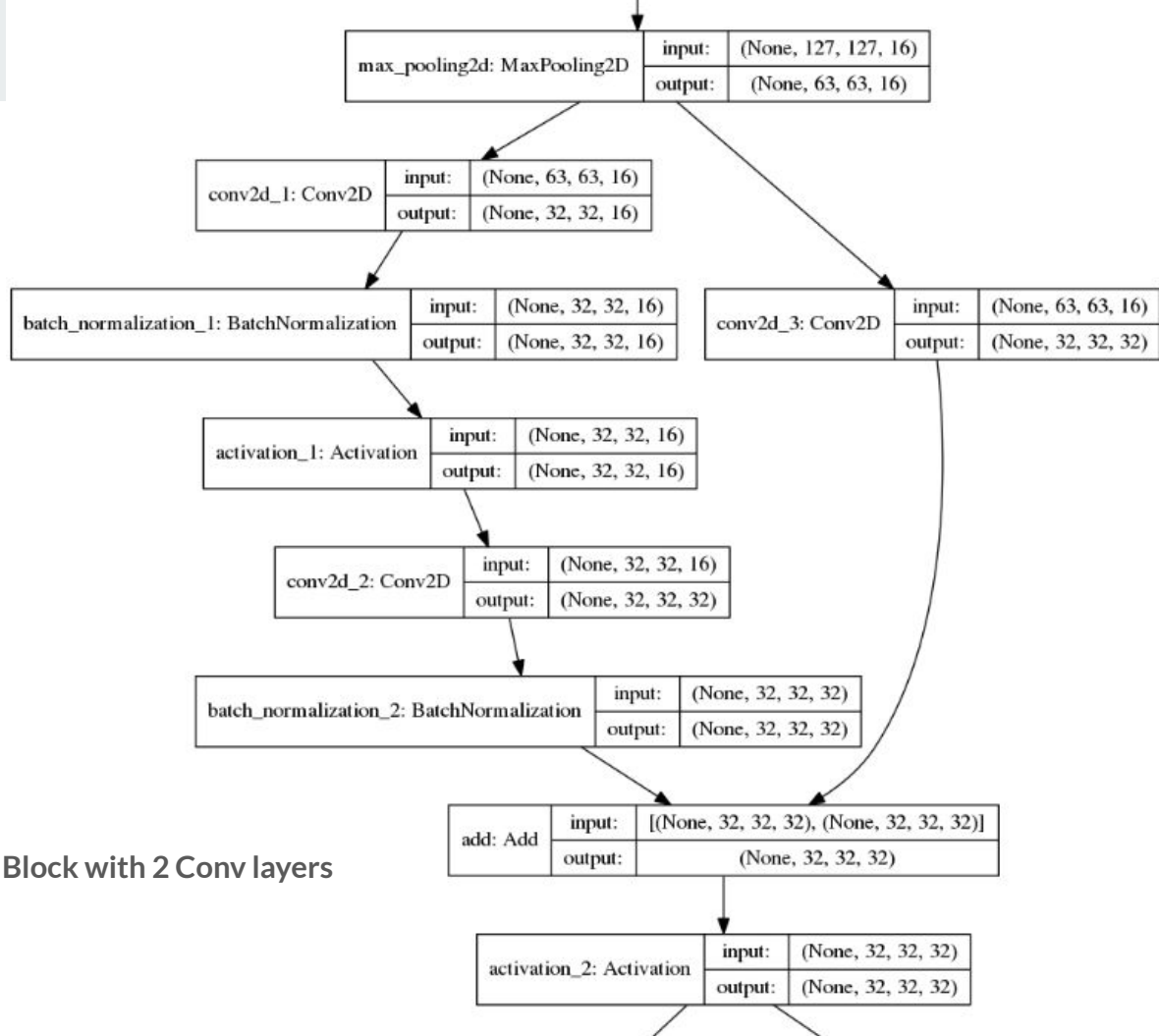




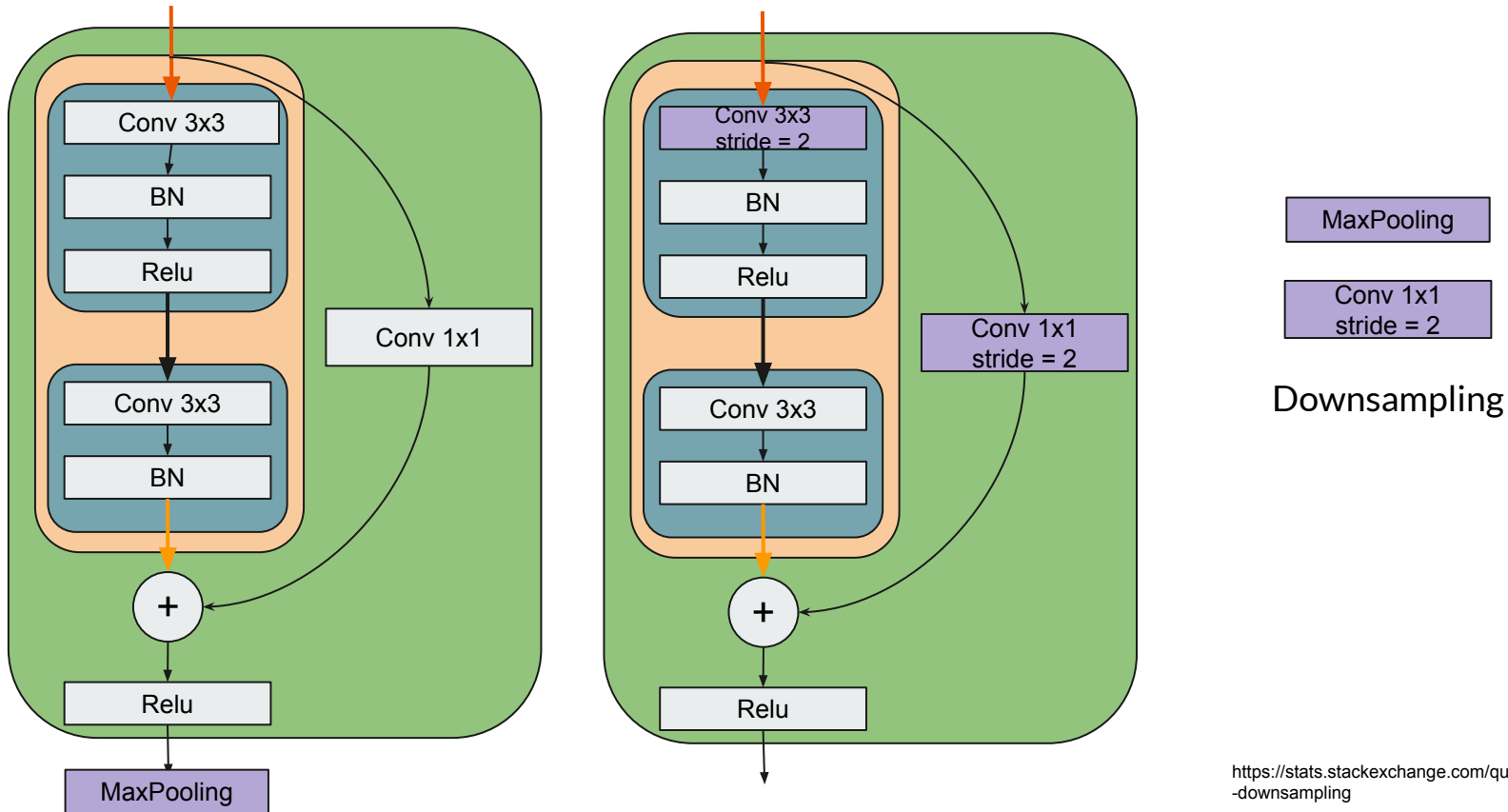
Detail of one residue block



