Weekly Report

Wangwon Lee, 2019/05/18

This week

- **Deconvolution Network**
  - What is DeconvNet
  - Apply VGG16 in Keras
  - Compare to CAM
  - Training 2D-Model for visualization

Next week

- **Deconvolution Network**
  - Apply to 2D (our data)
  - To improve performance
    - Fine tuning (simply)
    - Augmentation
  - Apply to 1D (our data)

Interesting and new finding

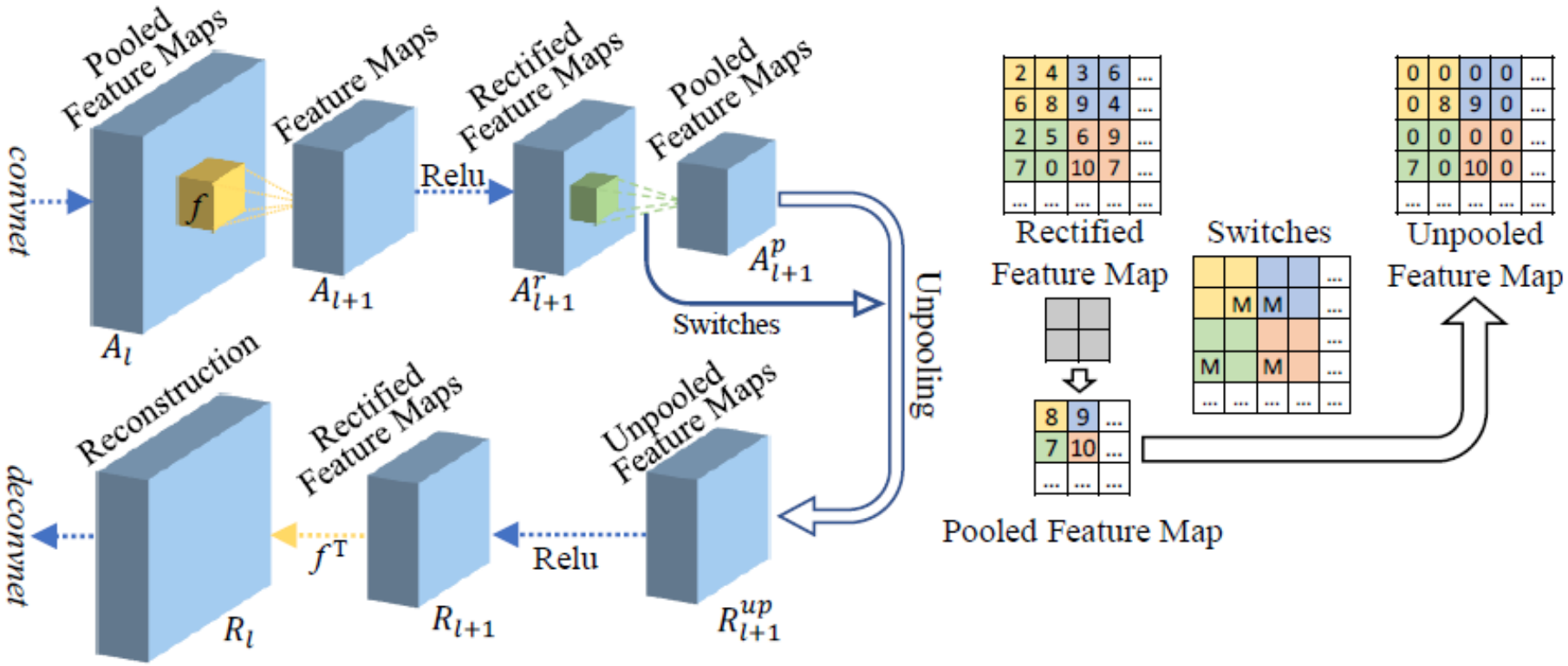
- DeconvNet

The aim of this month / Discussion

- **The aim of this month**: To study brain data.

What is DeconNet?

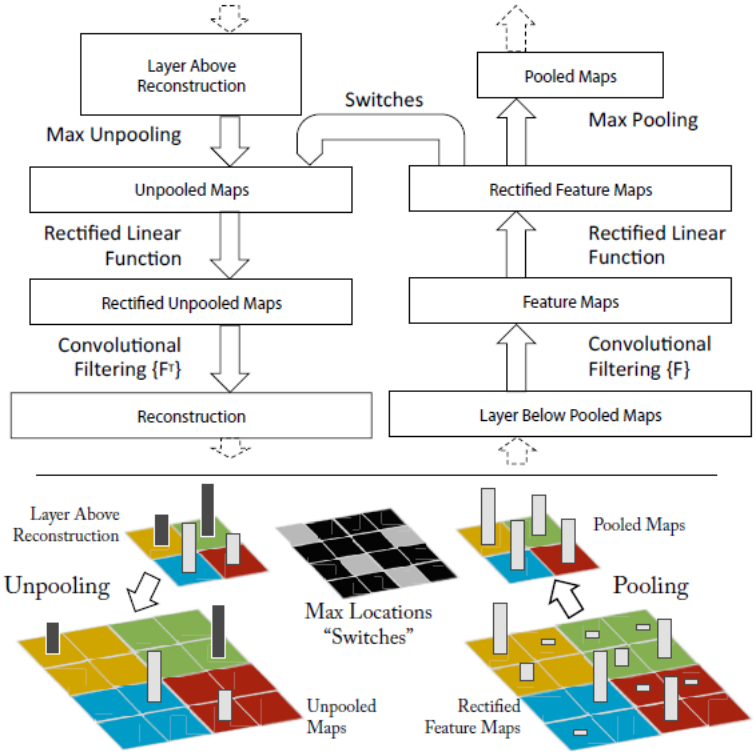
- Visualize each filter or layer (**how to see** the data)
- To target certain filter(Conv) or certain layer(Dense, Softmax)



