

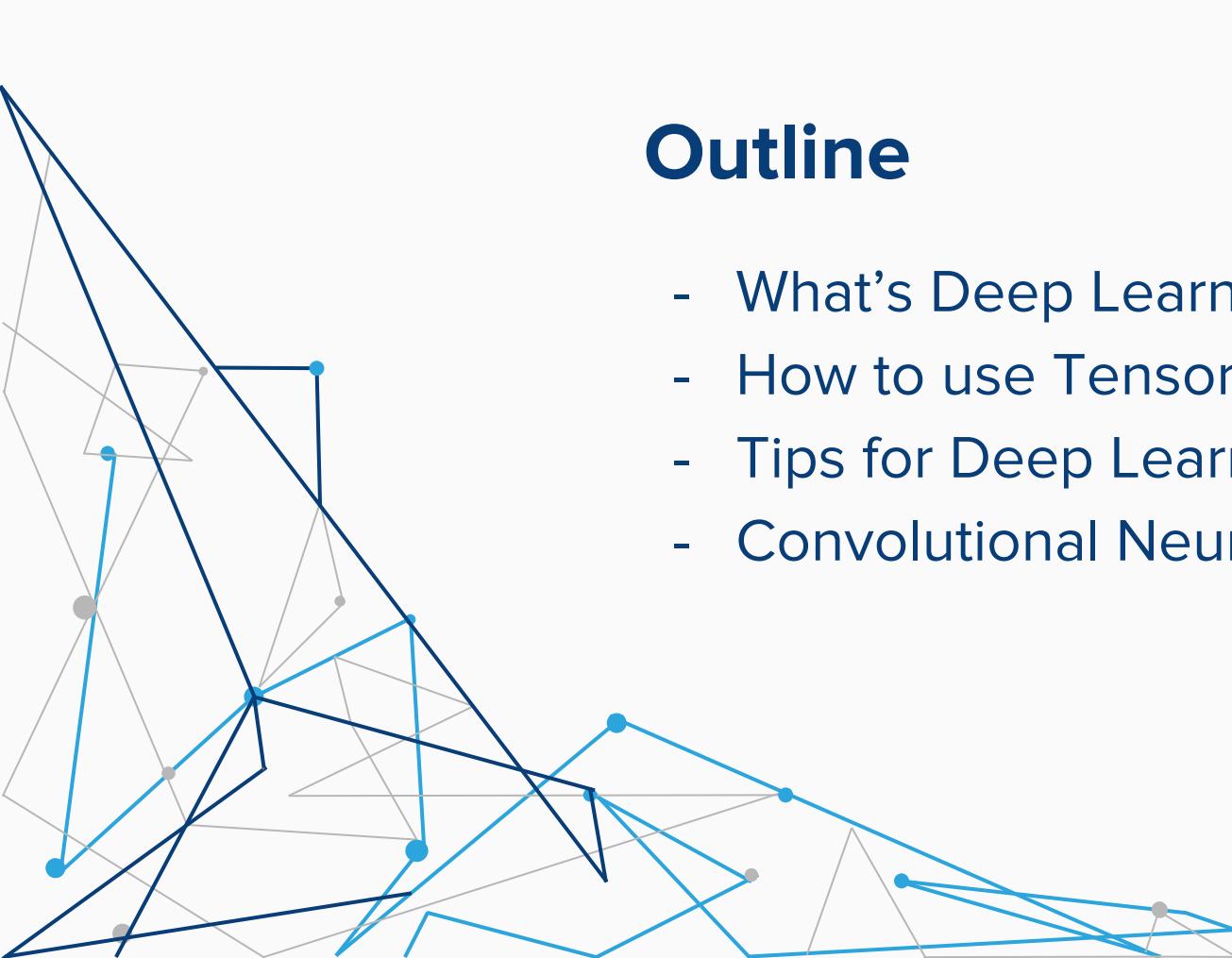
Introduction to Deep Learning Part I

Jeff Liu and Ching Lee

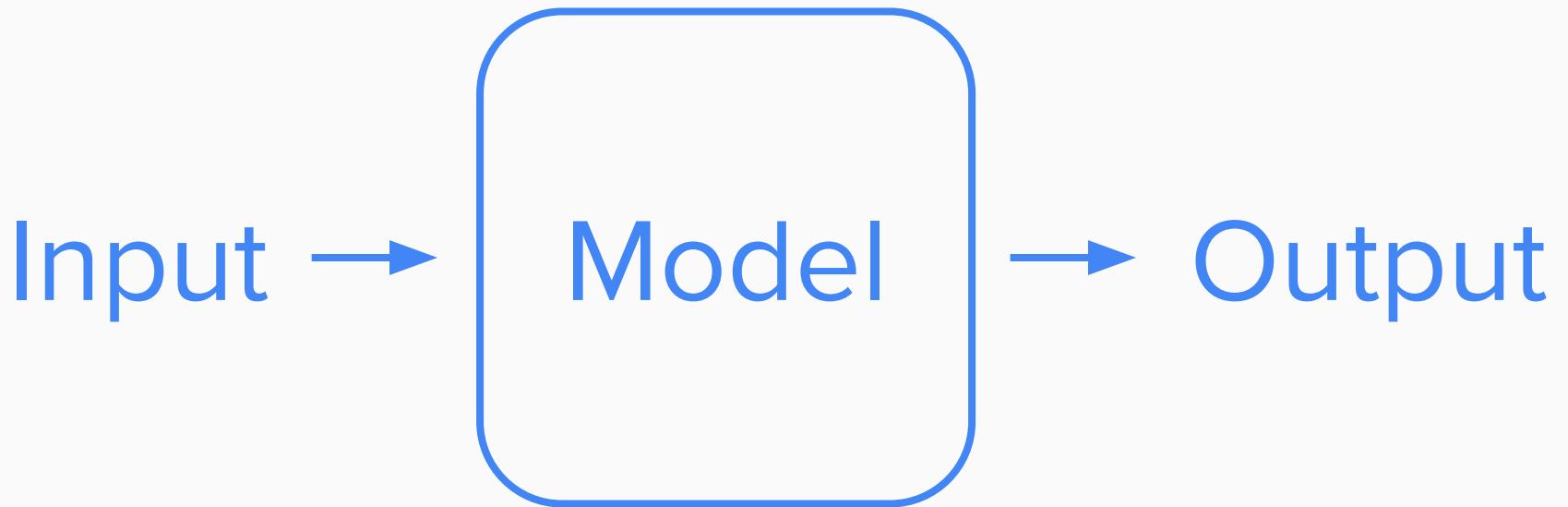
2018/10/16

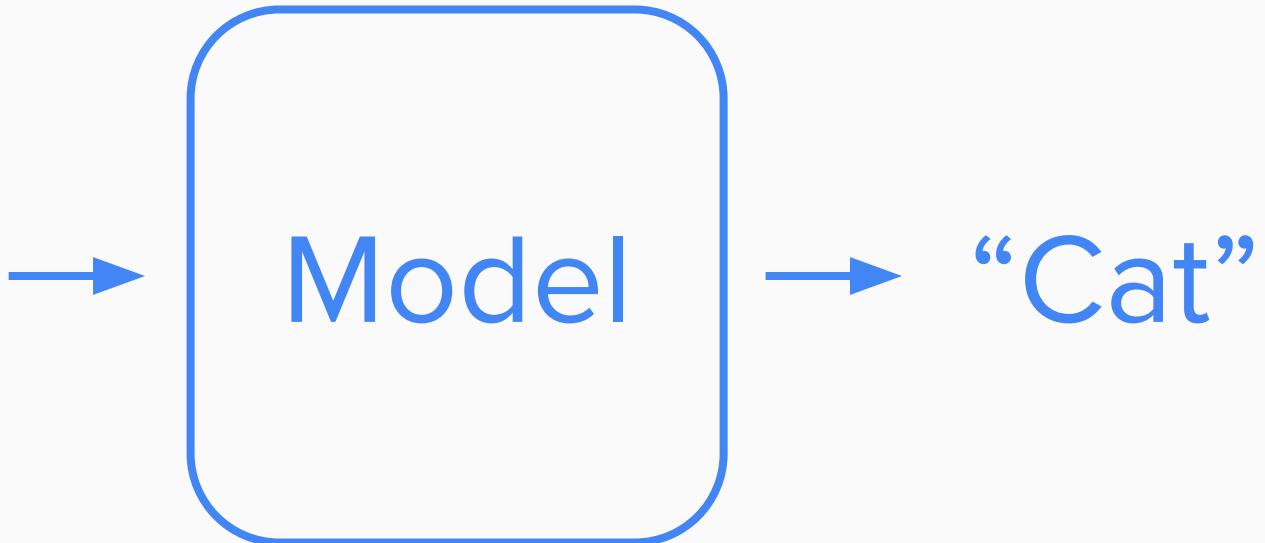
Outline

- What's Deep Learning?
- How to use TensorFlow and Keras
- Tips for Deep Learning
- Convolutional Neural Networks

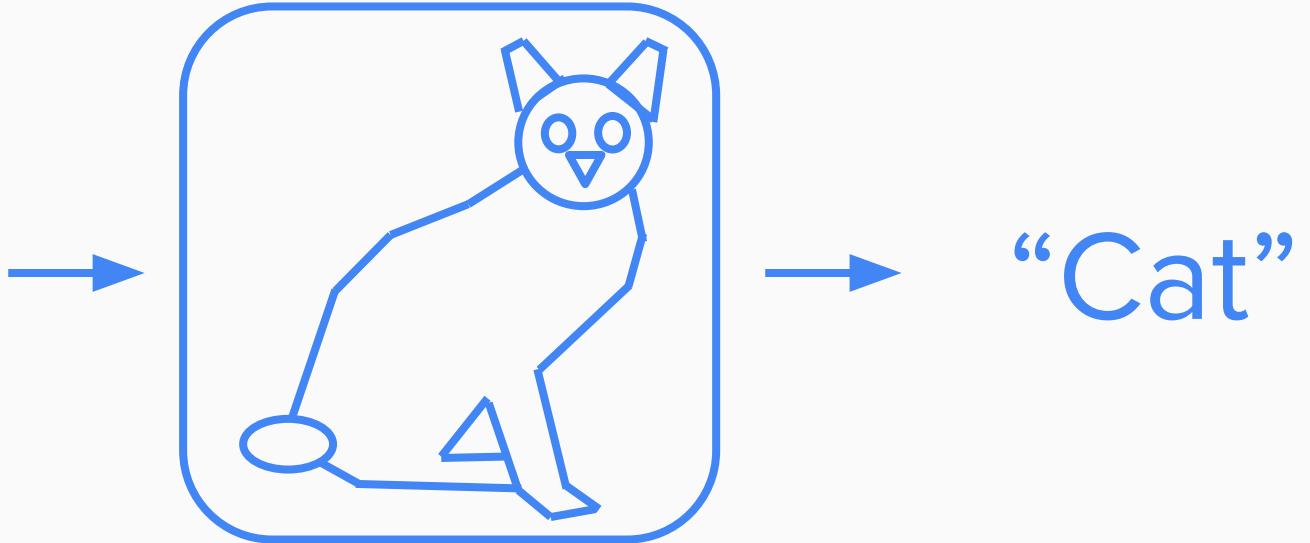


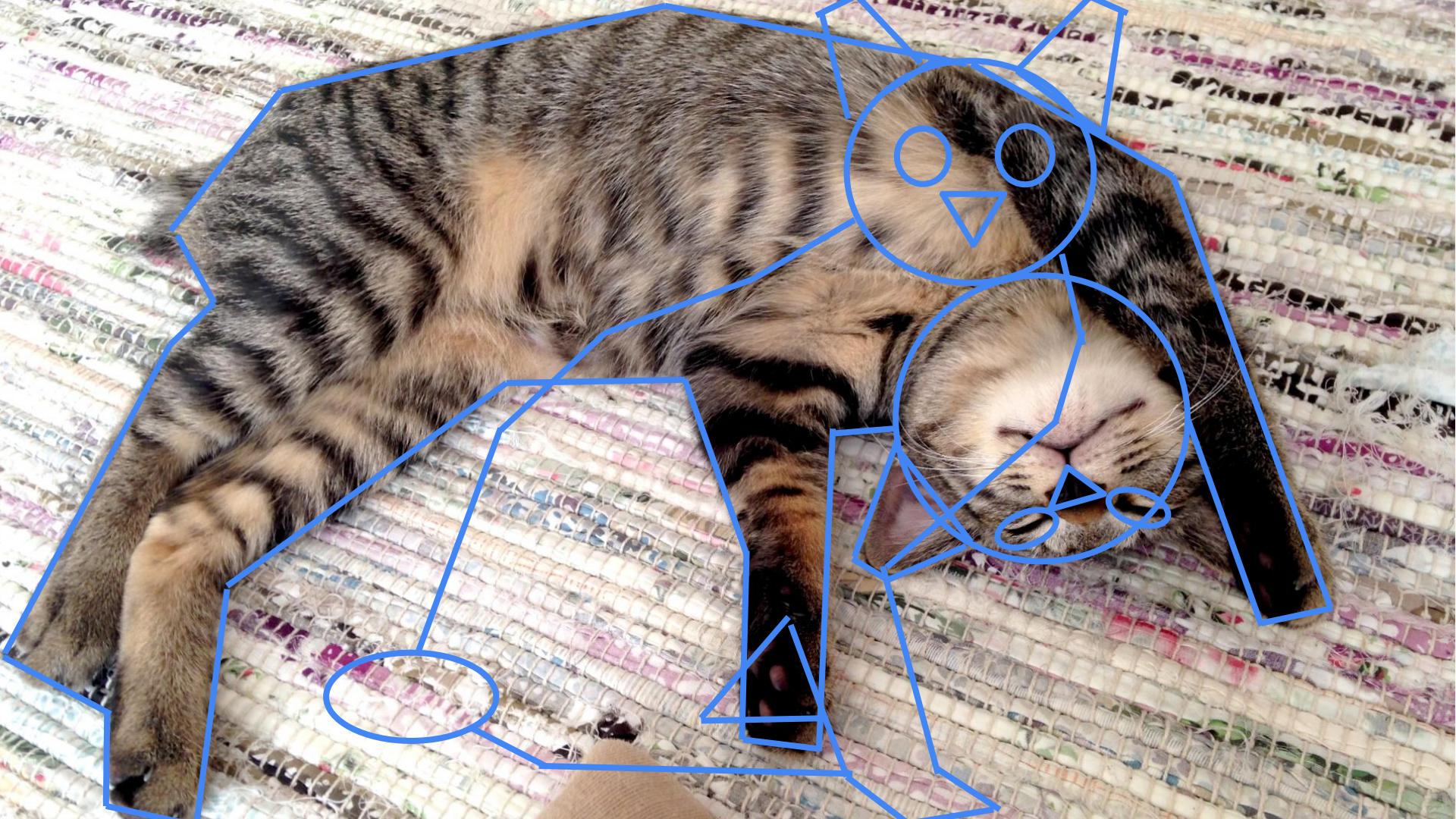










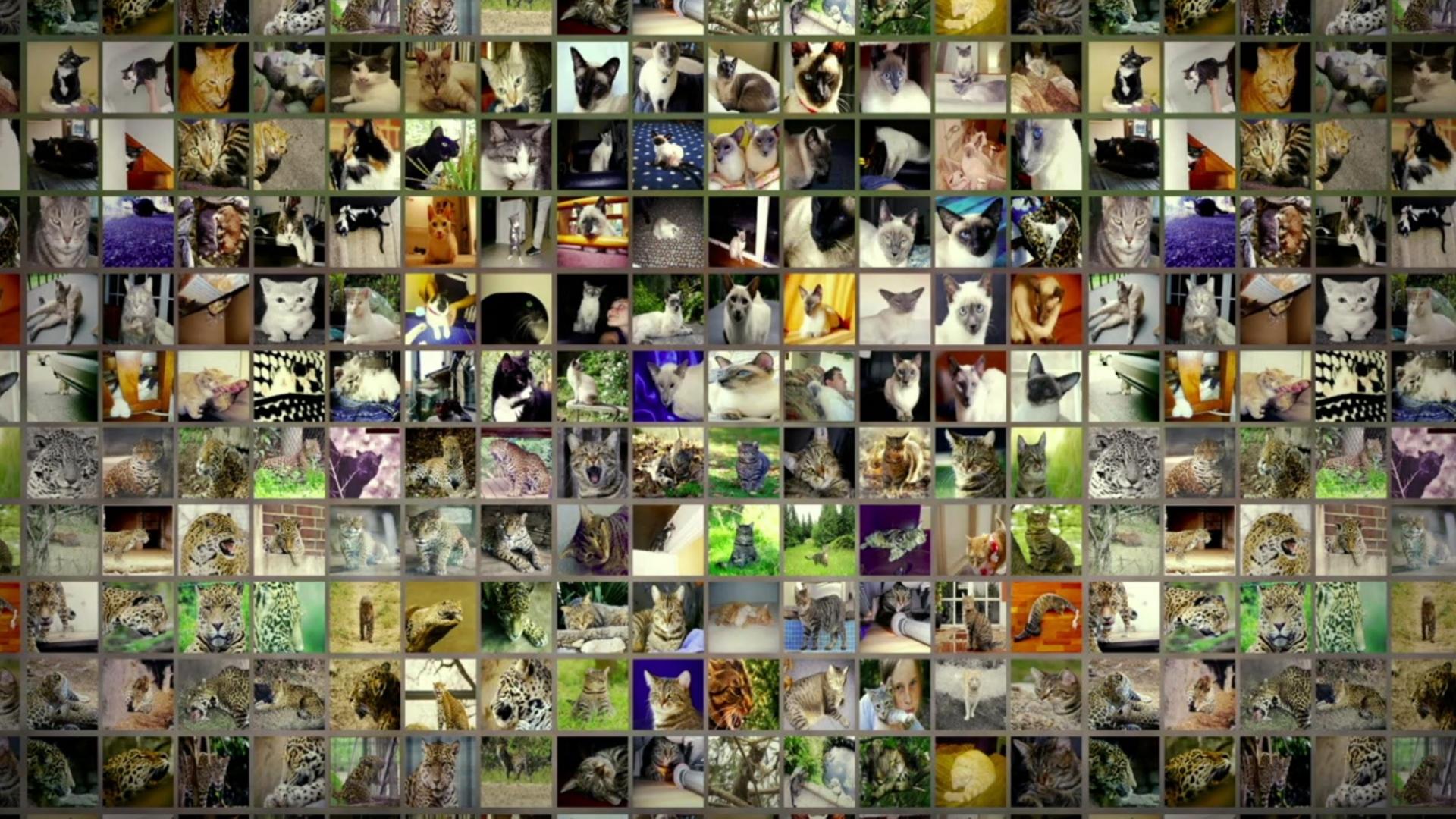


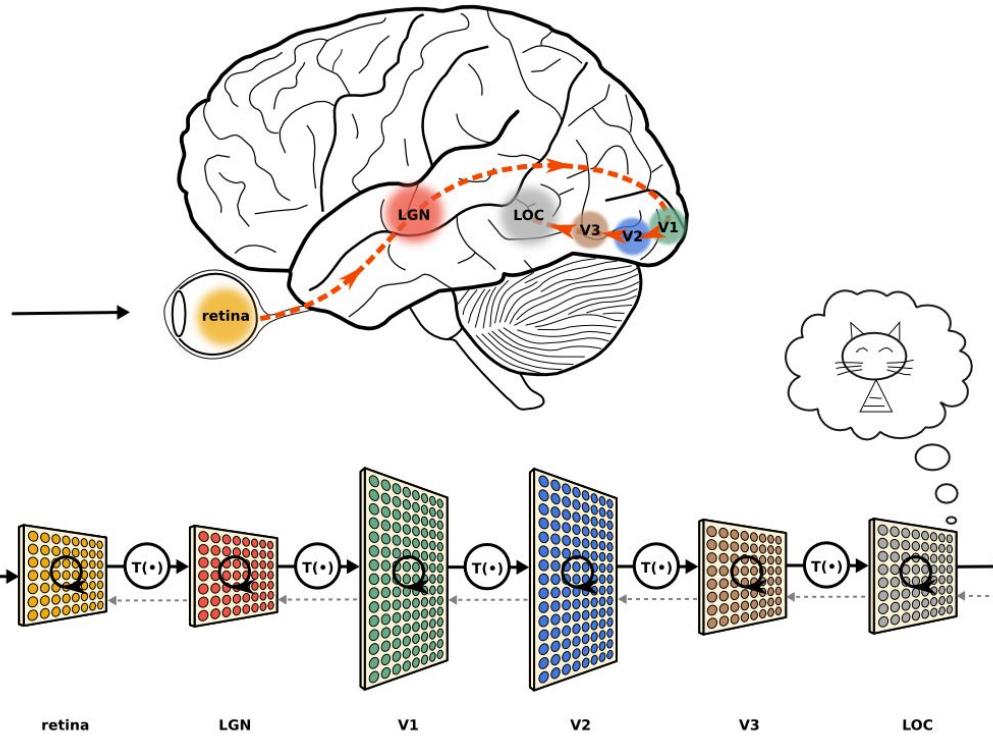












(Deep) Neural Networks

Artificial
Intelligence

目標



1950

Machine
Learning

手段



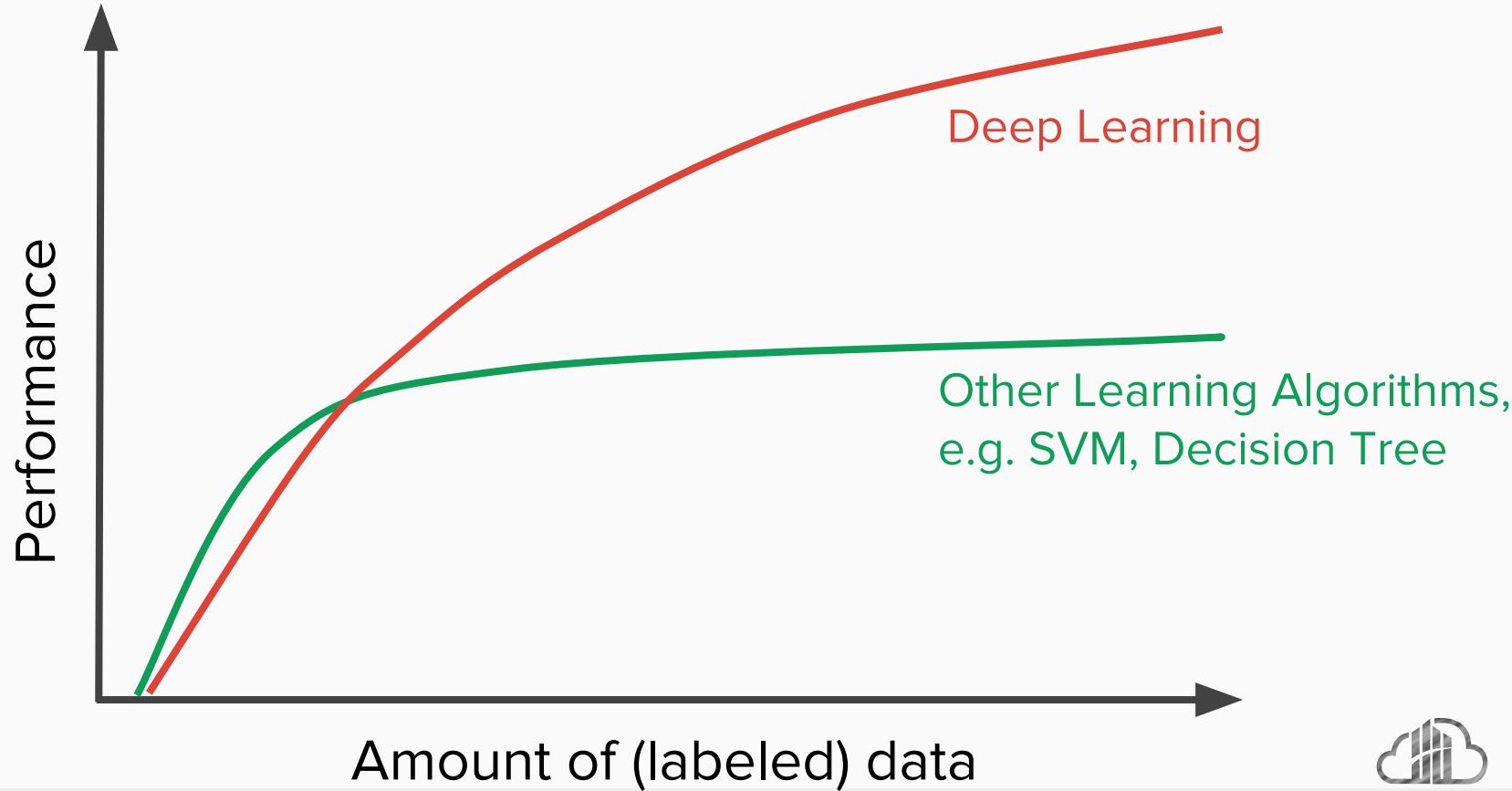
1980

Deep
Learning



2010

Why Deep Learning?