Residual Block with 2 Conv layers

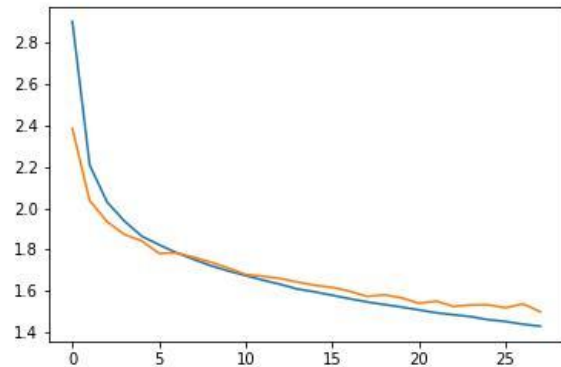
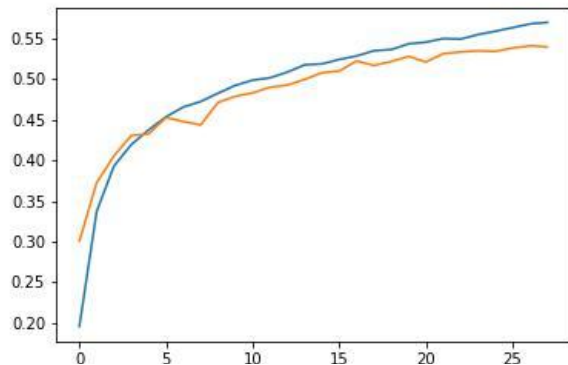


Different residual blocks in the model