The structure of the *Deconvolutional Network*

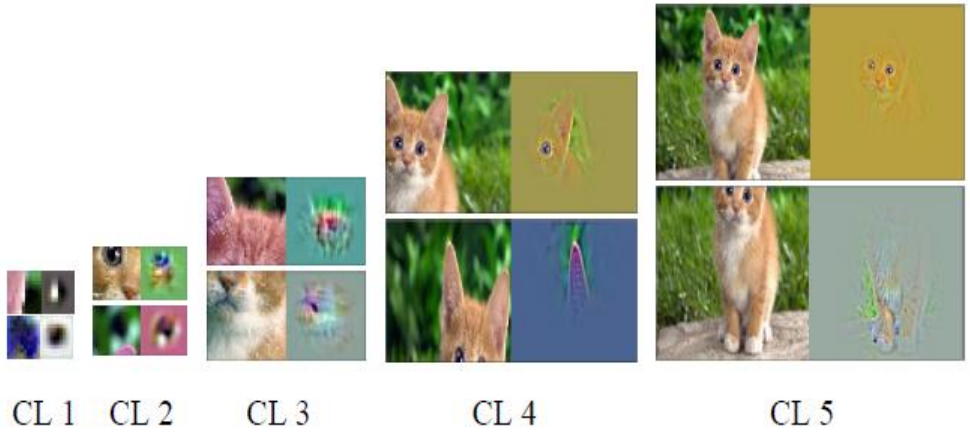
What is DeconNet?

- Visualize each filter or layer (**how to see** the data)
- To target certain filter(Conv) or certain layer(Dense, Softmax)



< The architecture >

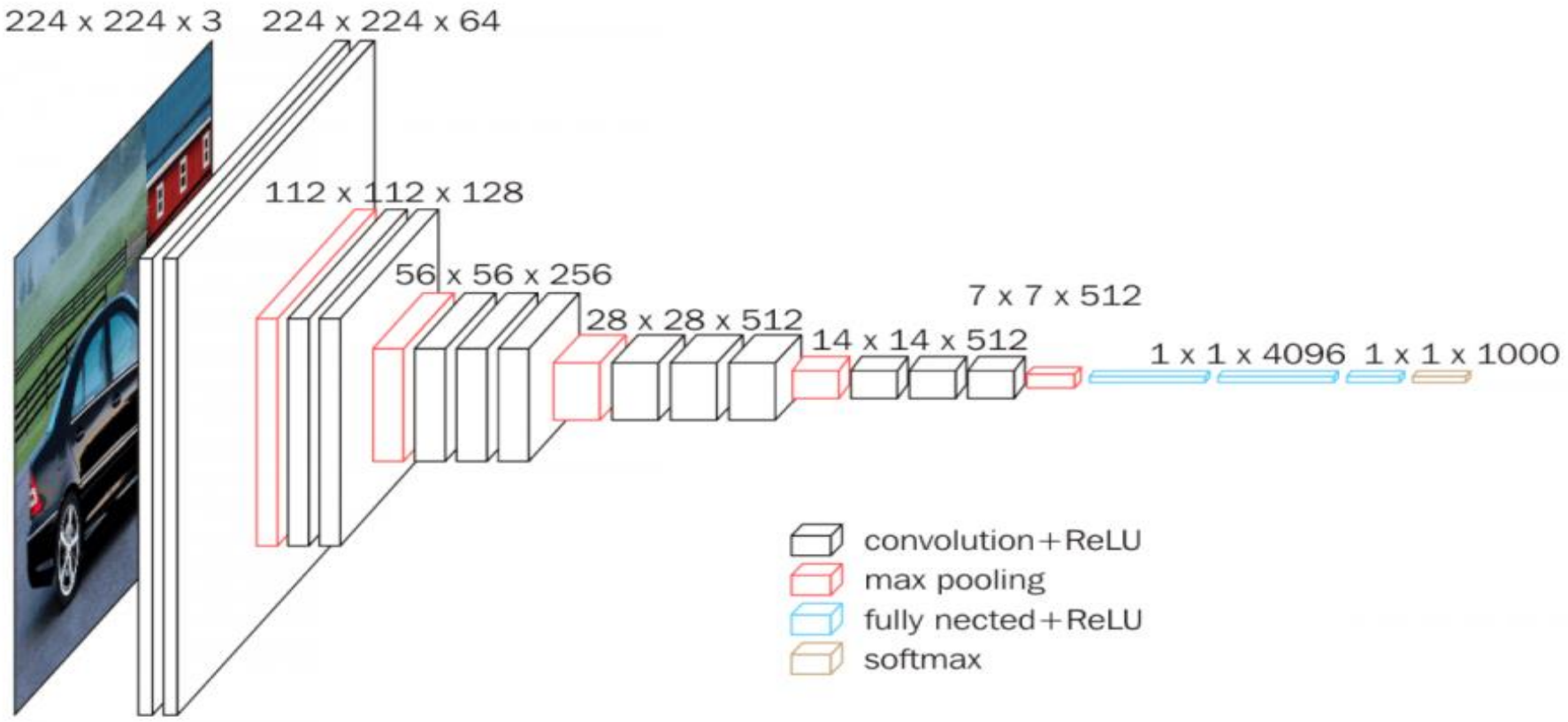
Left: input image
Right: visualization of each filter