The Basis of Artificial Intelligence



Data



Algorithm



Hardware



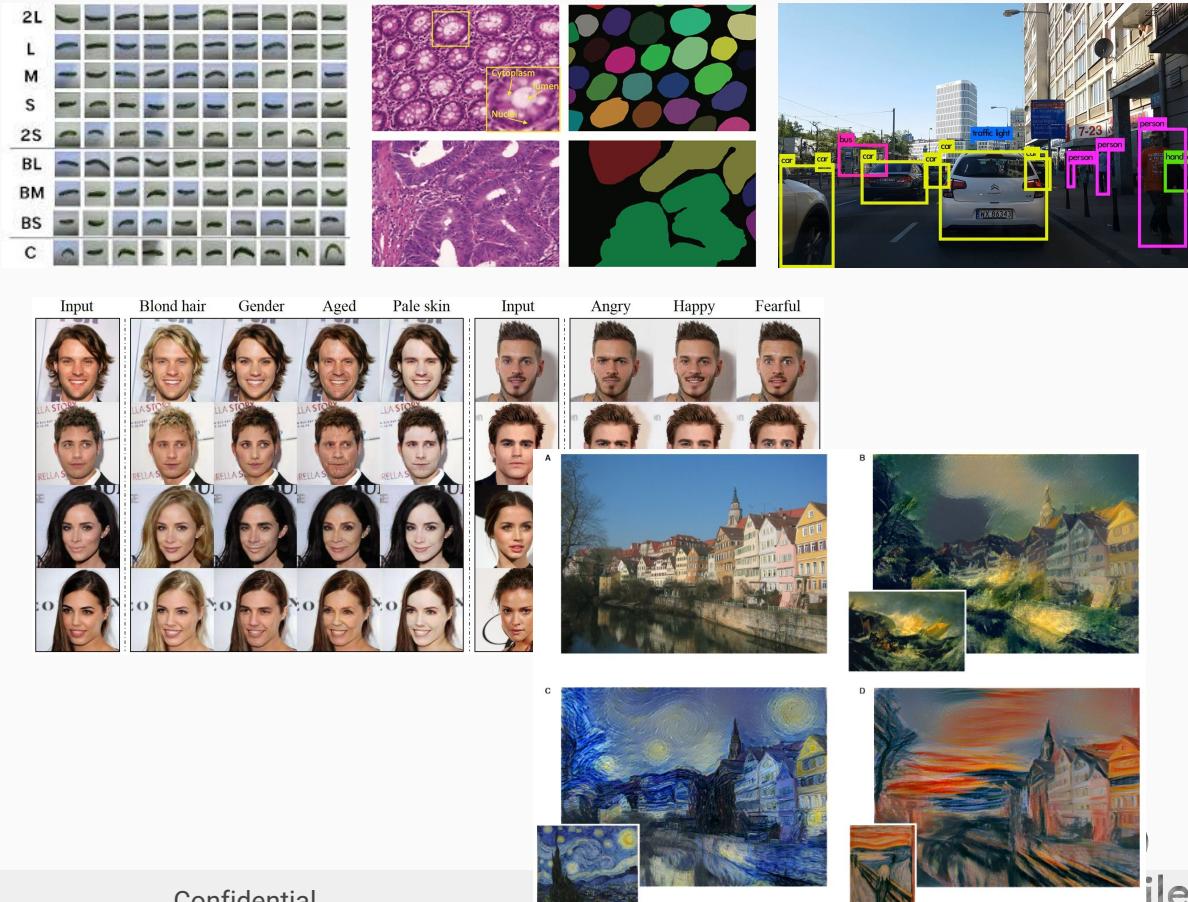
Structured Data

- Attributes (e.g. age, sex, income ...) as inputs
- Continuous (e.g. income) / Categorical (e.g. gender)

1 age Min 17. 1st Qu 28. Median 37. Mean 38.5816 3rd Qu 48. Max 90.	2 workclass Private 22 696 Self-emp-not-inc 2541 Local-gov 2093 ? 1836 State-gov 1298 Self-emp-inc 1116 (Other) 981	3 education HS-grad 10 501 Some-college 7291 Bachelor 5355 Masters 1723 Assoc-voc 1382 11th 1175 (Other) 5134	4 education-num Min 1. 1st Qu 9. Median 10. Mean 10.0807 3rd Qu 12. Max 16.	5 marital-status Married-civ-spouse 14 976 Never-married 10 683 Divorced 4443 Separated 1025 Widowed 993 Married-spouse-absent 418 Married-AF-spouse 23
6 occupation Prof-specialty 4140 Craft-repair 4099 Exec-managerial 4066 Adm-clerical 3770 Sales 3650 Other-service 3295 (Other) 9541	7 relationship Husband 13 193 Not-in-family 8305 Own-child 5068 Unmarried 3446 Wife 1568 Other-relative 981	8 race White 27 816 Black 3124 Asian-Pac-Islander 1039 Amer-Indian-Eskimo 311 Other 271	9 sex Male 21 790 Female 10 771	10 capital-gain 1st Qu 0. 3rd Qu 0. Median 0. Min 0. Mean 1077.65 Max 99 999.
11 capital-loss 1st Qu 0. 3rd Qu 0. Median 0. Min 0. Mean 87.3038 Max 4356.	12 hours-per-week Min 1. 1st Qu 40. Median 40. Mean 40.4375 3rd Qu 45. Max 99.	13 native-country United-States 29 170 Mexico 643 ? 583 Philippines 198 Germany 137 Canada 121 (Other) 1709	14 income =<50K 24 720 >50K 7841	

Images

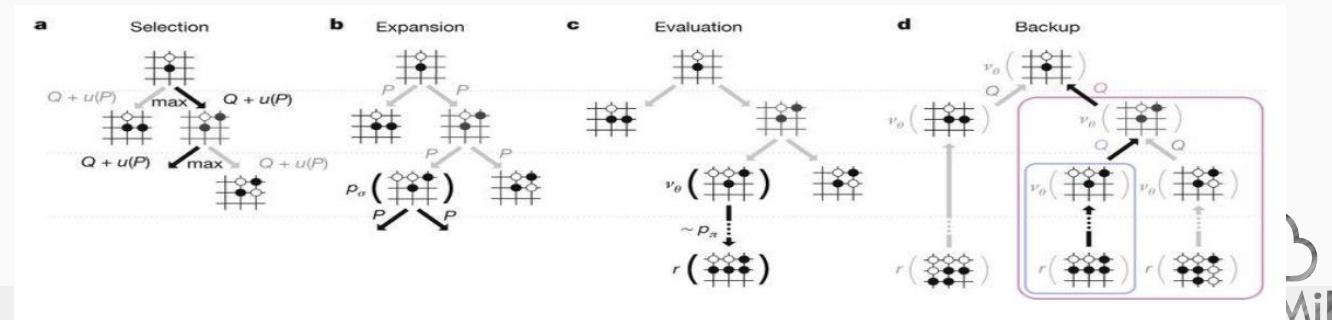
- Classification
- Segmentation
- Object Detection
- Generative Models
- Art



Sequential Data

- Text
- Speech, Audio
- Videos
- Sequential Decision Making (RL)

“CloudMile 成立於 2016 年，致力於 B2B 雲端與人工智慧應用。為客戶建立國際級雲端架構，並以機器學習及大數據分析技術為核心，協助企業進行商業預測與 產業升級。”



Learning ≈ Looking for a Function

- Image Recognition

$$f(\text{cat}) = \text{"cat"}$$



- Speech Recognition

$$f(\text{声波}) = \text{"你太skr了～"}$$



- Machine Translation

$$f(\text{母湯}) = \text{"不可以"}$$

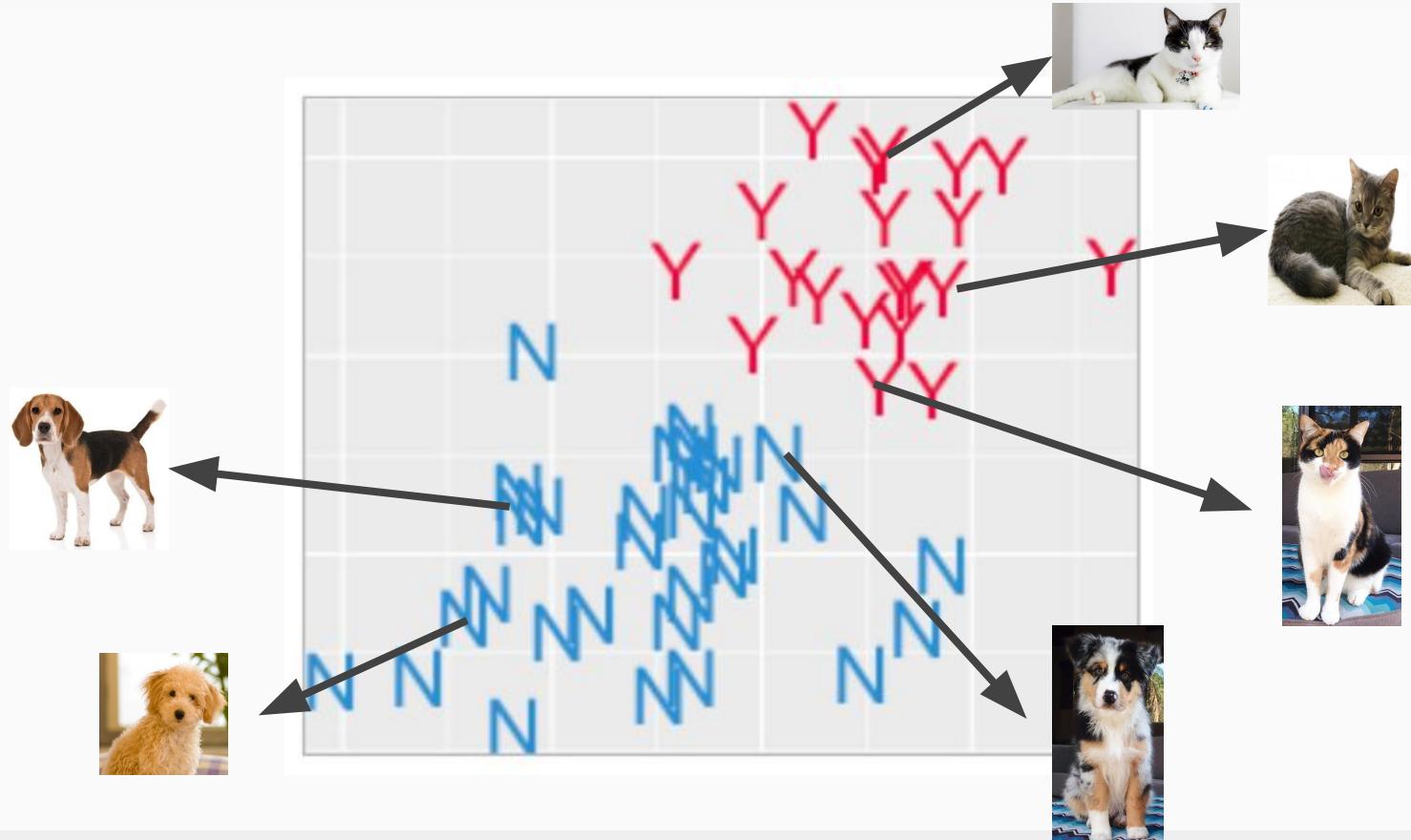
“母湯”

- Playing Go

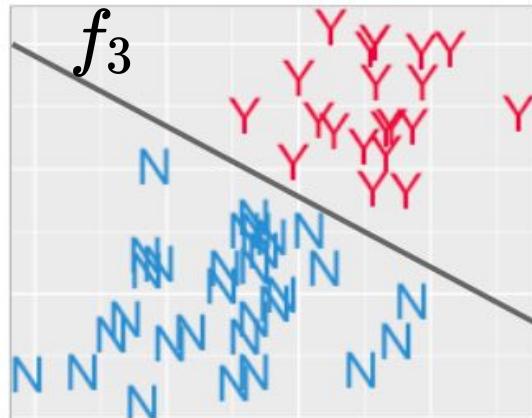
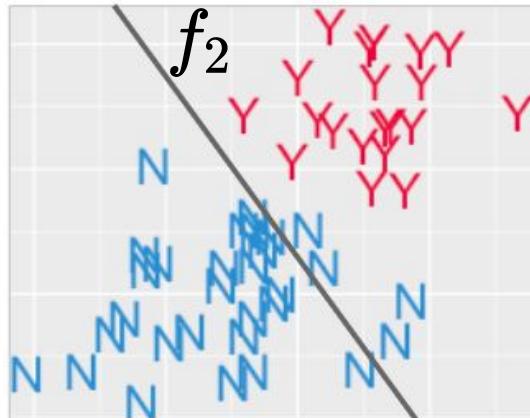
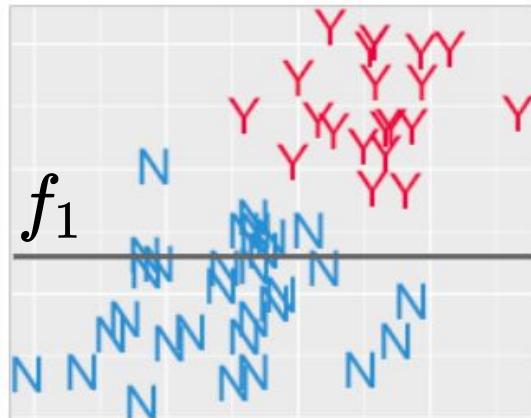
$$f(\text{棋盘}) = \text{"3-3" (next move)}$$



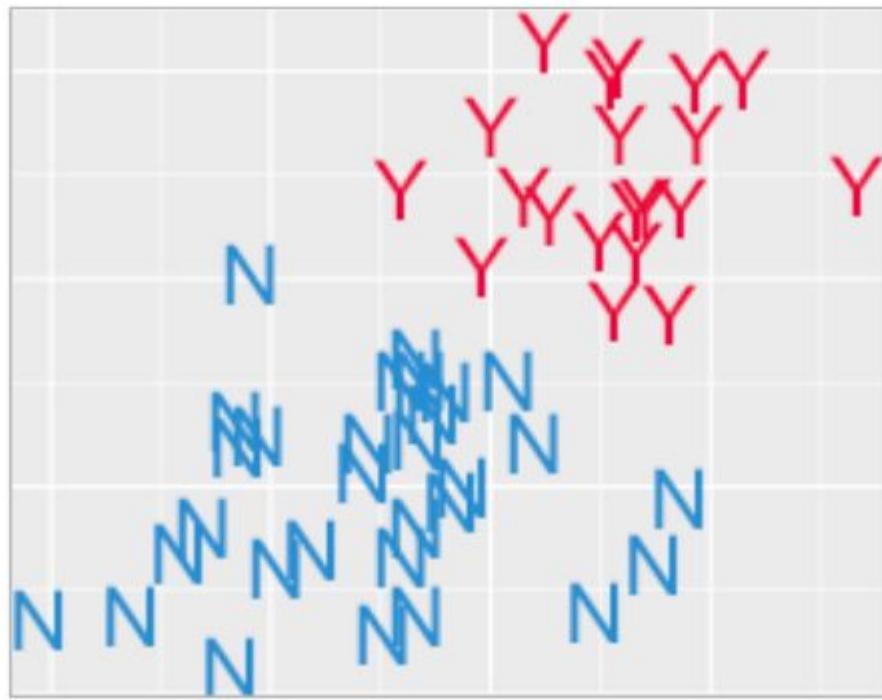
Cat/Dog Classification



Solutions may be something like ...



Select a Model



-  Support Vector Classifier
-  Decision Tree
-  Neural Network

Machine Learning Framework

Step 1

Neural Networks

Model

$f_1, f_2 \dots$

Training

Step 2

goodness of
function f

Step 3

pick the “best” function

f^*

Prediction

“Cat”

using f^*

training
data

Input:



“cat”



“cat”



“dog”



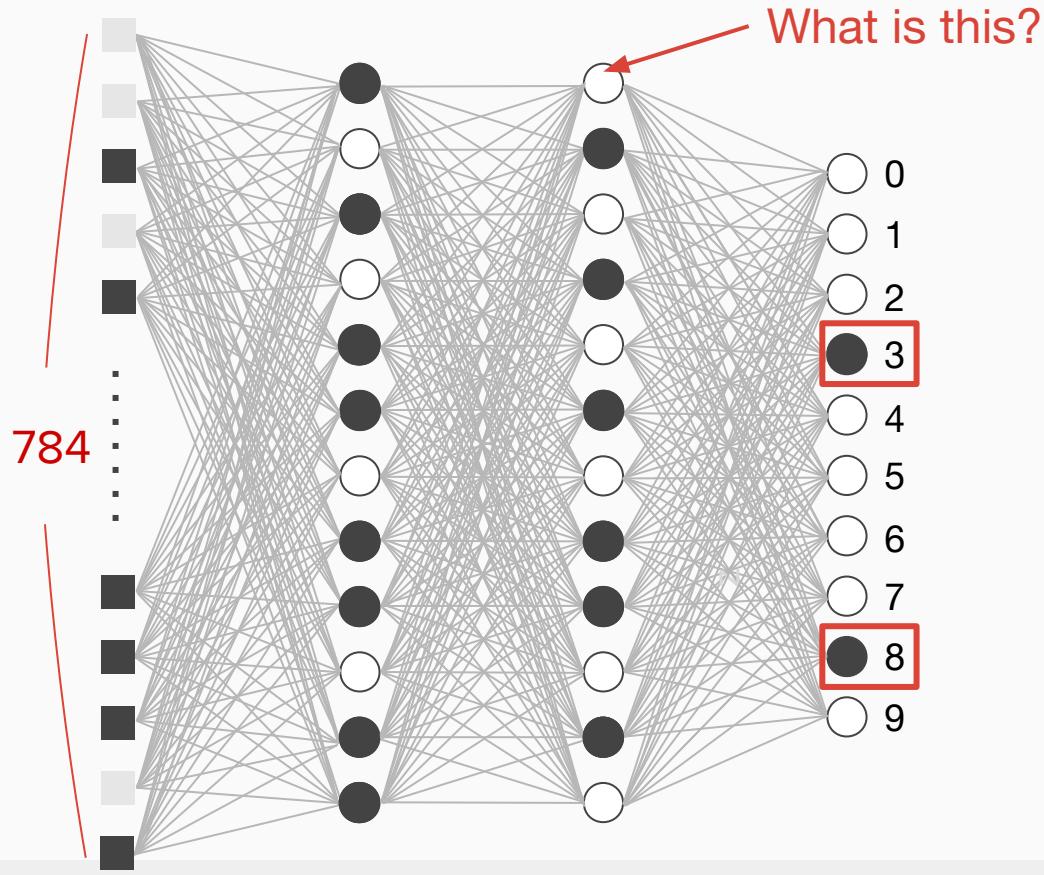
“dog”



Output:

