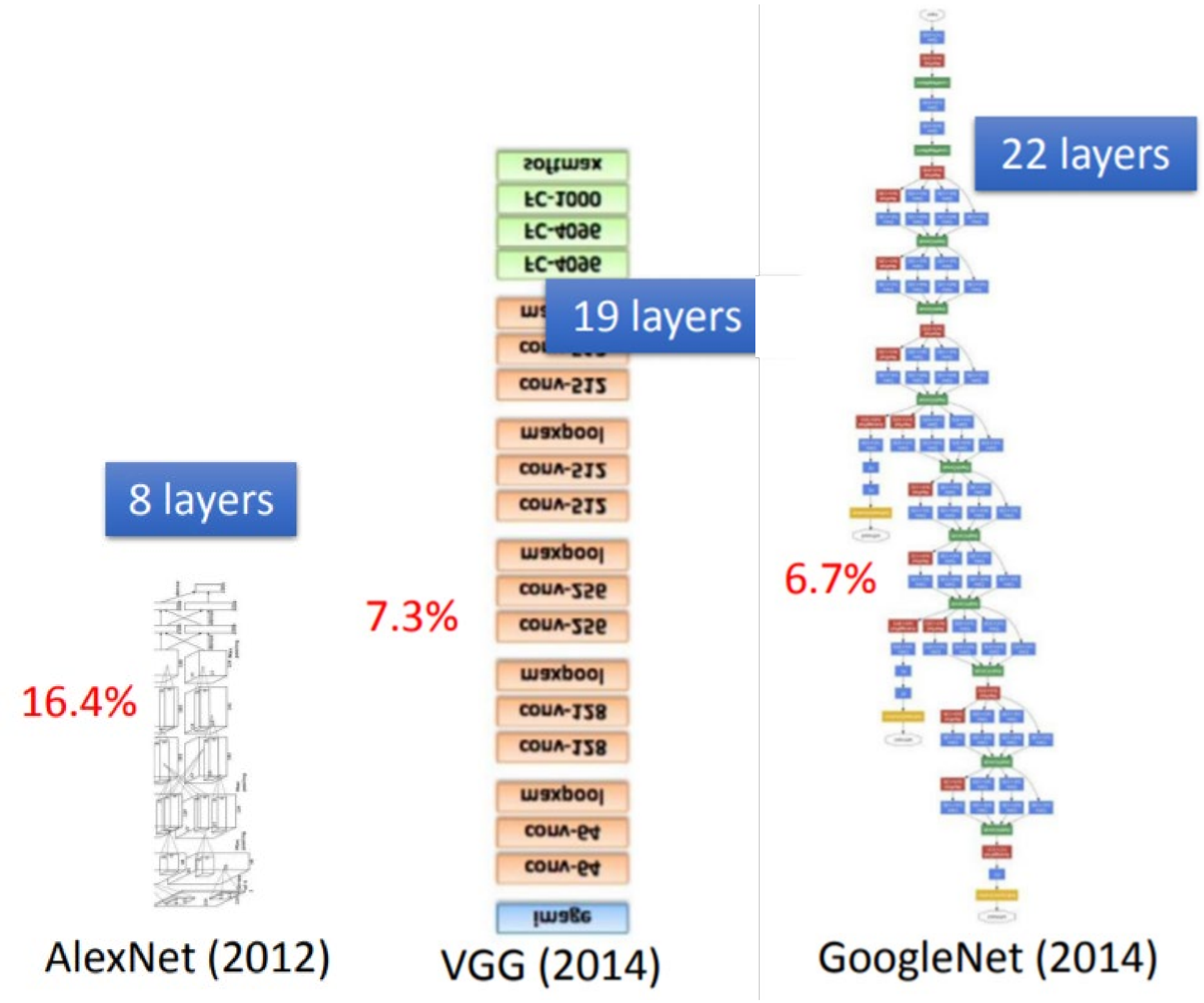
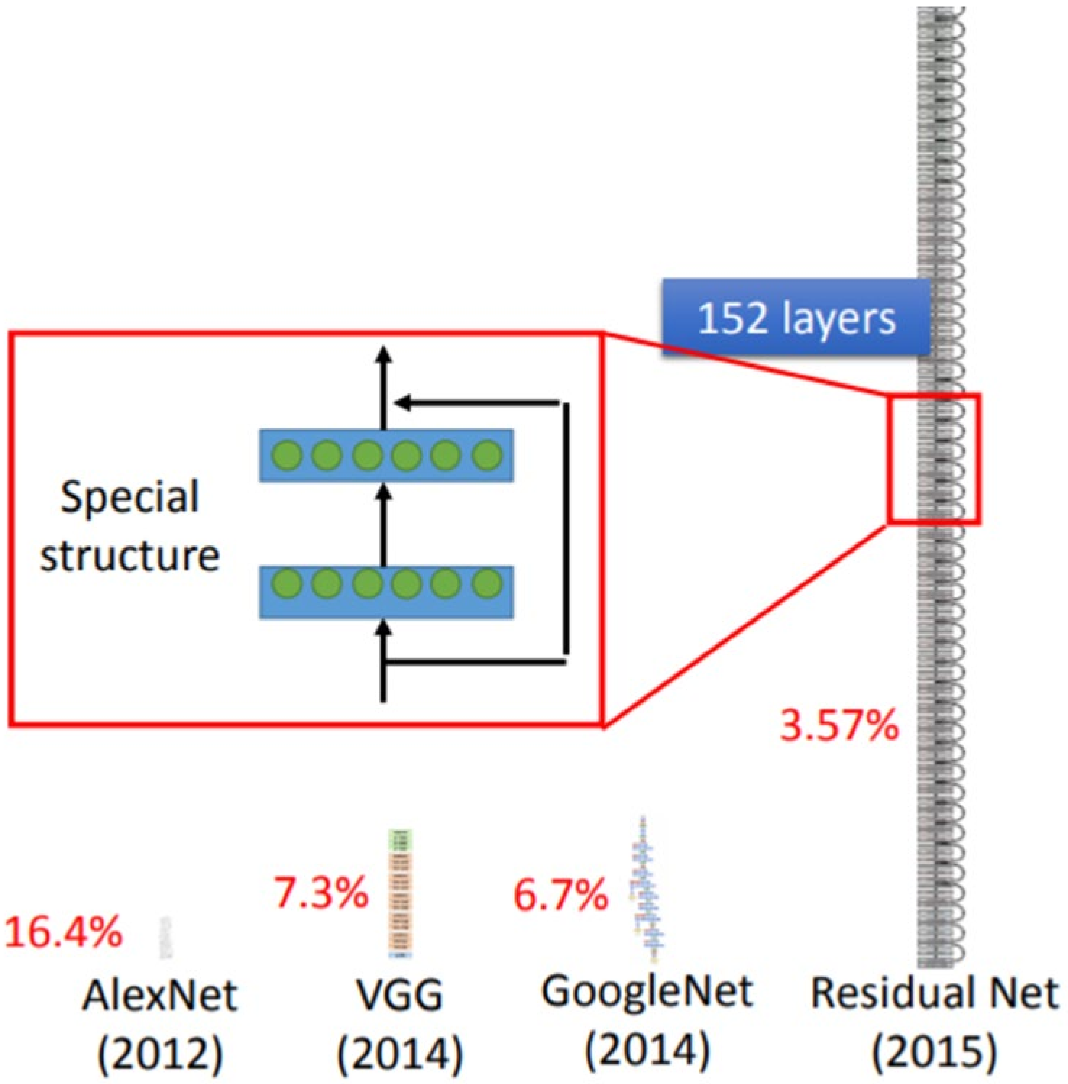
Why deep ?