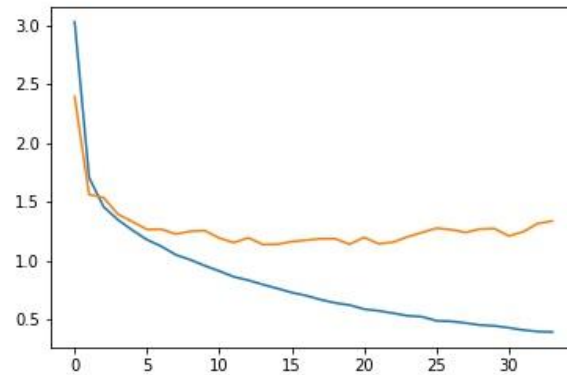
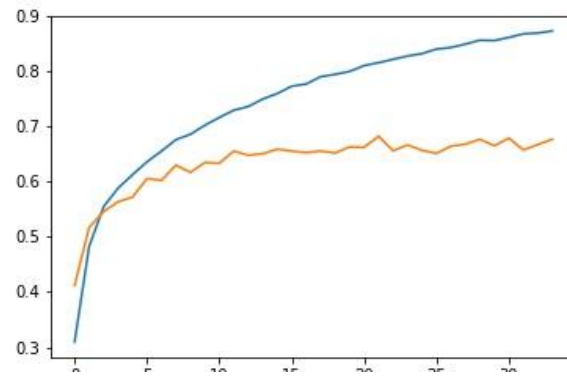
2 ResNet Blocks (/w 2 Conv/block)