Confidential

Neural Networks for MNIST



Confidential

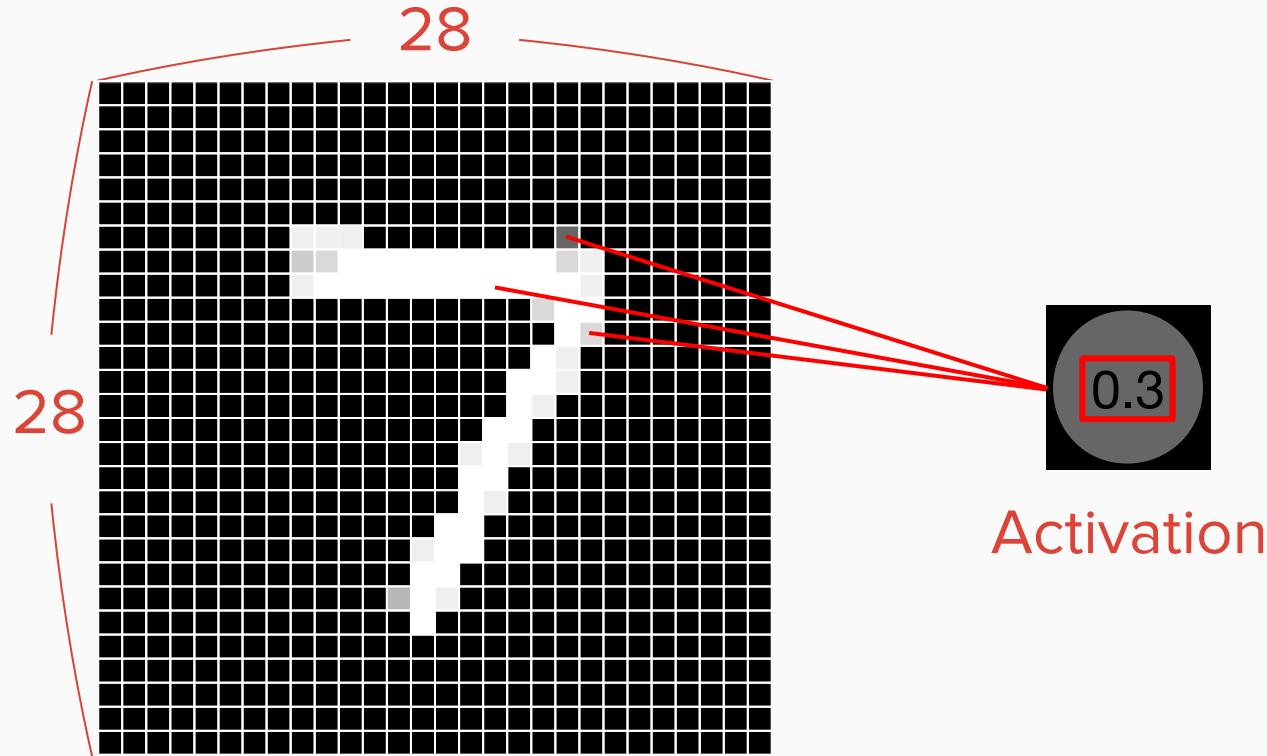
27

Neuron

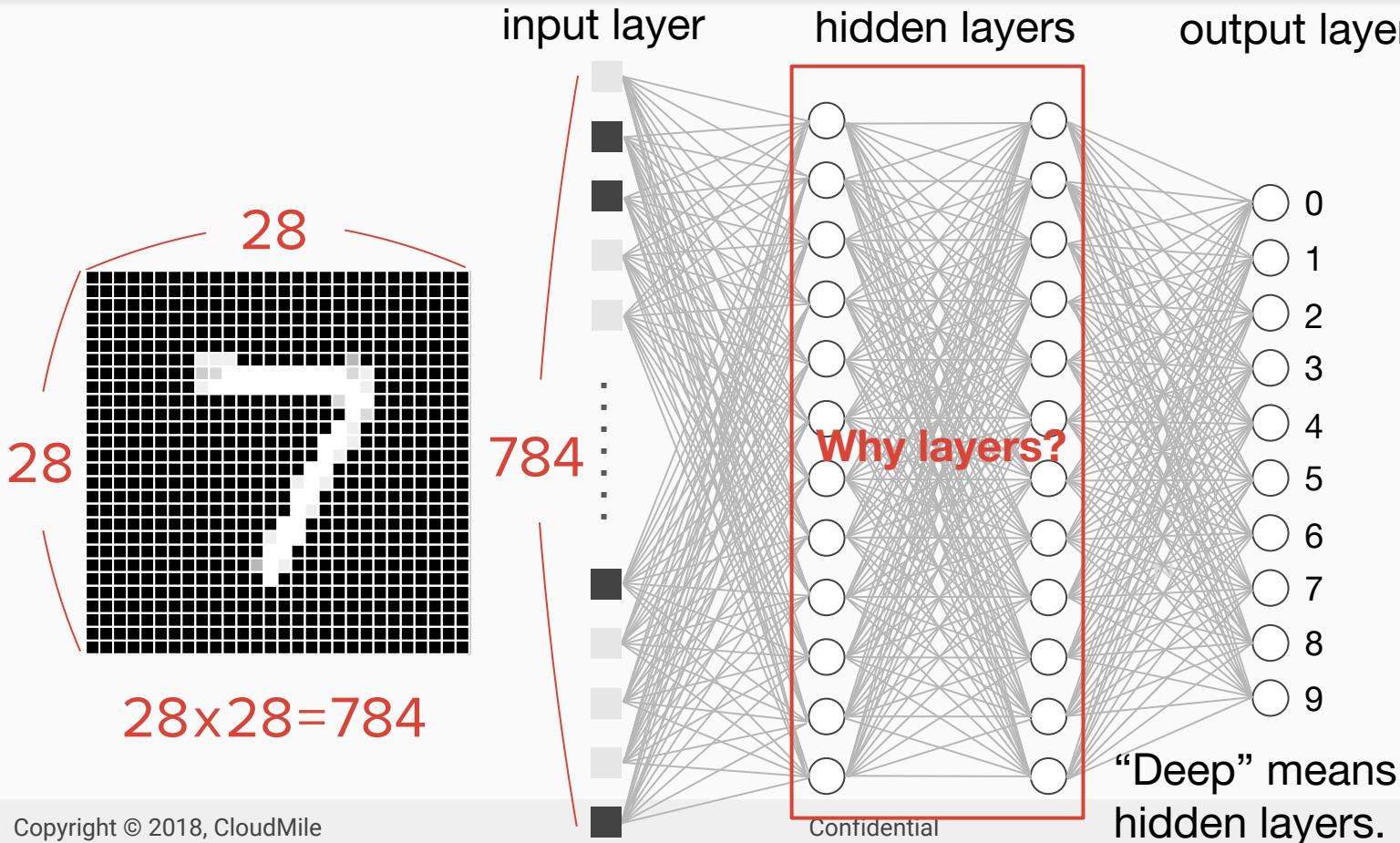
0.98

Neuron → Holding a number

Activation



Deep Neural Networks (Deep Learning)



Meaningfulness of Deep Learning

$$\begin{matrix} \text{9} \\ \hline \end{matrix} = \begin{matrix} \text{0} \\ \hline \end{matrix} + \begin{matrix} \text{1} \\ \hline \end{matrix}$$

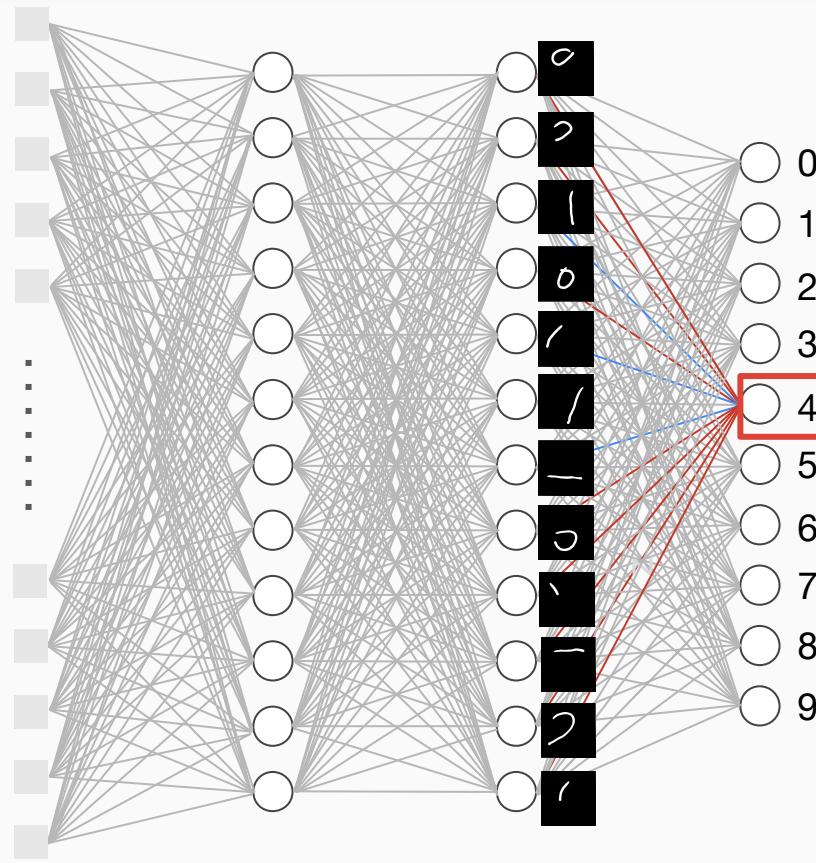
$$\begin{matrix} \text{8} \\ \hline \end{matrix} = \begin{matrix} \text{0} \\ \hline \end{matrix} + \begin{matrix} \text{0} \\ \hline \end{matrix}$$

$$\begin{matrix} \text{4} \\ \hline \end{matrix} = \begin{matrix} \text{1} \\ \hline \end{matrix} + \begin{matrix} \text{---} \\ \hline \end{matrix} + \begin{matrix} \text{1} \\ \hline \end{matrix}$$

Meaningfulness of Deep Learning (Cont'd)

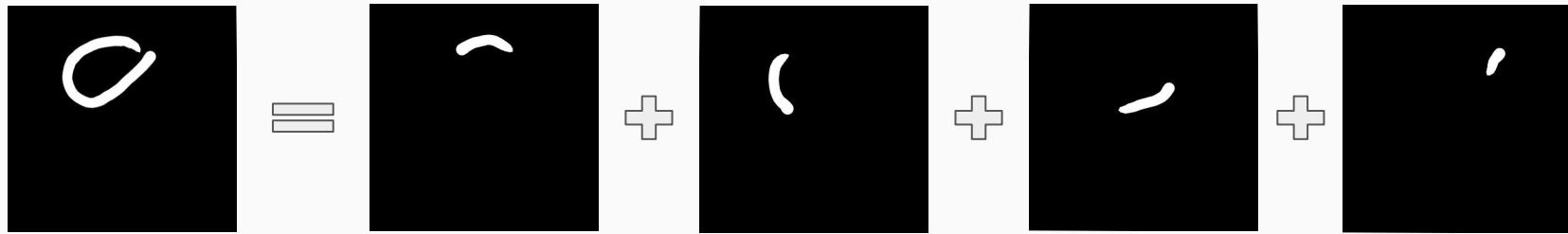


raw images

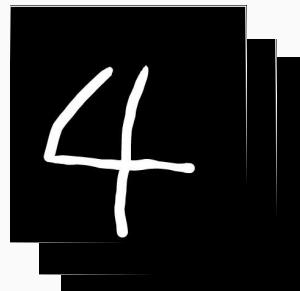


Confidential

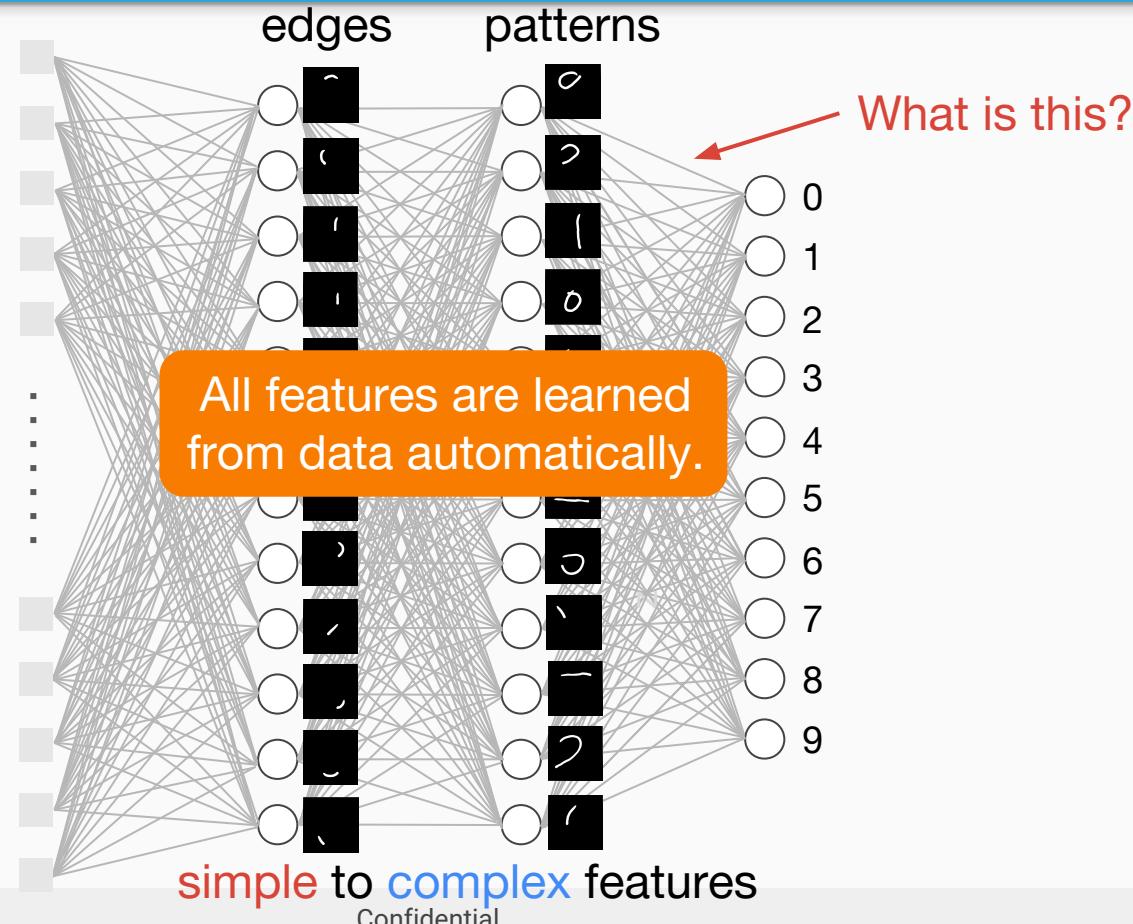
Meaningfulness of Deep Learning (Cont'd)



Meaningfulness of Deep Learning (Cont'd)

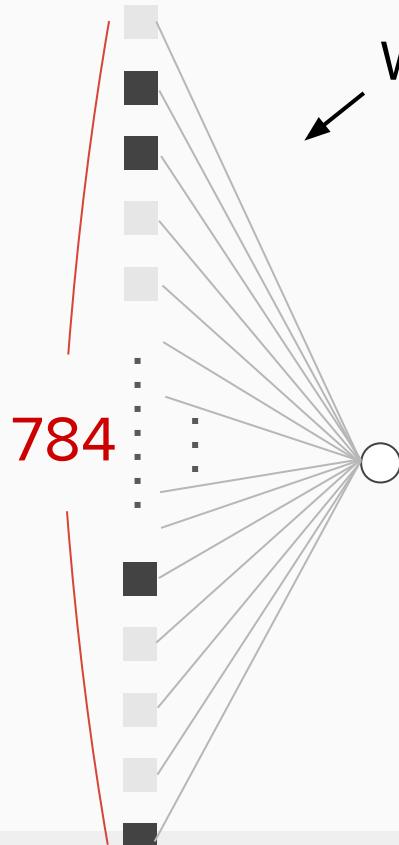


raw images



Confidential

How to Extract an Edge?



What parameters?

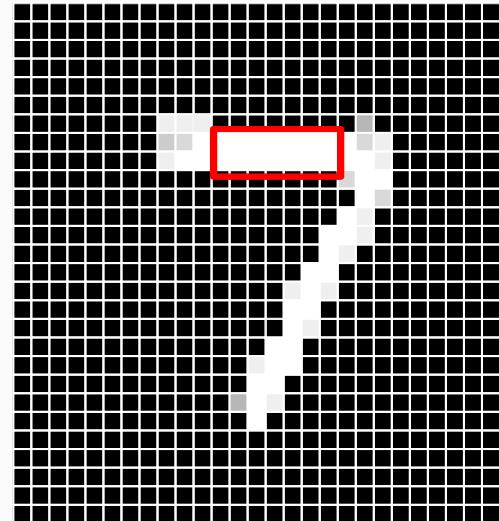
$$w_1 : 1.91$$

$$w_2 : -1.68$$

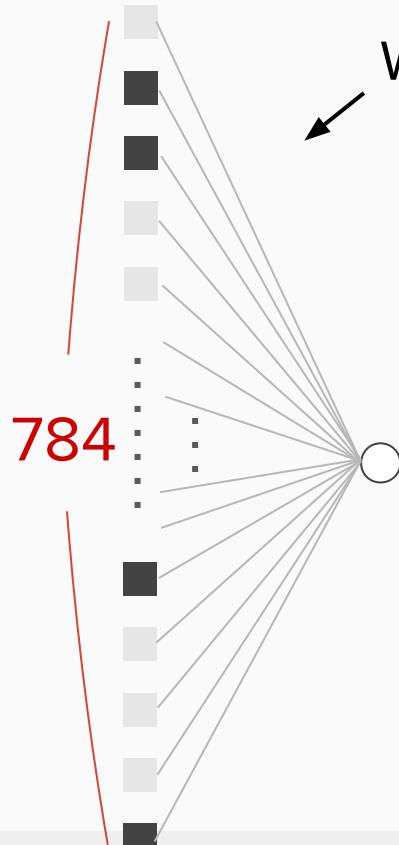
$$w_3 : 0.78$$

⋮

$$w_n : 0.23$$



How to Extract an Edge?



What parameters?

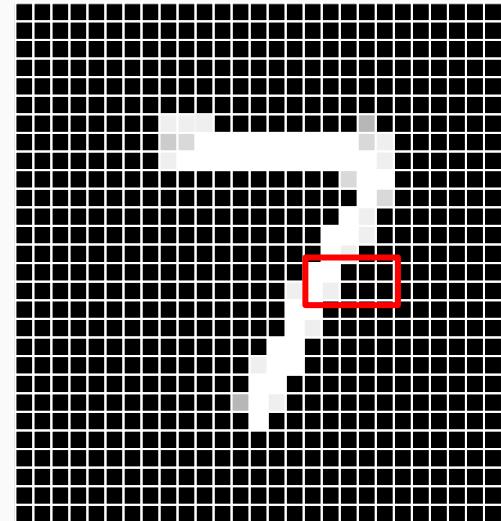
$$w_1 : 0.92$$

$$w_2 : -0.53$$

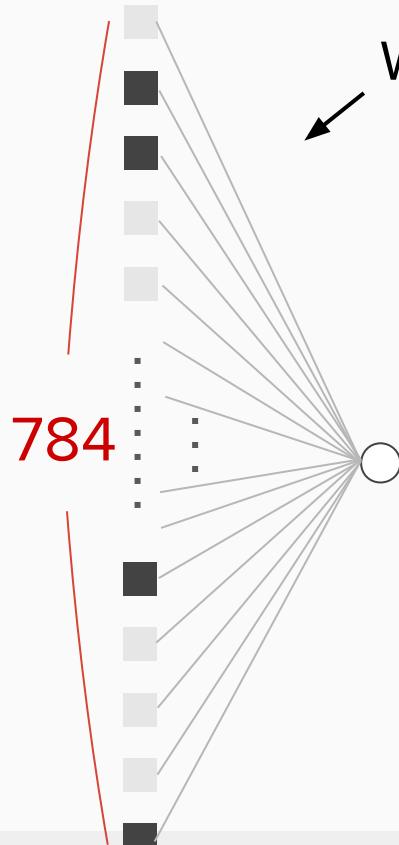
$$w_3 : -2.54$$

⋮

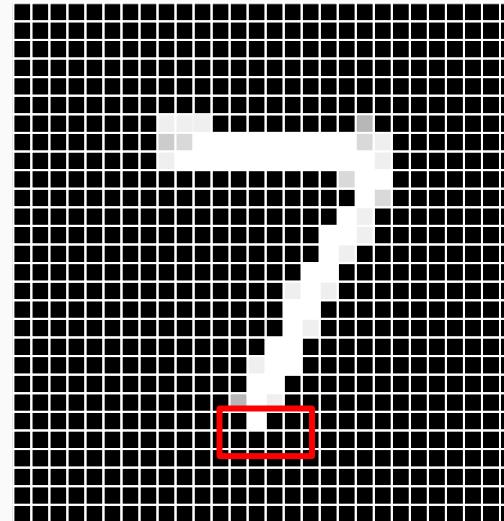
$$w_n : -1.05$$



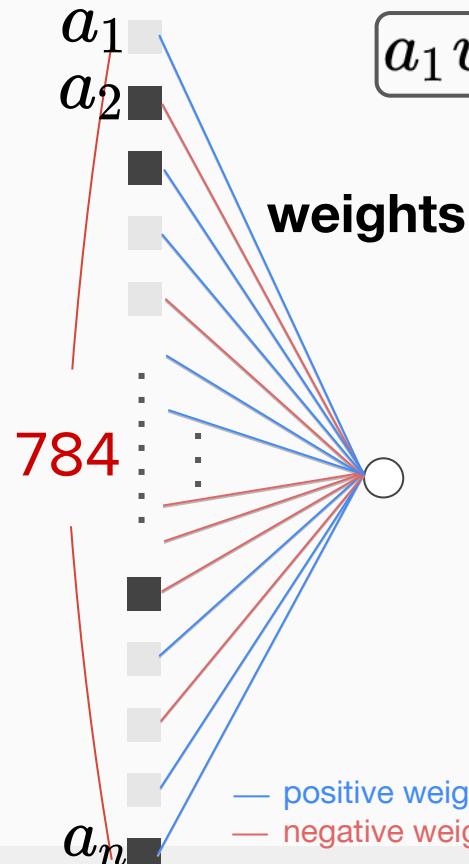
How to Extract an Edge?



What parameters?