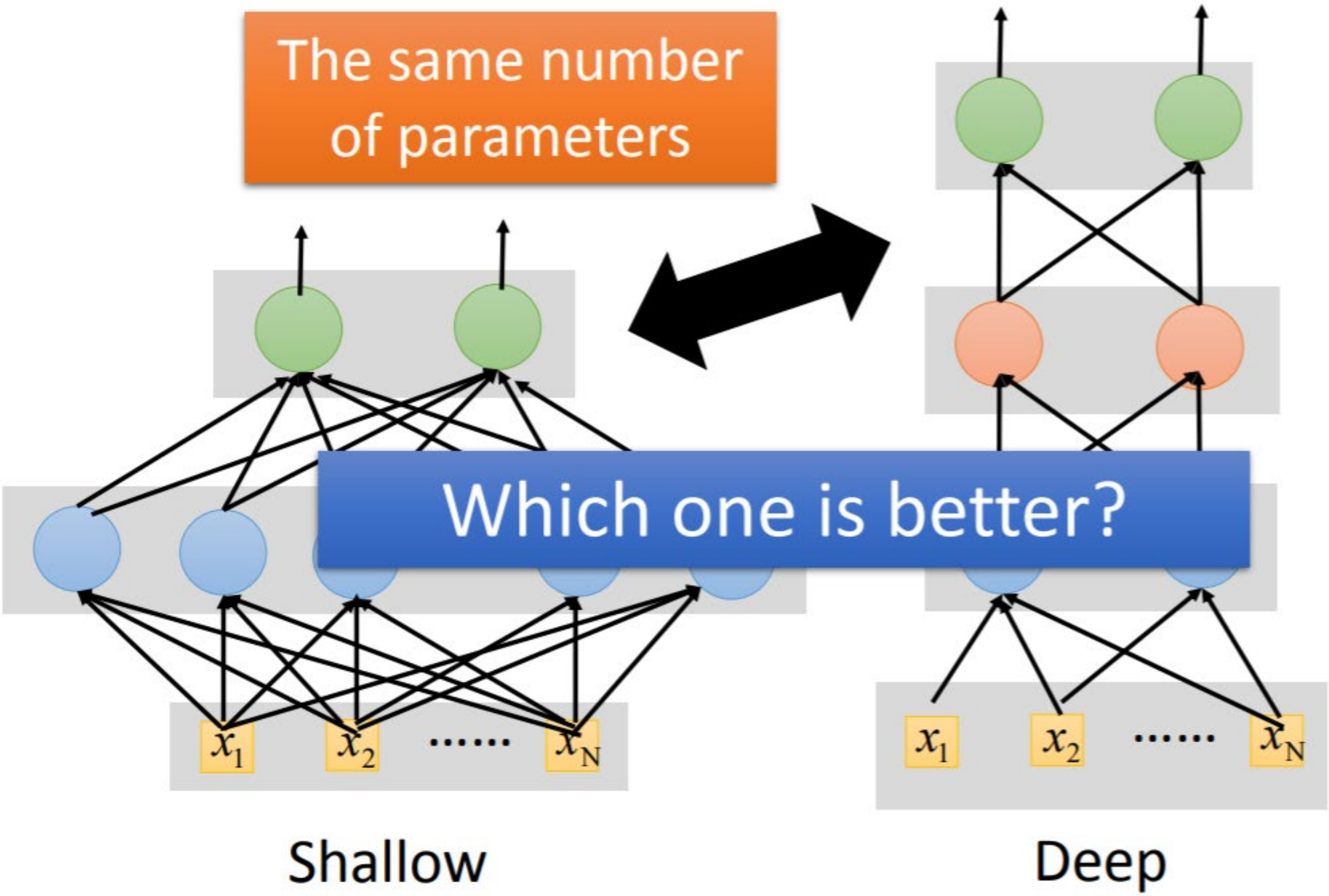
Going deeper and deeper...





Reference: 李弘毅 ML Lecture 6 https://youtu.be/Dr-WRlEFefw

With same number of parameters, which NN is better?



Reference: 李弘毅 ML Lecture 11 https://youtu.be/XsC9byQkUH8

Deep is better

Layer X Size	Word Error Rate (%)	Layer X Size	Word Error Rate (%)
1 X 2k	24.2		
2 X 2k	20.4		
3 X 2k	18.4		
4 X 2k	17.8		
5 X 2k	17.2	1 X 3772	22.5
7 X 2k	17.1	1 X 4634	22.6
		1 X 16k	22.1

Why?