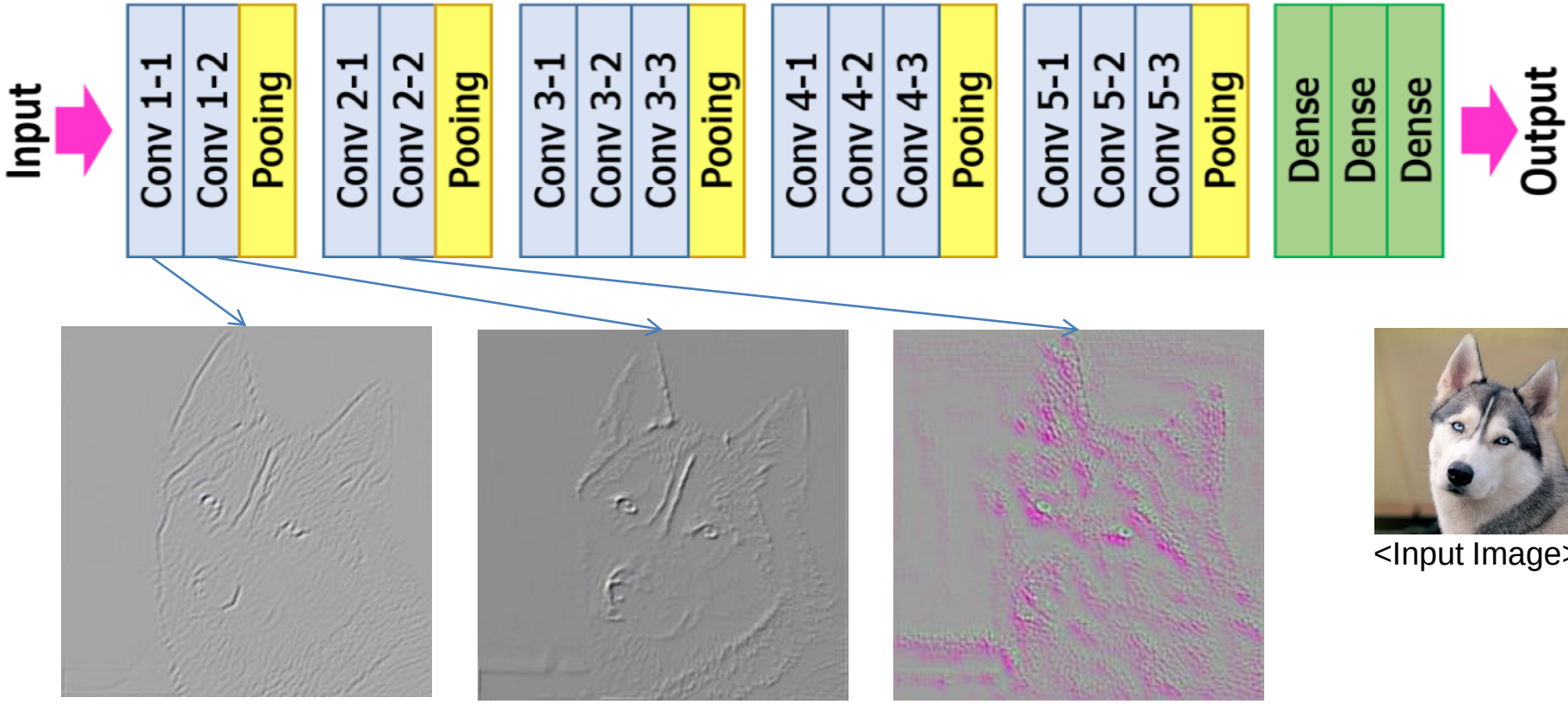
< Visualization of Convolution layer >

Apply VGG16 in Keras

- It’s hard to validate whether DeconvNet is implemented well in 1D-CNN
  Because we can not understand the result of DeconvNet visually
- So, Test in 2D-Model first (Target model: pretrained VGG16 provided by Keras)