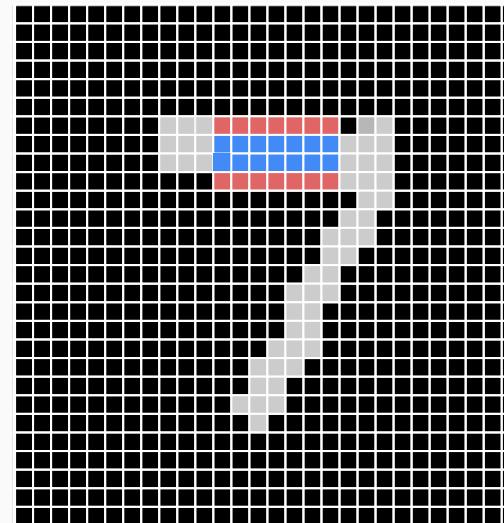
$$w_1 : -0.98$$
$$w_2 : -0.87$$
$$w_3 : 1.23$$
$$\vdots$$
$$w_n : 3.14$$


Weighted Sum



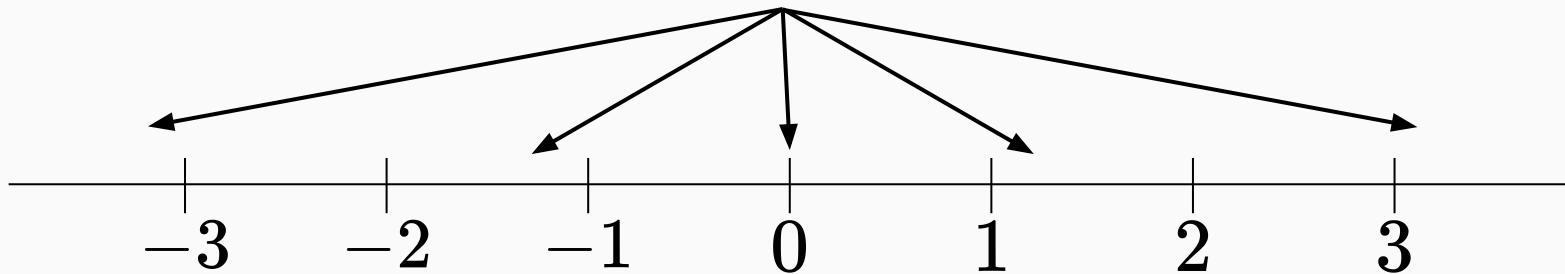
$$a_1 w_1 + a_2 w_2 + a_3 w_3 + \cdots + a_n w_n$$

$$\begin{aligned}w_1 &: 1.91 \\w_2 &: -1.68 \\w_3 &: 0.78 \\\vdots \\w_n &: 0.23\end{aligned}$$



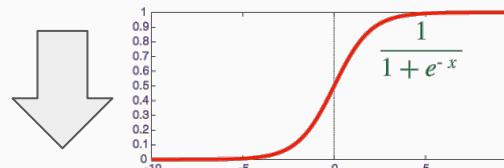
Activation Function

$$a_1 w_1 + a_2 w_2 + a_3 w_3 + \cdots + a_n w_n$$

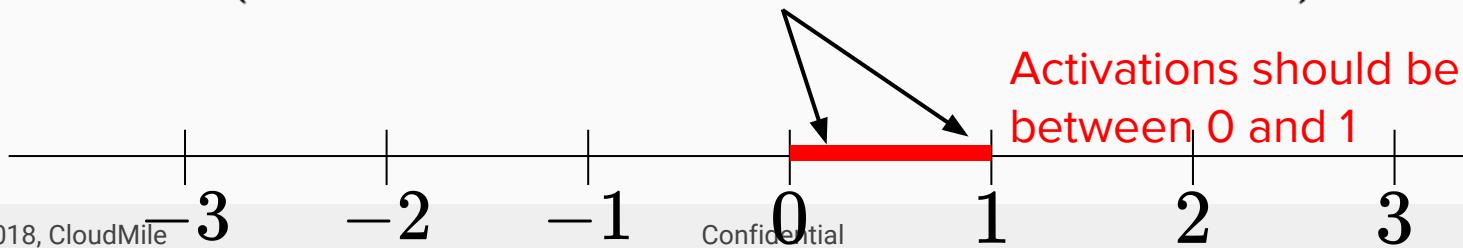


sigmoid

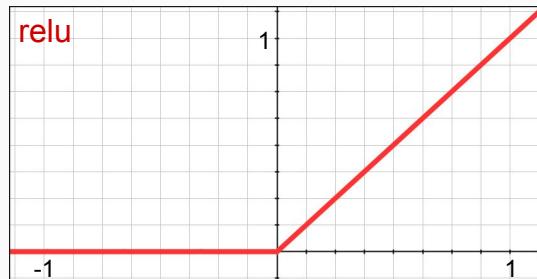
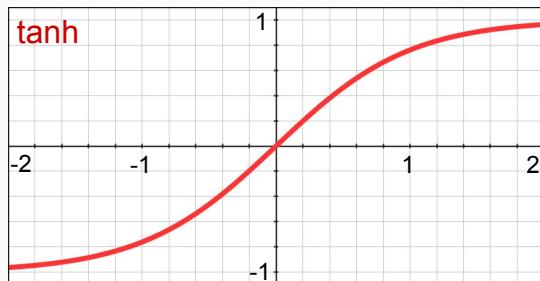
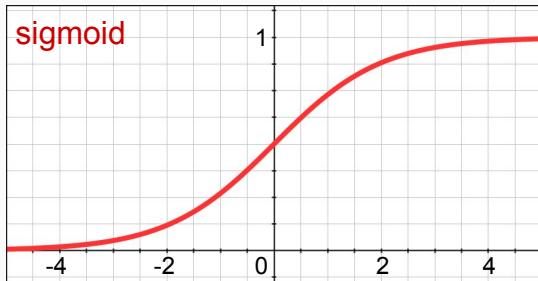
$$\sigma(a_1 w_1 + a_2 w_2 + a_3 w_3 + \cdots + a_n w_n)$$



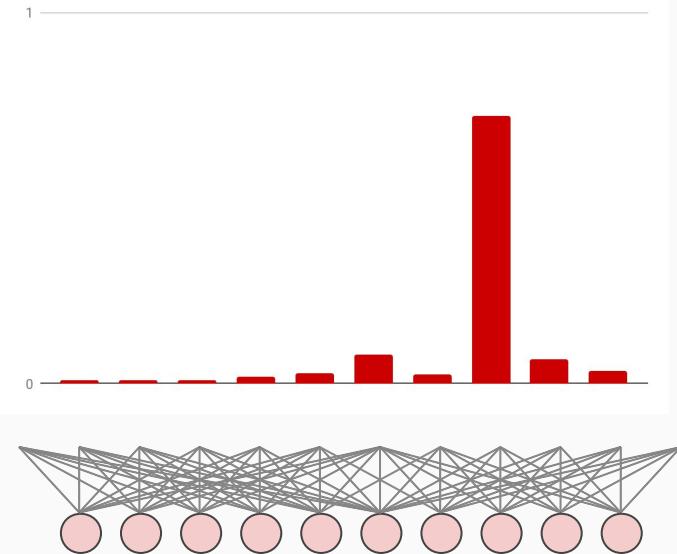
introduce
non-linear
complexities



Other Activation Functions



frequently used

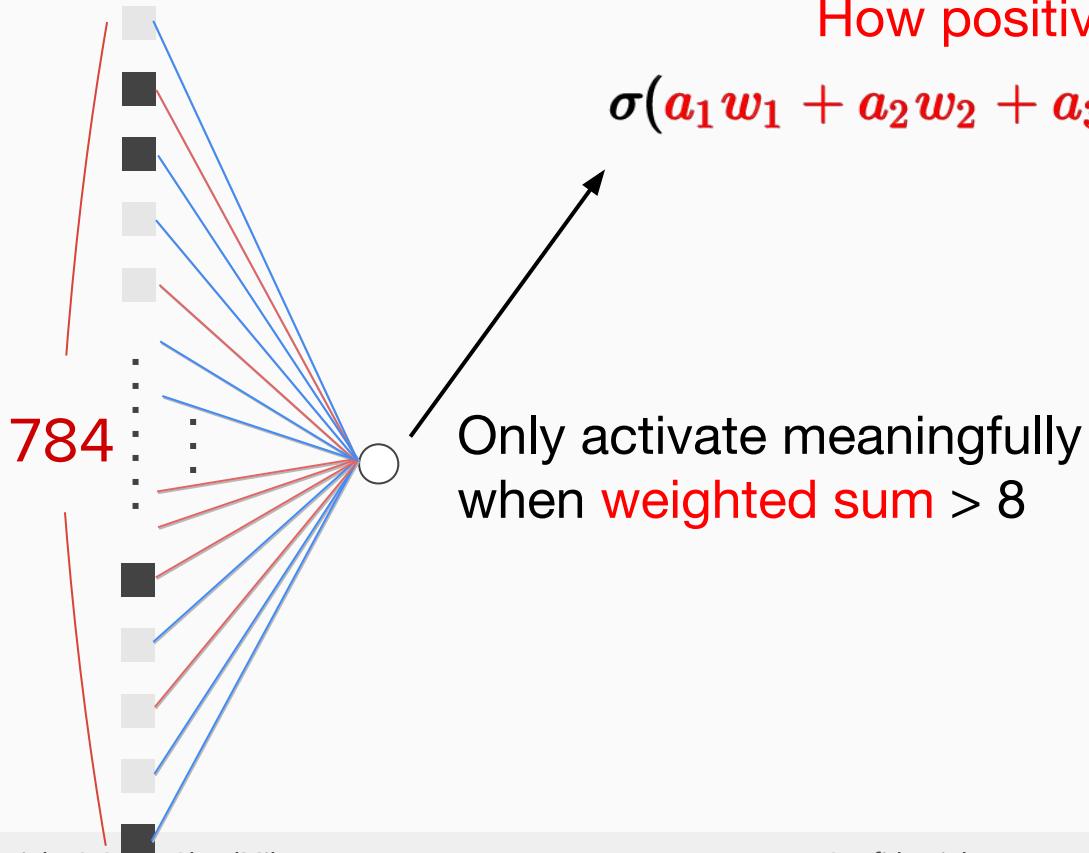


$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

weighted sum+b

classification output

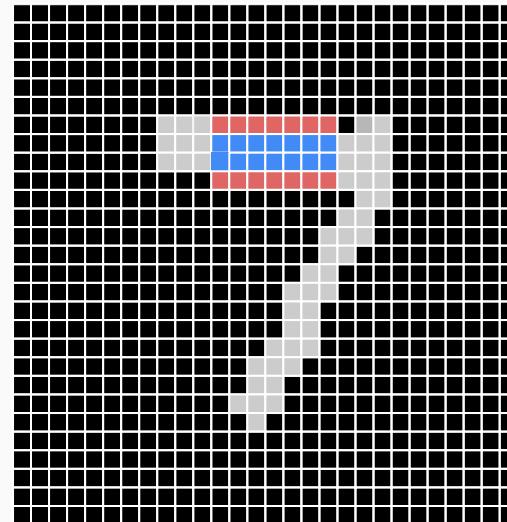
Bias for Inactivity



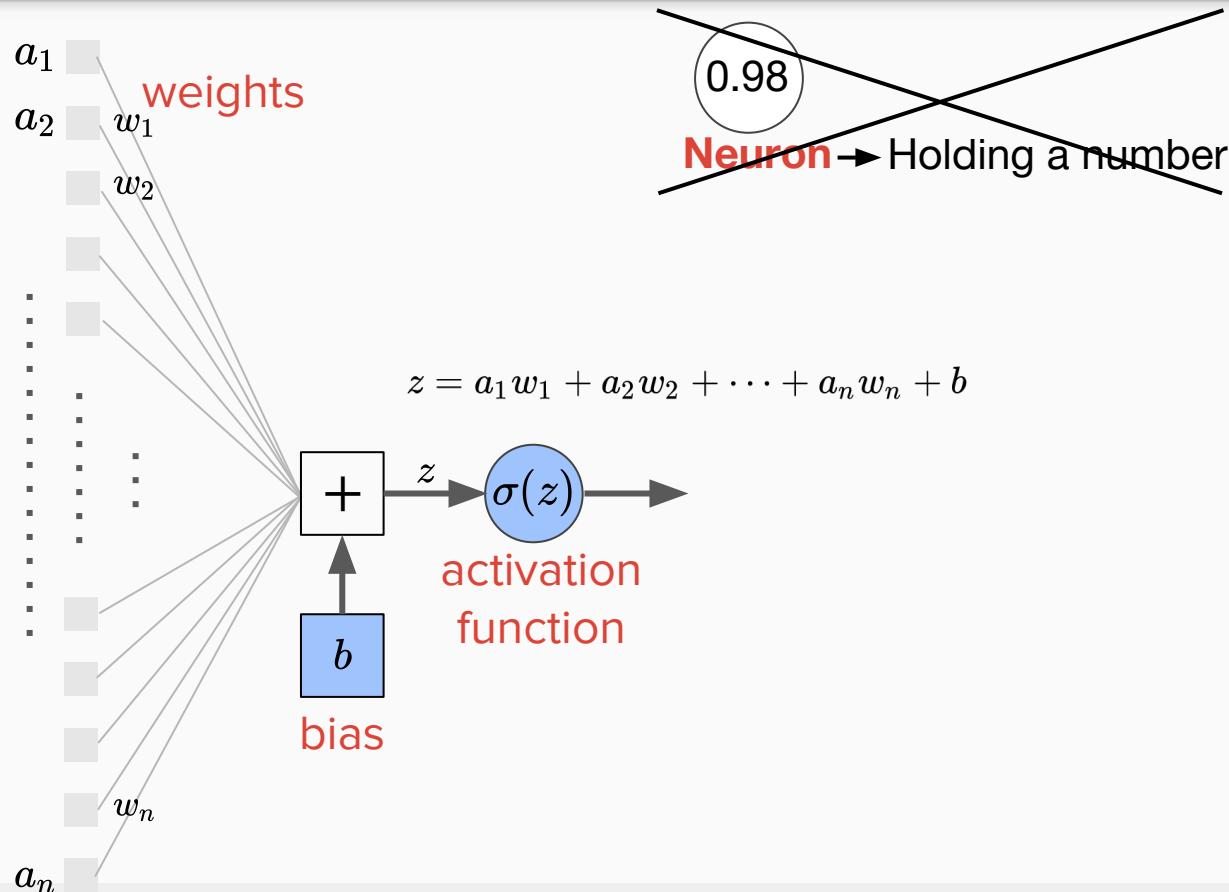
How positive is it?

$$\sigma(a_1 w_1 + a_2 w_2 + a_3 w_3 + \cdots + a_n w_n) - 8)$$

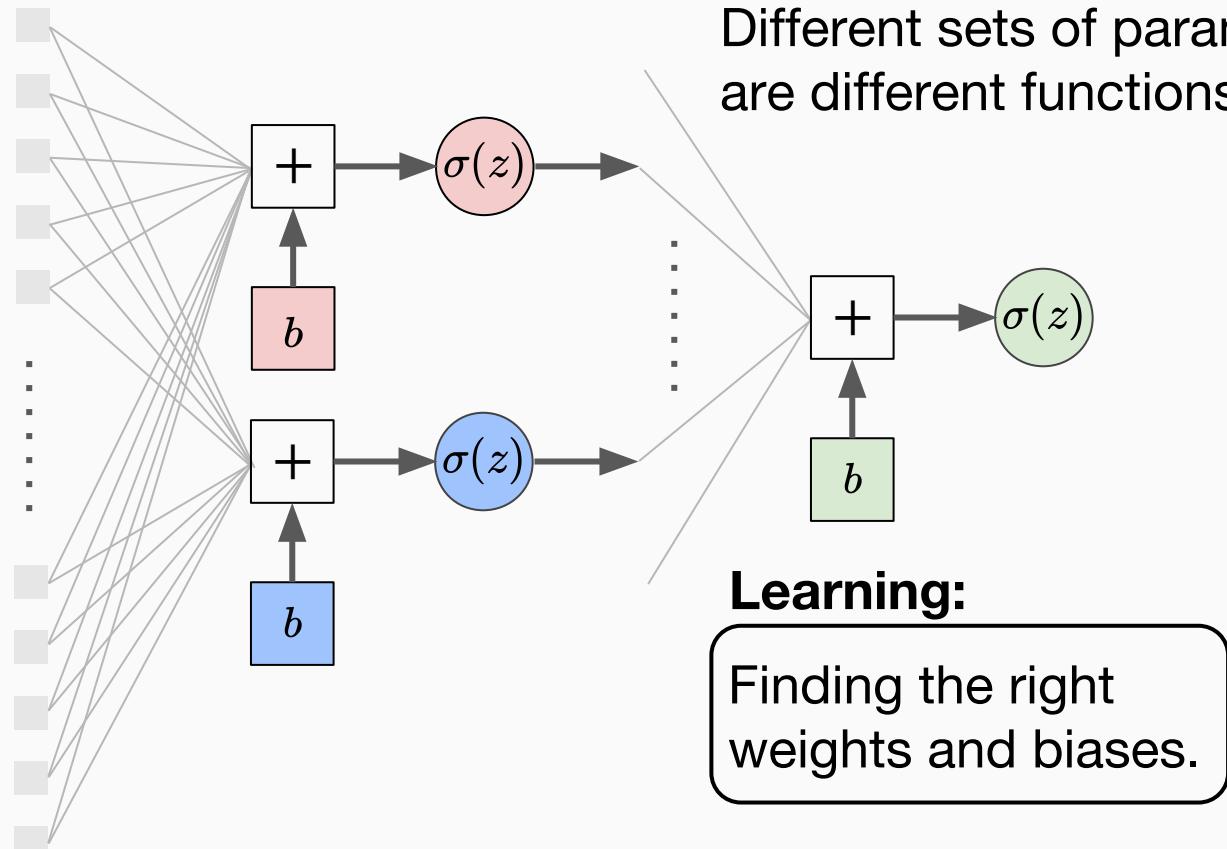
bias



Neuron is a Simple Function



Neural Network is a Function



Machine Learning Framework

Step 1

Neural Networks

Model

$f_1, f_2 \dots$

Step 2

goodness of function f

training data

Training

Prediction

Step 3

pick the “best” function

f^*

Input:
Output:



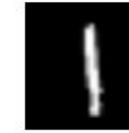
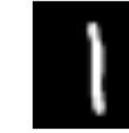
using f^*

“cat”

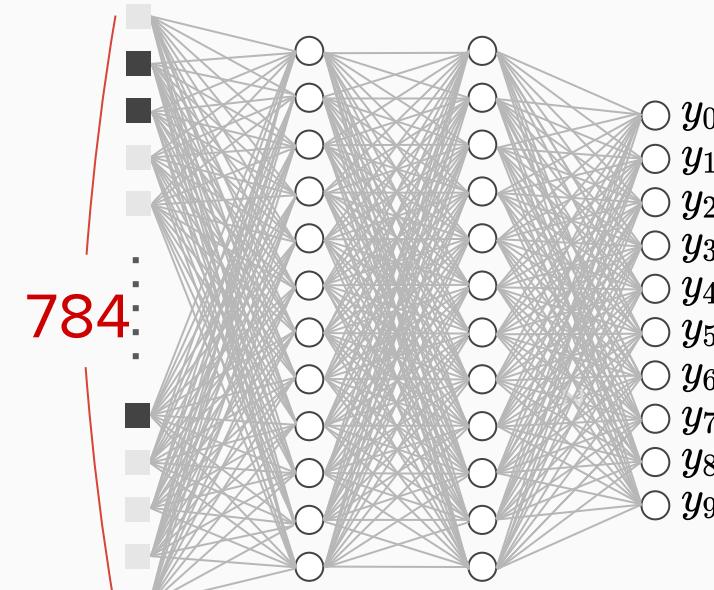
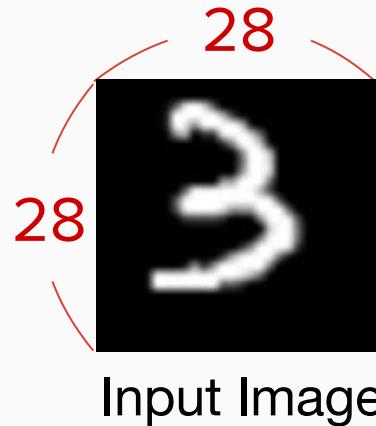
Confidential



Training Data

images →						...
labels →	“5”	“0”	“4”	“1”	“9”	...
						...
	“2”	“1”	“3”	“1”	“4”	...
						...
	“3”	“5”	“3”	“6”	“1”	...
						...
	“7”	“2”	“8”	“6”	“9”	

Goal of Learning

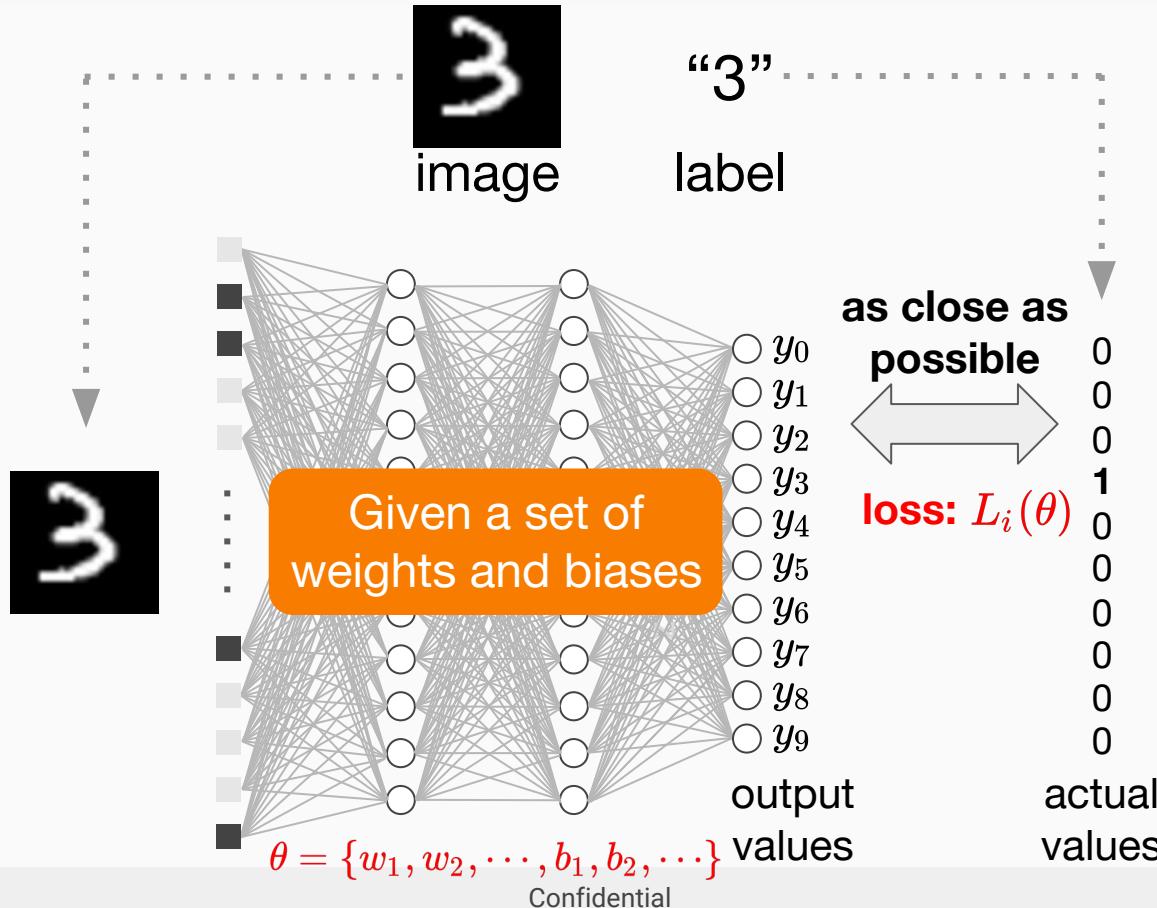


Goal

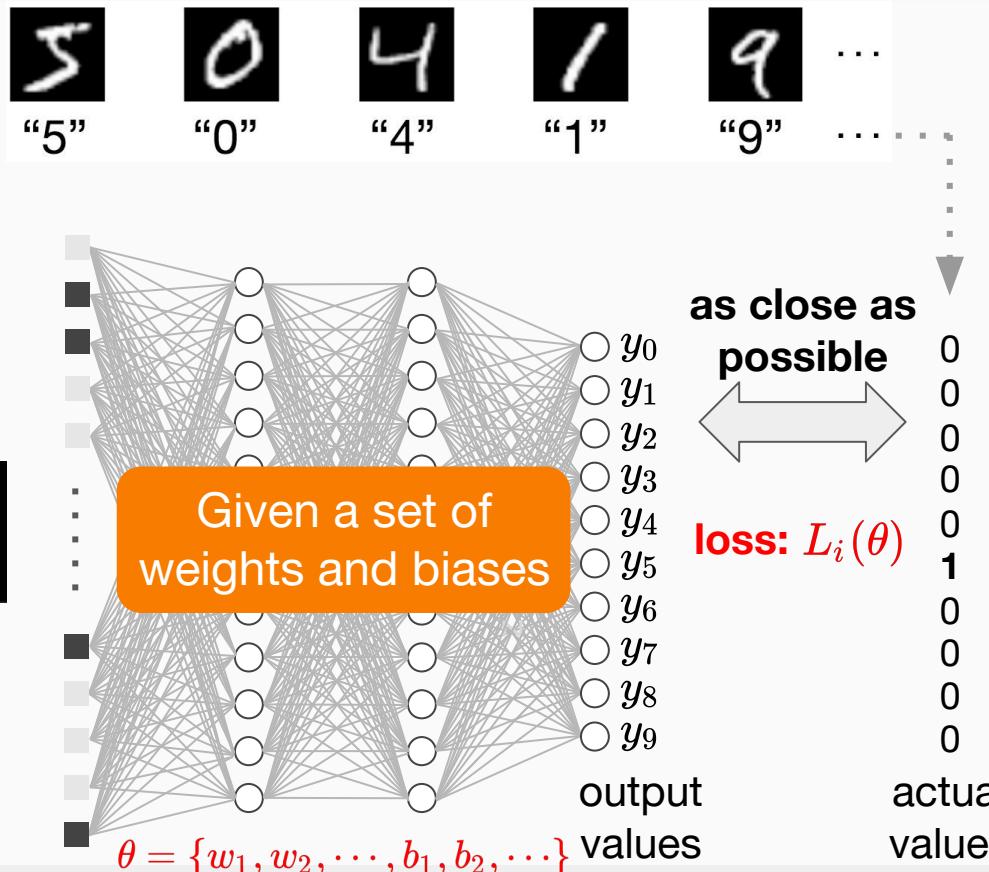
Input : y_3 has the maximum value

Input : y_8 has the maximum value

Loss



Average Loss of all training data ...