66.41% on test dataset



3 ResNet Blocks (/w 4 Conv/block)

66.86% on test dataset



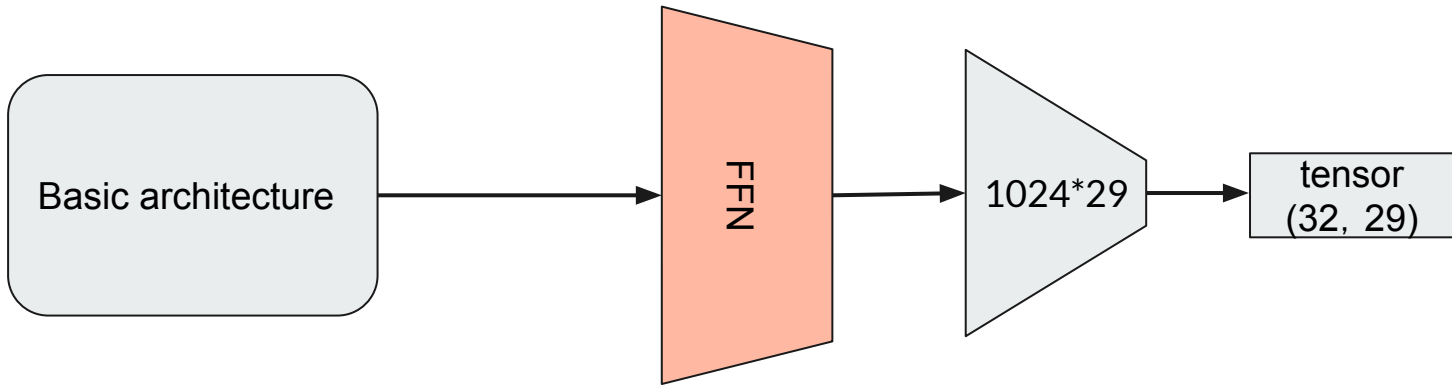
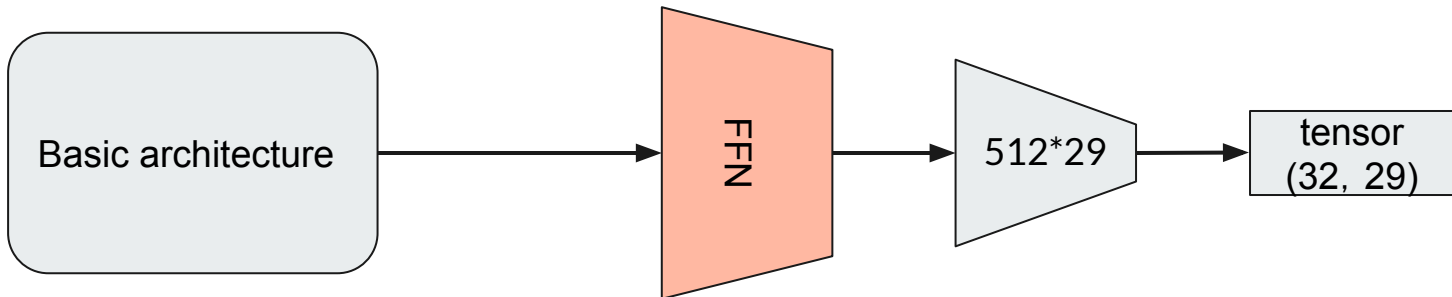
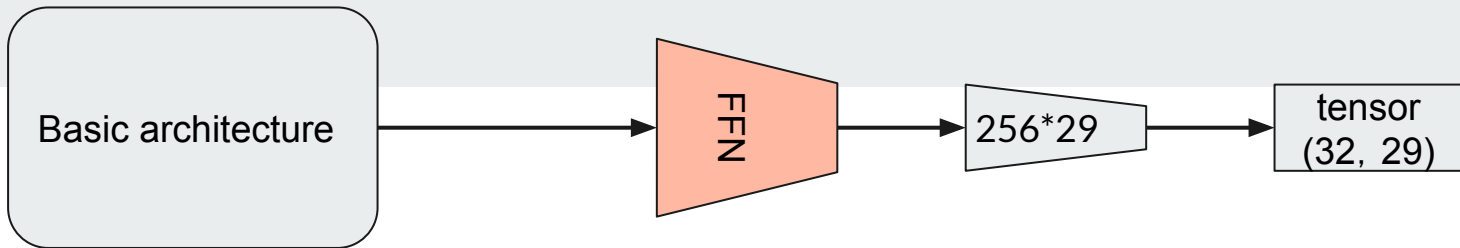


5. Extra Fully Connected Layer

In the previous CNNs, all the output from the convolutional layers are just connected to the a FCN with a transformation matrix (Conv_output * Num_class->29).

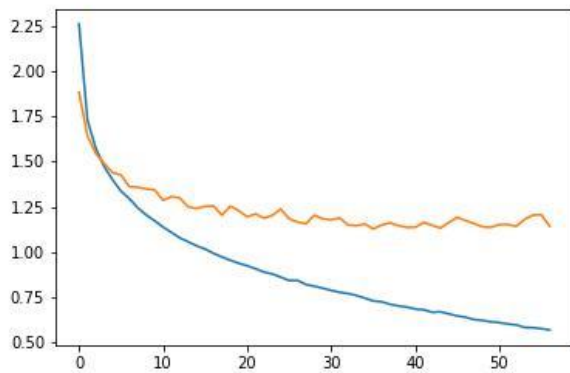
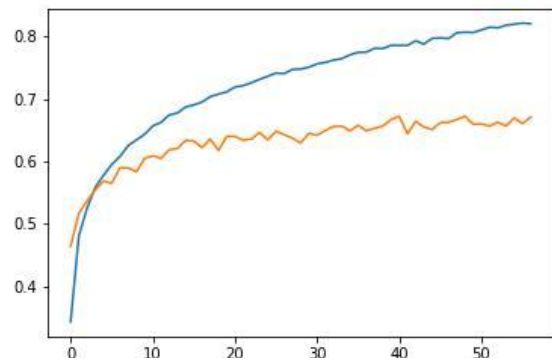
Then we made an attempt to add an FCN layer, with dim: 256/512/1024.