deep + thin

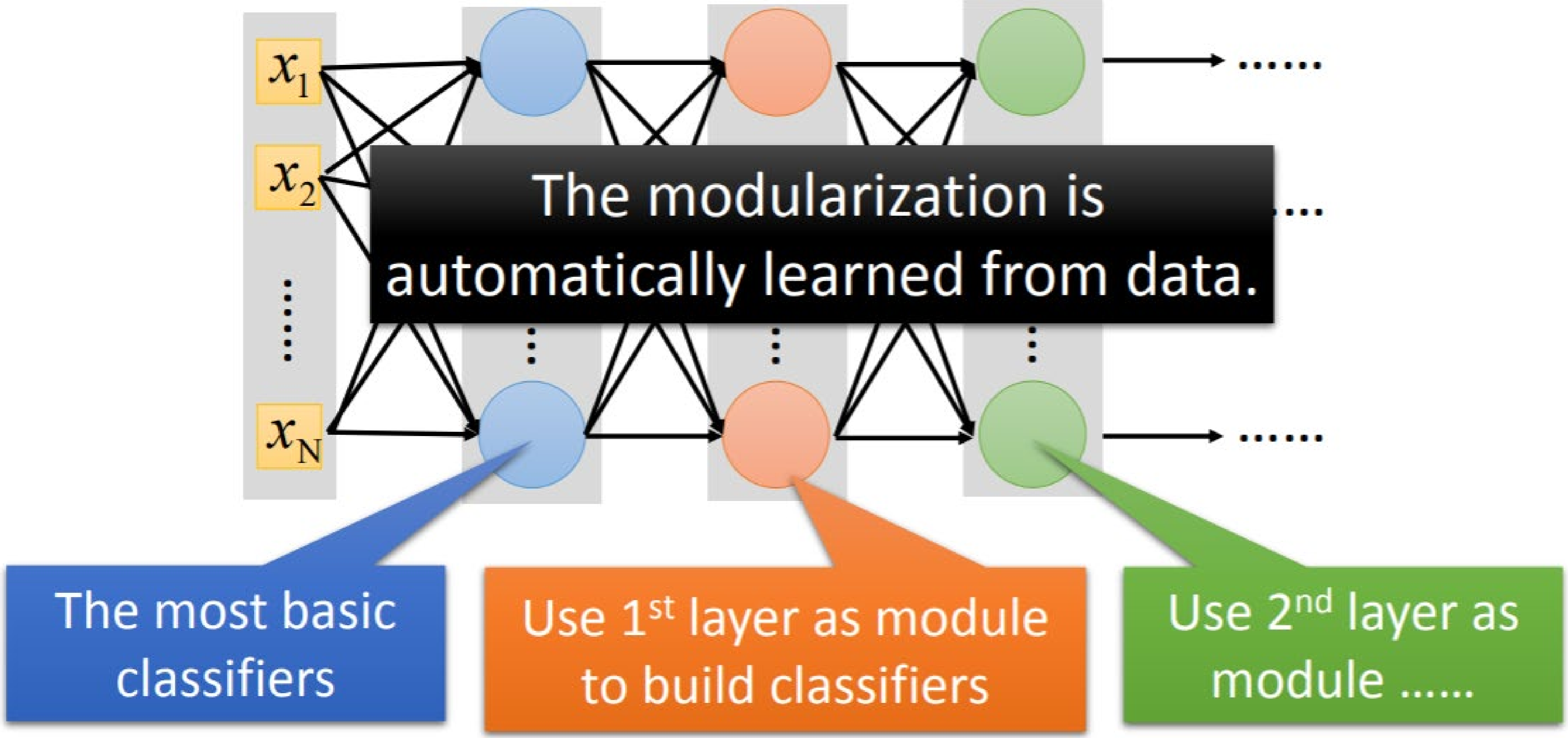
short + fat

Seide, Frank, Gang Li, and Dong Yu. "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." *Interspeech*. 2011.

Reference: 李弘毅 ML Lecture 11 https://youtu.be/XsC9byQkUH8

Reason 1 – Modularization

- Deep → Modularization → Less training data?



Reference: 李弘毅 ML Lecture 11 https://youtu.be/XsC9byQkUH8

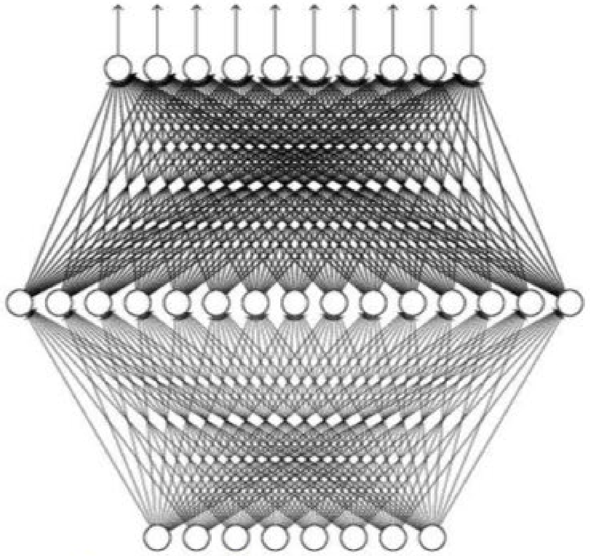
Universality theorem

Any continuous function f

$$f : R^N \rightarrow R^M$$

Can be realized by a network with one hidden layer

(given **enough** hidden neurons)



Reference for the reason: http://neuralnetworksanddeeplearning.com/chap4.html

Yes, shallow network can represent any function.

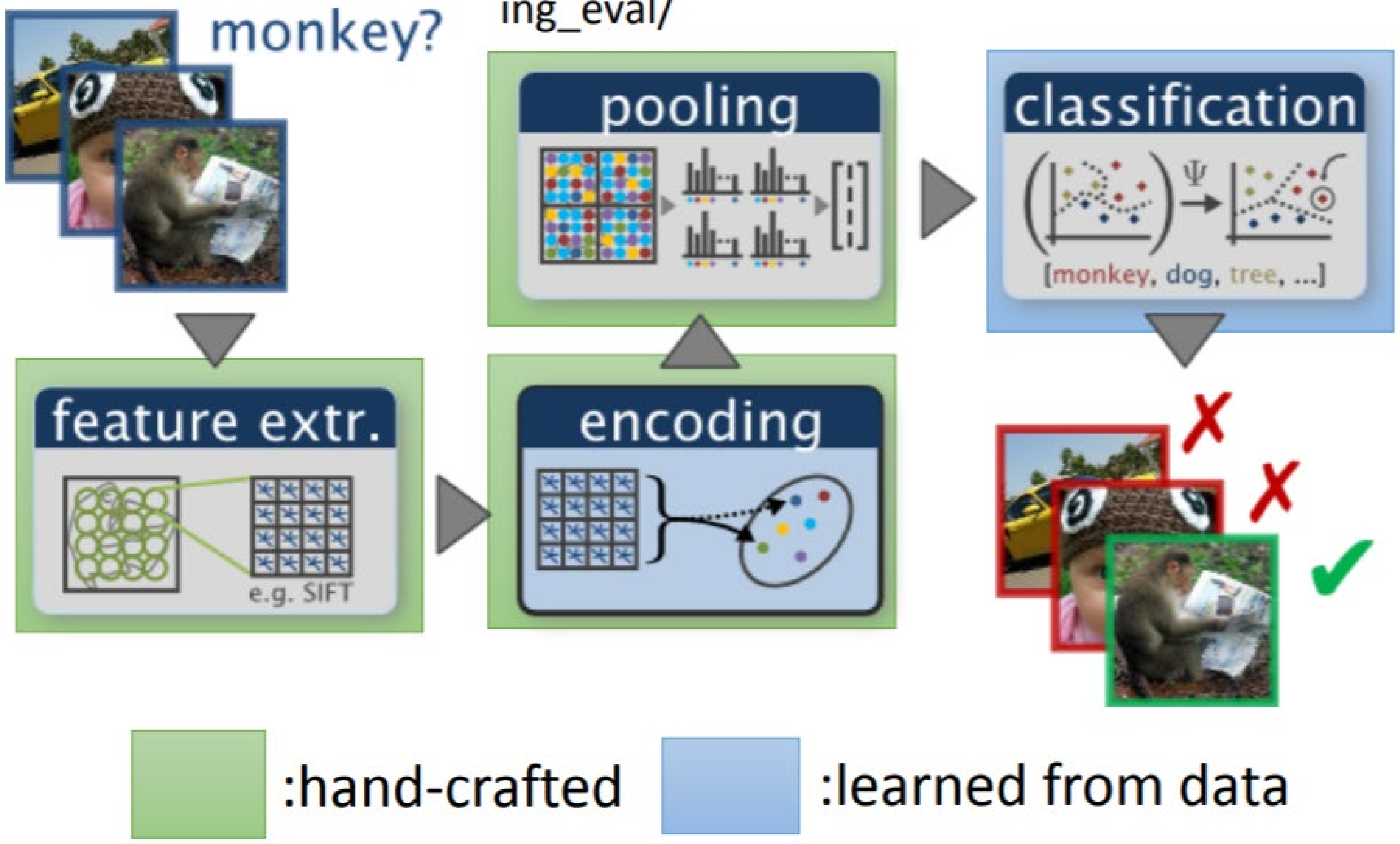
However, using deep structure is more effective.