as small as possible

average loss:

$$\frac{1}{N} \sum_{i=1}^N L_i(\theta)$$

Learning:

Finding the right weights and biases **that minimize average loss**

Some Loss Functions for (Binary) Classification

- Mean Squared Error (MSE): $-\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$

$$-\frac{(0.7-1)^2 + (0.1-0)^2 + (0.2-0)^2}{3} = 0.0467$$

output value	actual value
(0.7 ,	1)
(0.1 ,	0)
(0.2 ,	0)

- Cross Entropy: $-\frac{1}{N} \sum_{i=1}^N [\hat{y}_i \log(y_i) + (1 - \hat{y}_i) \log(1 - y_i)]$

$$-\frac{1 \times \log(0.7) + (1 - 0) \times \log(1 - 0.1) + (1 - 0) \times \log(1 - 0.2)}{3} = 0.2284$$

Machine Learning Framework

Step 1

Neural Networks

Model

$f_1, f_2 \dots$

Training

Step 2

goodness of
function f

Loss

training
data

Step 3

pick the “best” function

f^*

Prediction

“cat”

using f^*

Input:
Output:

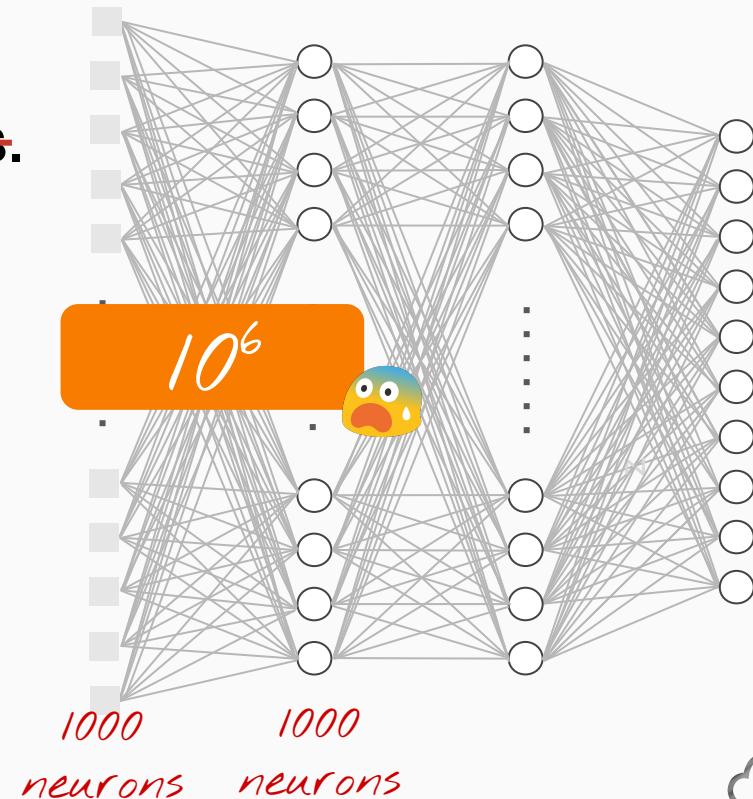
			
“cat”	“cat”	“dog”	“dog”



How to Find the Right Weights and Biases?

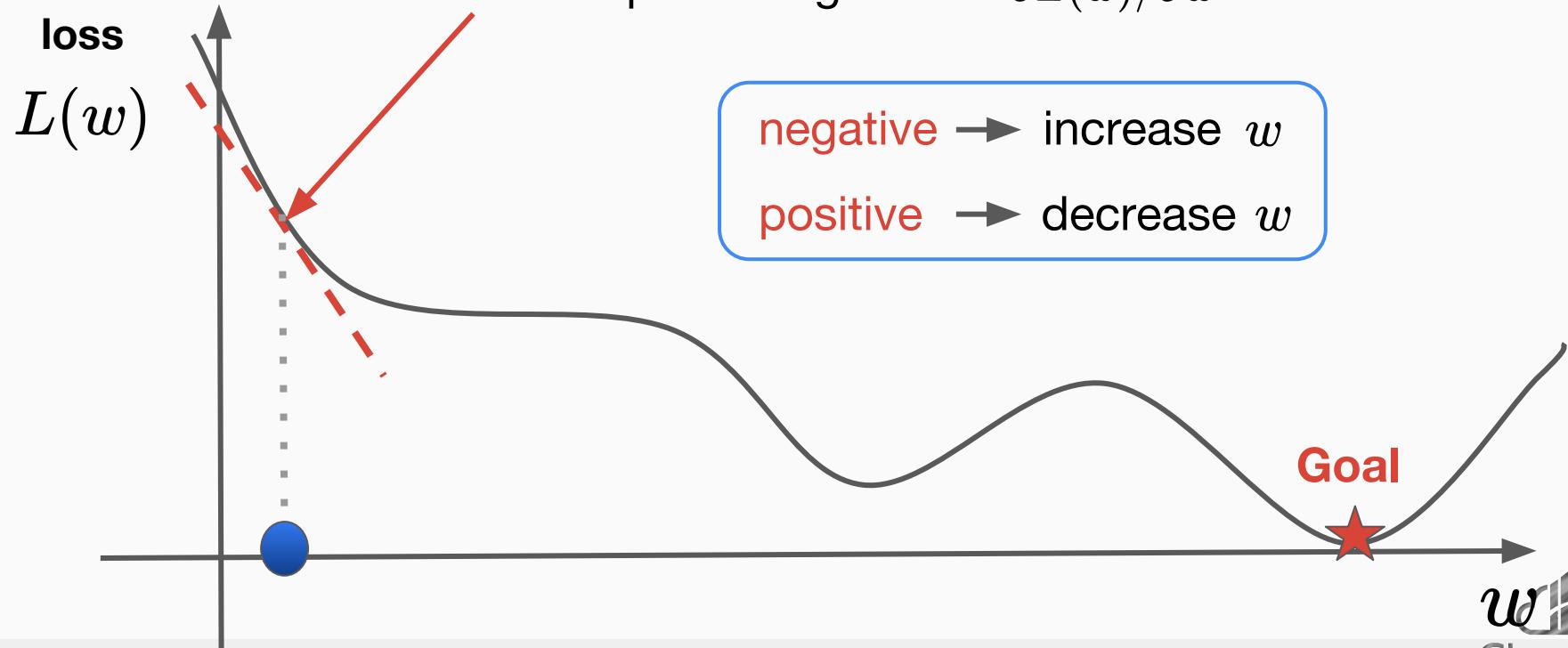
- ~~Enumerate all possible values.~~
- more efficient approach?

Gradient Descent

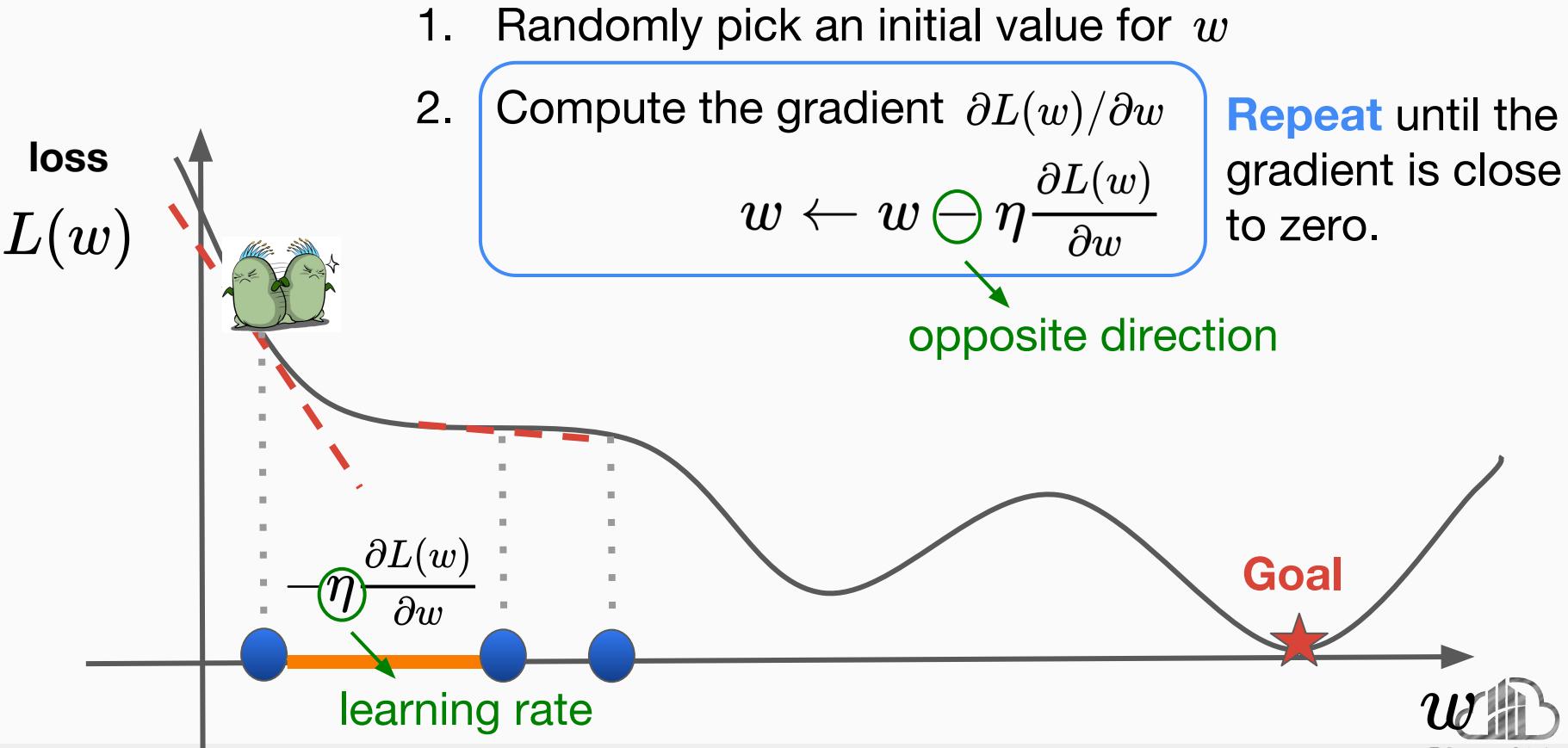


Gradient Descent

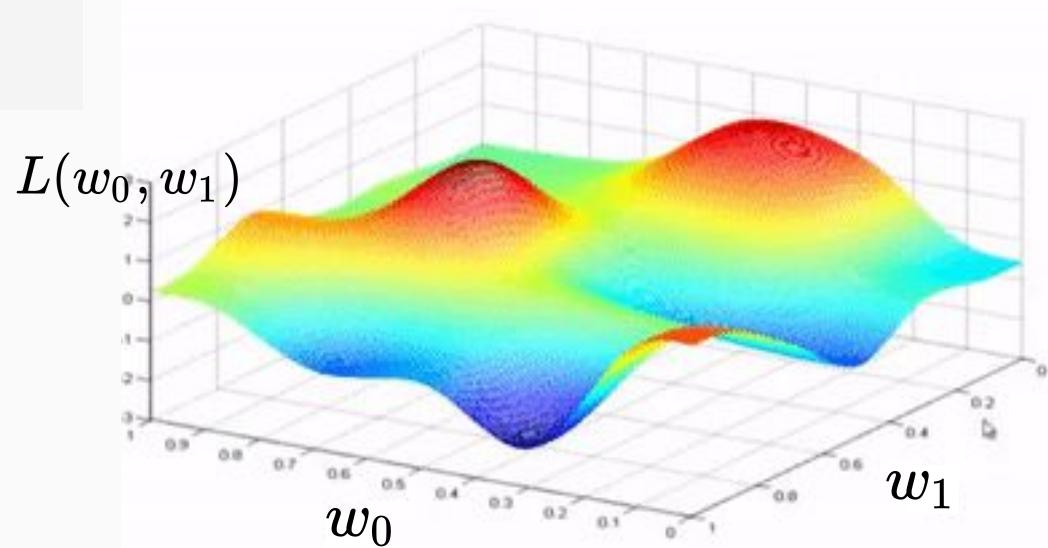
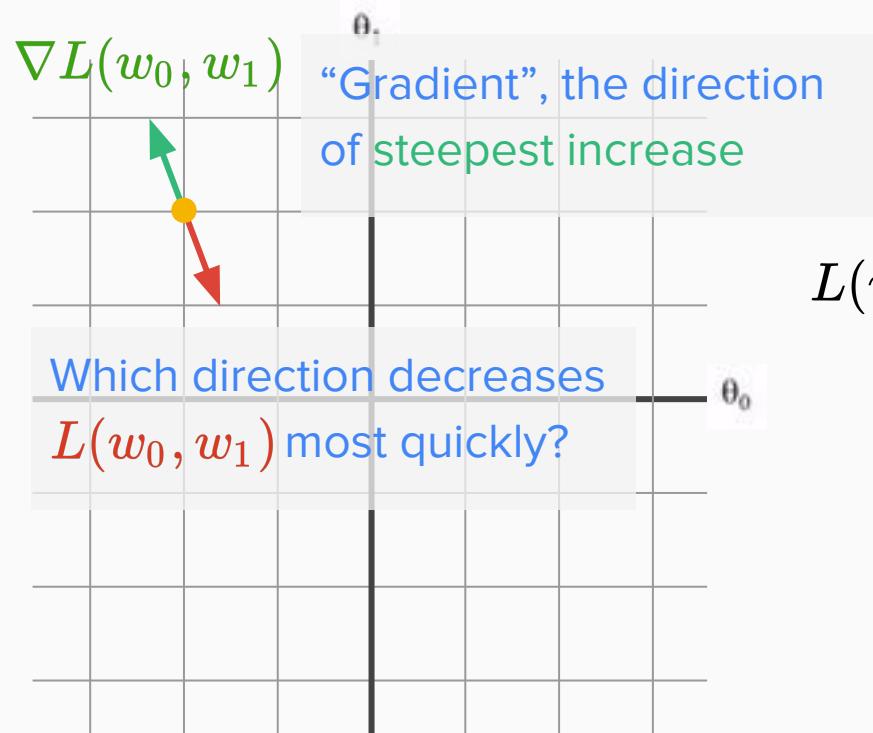
1. Randomly pick an initial value for w
2. Compute the gradient $\partial L(w)/\partial w$

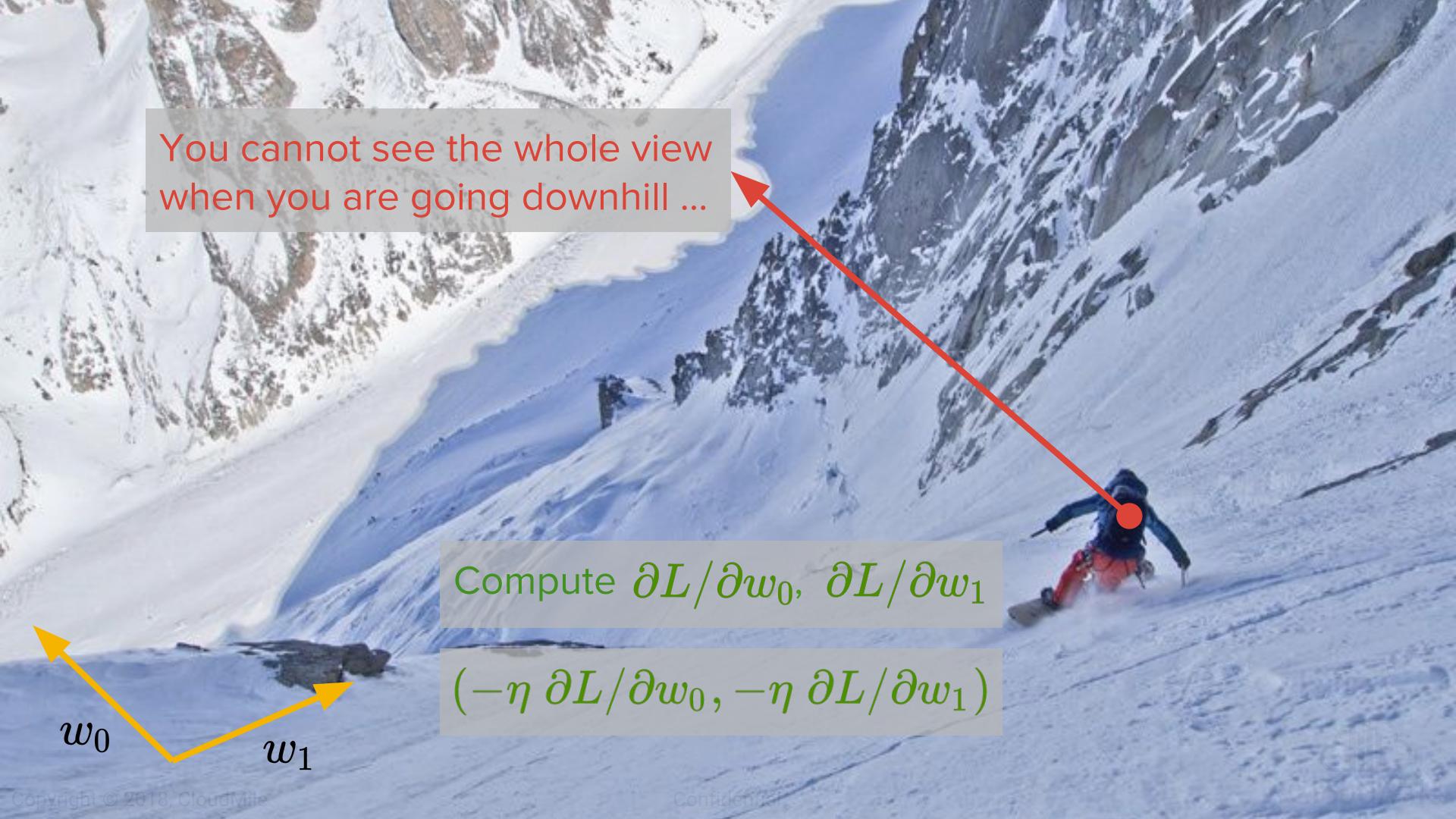


Gradient Descent (Cont'd)



Gradient Descent for Two Parameters





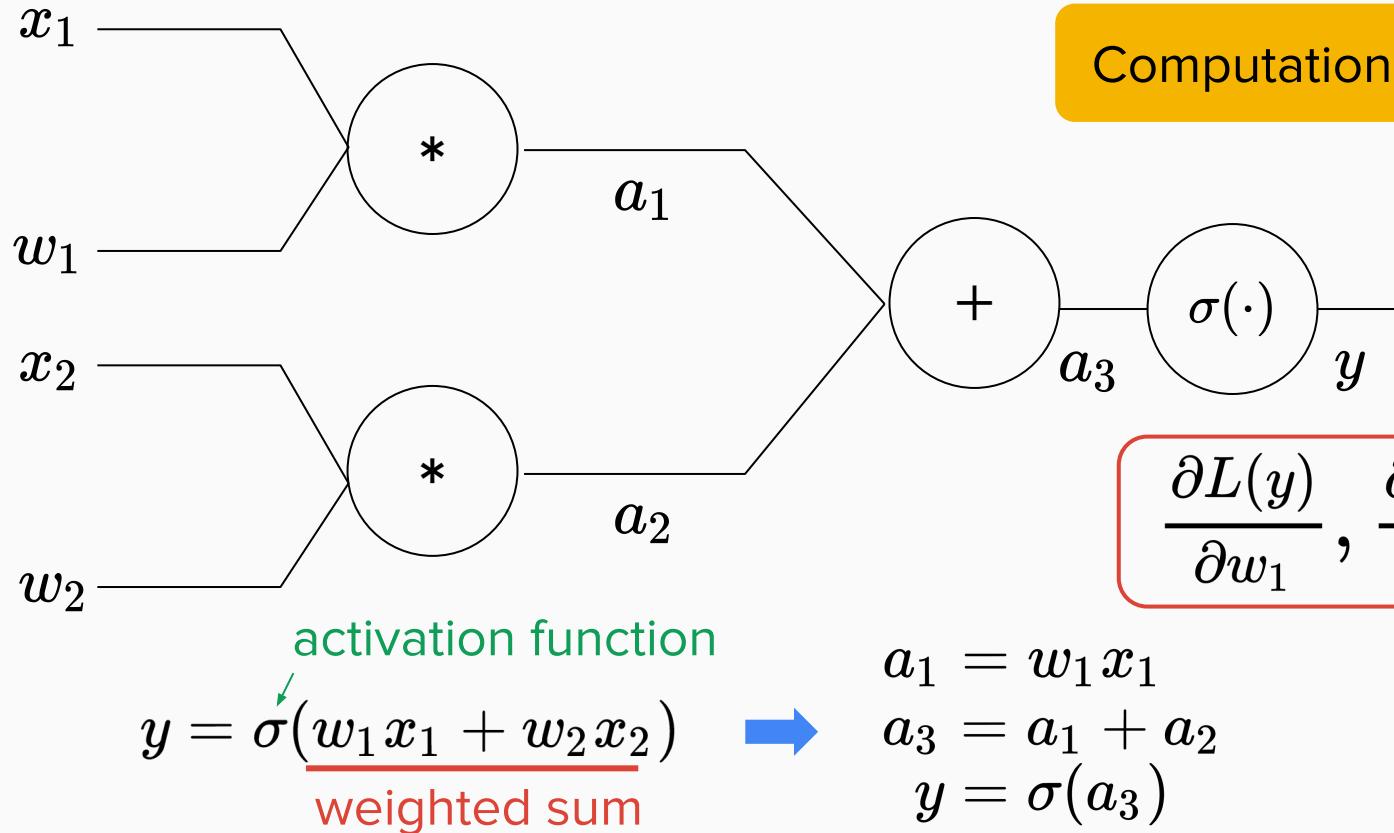
You cannot see the whole view
when you are going downhill ...