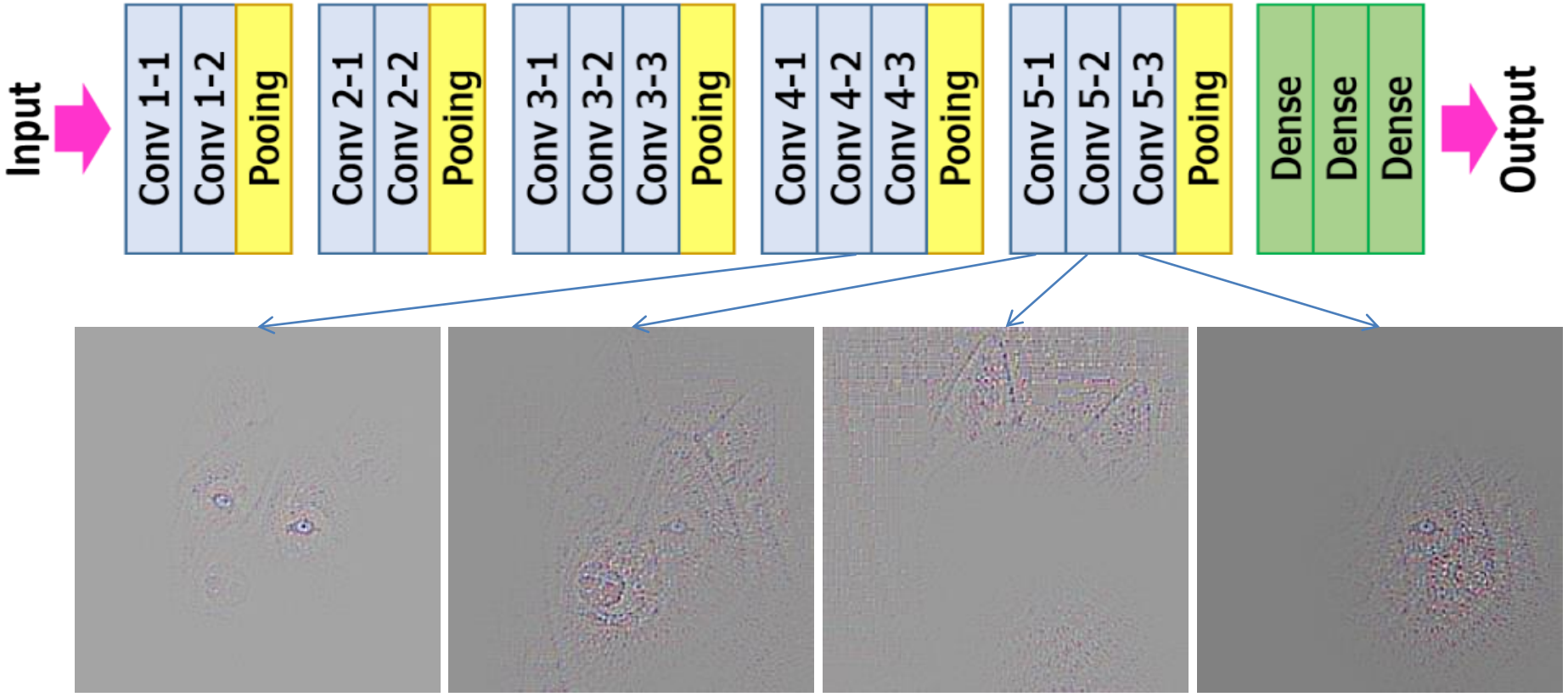


Apply VGG16 in Keras (based on 0th filter)



- In the 0th filter of the front layer, the model focus on the **edge** or **texture**
- Although there is no exist in the slide, each filter focus on individual feature.
- Front layer see the data entirely

Apply VGG16 in Keras (based on 0th filter)

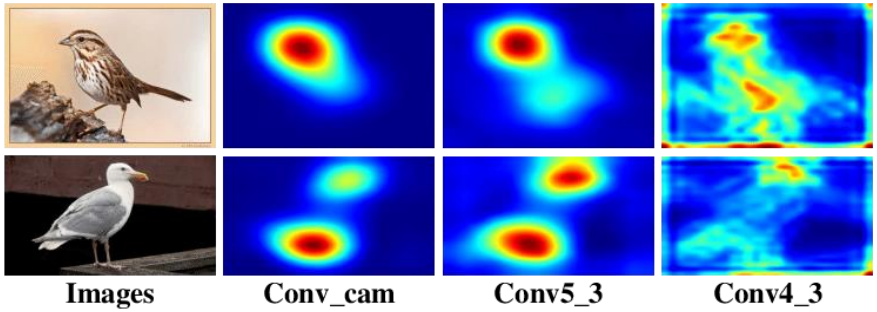


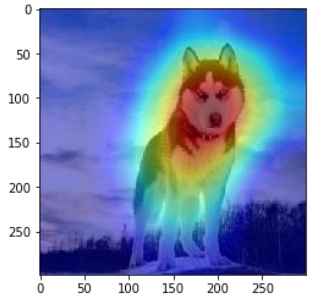
- But, In the 0th filter of the backward layer, the model focus on certain **area(eyes, nose, ear)**
- Backward layer see the data partially