Reference: 李弘毅 ML Lecture 11 https://youtu.be/XsC9byQkUH8

Reason 2: End-to-end learning

- Shallow Approach

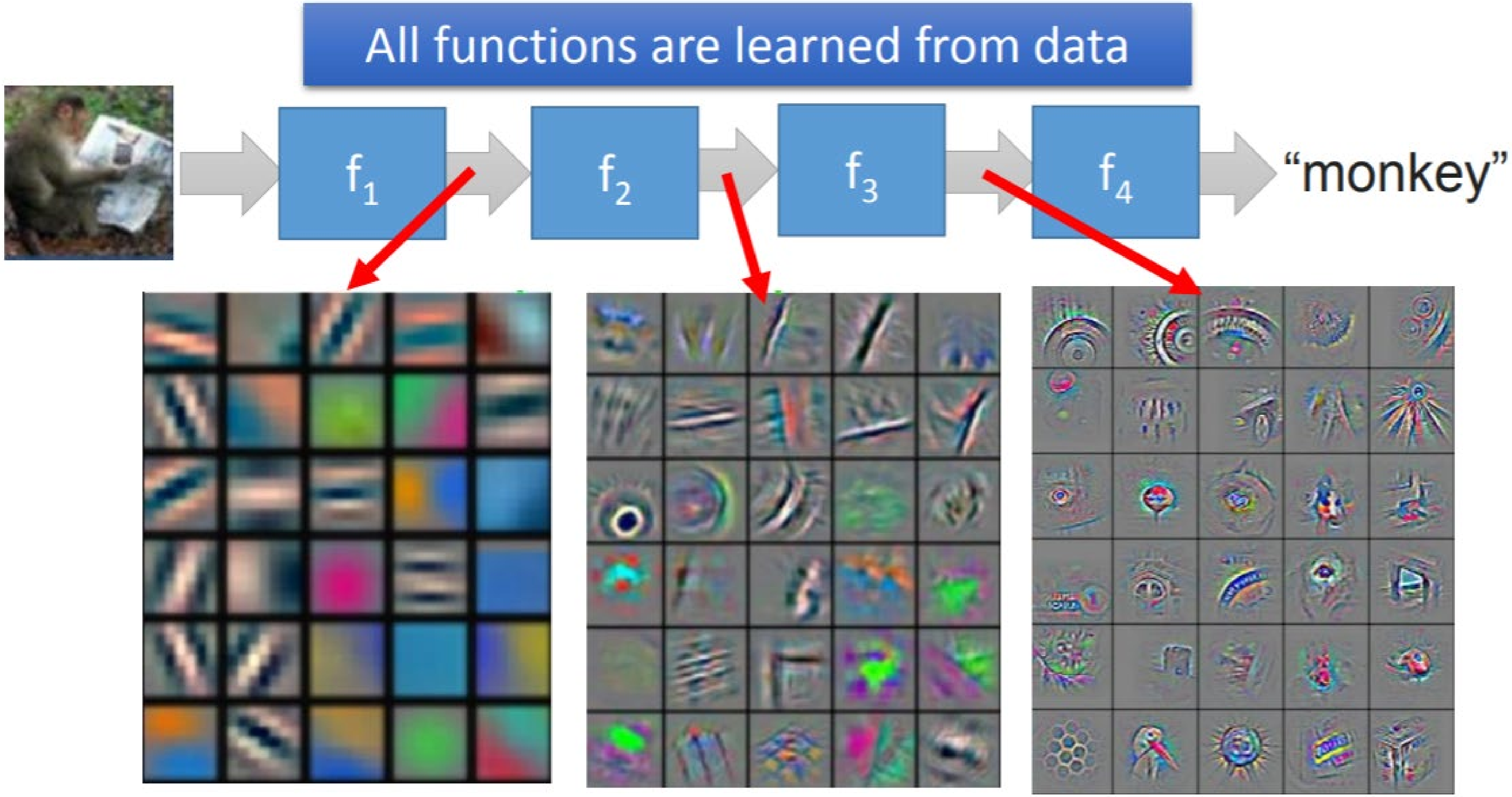
http://www.robots.ox.ac.uk/~vgg/research/encoding_eval/



Reference: 李弘毅 ML Lecture 11 https://youtu.be/XsC9byQkUH8

End-to-end learning

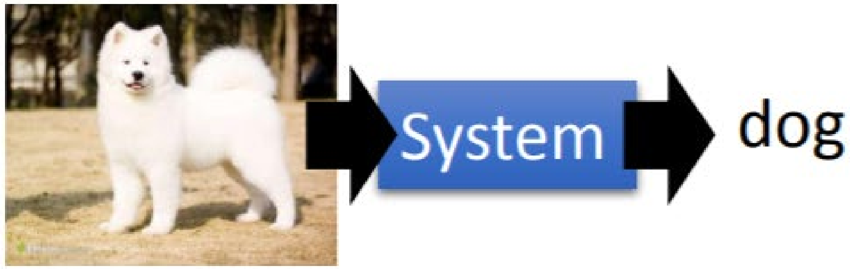
- Deep Learning



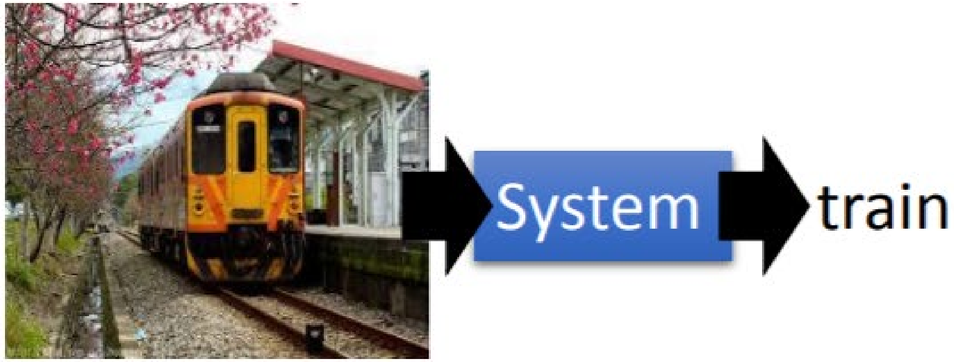
Reference: 李弘毅 ML Lecture 11 https://youtu.be/XsC9byQkUH8