The basic architecture is the one with 3 Conv layers(32->64->128), data_aug and adam.



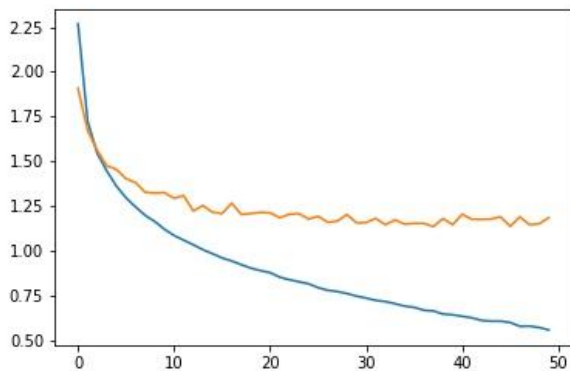
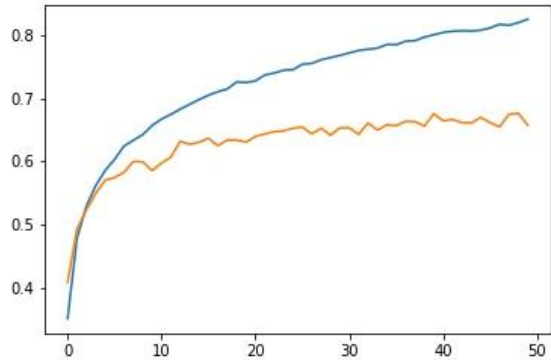
dim->256

66.47% on test dataset



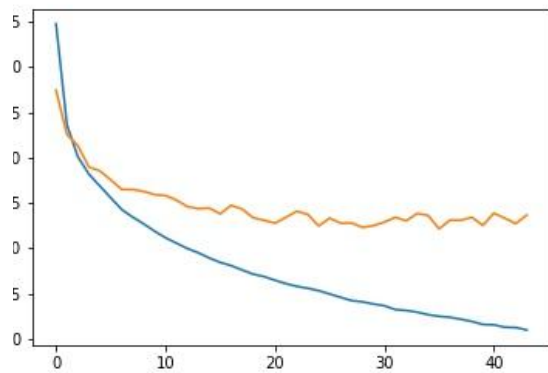
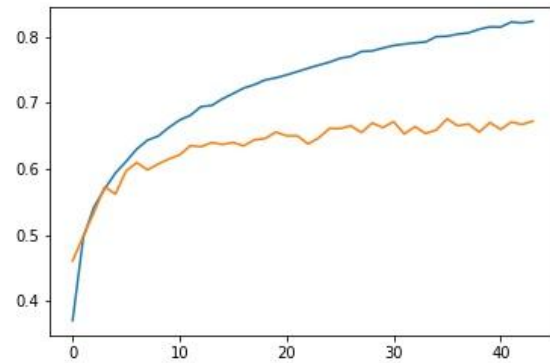
dim->512