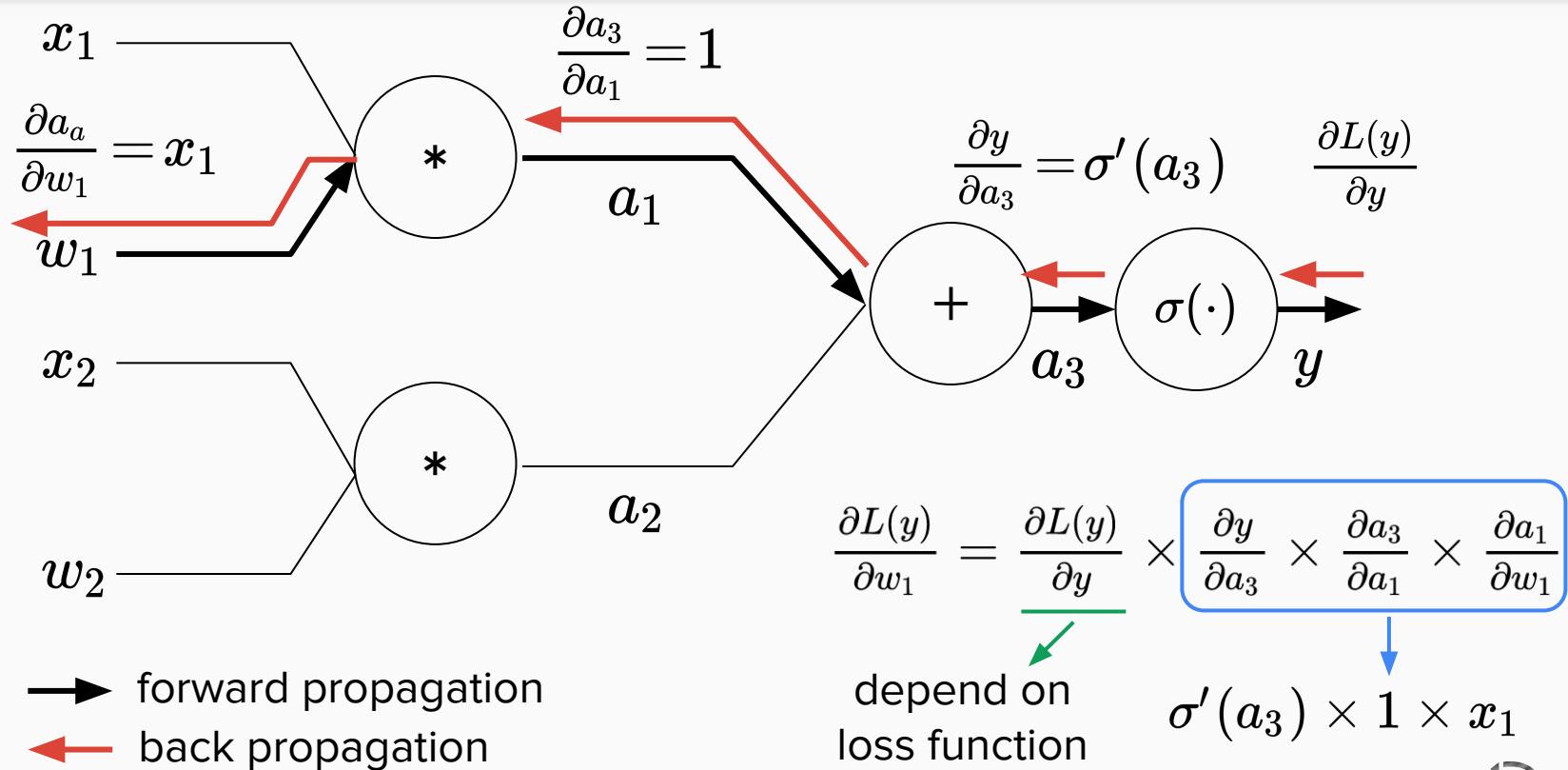
Compute $\partial L / \partial w_0, \partial L / \partial w_1$

$(-\eta \partial L / \partial w_0, -\eta \partial L / \partial w_1)$

Let's see a simple case when there is only one neuron ...



Back propagation: an efficient way to compute the gradient in NN



Don't worry about the gradient since the toolkits will handle it :)



TensorFlow



Caffe

Ref: <https://youtu.be/ibJpTrp5mcE>

Copyright © 2018, CloudMile

Confidential



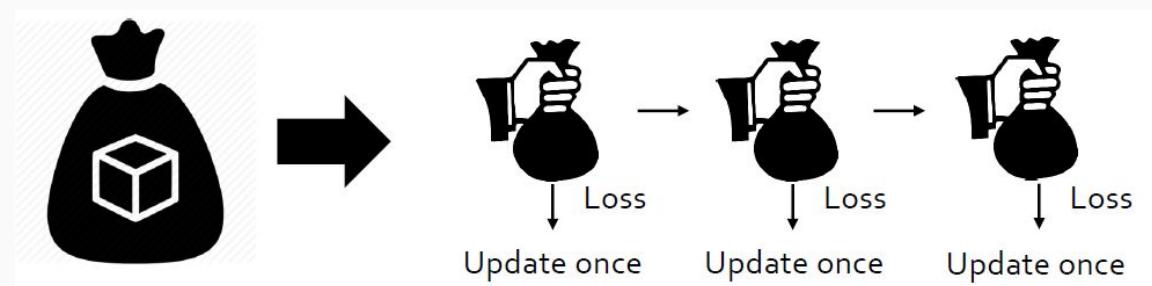
Stochastic/Mini-batch Gradient Descent

Learning:

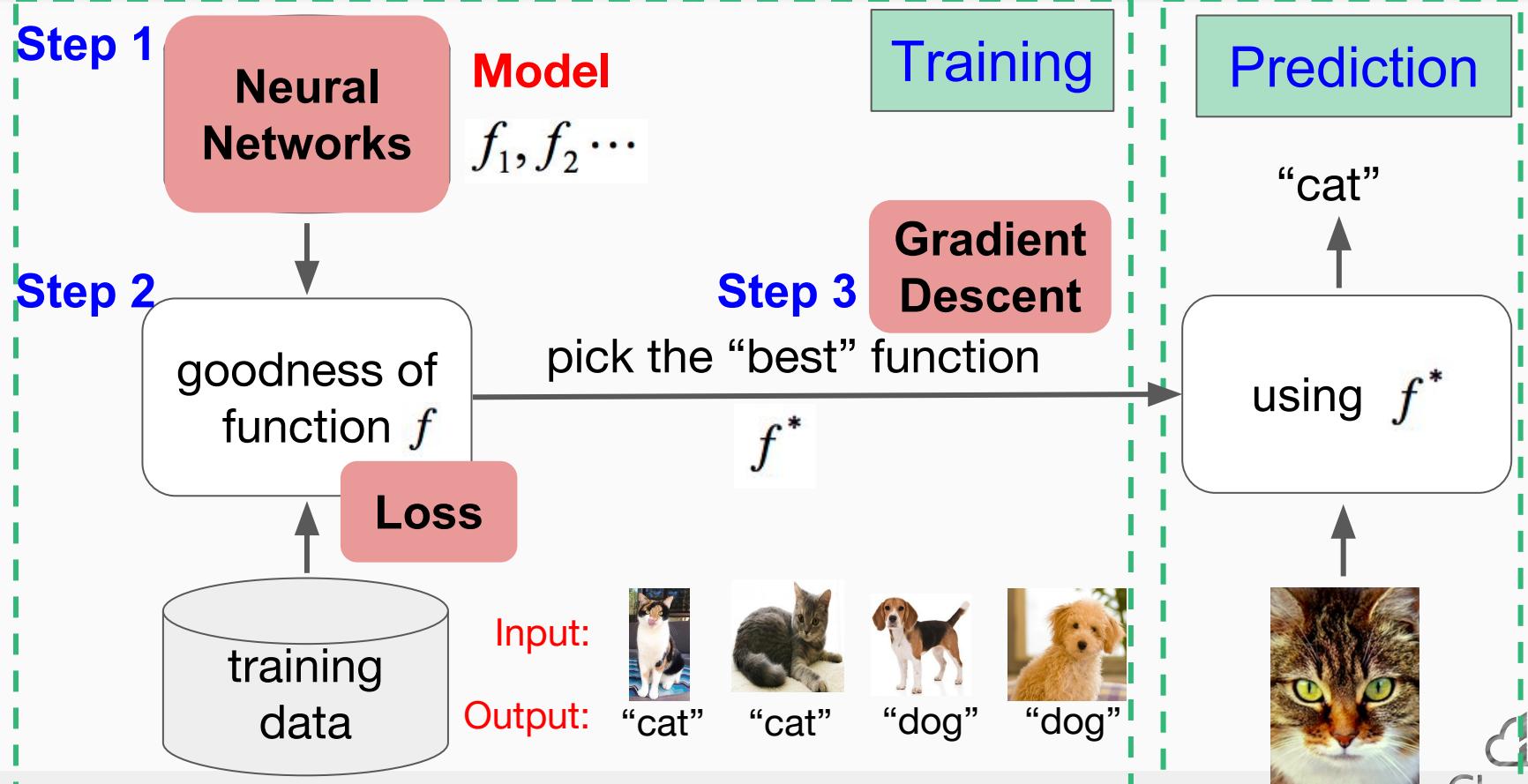
Finding the right weights
and biases that minimize
average loss

But, averaging the loss
of **all** training data is
very inefficient ...

- Stochastic: pick a **single** training data, then perform a parameter update.
- Mini-batch: pick a **batch** training data, then perform a parameter update.
 - e.g., batch size: 16, 32, 64, ...



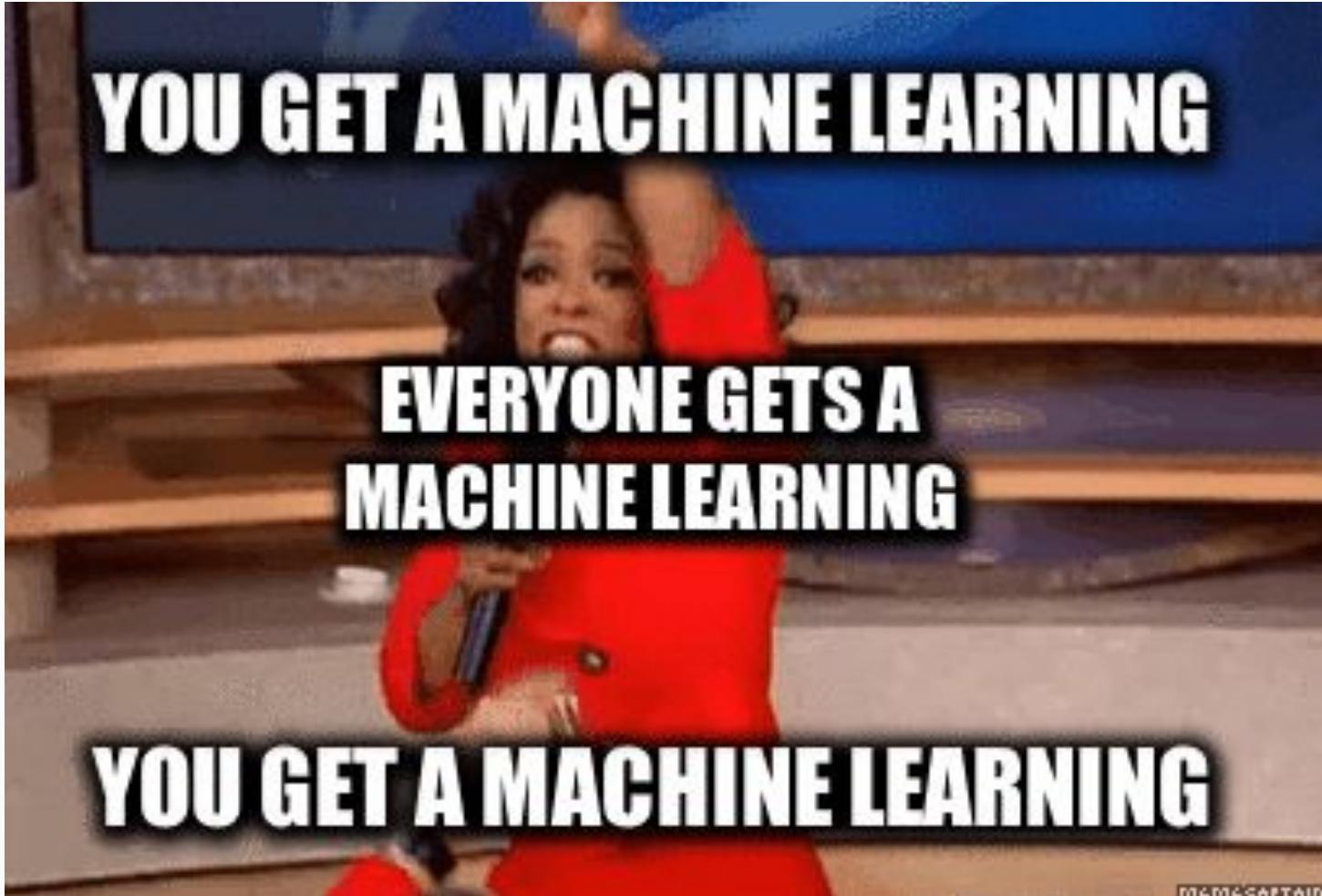
Machine Learning Framework



YOU GET A MACHINE LEARNING

**EVERYONE GETS A
MACHINE LEARNING**

YOU GET A MACHINE LEARNING



MEME CAPTION

Build your first neural network!



CloudMile

Tools that we'll use today!

colab



TensorFlow



Keras

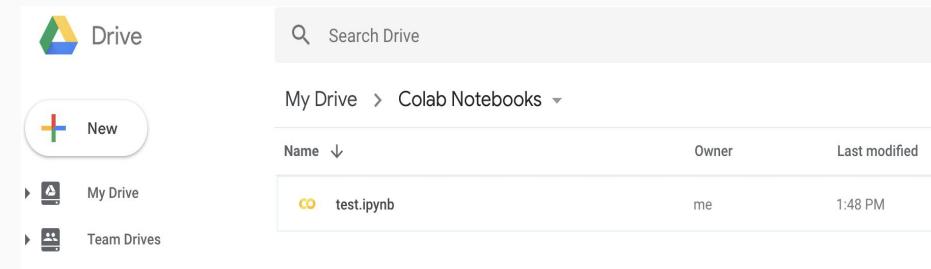
colab



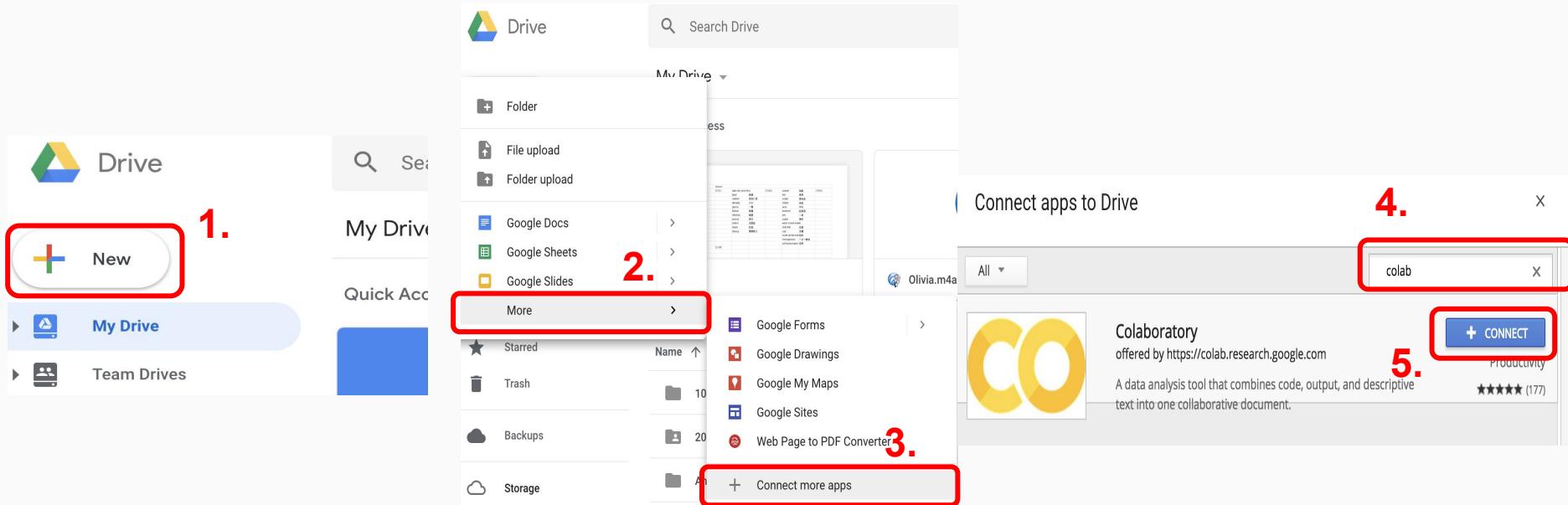
K Keras

Colab

- 類似Jupyter Notebook，互動式介面，及時輸出結果
- 儲存在Drive 上，可以共同編輯
- 背後有VM 在運作
- 可連續使用**免費的 Tesla K80 GPU 12小時(每次)**
- 許多套件已事先安裝，例如：TensorFlow, scikit-learn, numpy, matplotlib

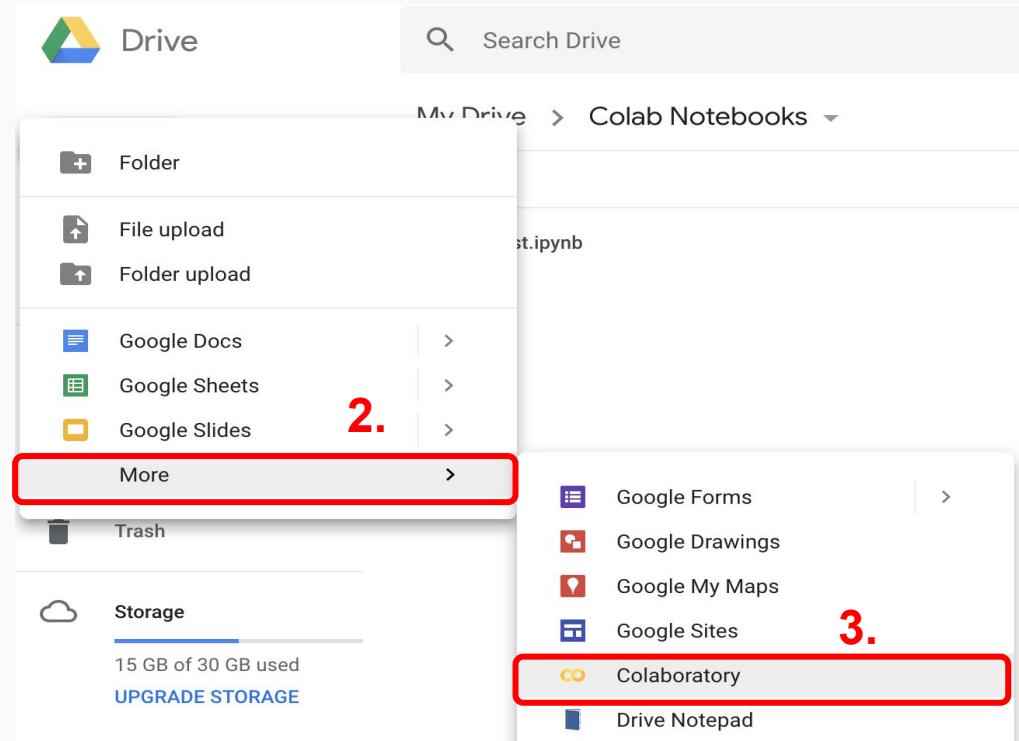
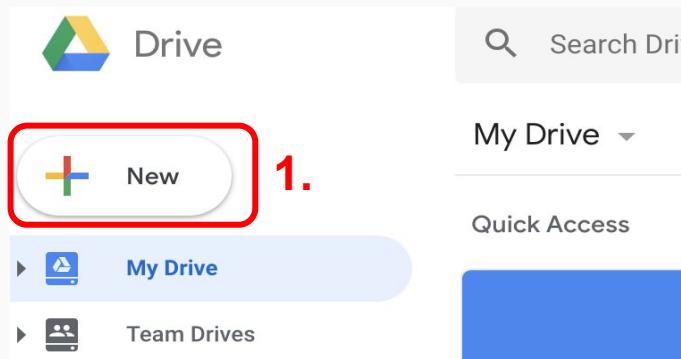


- 進入Google Drive 連接到Colaboratory

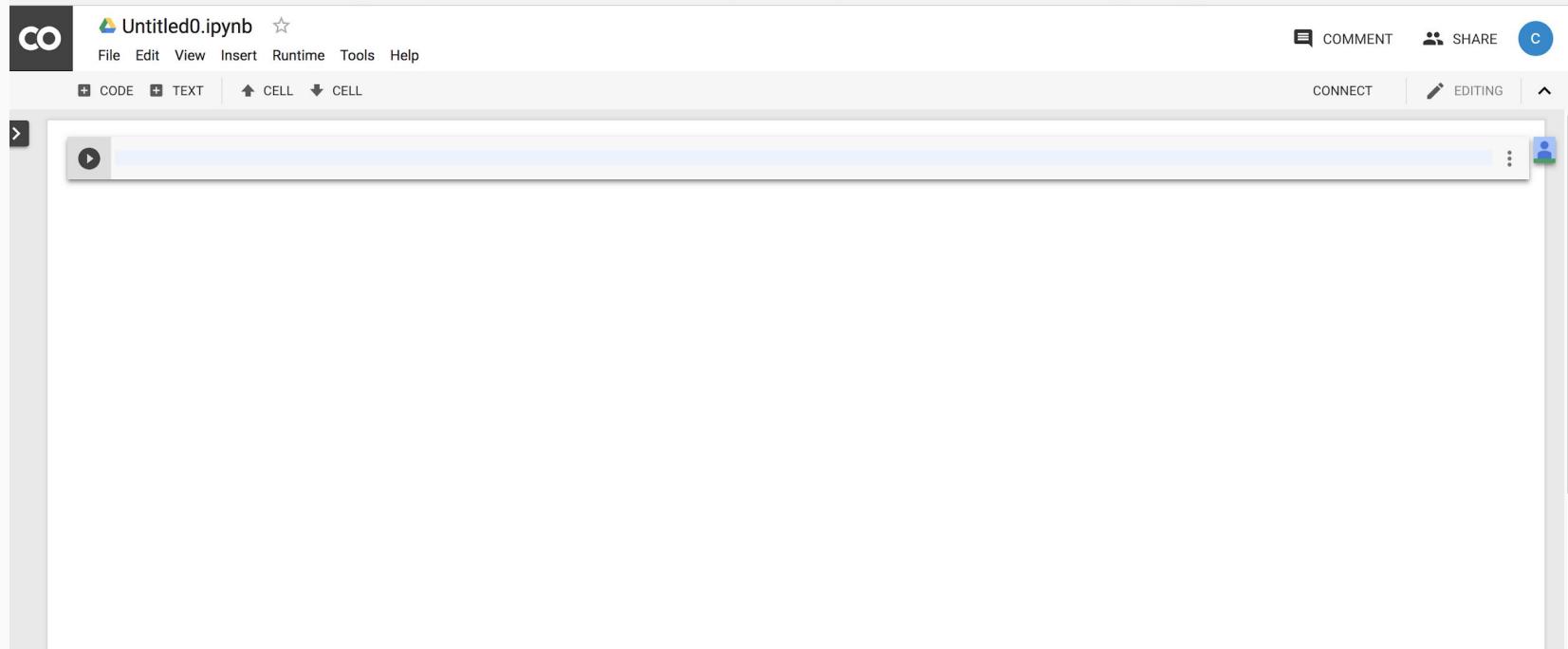


Colab

- 建立一個新的Colaboratory



Colab



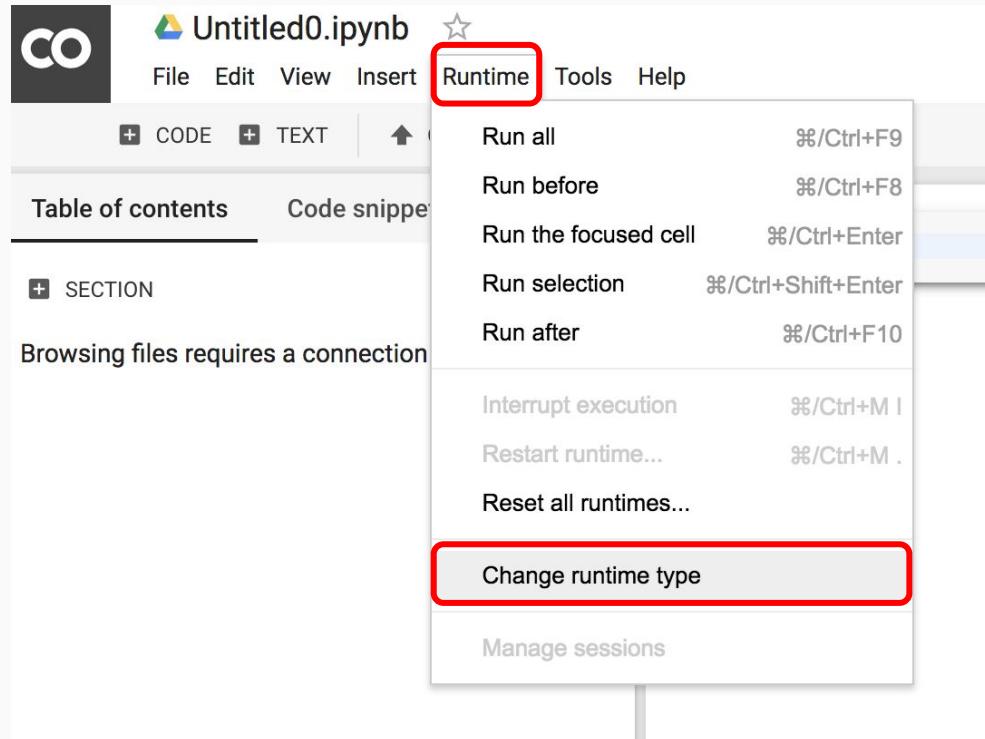
Colab

The screenshot shows the Google Colab interface. At the top left is the CloudMile logo (a stylized 'CO' in white). Next to it is the file name "Untitled0.ipynb" with a star icon. The top menu bar includes File, Edit, View, Insert, Runtime, Tools, and Help. Below the menu is a toolbar with buttons for CODE, TEXT, and CELL operations. On the left side, there's a sidebar with "Table of contents" (underlined), "Code snippets", "Files", and an "X" button. A "SECTION" button is also present. The main workspace is currently empty, showing a play button icon.