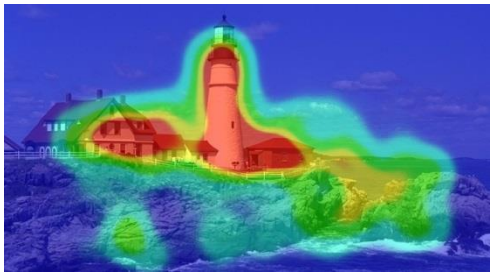
<Input Image>

Compare to CAM(Class Activation Map)

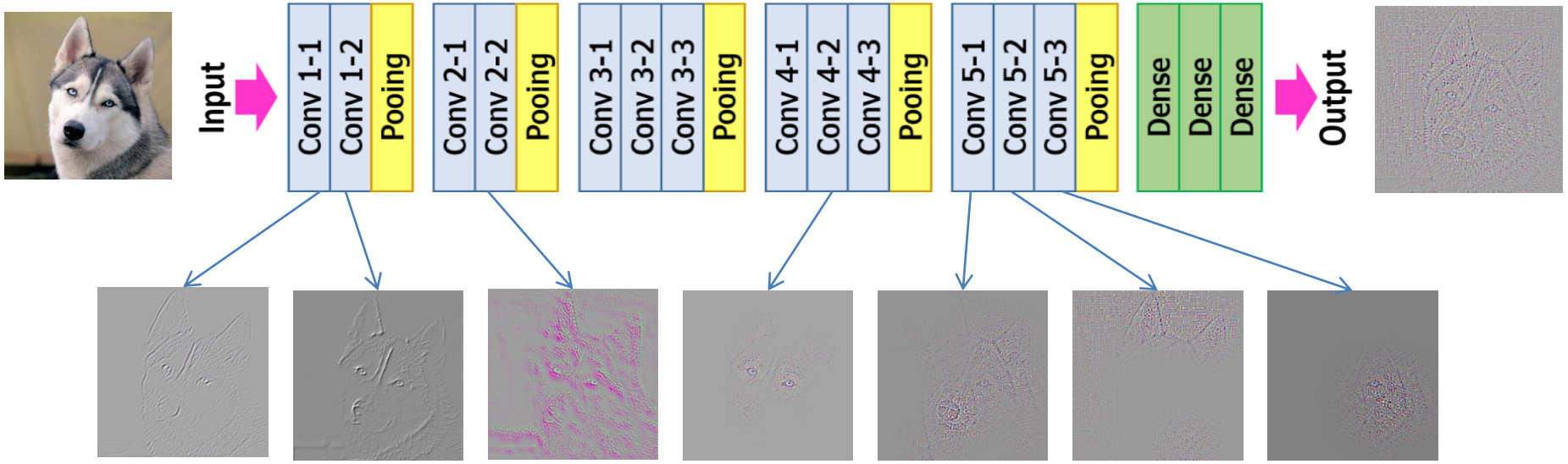
- CAM visualize the model **where to see**







- DeconvNet visualize the model **how to see**

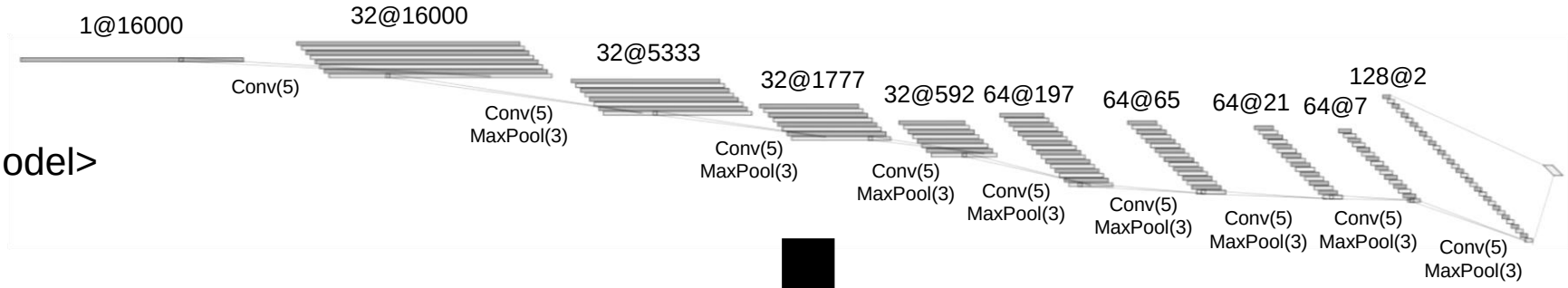


Training 2D-Model for visualization (MNIST)

