SUMMER SCHOOL 2020

MACHINE LEARNING DAY 2

DEEP LEARNING

Session IV: Convolutional Neural Network

Isaac Ye, HPTC @ York University

Isaac@sharcnet.ca

Session IV

- Backward propagation
- Model capacity / overfitting
- MNIST classification
- Vanishing gradient problem
- *Lab 4A: Multi-Layer Perceptron (MNIST)*
- Issue with MLP
- Convolutional Neural Network
- Techniques
- *Lab 4B: Convolutional Neural Network (CIFAR10)*

Supervised learning

Supervised

Discrete	**Classification**
Continuous	**Regression**

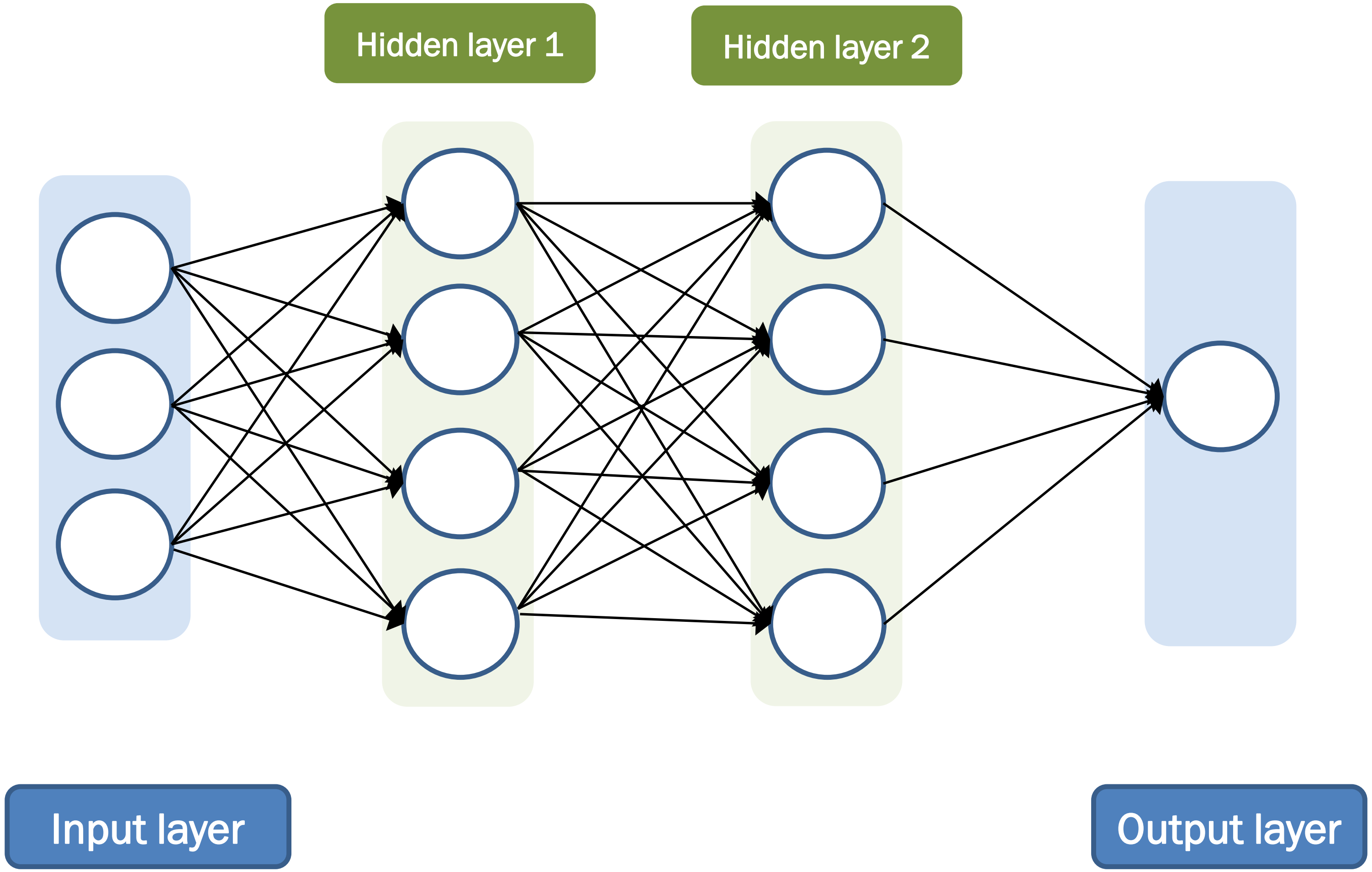
Supervised

Input	Data (x, y) x: data, y: label
Goal	learn a function to map x to y
Examples	classification, regression, object detection semantic segmentation, image captioning

Categories of ML problems

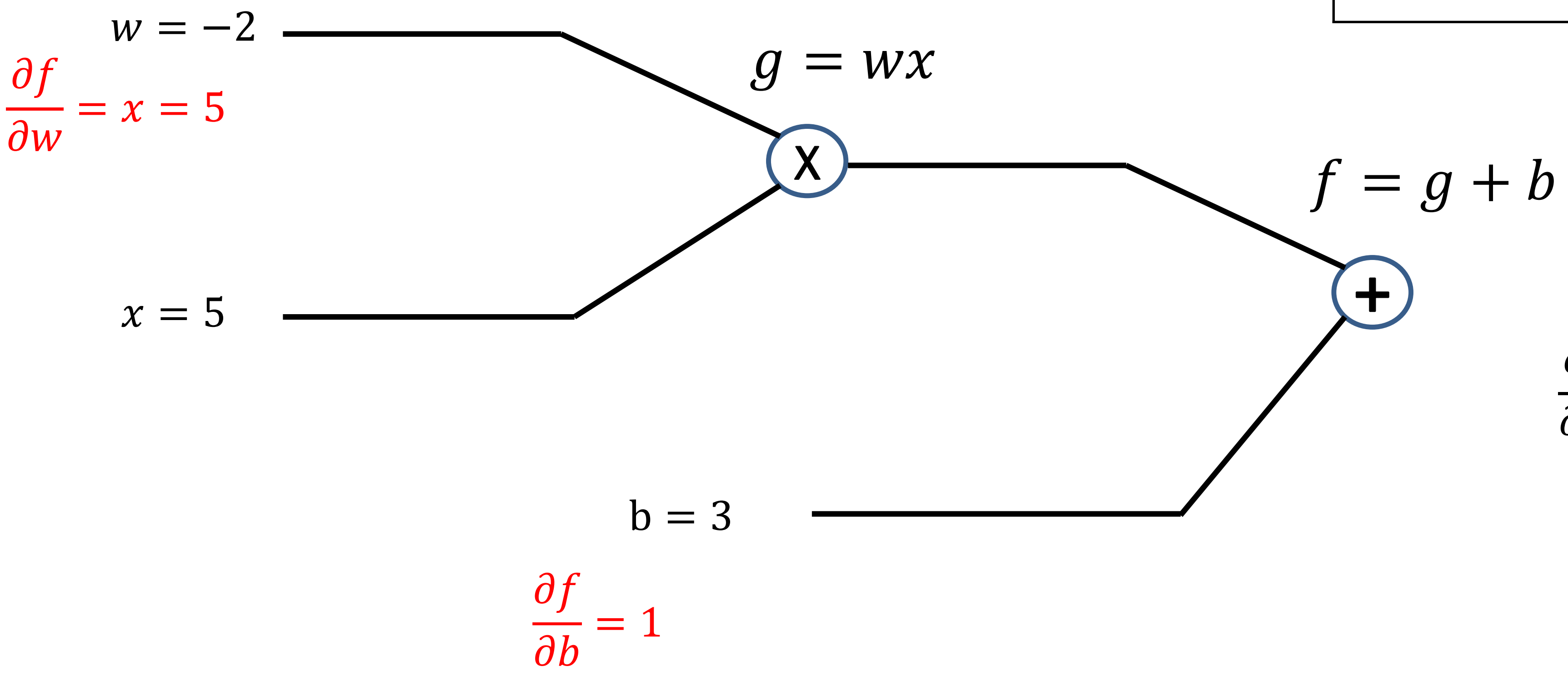
	Supervised	Unsupervised	Reinforcement
Discrete	Classification	Clustering	Action space agent
Continuous	Regression	Dimensionality reduction	Action space agent

Multi-Layer Perceptron



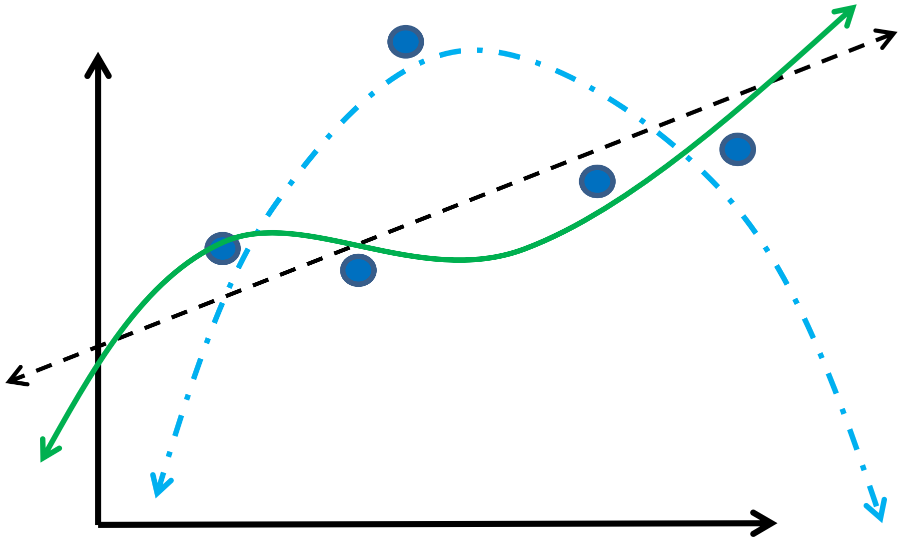
Backward propagation

$g = wx$	$f = g + b$
$\frac{\partial g}{\partial w} = x$, $\frac{\partial g}{\partial x} = w$	$\frac{\partial f}{\partial g} = 1$, $\frac{\partial f}{\partial b} = 1$



$$\frac{\partial f}{\partial w} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial w} = 1x$$

Model capacity



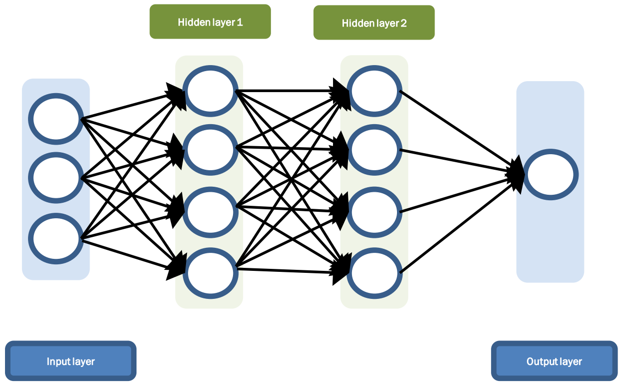
$$y = w_1 x + b$$

$$y = w_2 x^2 + w_1 x + b$$

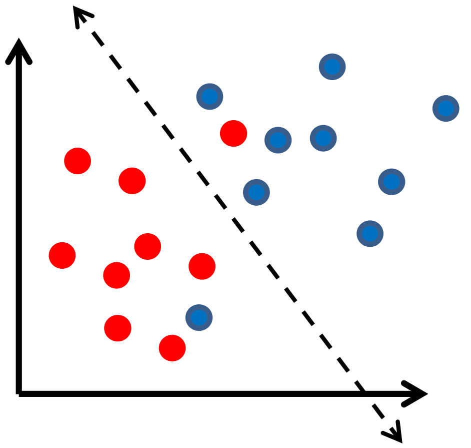
$$y = w_3 x^3 + w_2 x^2 + w_1 x + b$$

The more hidden layers and units,
the higher model capacity it will have.