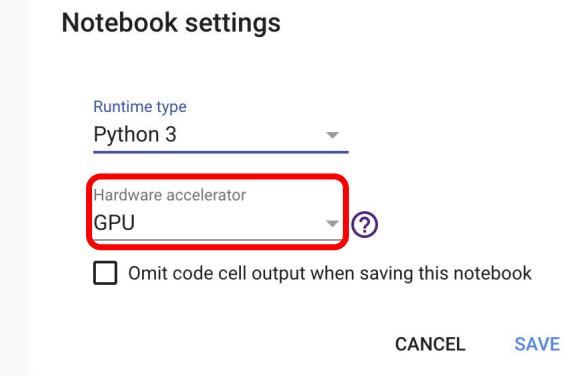
Colab

The screenshot shows the Google Colab interface. At the top, there's a navigation bar with a 'CO' logo, the file name 'Untitled0.ipynb', and icons for Comment, Share, and a user profile. Below the bar are menu options: File, Edit, View, Insert, Runtime, Tools, Help. A toolbar below the menu includes buttons for CODE, TEXT, CELL UP, and CELL DOWN. The main area has tabs for 'Table of contents', 'Code snippets', 'Files', and 'X'. The 'Table of contents' tab is highlighted with a red border. On the left, there's a sidebar with a 'SECTION' button and a message: 'Browsing files requires a connection to a runtime.' A large central workspace is currently empty.

Colab



● 使用 Python 3



Colab

A screenshot of the Google Colab interface. At the top, there's a blue header bar with the word "Colab". Below it is a dark toolbar with a "CO" logo, file navigation, and search functions. The main workspace shows a notebook titled "Untitled0.ipynb". The top menu bar includes "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". Below the menu is a toolbar with buttons for "CODE", "TEXT", "CELL UP", "CELL DOWN", and "COMMENT". A "SHARE" button is also present. On the left, there's a sidebar with "Table of contents", "Code snippets", "Files", and "SECTION" buttons. The main area has a play button icon. To the right, there's a "CONNECT" dropdown menu with options "Connect to hosted runtime" and "Connect to local runtime...". A large red arrow points from the left towards this dropdown menu. The bottom right corner of the screen shows a "CloudMile" watermark.

Colab

The screenshot shows the Google Colab interface. At the top, there's a navigation bar with a 'CO' logo, the title 'Untitled0.ipynb', and icons for 'COMMENT', 'SHARE', and a user profile. Below the title, the menu bar includes 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. Underneath the menu is a toolbar with buttons for '+ CODE', '+ TEXT', 'CELL UP', and 'CELL DOWN'. To the right of the toolbar are status indicators: a green checkmark for 'CONNECTED', a pencil icon for 'EDITING', and a person icon. On the left side, there's a sidebar with 'Table of contents', 'Code snippets', and a 'Files' tab which is currently selected. The 'Files' tab has options for 'UPLOAD' and 'REFRESH'. The main area displays a file tree under the 'sample_data' folder, which contains 'README.md', 'anscombe.json', 'california_housing_test.csv', 'california_housing_train.csv', 'mnist_test.csv', and 'mnist_train_small.csv'. A large, empty workspace is visible on the right.



CloudMile

基本語法 - Python

- 輸入 Python 程式碼，點擊  或是「Shift + Enter」執行程式碼區塊。



The screenshot shows the Google Colab interface. At the top, there's a navigation bar with a 'co' logo, the file name 'Untitled0.ipynb', and a star icon. Below the bar are menu options: File, Edit, View, Insert, Runtime, Tools, Help. Underneath the menu is a toolbar with buttons for '+ CODE', '+ TEXT', and navigation arrows for 'CELL'. On the left side, there's a sidebar with 'Table of contents', 'Code snippets', and a 'Files' tab which is currently selected. Below the sidebar are buttons for 'UPLOAD' and 'REFRESH', and a folder named 'sample_data'. The main area contains a code cell with the following content:

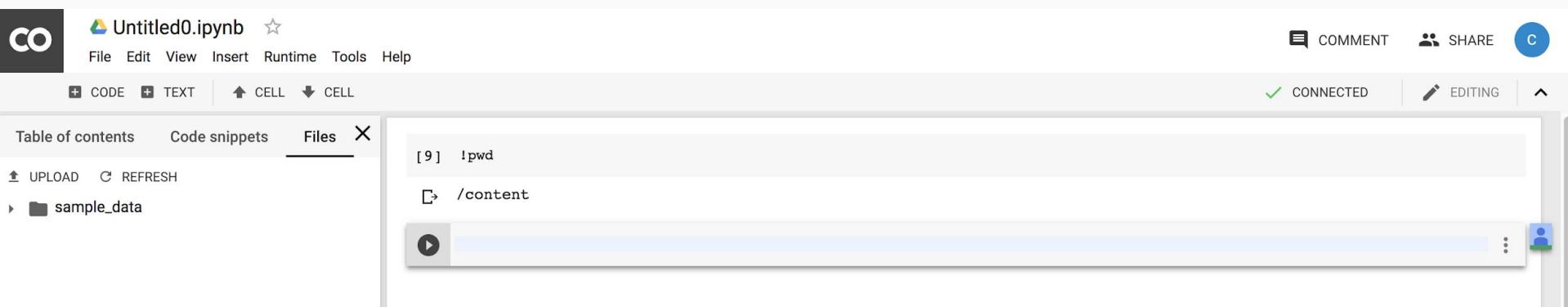
```
[8] print('Hello World')
```

The cell has executed and produced the output:

```
Hello World
```

基本語法 - Shell

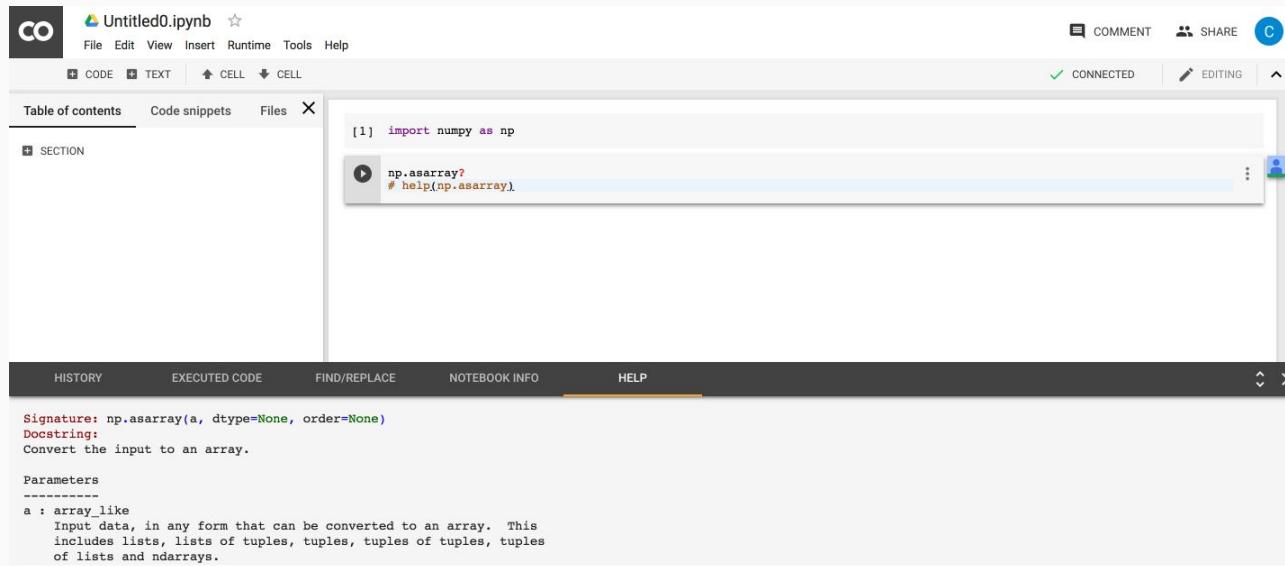
- 前面加上 !



The screenshot shows the Google Colab interface. At the top, there's a navigation bar with a 'CO' logo, the file name 'Untitled0.ipynb', and icons for Comment, Share, and a user profile. Below the bar are tabs for File, Edit, View, Insert, Runtime, Tools, Help, CODE, TEXT, CELL, and FILES. The FILES tab is currently selected. On the left, there's a sidebar with 'Table of contents', 'Code snippets', and a 'Files' section containing 'sample_data'. The main area shows a terminal cell with the command '!pwd' and its output '/content'. A play button icon is at the bottom of the cell.

特殊語法 - 查看

- 在function 後面加上? 或是用help()可以查看documentation



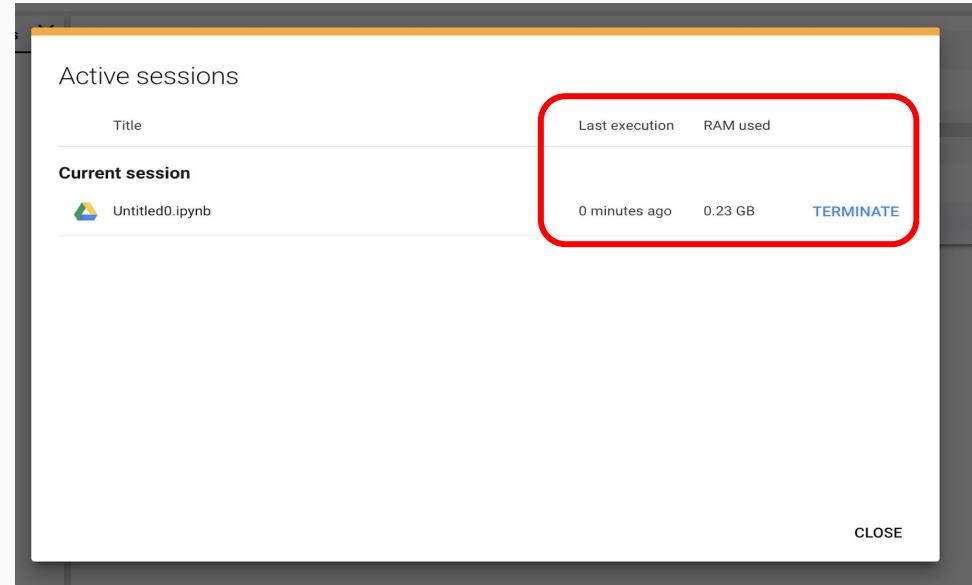
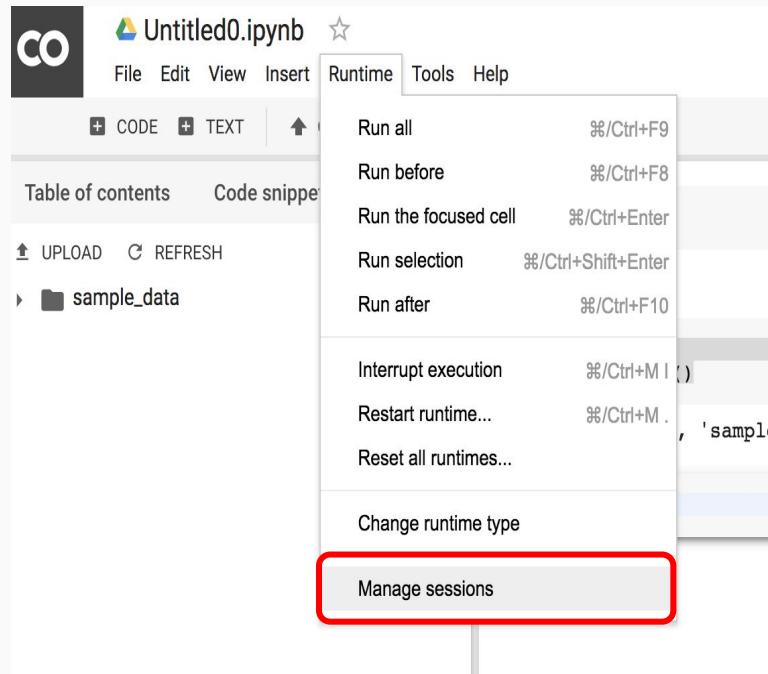
The screenshot shows a Google Colab notebook titled "Untitled0.ipynb". In the code cell, the user has typed "np.asarray?" followed by a question mark. A tooltip appears over the question mark, displaying the command "# help(np.asarray)". Below the cell, the executed code shows the function signature and docstring for np.asarray. The docstring indicates that it converts input to an array. The "Parameters" section defines 'a' as an array-like input that can be converted to an array, including lists, tuples, and ndarrays.

```
[1] import numpy as np
[2] np.asarray?
# help(np.asarray)

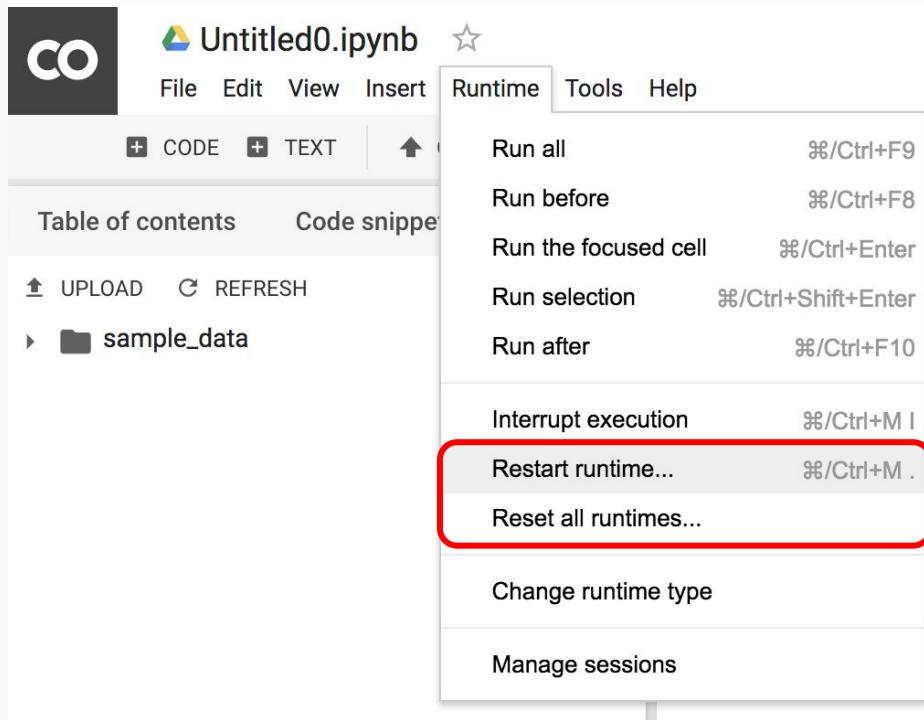
Signature: np.asarray(a, dtype=None, order=None)
Docstring:
Convert the input to an array.

Parameters
-----
a : array_like
    Input data, in any form that can be converted to an array. This
    includes lists, lists of tuples, tuples, tuples of tuples, tuples
    of lists and ndarrays.
```

Manage Session



Restart runtime/ Reset runtime



Colab

從網站下載 code 到本機 -> 網址 : https://github.com/CloudMile/allianz_courses

CloudMile / allianz_courses

Unwatch 6 Star 0 Fork 2

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

CloudMile course materials for Allianz Edit

Manage topics

30 commits 1 branch 0 releases 4 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

Ching-Yi Lee first commit Latest commit 47e4518 37 seconds ago

Intro_to_ML_NLP_workshop	Merge branch 'master' of github.com:CloudMile/allianz_courses	2 days ago
deep_learning1	first commit	37 seconds ago
insurance_use_case	#	12 hours ago
.gitignore	#	16 days ago
README.md	Update README.md	16 days ago



Colab

CloudMile / allianz_courses

Unwatch 6 Star 0 Fork 2

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Branch: master → allianz_courses / deep_learning1 / lab2.ipynb Find file Copy path

Ching-Yi Lee first commit 47e4518 5 minutes ago

0 contributors

Raw

1 lines (1 sloc) | 474 KB

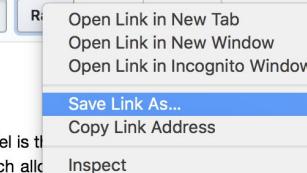
Getting started: Introduction to Keras

The core data structure of Keras is a [model](#), a way to organize layers. The simplest type of model is the stack of layers. For more complex architectures, you should use the [Keras functional API](#), which allows layers.

Here is the Sequential model:

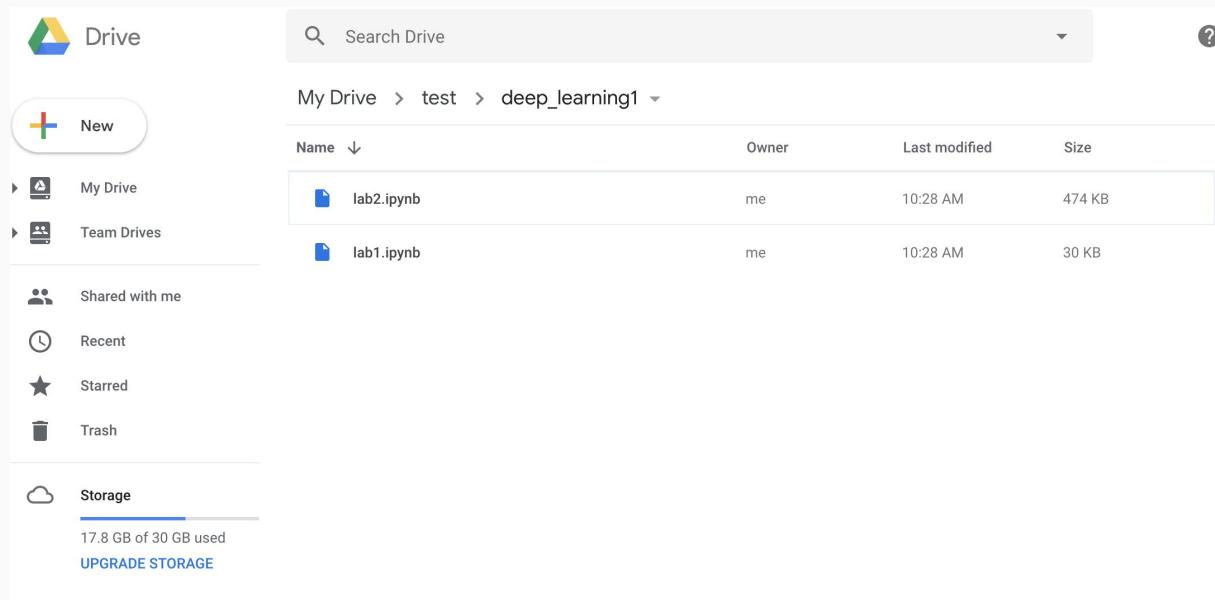
```
from keras.models import Sequential  
  
model = Sequential()  
  
Stacking layers is as easy as .add():  
  
from keras.layers import Dense  
  
model.add(Dense(units=64, activation='relu', input_dim=100))  
model.add(Dense(units=10, activation='softmax'))
```

Once your model looks good, configure its learning process with .compile():

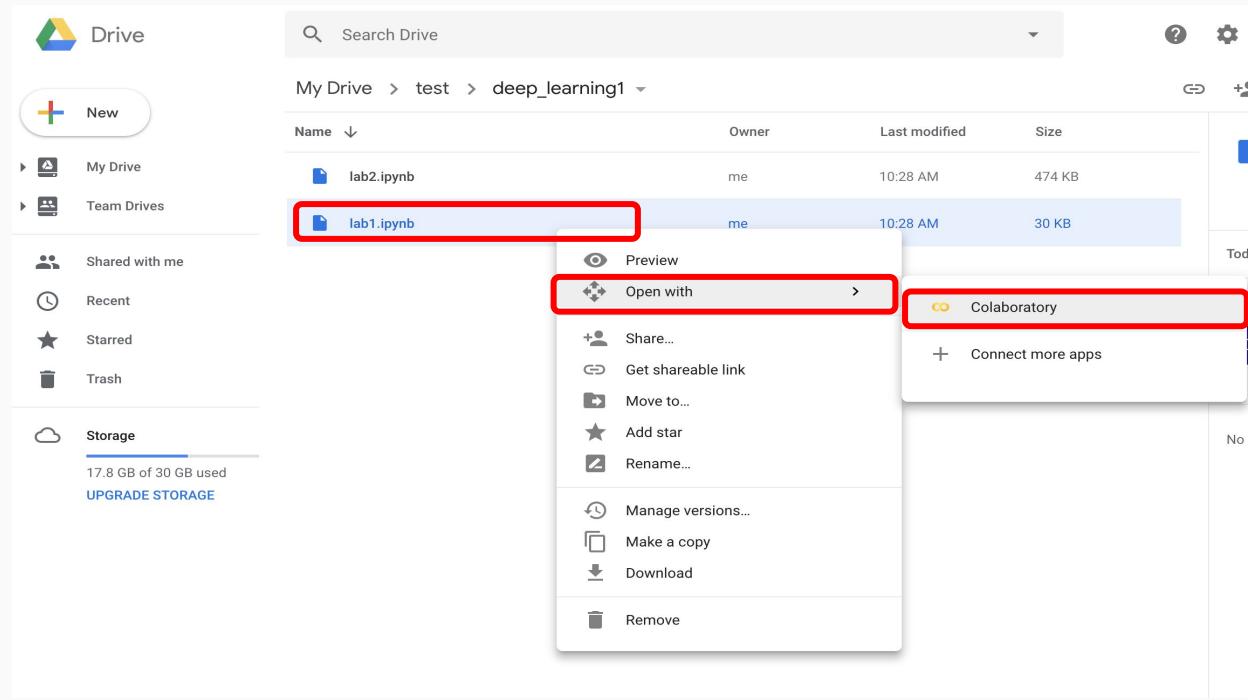


Open jupyter notebook from Colab

- Upload files to drive



Open jupyter notebook from Colab



Colab

lab2.ipynb ★

File Edit View Insert Runtime Tools Help

+ CODE + TEXT ↑ CELL ↓ CELL

CONNECT ▾ EDITING ▾

Table of contents Code snippets Files X

Getting started: Introduction to Keras

Import Necessary Library

Download Dataset

Data Exploration

Data Preprocessing

Normalizing(feature_scaling)

One-hot encoding

Train Model

(Lab 2-a) Basic Model: model_sig_sgd_001

(Lab 2-b) Change activation function to Relu: model_relu_sgd_001

(Lab 2-c) Change optimizer to Adam: model_relu_adam_001

(Lab 2-d) Change Learning Rate to 0.01: model_relu_adam_01

(Lab 2-e) Add more neurons in hidden layers(overfit): model_large_relu_adam_001

Overfit Solution

Getting started: Introduction to Keras

The core data structure of Keras is a `model`, a way to organize layers. The simplest type of model is the [Sequential](#) model, a linear stack of layers. For more complex architectures, you should use the [Keras functional API](#), which allows to build arbitrary graphs of layers.