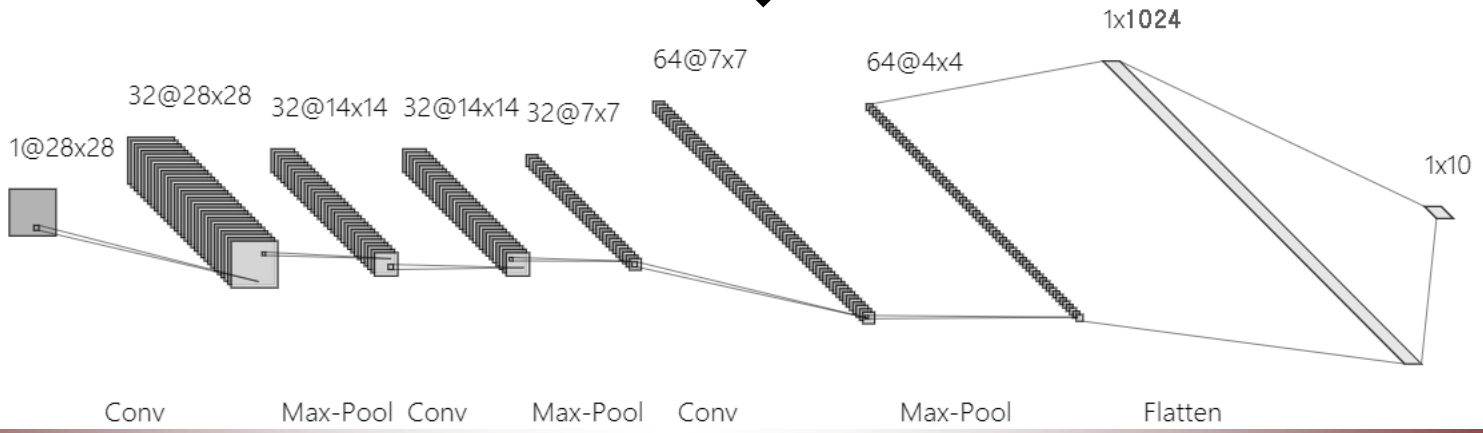
- Change (Because of input data's shape)
  Input: 16000X1 → 28X28X1, Output: 16 → 10 (0 ~ 9)
  filter size: 1X5 → 3X3, pool size: 1X3 → 2X2, number of conv layer: 8 → 3

- Train: 40000, Val: 20000, Test:10000

<1D Model>



<2D Model>

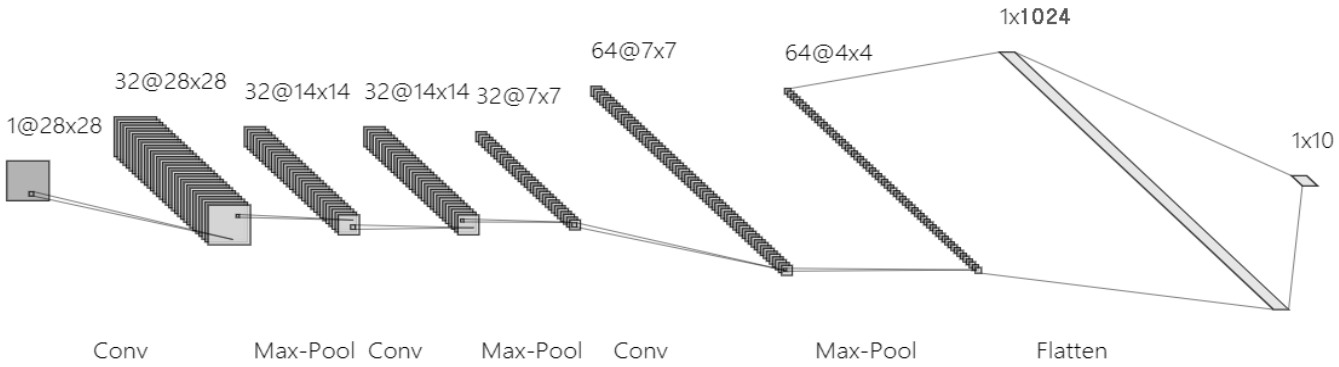


Training 2D-Model for visualization (MNIST)

- To find best fit model, Tuning the start channel(16, 32) and dropout rate(0.5, 0.75)
- When the number of convolution filter is 3, performance is best.

Architecture	CH16+DO0.5	CH16+DO0.75	CH32+DO0.5	CH32+DO0.75
1 CONV(3X3)	0.9835	0.9828	0.9837	0.9839
2 CONV(3X3)	0.9893	0.9875	0.9891	0.9901
3 CONV(3X3)	0.9901	0.9902	0.9921	0.9934
4 CONV(3X3)	0.9892	0.9891	0.9917	0.9900
5 CONV(3X3)	0.9863	0.9910	0.9901	0.9904

<Best Architecture>



Training 2D-Model for visualization (MNIST)

- In the best architecture, It seems that the model classify well
- So, I will visualize this as DeconvNet in next week.