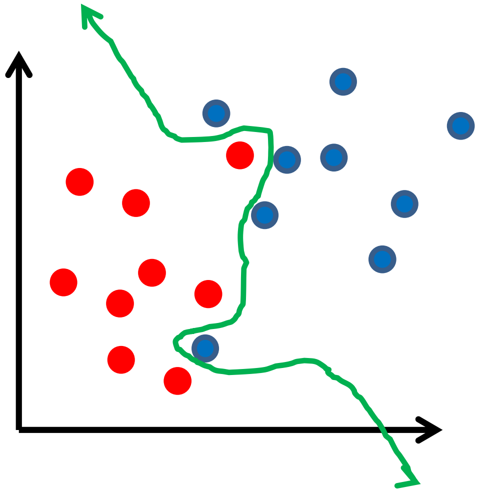
<u>**Does it guarantee better accuracy?**</u>



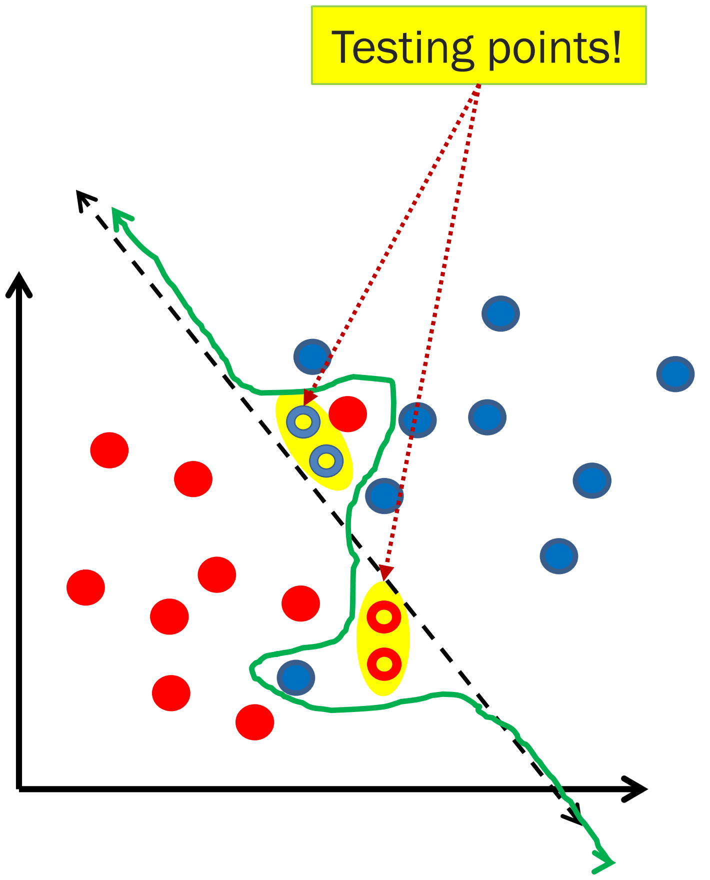
Overfitting

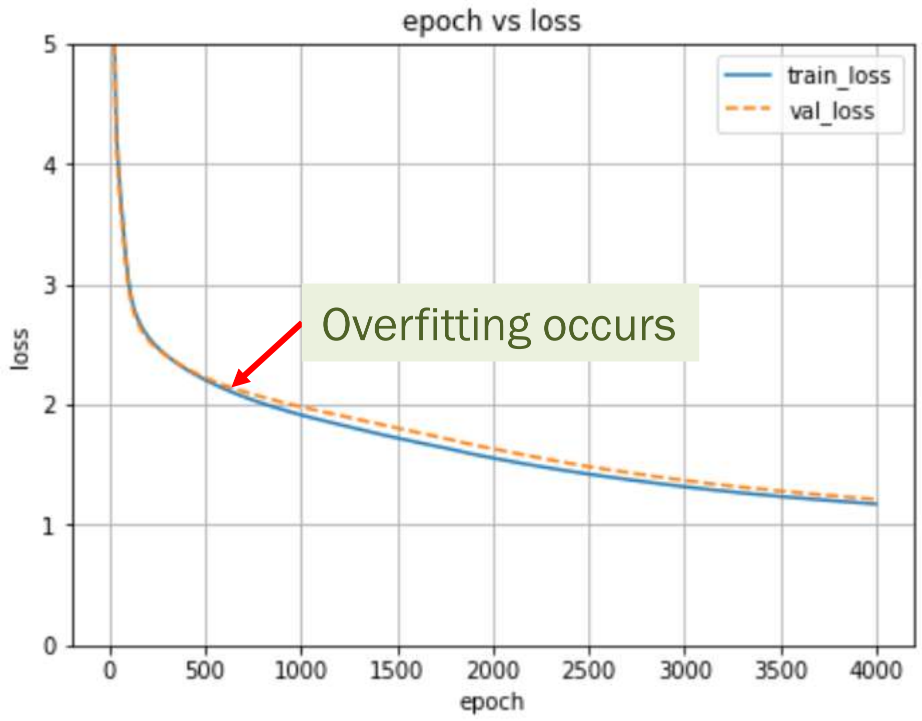


Lower model capacity

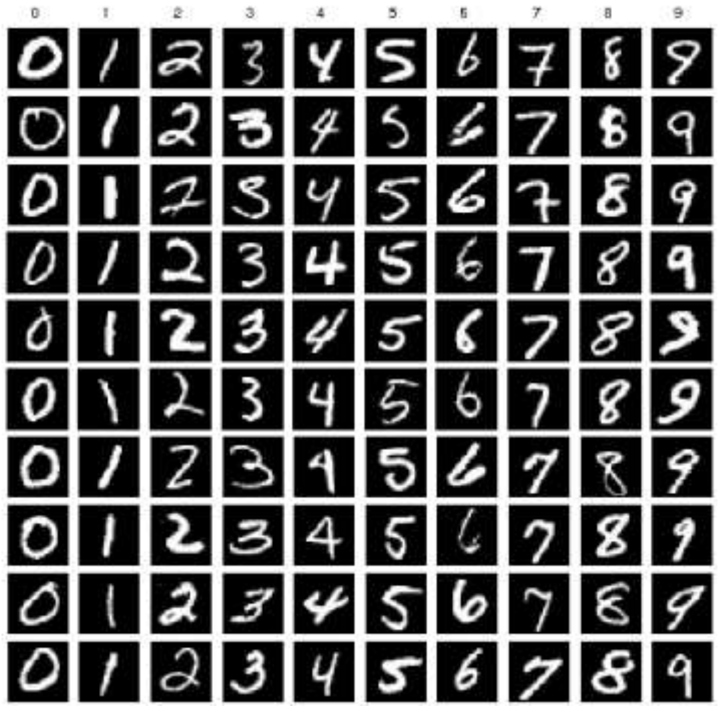


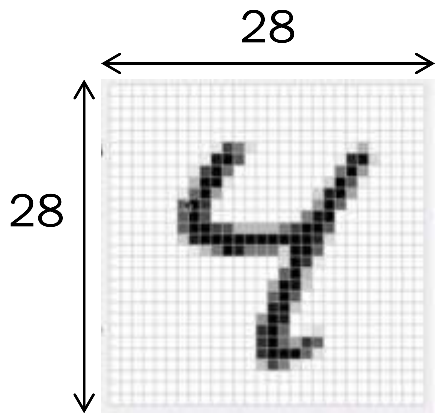
Higher model capacity

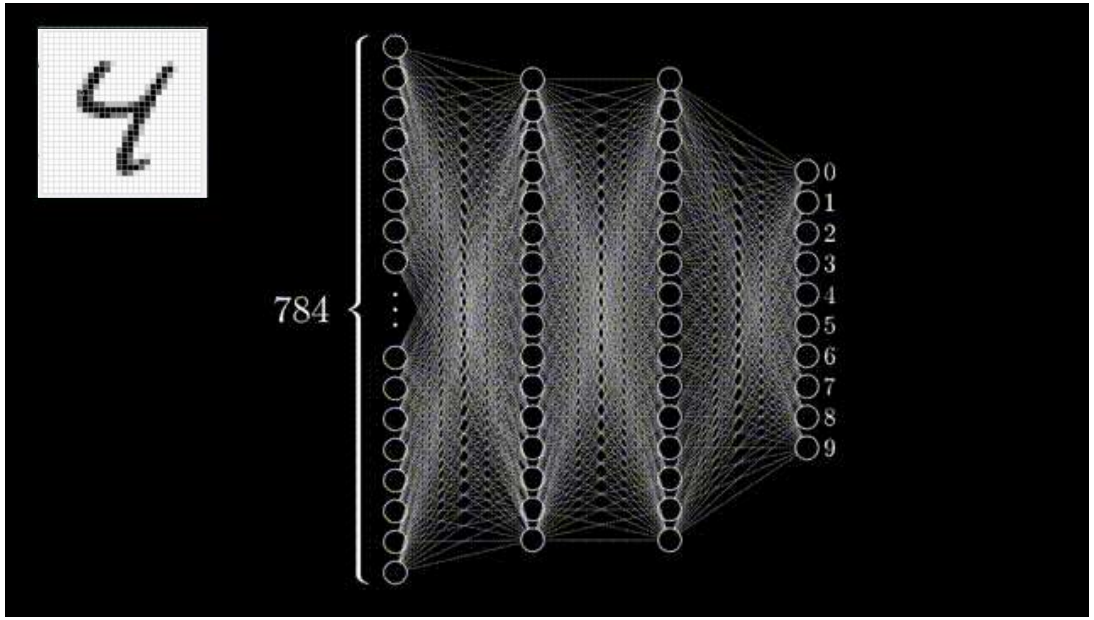




Classification problem: MNIST