67.68% on test dataset

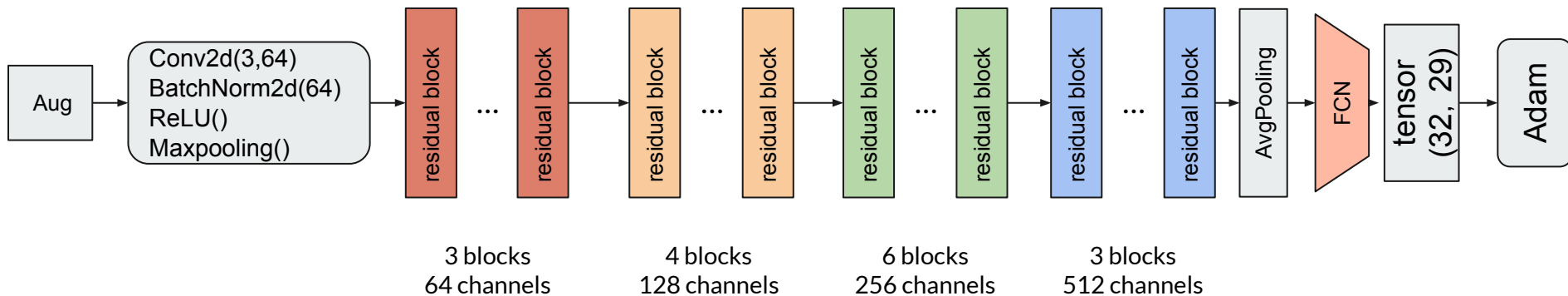


dim->1024

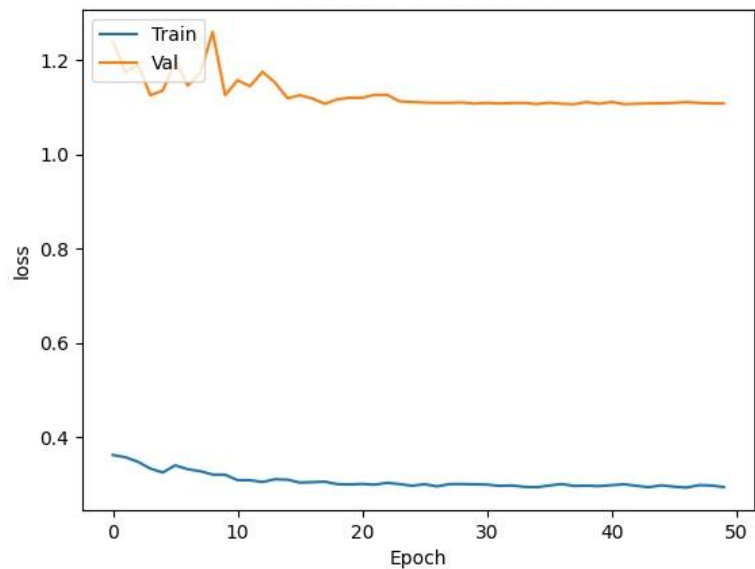
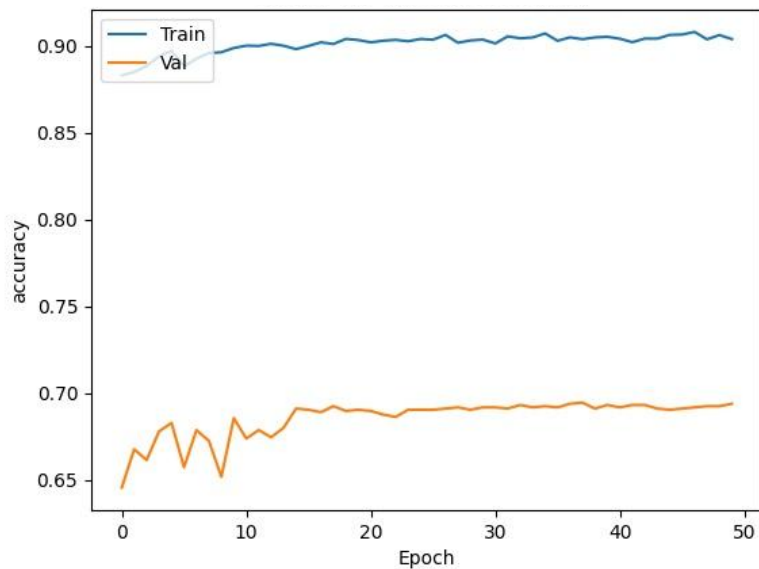
66.85% on test dataset