Actual class

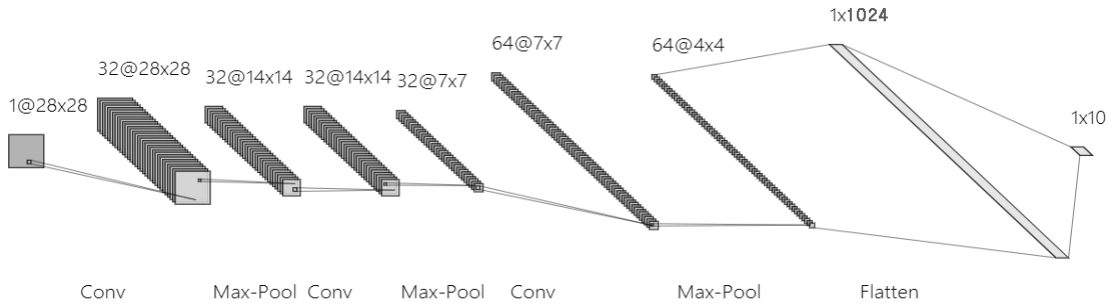
Predict Class

```
[[ 975    0    1    0    0    0    2    1    1    0]
 [   0 1133    1    1    0    0    0    0    0    0]
 [   1    0 1028    0    1    0    0    1    0    1]
 [   0    0    0 1008    0    1    0    0    1    0]
 [   0    0    0    0  975    0    0    0    1    6]
 [   1    0    0    3    0  887    1    0    0    0]
 [   4    1    0    0    2    1  946    0    4    0]
 [   0    3    4    0    0    0    0 1019    1    1]
 [   0    0    0    1    1    0    0    0  969    3]
 [   0    0    1    0    5    4    0    4    1  994]]
```

	precision	recall	f1-score	support
0	0.99	0.99	0.99	980
1	1.00	1.00	1.00	1135
2	0.99	1.00	0.99	1032
3	1.00	1.00	1.00	1010
4	0.99	0.99	0.99	982
5	0.99	0.99	0.99	892
6	1.00	0.99	0.99	958
7	0.99	0.99	0.99	1028
8	0.99	0.99	0.99	974
9	0.99	0.99	0.99	1009
weighted avg	0.99	0.99	0.99	10000

<Best Architecture>

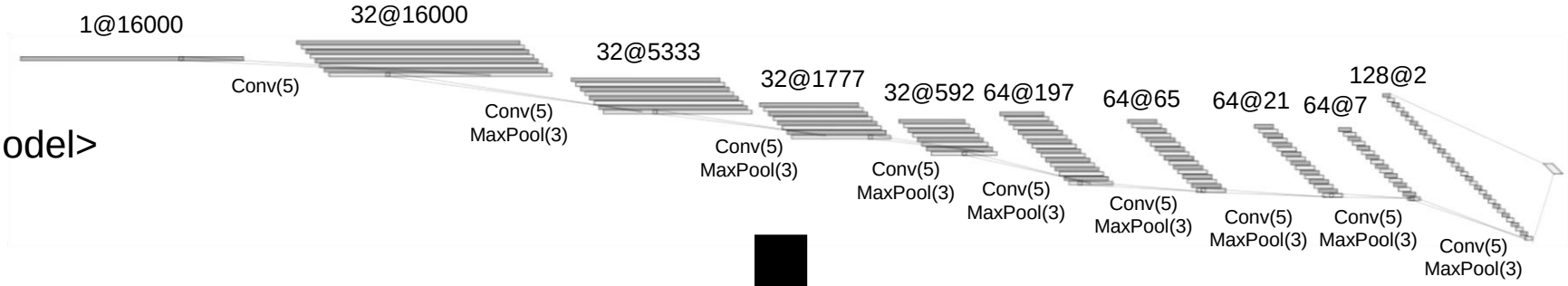
(Acc: 0.9934)