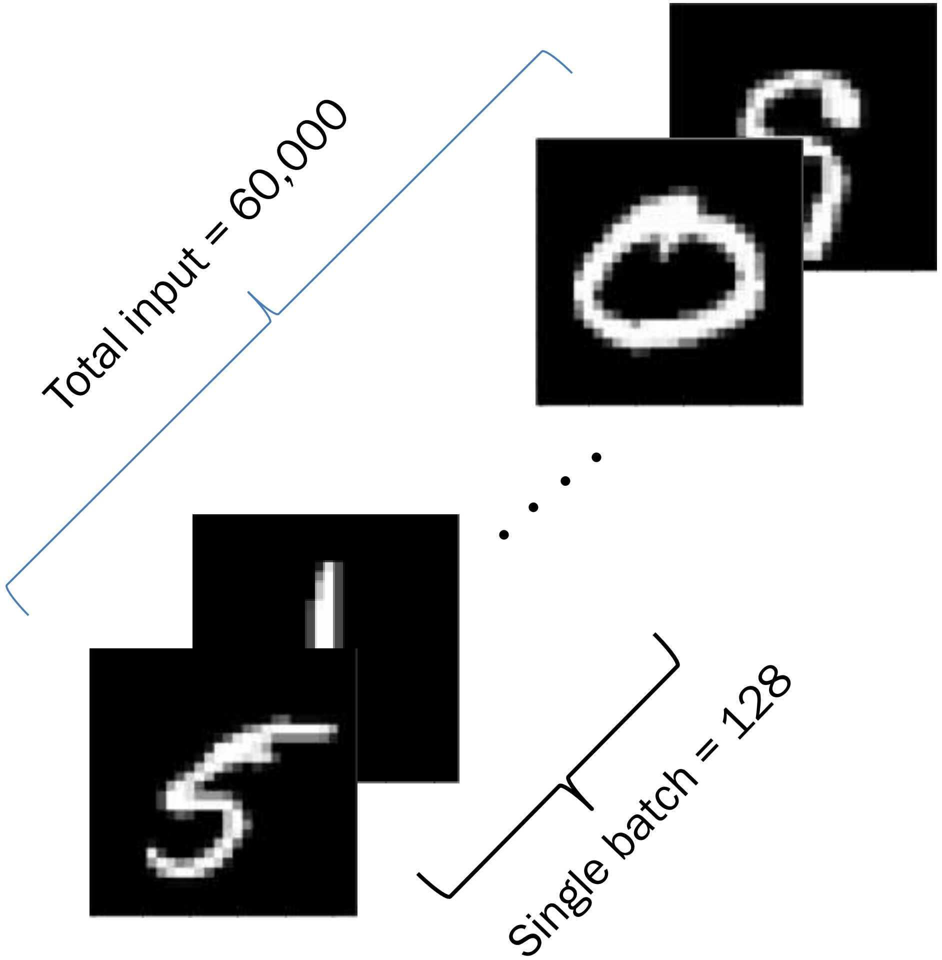




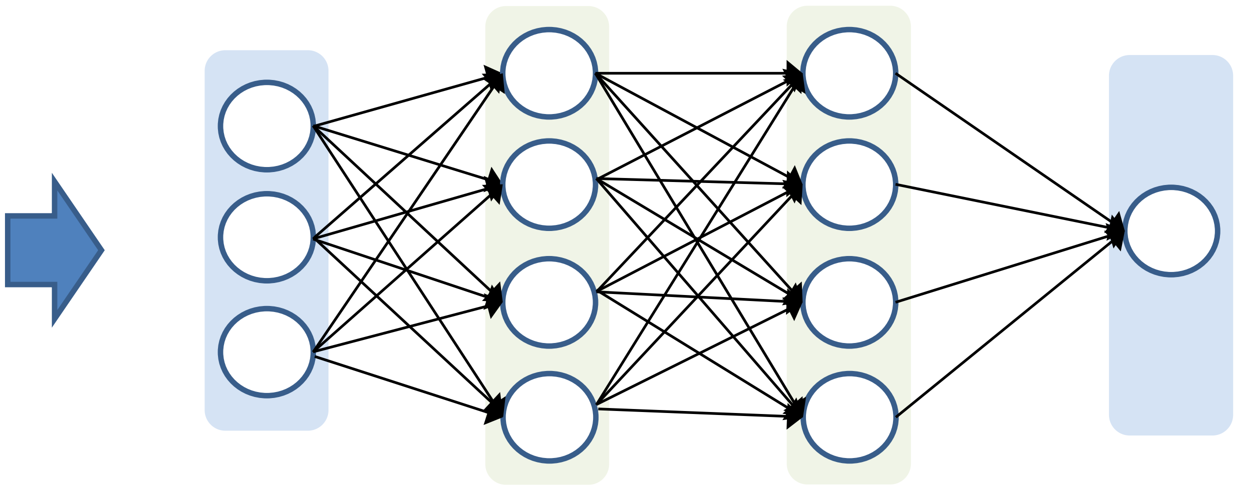


Handwritten data
60K train set and 10K test set
Each image has a size of 28x28 (=784)

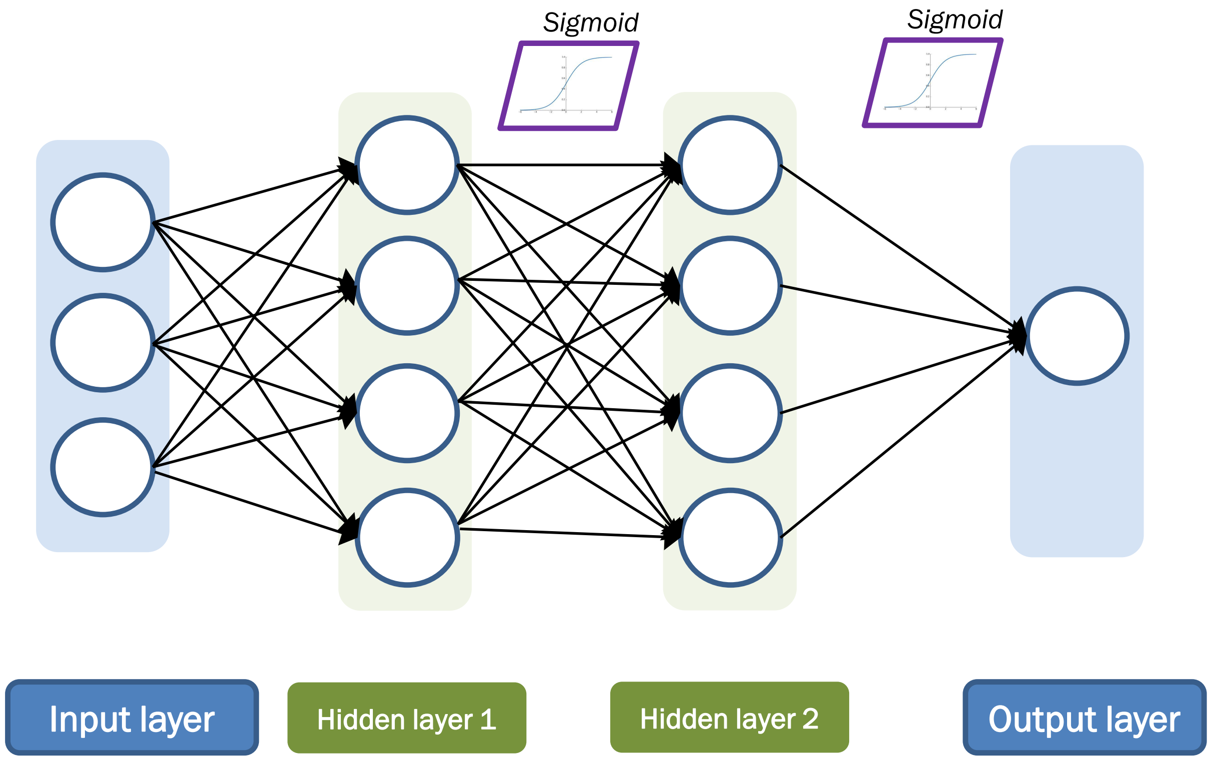
Batch, Iteration, Epoch

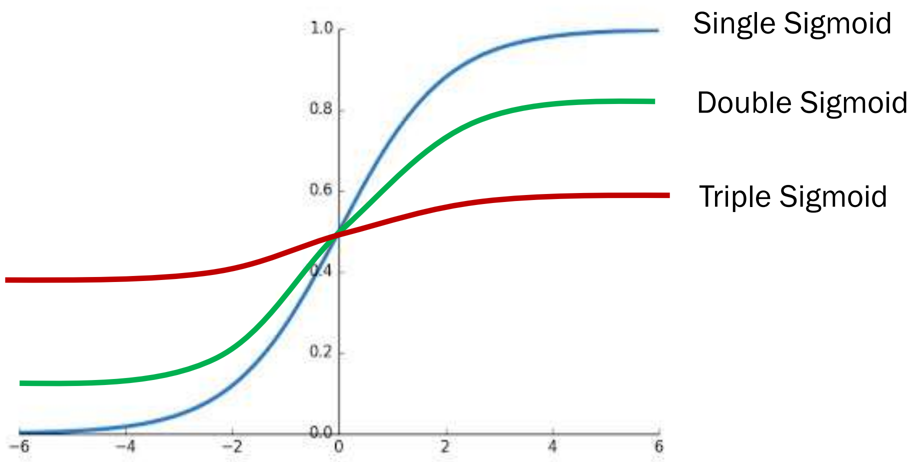


Epoch: one complete run of total input
Batch size: the amount of input for each iteration
# of iteration = # total input / batch size