Deep Residual Network (34 layers) + FCN



34 layers resnet + adam + 512 fcn
68.68% on test dataset



Conclusion



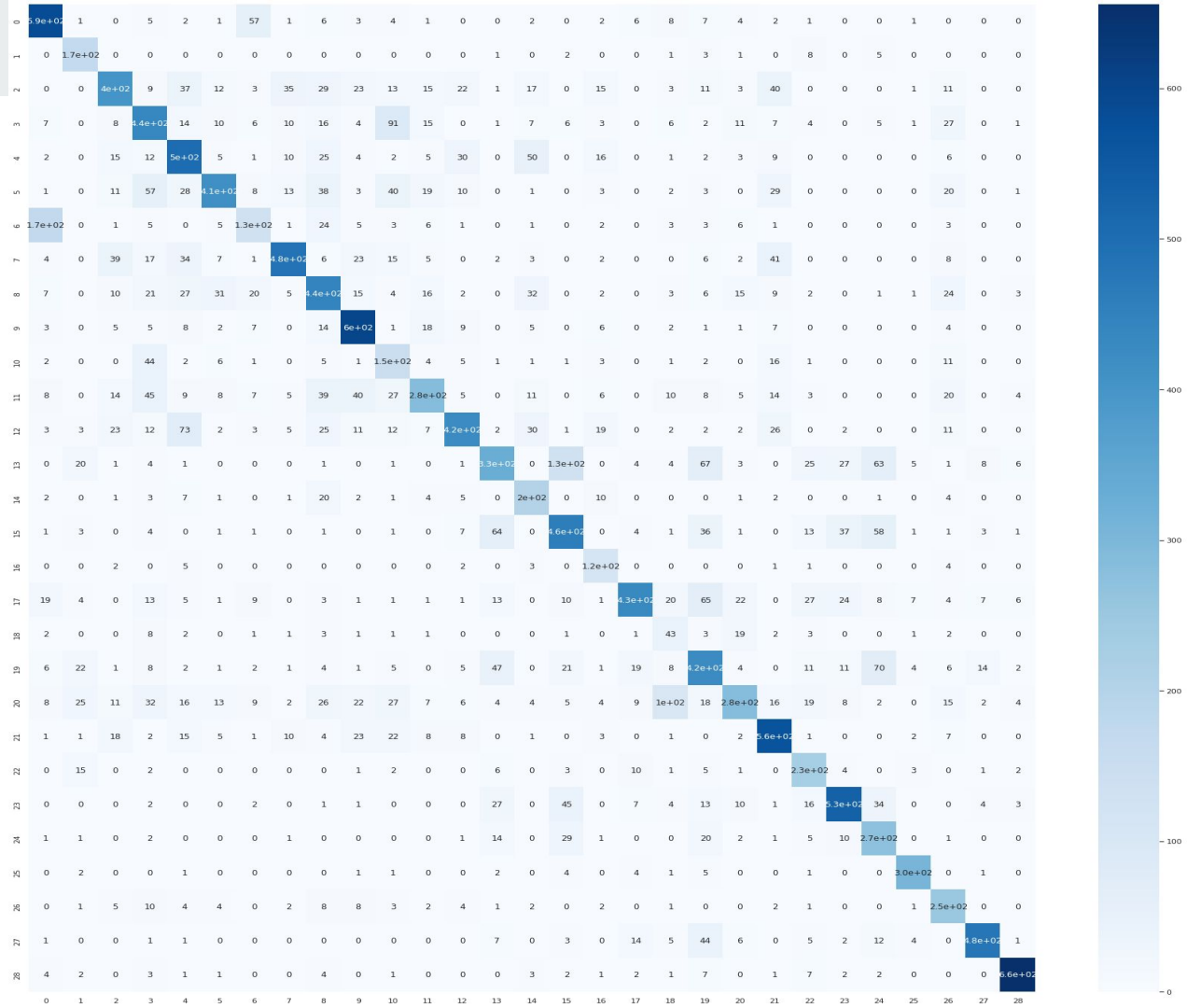
Top3 model with its acc

Model	Accuracy
34 layers resnet + adam + 512 fcn	68.68%
3 layers(3*32)+aug+adam+512 fcn	67.68%
3 ResNet Blocks (/w 4 Conv/block)	66.86%
3 layers(3*32)+aug+adam+1024 fcn	66.85%

Confusion Matrix



34 residual layers
+aug
+adam
+512 fcn





Future Work

In this lab, it is found that the numbers of convolutional channels or blocks, adding fully connected layers, or deeping the architecture (using ResNet if possible), all these operations could improve the model performance .

The top3 models in this lab are displayed in the previous page. Deep architecture training with ResNet does cost amount of computing resources and time, hence all of these aspects need to be taken into consideration.

We plan to take advantage of well known architectures and fine tune them in order to acquire higher accuracy, and enable us to train on higher resolution images.