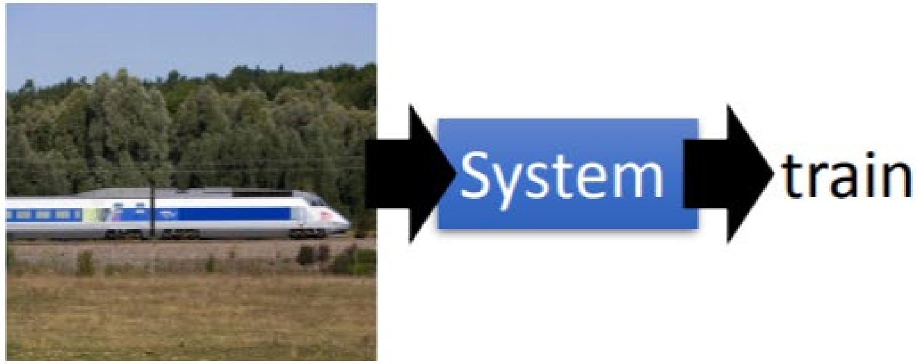
Reason 3 - Easier to handle complex task

- Very similar input, different output



- Very different input, similar output

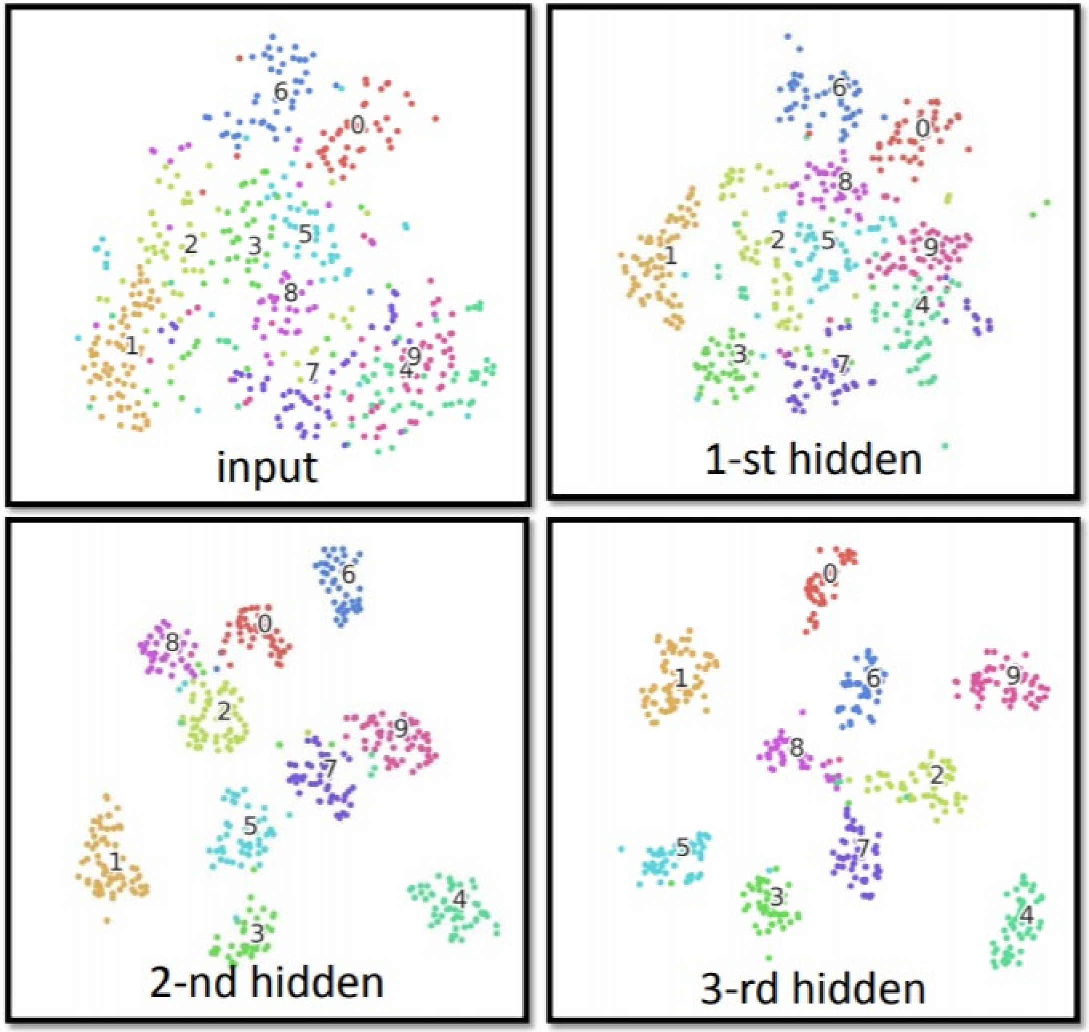




Reference: 李弘毅 ML Lecture 11 https://youtu.be/XsC9byQkUH8

Easier to handle complex task with DL

MNIST



How to implement this in PyTorch?

Reference: 李弘毅 ML Lecture 11 https://youtu.be/XsC9byQkUH8

What does CNN learn?

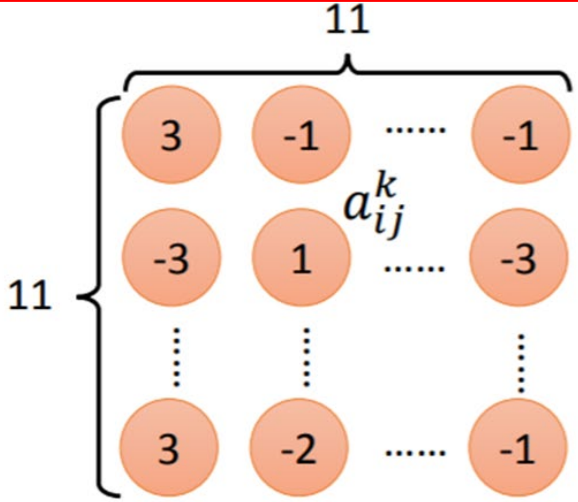
Only the weight of the 1st convolution filters can be directly visualized. How to interpret the filter weights of other convolution layers?

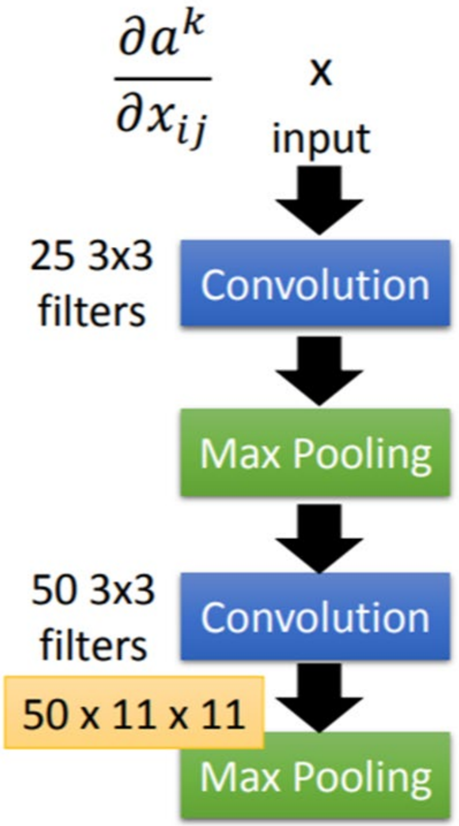
How to use gradient ascent to implement this in PyTorch?

The output of the k-th filter is a 11 x 11 matrix.