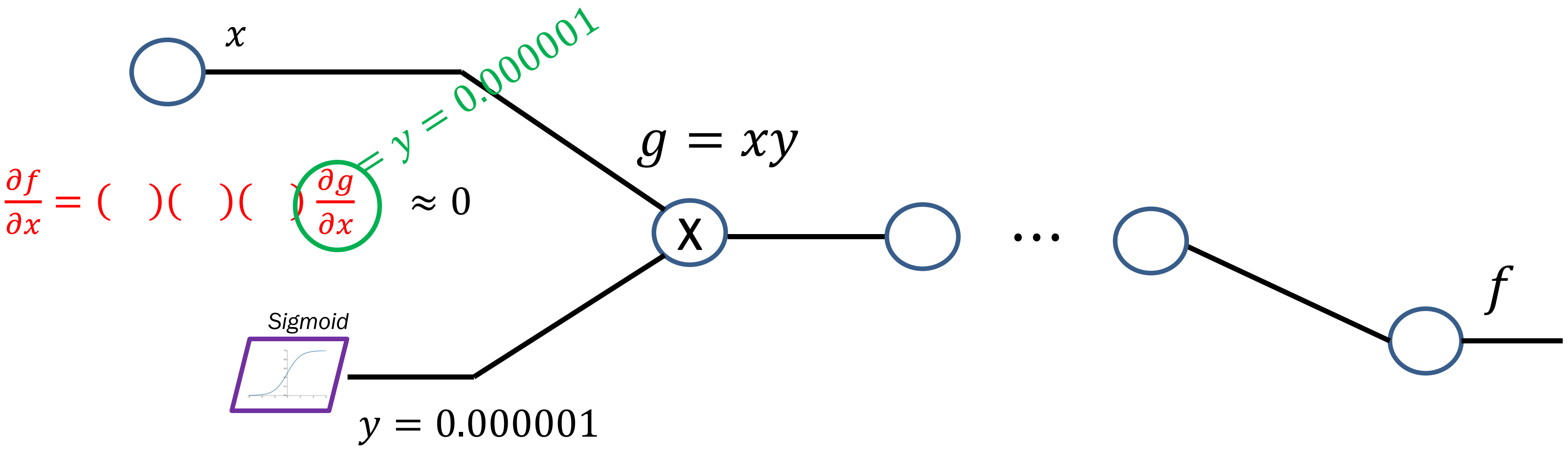


Activation function: Sigmoid problem

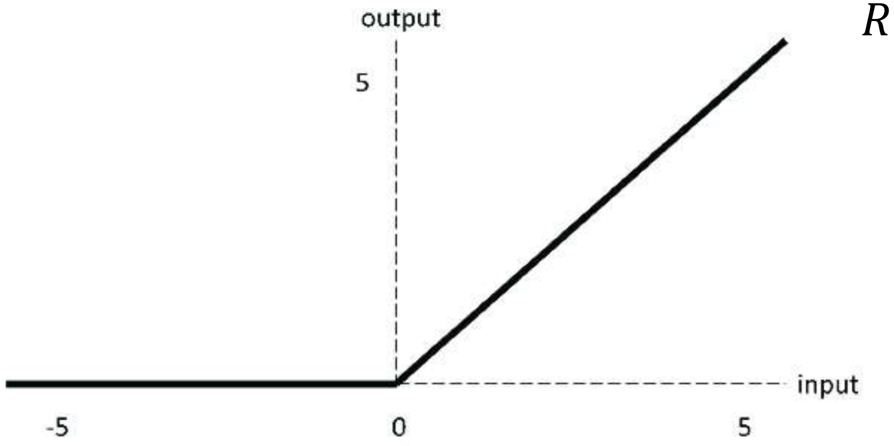




Vanishing gradient problem



Activation functions: Rectified Linear Unit (ReLU)



$$R(z) = \max(0, z)$$

Lab 4A: MNIST classification - MLP

Exercise 1: MLP vs. Linear model	Model
Run 1	MLP
Run 2	Linear model

Exercise 2: more layers w/ 100 units	# of layers
Run 1	2
Run 2	4

Exercise 3: Different units	# of units
Run 1	100
Run 2	200

Exercise 4: different learning rate	Learning rate
Run 1	0.002
Run 2	0.02

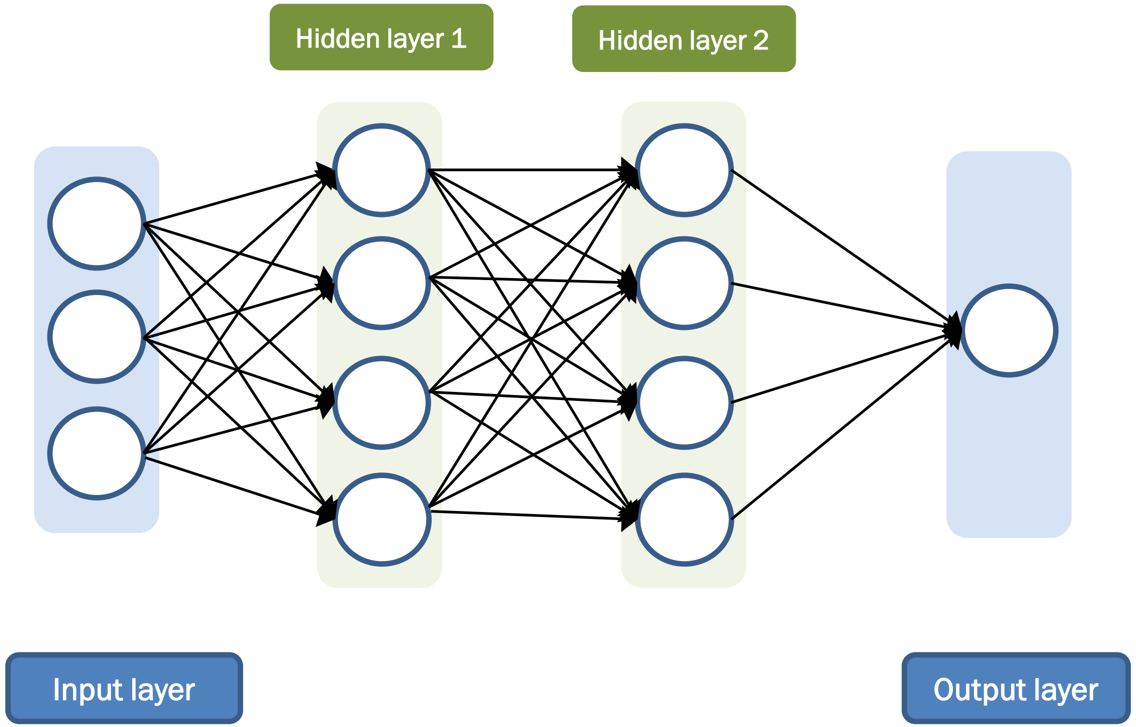
https://pytorch.org/docs/stable/nn.html
#non-linear-activations-weighted-sum-nonlinearity

You may want to try

1. Use different cost function
2. Use different optimizers
   https://pytorch.org/docs/stable/optim.html?highlight=optimizer
   #torch.optim.Optimizer

Break room

Issue with MLP

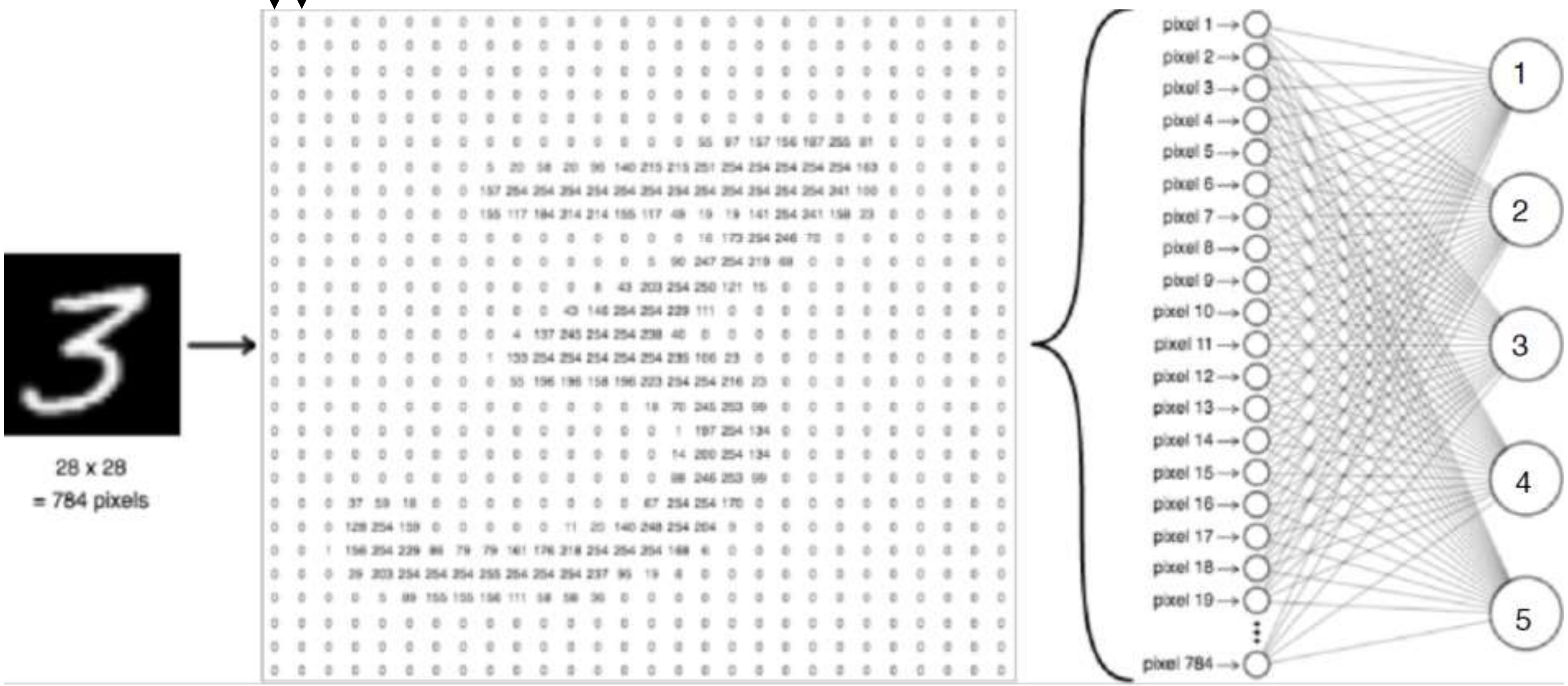


A neuron is connected with every neuron in next layer (fully connected)