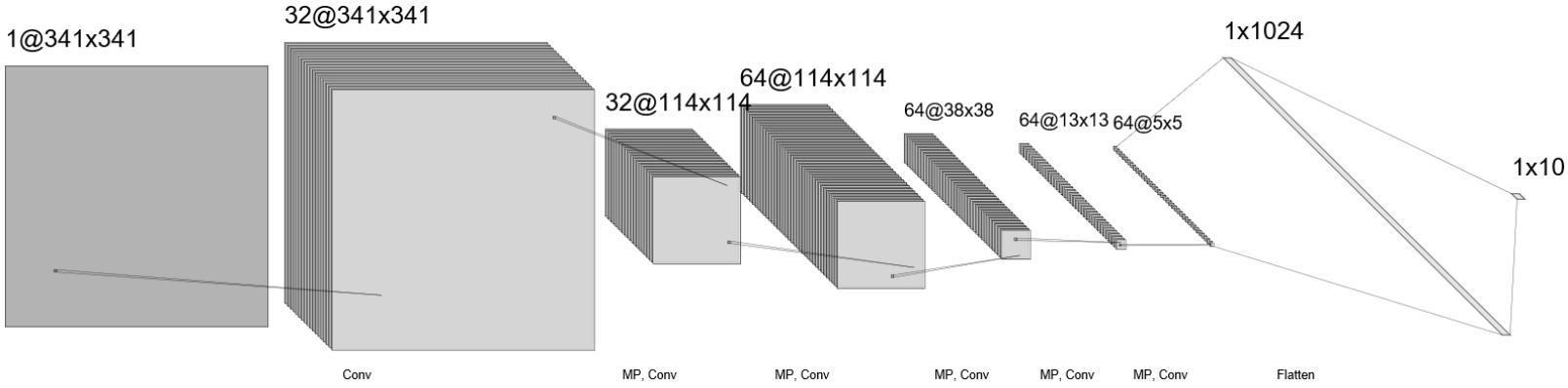
Training 2D-Model for visualization (Imagenet)

- Change (Because of input data's shape)
  Input: 16000X1 → 341X341X3, Output: 16 → 6 ('bed', 'bird', 'cat', 'dog', 'house', 'tree')
  filter size: 1X5 → 5X5, pool size: 1X3 → 3X3, number of conv layer: 8 → 5
- Train: 4680(60%), Val: 1560(20%), Test: 1560(20%)

<1D Model>

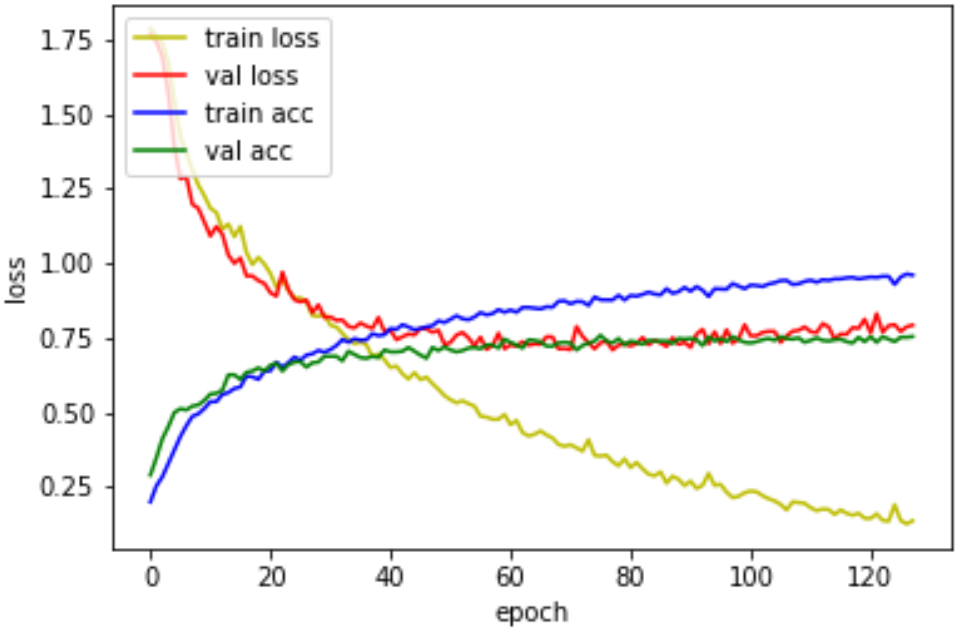


<2D Model>



Training 2D-Model for visualization (Imagenet)

- It is still training now… For the report, I use only **30%** of each set.
  - Train: 1404(20%), Val: 468(6%), Test: 468(6%)
- Despite val loss increase, It seems that val accuracy increase.
- Maybe It is caused by a lack of data. I will try again as more data.



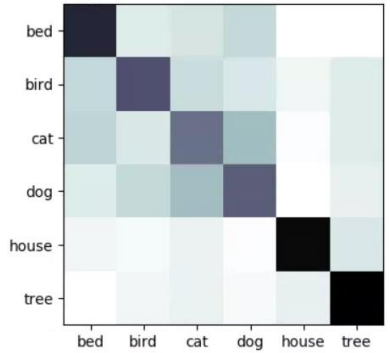
< Accuracy: 0.6217 >

Actual class

Predict Class

```
[[53  6  8 11  0  0]
 [11 41 10  7  3  6]
 [12  7 34 18  1  6]
 [ 6 11 18 38  0  5]
 [ 3  2  4  1 61  7]
 [ 0  3  4  2  5 64]]
```

=



	precision	recall	f1-score	support
0	0.62	0.68	0.65	78
1	0.59	0.53	0.55	78
2	0.44	0.44	0.44	78
3	0.49	0.49	0.49	78
4	0.87	0.78	0.82	78
5	0.73	0.82	0.77	78
weighted avg	0.62	0.62	0.62	468

Any Question?

Thank you