Degree of the activation of the k-th filter: $a^k = \sum_{i=1}^{11}\sum_{j=1}^{11} a_{ij}^k$

$$\boxed{x^* = arg\max_x a^k \quad \text{(gradient ascent)}}$$





Reference: 李弘毅 ML Lecture 10 https://youtu.be/FrKWiRv254g

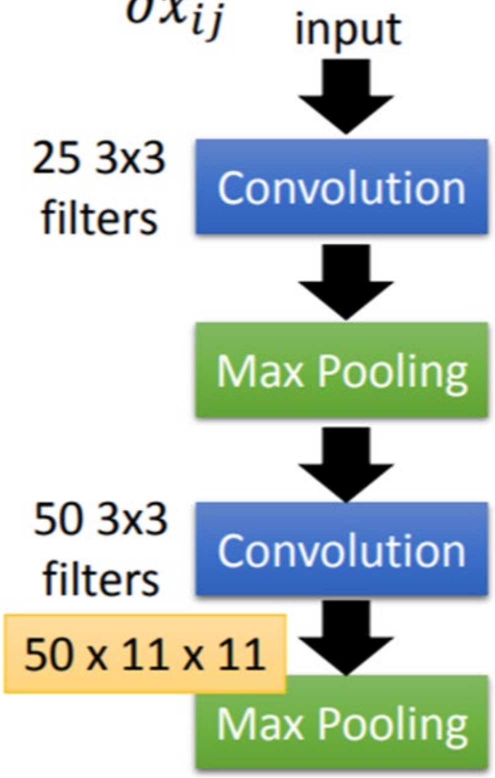
With MNIST data set, in the convolution layer, the filters detects a particular texture pattern.

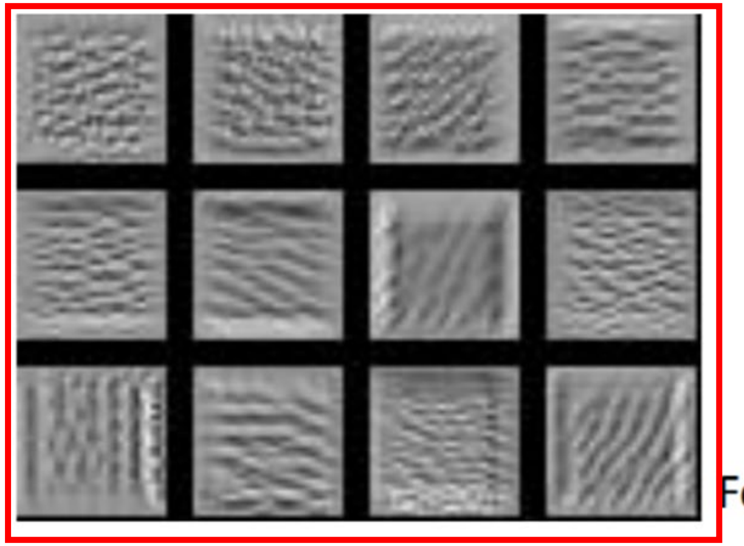
The output of the k-th filter is a 11 x 11 matrix.

Degree of the activation of the k-th filter: $a^k = \sum_{i=1}^{11} \sum_{j=1}^{11} a_{ij}^k$

$x^* = arg \max_x a^k$ (gradient ascent)

$\frac{\partial a^k}{\partial x_{ij}}$





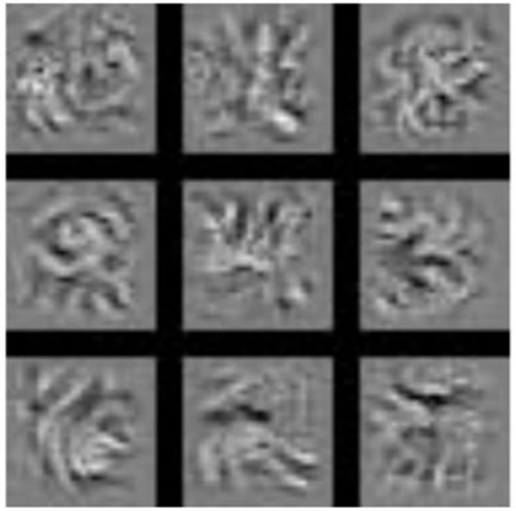
For each filter

How to implement this in PyTorch?

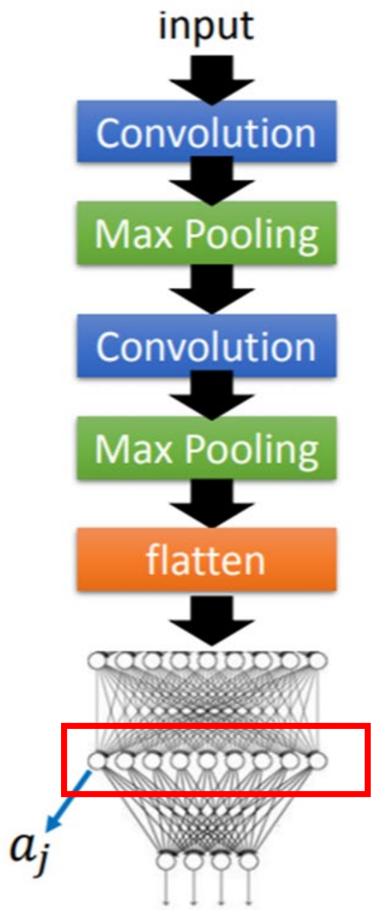
Reference: 李弘毅 ML Lecture 10 https://youtu.be/FrKWiRv254g

In the hidden layer of the fully-connected NN, each neuron detects an overall pattern in the picture rather than a particular texture pattern.

Find an image maximizing the output of neuron: $x^* = arg \max_x a^j$



Each figure corresponds to a neuron

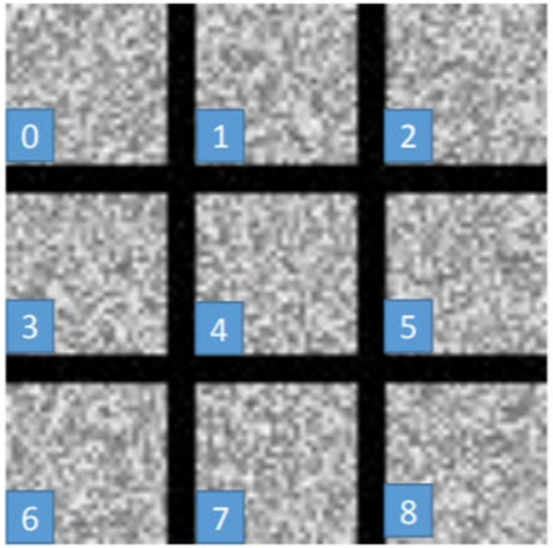


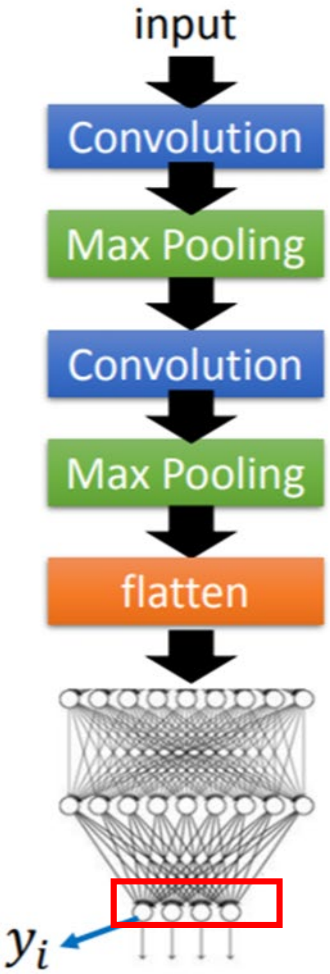
Reference: 李弘毅 ML Lecture 10 https://youtu.be/FrKWiRv254g