# of parameters increases explosively

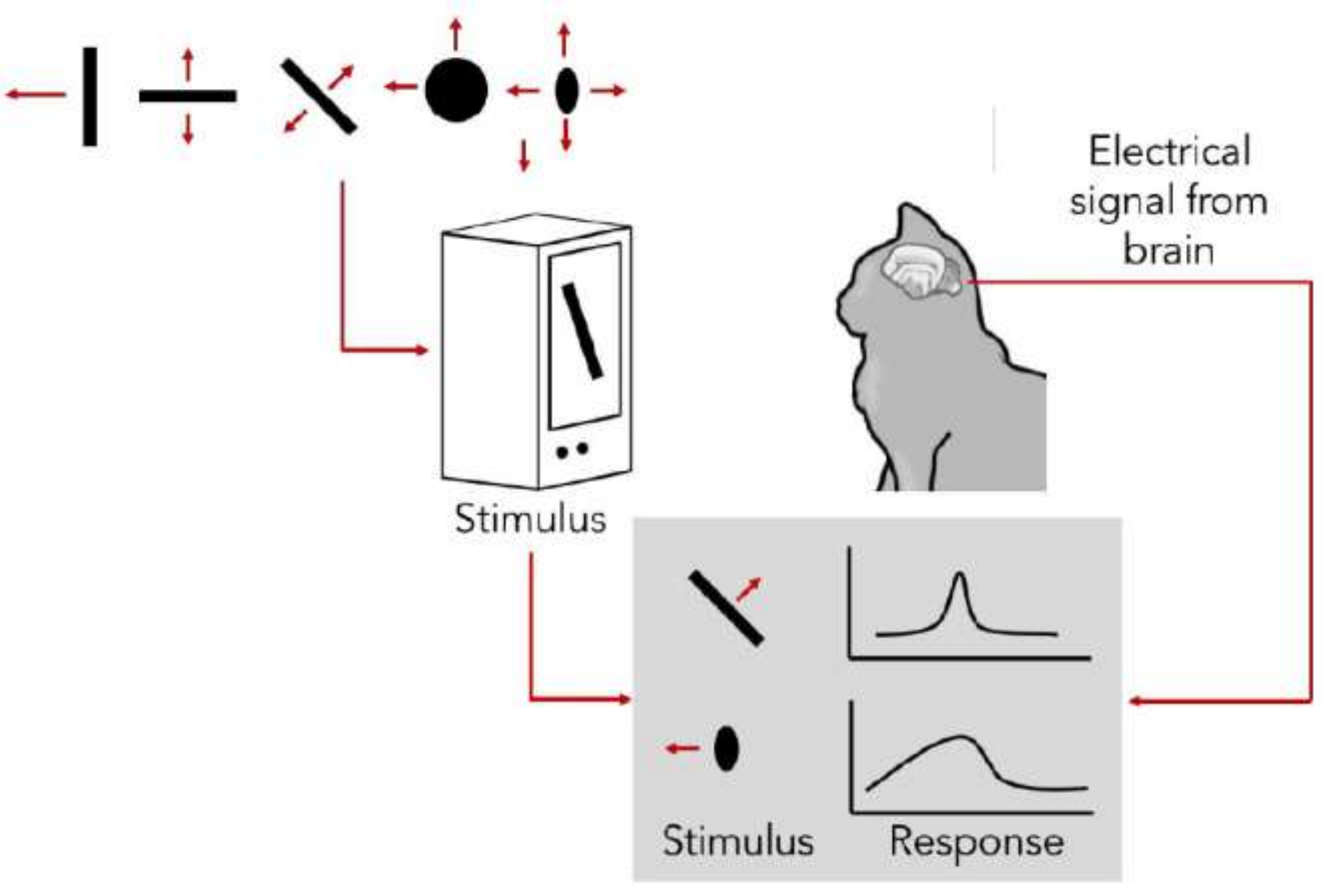
Issue with MLP

Not much meaningful info



Some of parameters are meaningless!

How to recognize an image?



Hierarchical structure

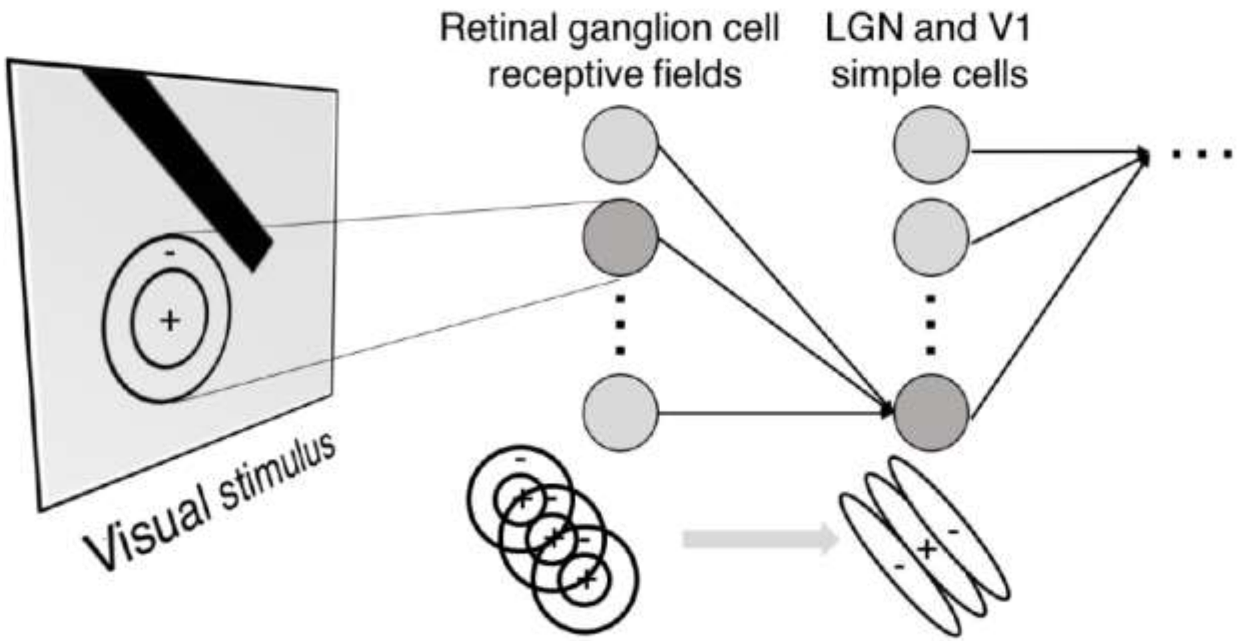
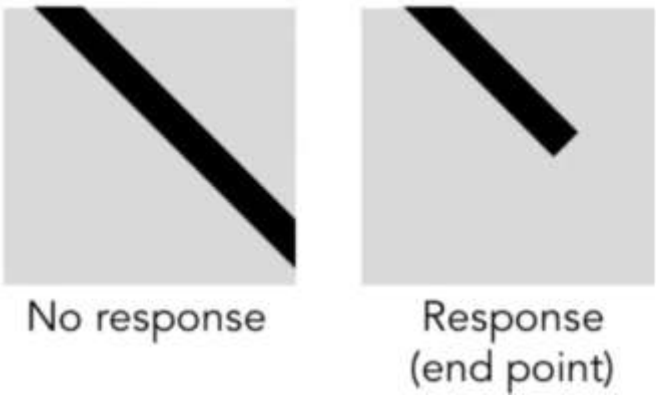


Illustration of hierarchical organization in early visual pathways by Lane McIntosh, copyright CS231n 2017

Simple cells: Response to light orientation

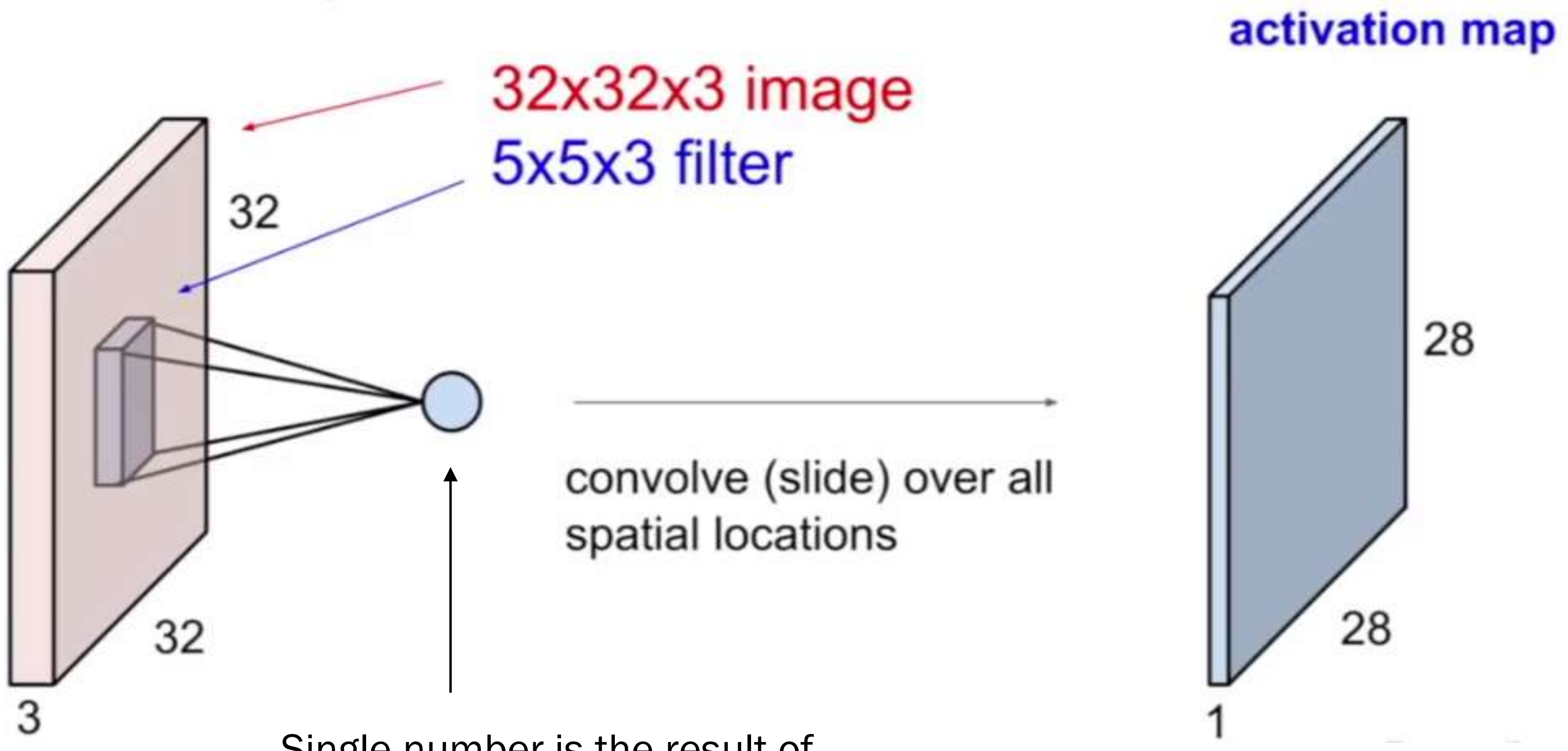
Complex cells: Response to light orientation and movement

Hypercomplex cells: Response to movement with an end point



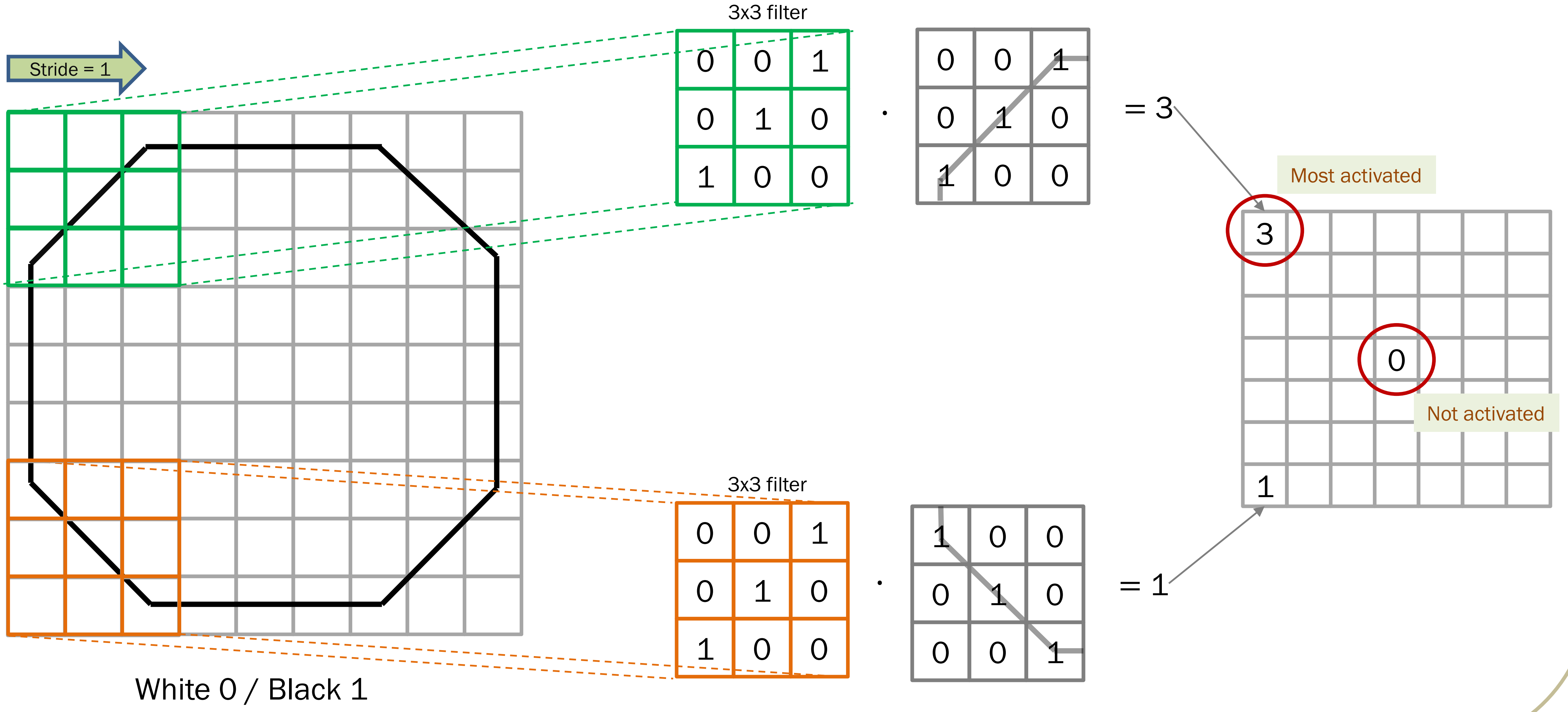
https://erickimphotography.com/blog/2018/09/30/hypercomplex-cells-how-does-visual-processing-work-in-our-brains/

Convolutional layer

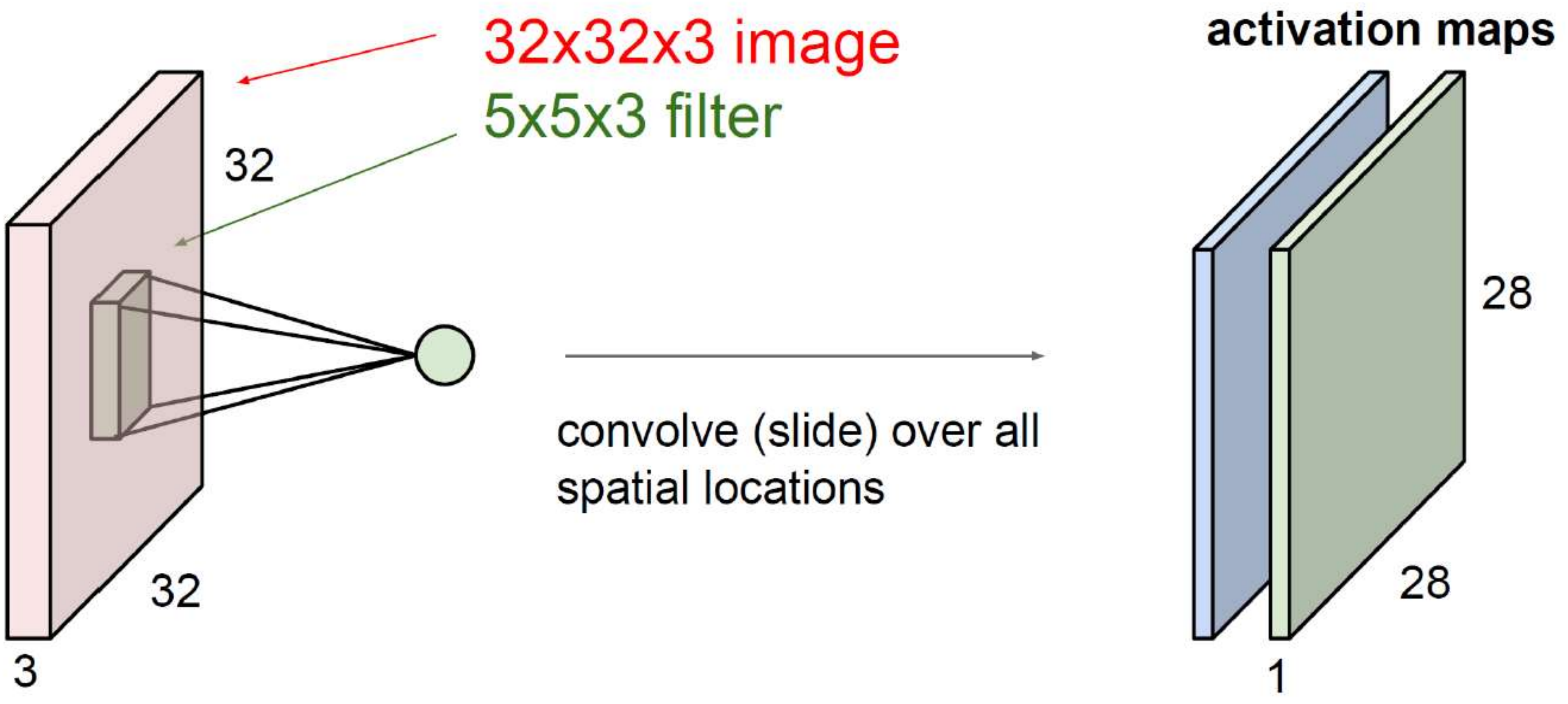


Single number is the result of taking a dot product between a filter and a small chunk (5x5x3) of the image plus bias