If we watch the output layer node, it is easy to see that CNN is easily fooled.

$$x^* = arg\max_x y^i$$

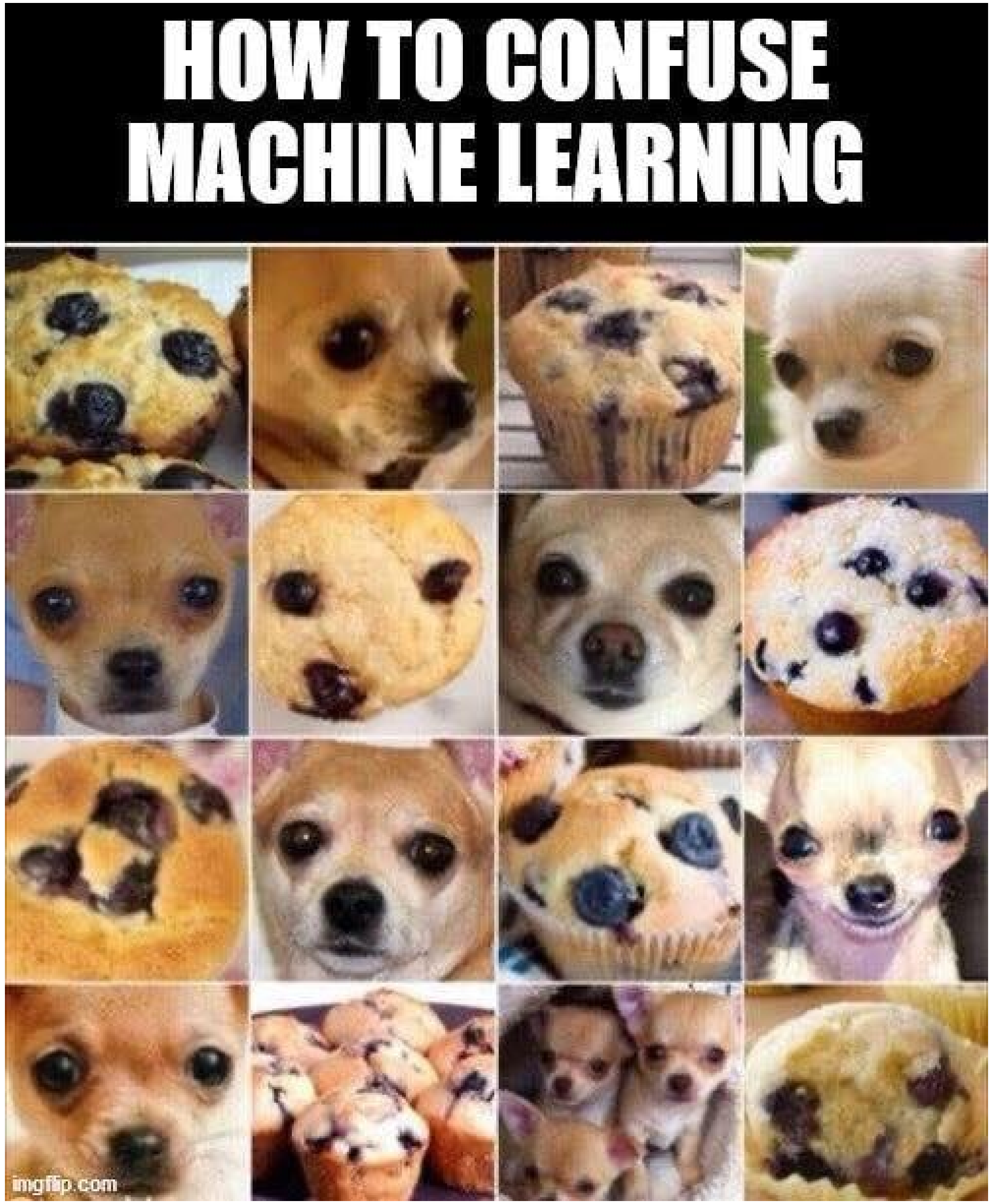
Can we see digits?





Deep Neural Networks are Easily Fooled
https://www.youtube.com/watch?v=M2IebCN9Ht4

Reference: 李弘毅 ML Lecture 10 https://youtu.be/FrKWiRv254g

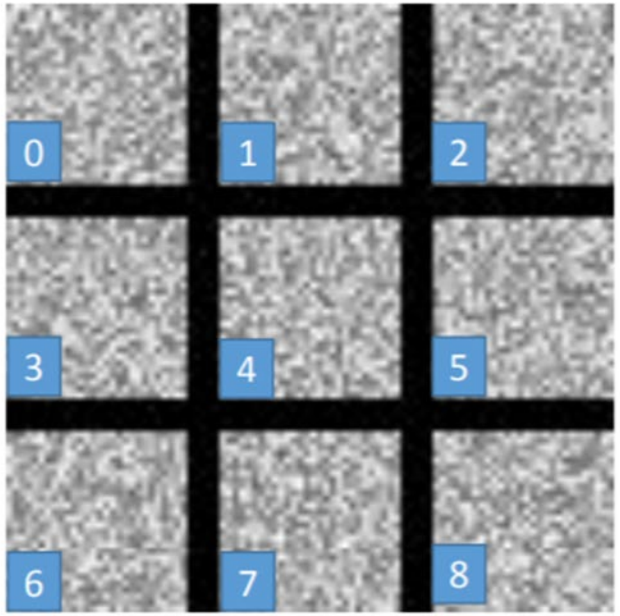


https://www.facebook.com/105366834575253/photos/a.106433371135266/217334733378462/

Adding regularization to the objective function to force most pixels be "NO INK"

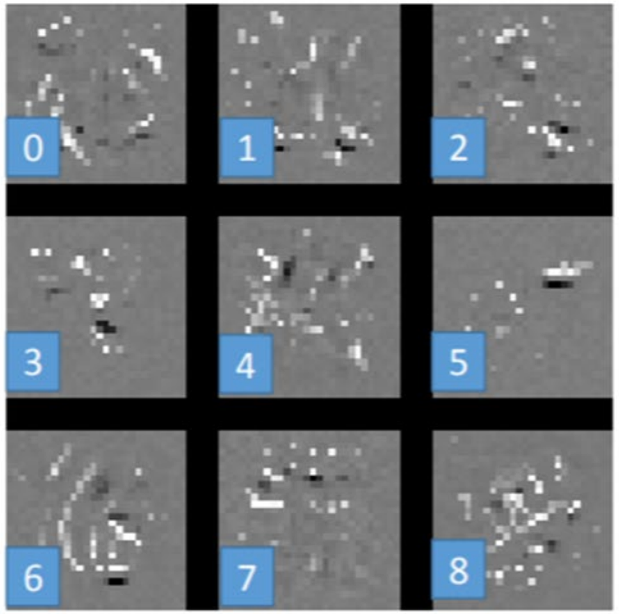
Here white pixels indicate ink, and black pixels indicate “NO INK”.