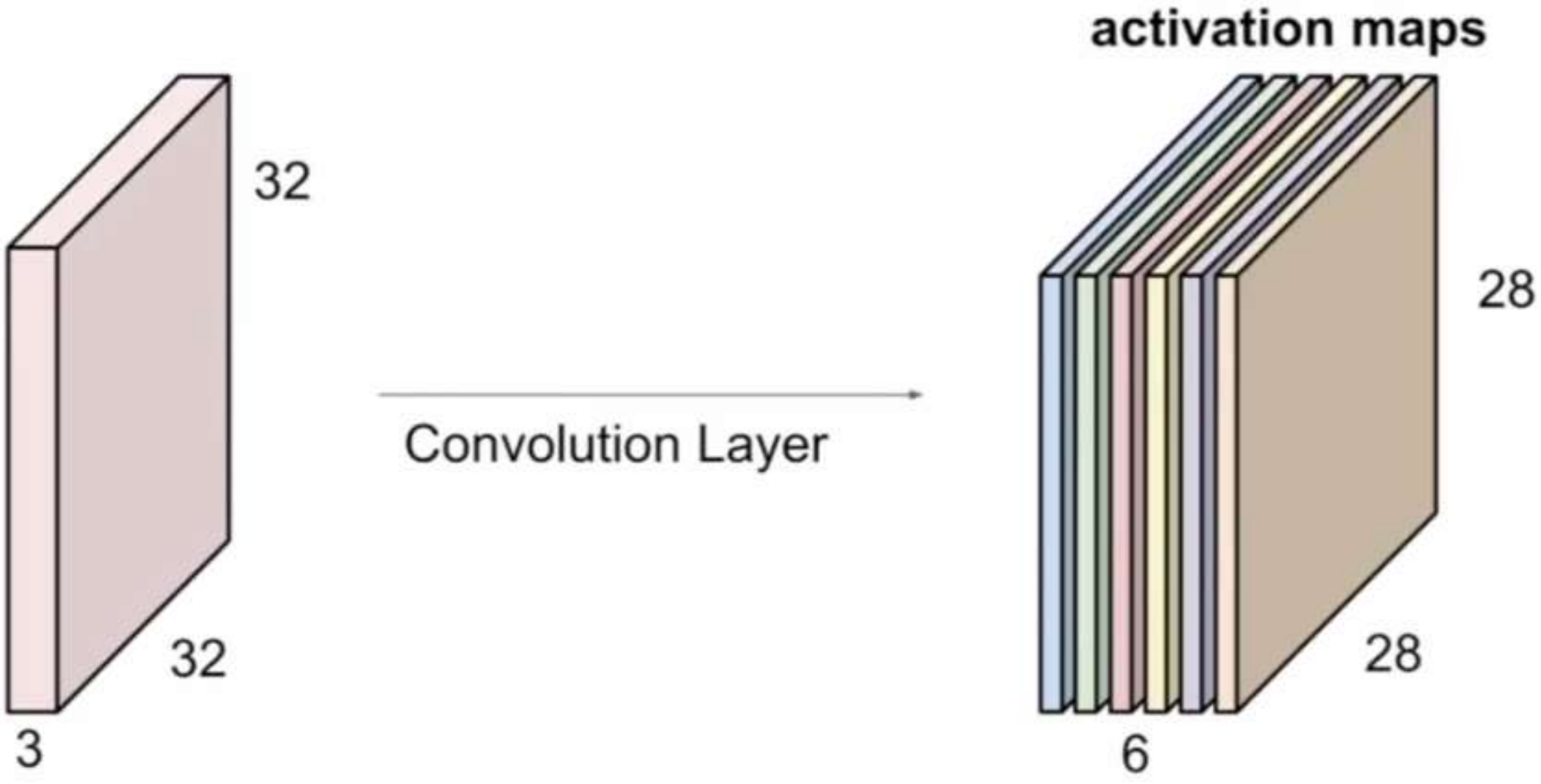
Convolutional operation



Convolutional layer

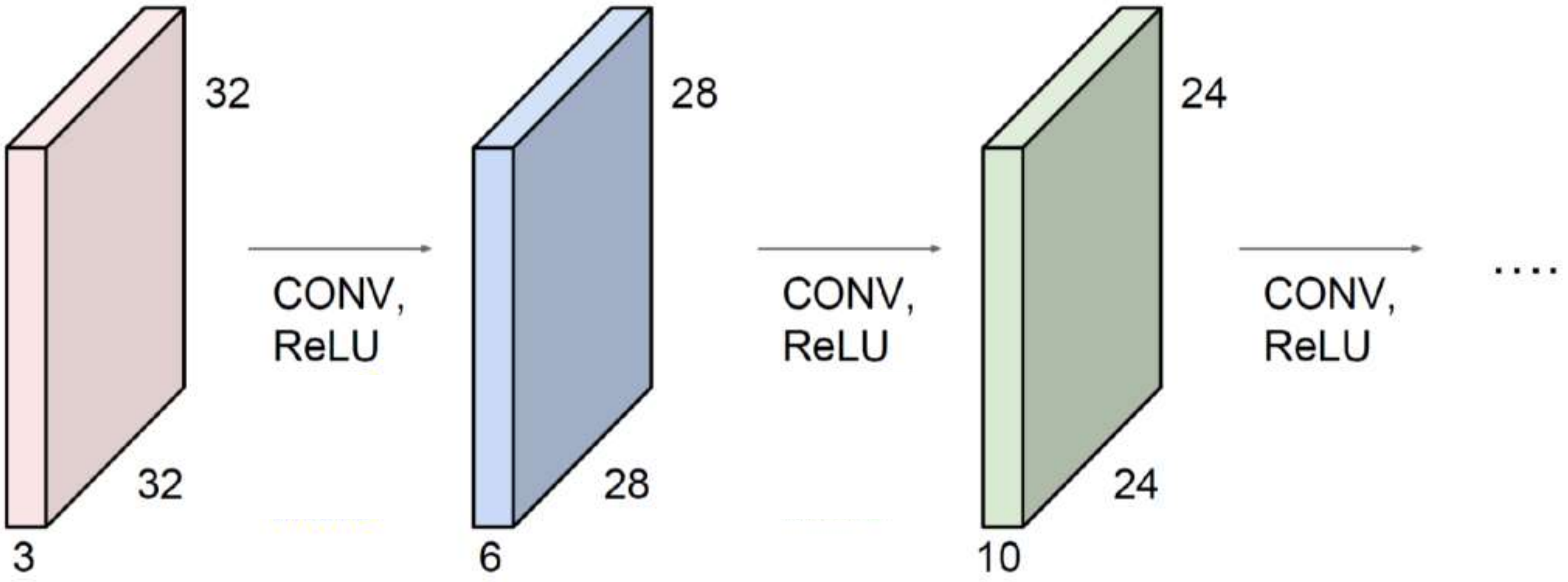


Convolutional layer



https://medium.com/dataseries/basic-overview-of-convolutional-neural-network-cnn-4fcc7dbb4f17

Convolutional Neural Network



Pooling

1	1	2	4
5	6	3	8
3	2	0	1
1	2	7	5

4x4

Max pooling

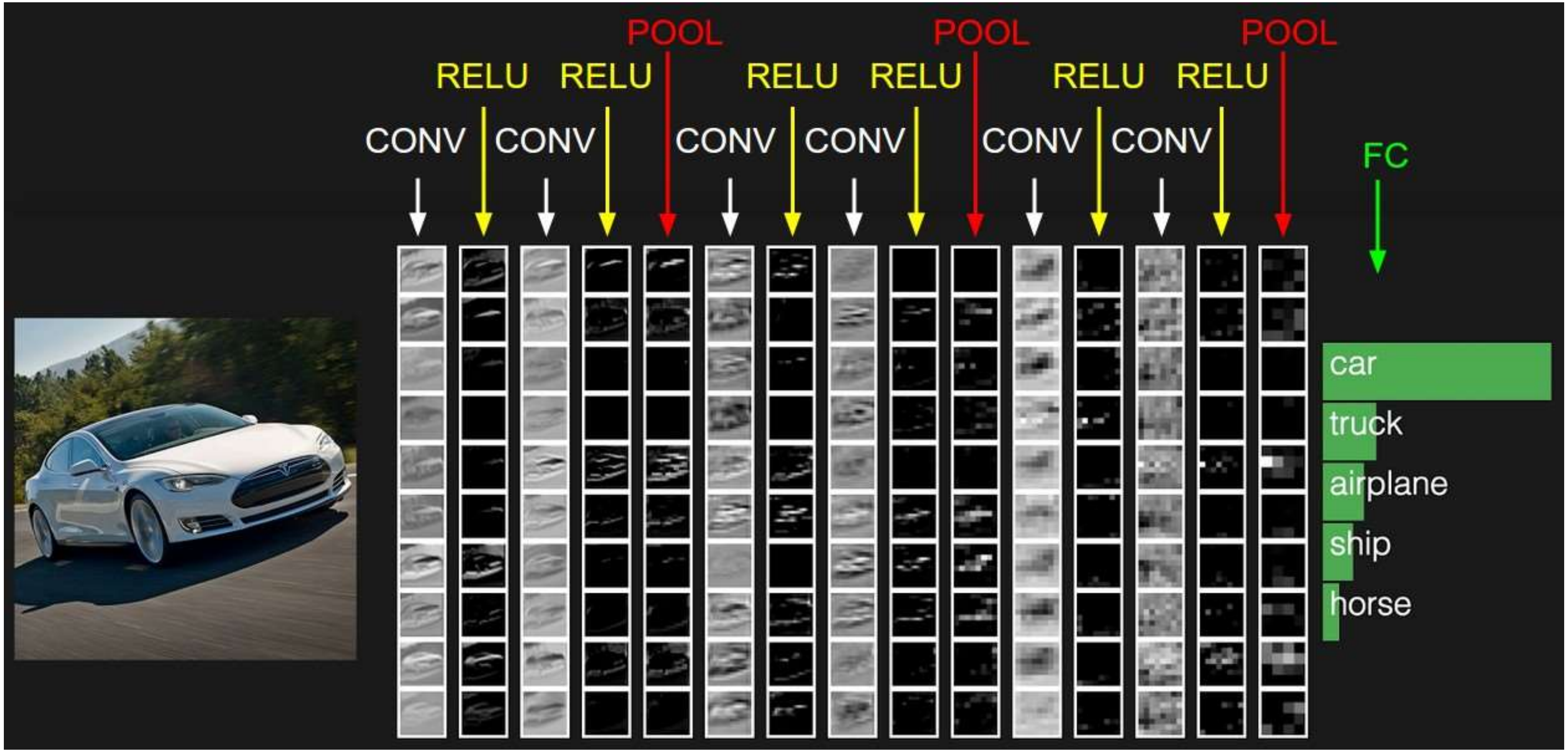
filter: 2x2

Stride: 2

6	8
3	7

2x2

ConvNet Architecture

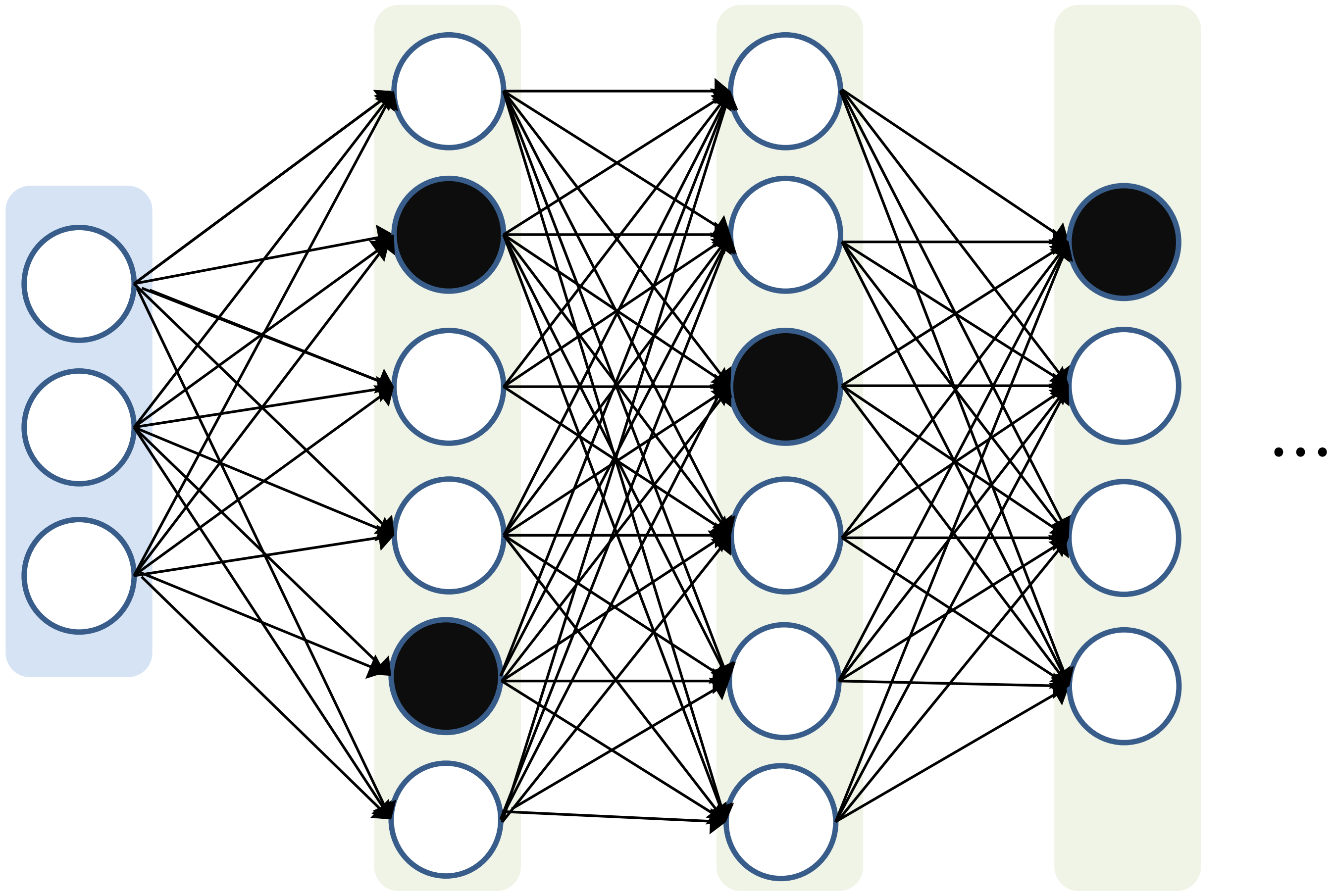


Hyperparameter?