$$x^* = arg\max_x y^i$$



Over all pixel values

$$x^* = arg\max_x \left(y^i - \boxed{\sum_{i,j} |x_{ij}|} \right)$$

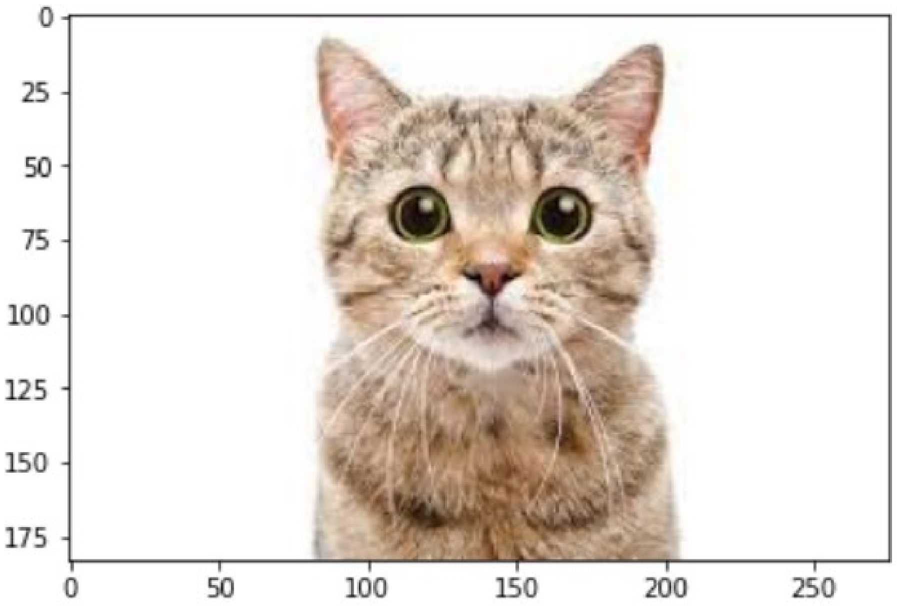


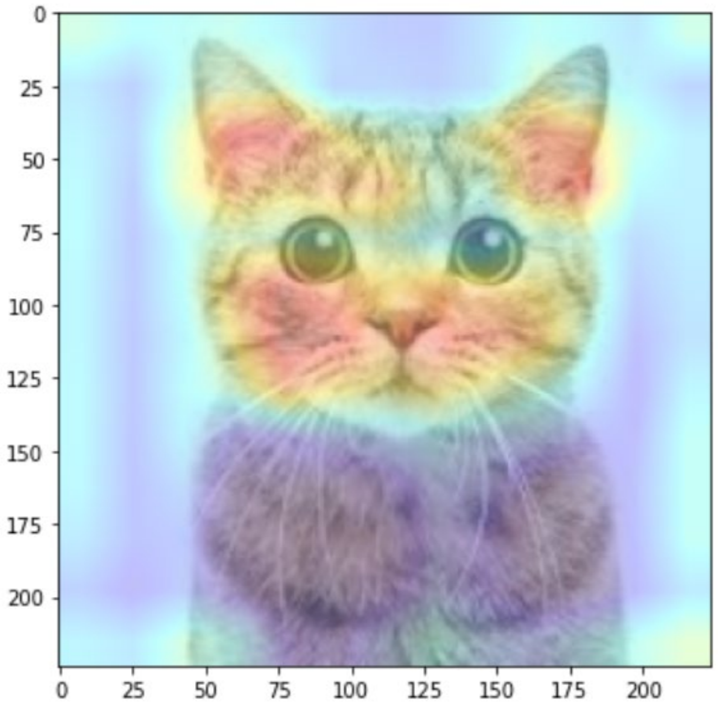
L1 regularization to force xij=0, i.e., force most pixels to be black, NO INK (as only small part of the image has ink)

Reference: 李弘毅 ML Lecture 10 https://youtu.be/FrKWiRv254g

Practice – What does CNN learn?

- Run "7.3 GradCAM.ipynb"





HW5 (3)

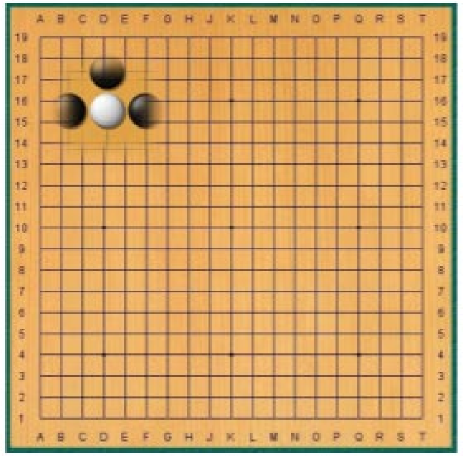
	Class index predicted by the model	Class index you assigned
AlexNet		
VGG		
ResNet18		

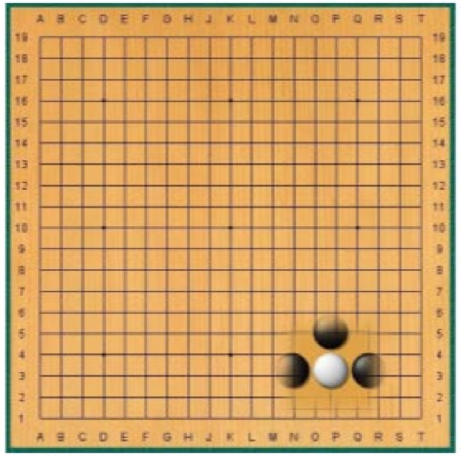
Use CNN in Alpha GO

- Some patterns are much smaller than the whole image

Alpha Go uses 5 x 5 for first layer

- The same patterns appear in different regions.





Reference: 李弘毅 ML Lecture 10 https://youtu.be/FrKWiRv254g

Use CNN in Alpha GO

Neural network architecture. The input to the policy network is a $19 \times 19 \times 48$ image stack consisting of 48 feature planes. The first hidden layer zero pads the input into a 23×23 image, then convolves k filters of kernel size 5×5 with stride 1 with the input image and applies a rectifier nonlinearity. Each of the subsequent hidden layers 2 to 12 zero pads the respective previous hidden layer into a 21×21 image, then convolves k filters of kernel size 3×3 with stride 1, again followed by a rectifier nonlinearity. The final layer convolves 1 filter of kernel size 1×1 with stride 1, with a different bias for each position, and applies a softmax function. The

Alpha Go does not use Max Pooling ……

Extended Data Table 3 additionally show the results of training with $k = 128$, 256 and 384 filters.