- Non-learnable parameters

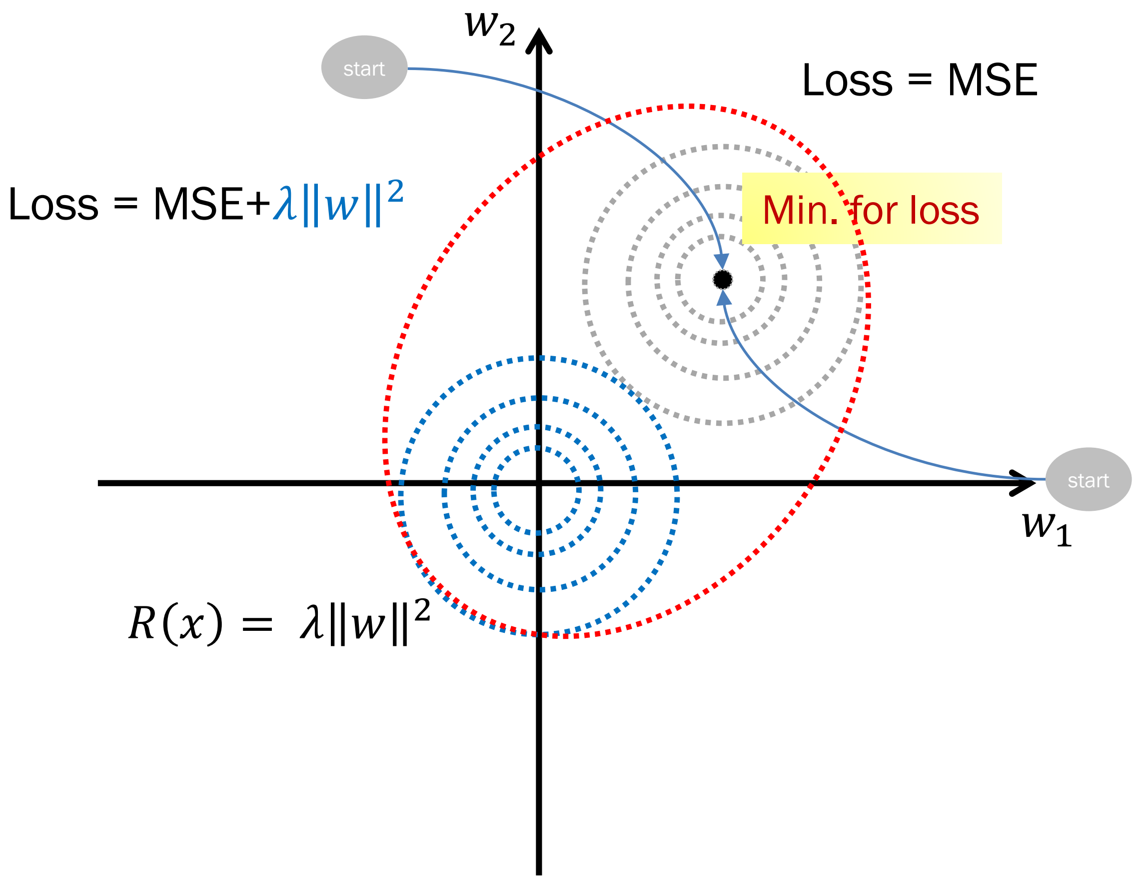
Category	Parameters
Model capacity	# of hidden layers # of hidden units Activation function
Regularization	Dropout rate Batch normalization L2 regularization Xavier initialization
Optimizing	Optimizer Learning rate # of Epoch Batch_size
Device	CPU/GPU
Post processing	Saving/filename

Dropout



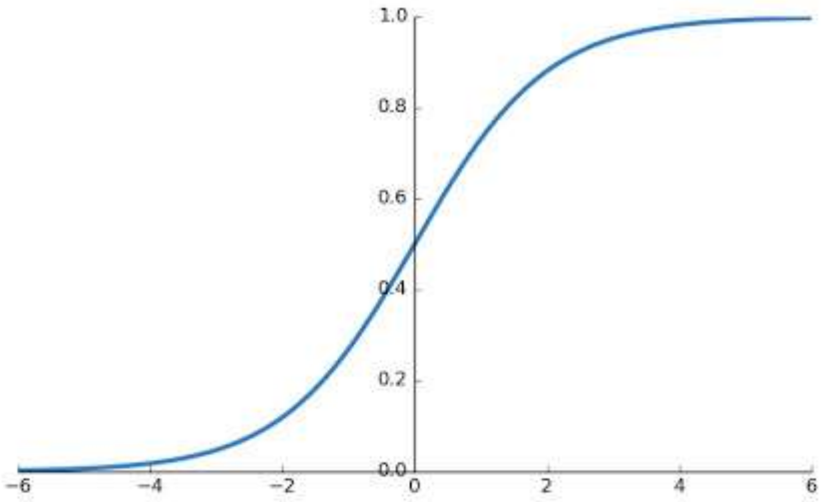
Intentionally turn off nodes with probability when training

L2 Regularization



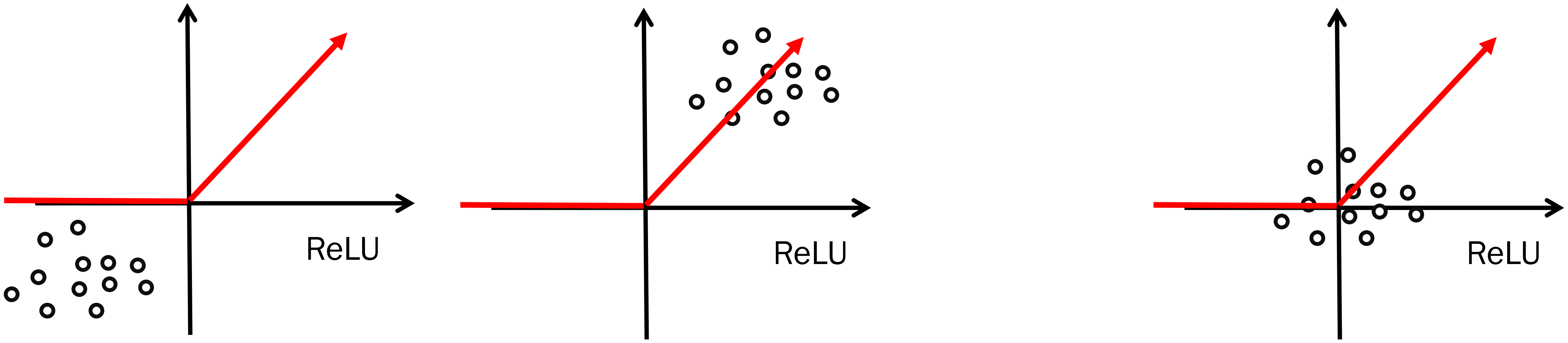
Xavier initialization

Models are sensitive to weight initialization
Keeping the shape of initialization valid to initiate parameters with better values
randomizing the initial weights, so that the inputs of each activation function fall within the sweet range of the activation function.
Ideally, none of the neurons should start with a trapped situation.



Batch normalization

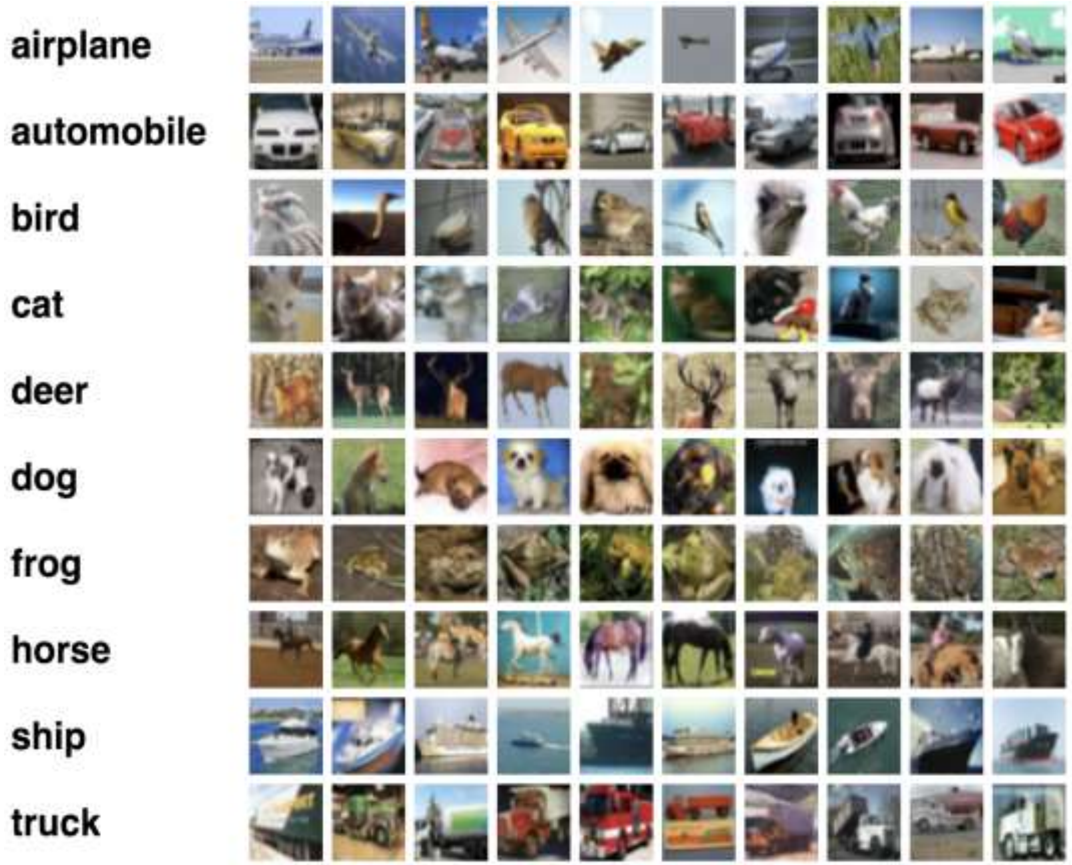
Normalize distribution of each input feature in each layer across each minibatch to Normal distribution $N{\sim}(0,1)$



Nothing happens!

Better(greater) learning rate
Faster convergence

CIFAR10



- 60K 32x32 colour images
- 10 classes (6 K images per class)
- 50K training images
- 10K test images

https://www.cs.toronto.edu/~kriz/cifar.html

Lab 4B: CIFAR10 classification - CNN

Exercise 1: More conv2D layer	# of conv2D
Run 1	2
Run 2	4

Exercise 2: different optimizer	Optimizer
Run 1	RMSprop
Run 2	SGD

Exercise 3: different epoch	# of epoch
Run 1	10
Run 2	30

You may want to try

1. Use different cost function
2. Use different learning rate

https://pytorch.org/docs/stable/optim.html?highlight=optimizer#torch.optim.Optimizer

Break room

Session ends:

Thank you!