Here is the Sequential model:

```
from keras.models import Sequential

model = Sequential()
```

Stacking layers is as easy as `.add()`:

```
from keras.layers import Dense

model.add(Dense(units=64, activation='relu', input_dim=100))
model.add(Dense(units=10, activation='softmax'))
```

Once your model looks good, configure its learning process with `.compile()`:

```
model.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])
```

If you need to, you can further configure your optimizer. A core principle of Keras is to make things reasonably simple, while allowing the user to be fully in control when they need to (the ultimate control being the easy extensibility of the source code).

```
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.SGD(lr=0.01, momentum=0.9, nesterov=True))
```

You can now iterate on your training data in batches:

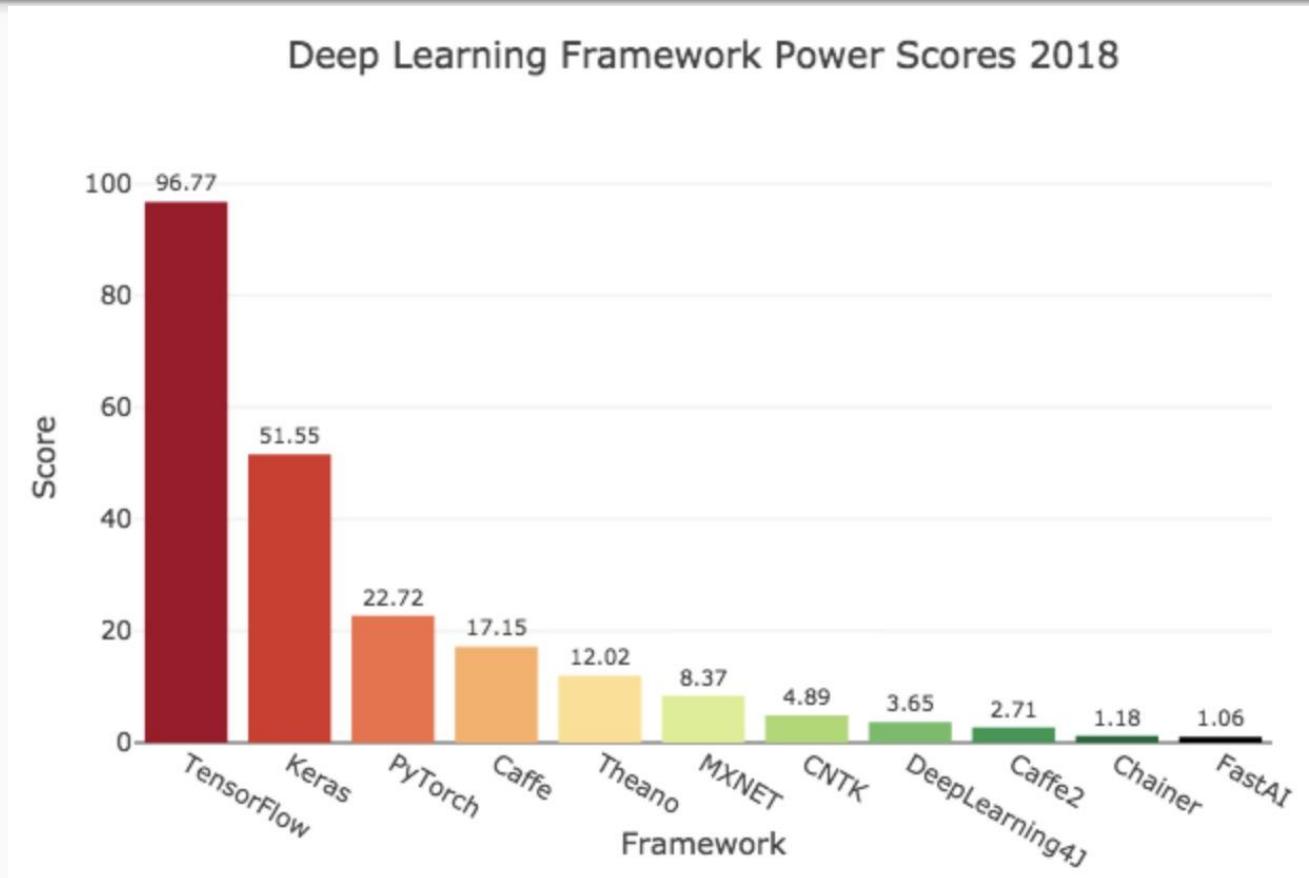
```
# x train and y train are Numpy arrays --just like in the Scikit-Learn API.
```



TensorFlow

colab K Keras

Different Deep Learning Frameworks



- Open source software library created by Google
- Google 開發的開源工具
- 透過 Computational Graph 運算



TensorFlow

Pre-made Estimators

models in a box

Estimator

Keras

train and evaluate models

Layers

Dataset

build models

Python Frontend

C++

Java

JavaScript

...

TensorFlow Distributed Execution Engine

CPU

GPU

TPU

iOS

Android

...

TensorFlow Fundamental

```
import numpy as np
import tensorflow as tf

x_data = np.random.rand(100).astype(np.float32)
y_data = x_data * 0.1 + 0.3

Input data

W = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
b = tf.Variable(tf.zeros([1]))
y = W * x_data + b

Graph

loss = tf.reduce_mean(tf.square(y - y_data))
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.5)
train = optimizer.minimize(loss)

init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)

Session

for step in range(201):
    sess.run(train)
    if step % 20 == 0:
        print(step, sess.run(W), sess.run(b))

sess.close()
```

Choose Input Data



Create graph



Tensorflow Session

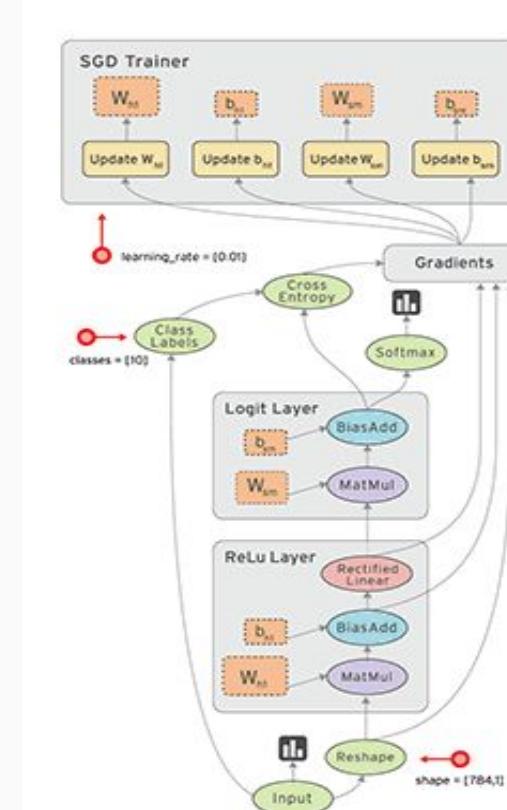
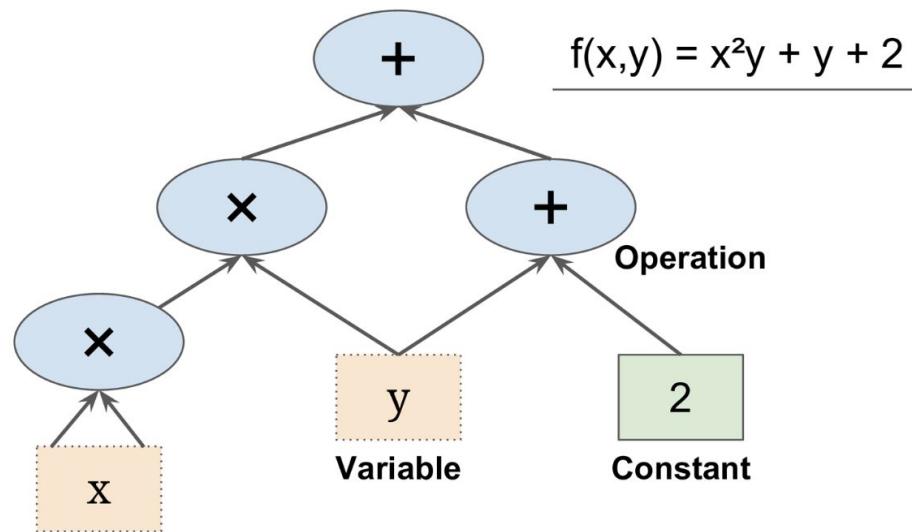


Run

TensorFlow Fundamental

- TensorFlow 是由 **Tensor** 和 **Flow** 組成

Computational Graph



TensorFlow Fundamental

- Build Graph

```
import tensorflow as tf

# 1x2 matrix
matrix1 = tf.constant([[3, 3]])

# 2x1 matrix
matrix2 = tf.constant([[2],
                      [2]])

# matmul() is multiplication of matrixs
product = tf.matmul(matrix1, matrix2)
```

product

Must run session to see the result.

Output:

<tf.Tensor 'MatMul_5:0' shape=(1, 1) dtype=int32>

TensorFlow Fundamental

- Run session

```
import tensorflow as tf

# 1x2 matrix
matrix1 = tf.constant([[3, 3]])

# 2x1 matrix
matrix2 = tf.constant([[2],
                      [2]])

# matmul() is multiplication of matrixs
product = tf.matmul(matrix1, matrix2)

# open session

with tf.Session() as sess:
    result = sess.run(product)
    print(result)
```

另一種寫法：

```
# open session
sess = tf.Session()
result = sess.run(product)
print(result)

# close session
sess.close()
```

Output: [[12]]

TensorFlow Fundamental

- **Input Data**

```
import tensorflow as tf

# Create placeholder for input data

input1 = tf.placeholder(tf.float32)
input2 = tf.placeholder(tf.float32)
output = tf.add(input1, input2)

# Use 8.0 for input1 and 3.0 for input2

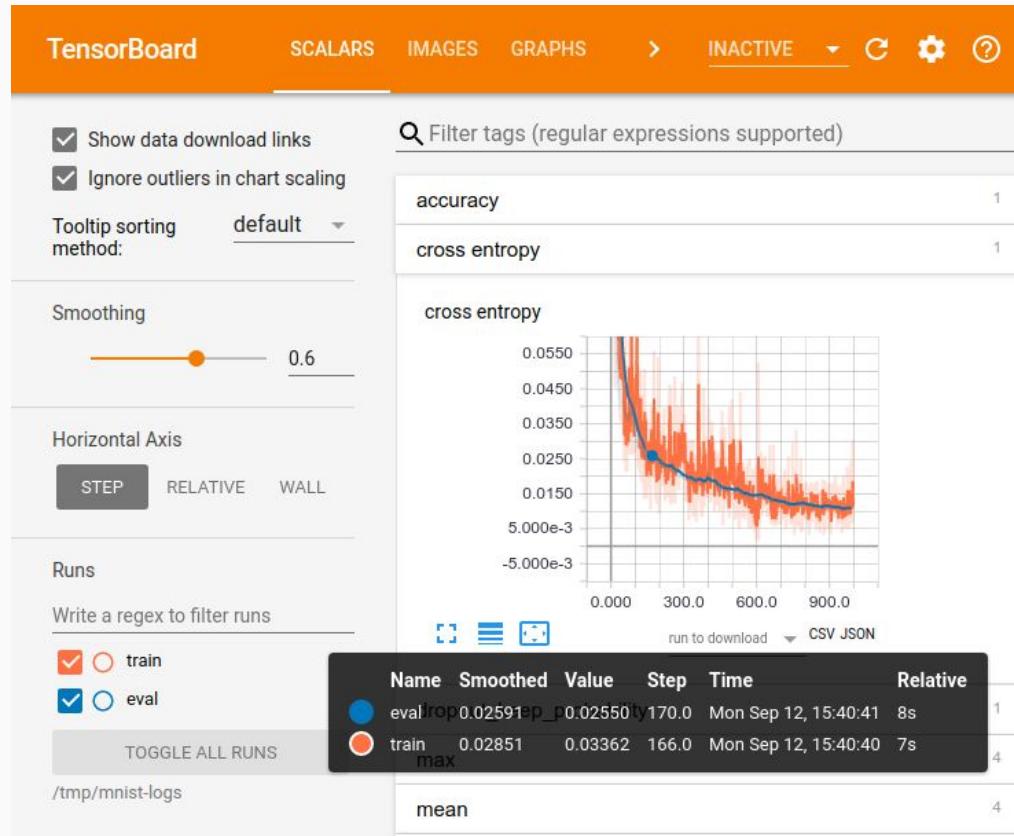
with tf.Session() as sess:
    result = sess.run([output], feed_dict = {input1: [8.0] , input2: [3.0]})
    print(result)
```

Output: [array([11.], dtype=float32)]

Tensorboard

Visualize Learning

- Scalars
- Images
- Graphs
- Audio
- Histograms

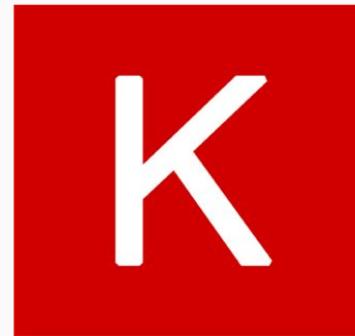


Lab 1 : TensorFlow Basics

Topic : TensorFlow Linear Regression Example

Start coding !

Filename	lab1.ipynb
x	10000 of random generated value between 0 and 1
Target function	$y = wx + b$
Loss function	mean square error
Learning rate	0.5
optimizer	<code>tf.train.GradientDescentOptimizer</code>



Keras

colab



TensorFlow

Keras

Pre-made Estimators

models in a box

Estimator

Keras

train and evaluate models

Layers

Dataset

build models

Python Frontend

C++

Java

JavaScript

...

TensorFlow Distributed Execution Engine

CPU

GPU

TPU

iOS

Android

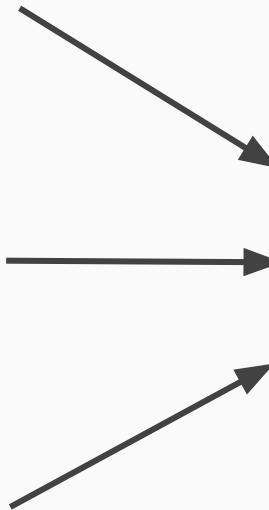
...

Keras

Backend



⋮
⋮

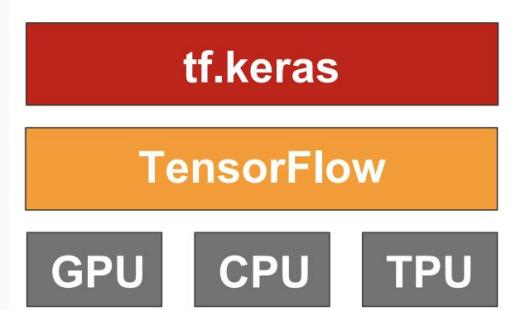


Keras

High-level API

Keras

- Keras is an API designed for human beings, not machines.
- Part of core Tensorflow (tf.keras)
- Strong multi-GPU support and distributed training support



	Keras	TensorFlow
Difficulty	Easy	More Difficult
Flexibility	Medium	Flexible
Rapid Prototyping	Fast	Medium
Suitable User	Beginner	Advanced User

Implementing a NN in Keras

- **Steps to build a model**

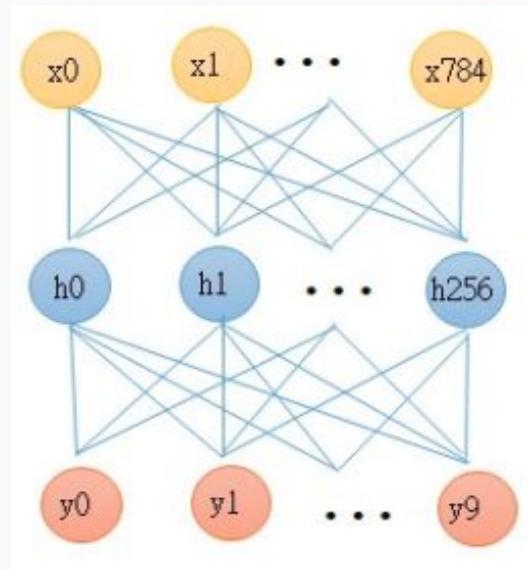
- Prepare the input
- Define the model architecture
- Specify the optimizer and learning rate
- Specify the loss function
- Train and test the model on the dataset



Keras Fundamental

- Building neural network is like making a cake

Input layer (x) **784**



Hidden layer(h1) **256**

Output layer(y) **10**



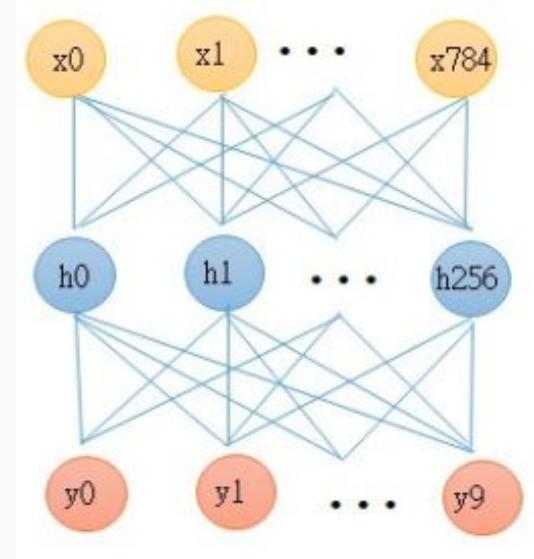
Keras Fundamental

- Import **Sequential** class from **keras.model**

```
from tensorflow.keras import Sequential  
  
model = Sequential()
```

- Stacking layers using **.add()** method

```
from tensorflow.keras.layers import Dense  
  
model.add(Dense(256,  
                input_dim = 784,  
                activation = 'sigmoid'))  
  
model.add(Dense(10,  
                activation = 'sigmoid'))
```



Keras Fundamental

- Configure learning process using `.compile()` method

```
model.compile(loss='mean_squared_error',  
              optimizer='sgd')
```

- Train the model on train dataset using `.fit()` method

```
model.fit(x_train, y_train, epochs=5, batch_size=32)
```

- Evaluate the model on test dataset using `.evaluate()` method

```
loss_and_metrics = model.evaluate(x_test, y_test, batch_size=128)
```

- Predict the data on test dataset using `.predict()` method

```
model.predict(x_test)
```

Discussion (~5 mins)

Term	Meaning
Label	
Input	
Example	
Model	
Training	
Prediction	

Discussion (~5 mins)

Term	Meaning
Label	True Answer
Input	Predictor variables, what you can use to predict the label
Example	Input and corresponding label
Model	Math function that takes variables and predict label
Training	Adjust model to minimize error
Prediction	Using model on unlabeled data

Lab 2-a

Topic : Fashion MNIST classification

Filename	lab2.ipynb
Data	Fashion MNIST
Target	<ul style="list-style-type: none">→ Import dataset→ Data exploration→ Data preprocessing→ Train a classification model

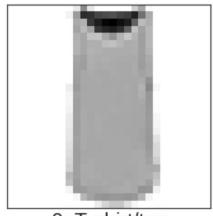
Lab 2a - Data Exploration



9: Ankle boot



0: T-shirt/top



0: T-shirt/top



3: Dress

Train image shape: (60000, 28, 28)

Test image shape: (10000, 28, 28)

Train class: [0 1 2 3 4 5 6 7 8 9]

Test class: [0 1 2 3 4 5 6 7 8 9]

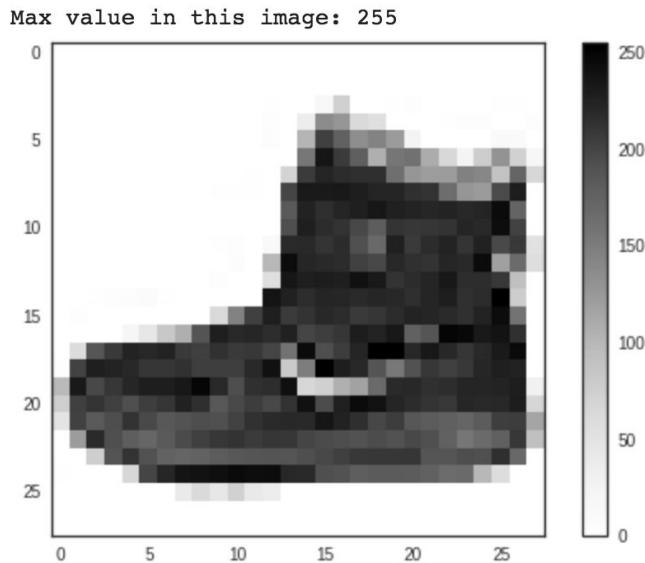
- There are 60000 images for training, and 10000 images for testing.
- The image size is 28 pixels x 28 pixels
- 10 classes in total

Label	Class
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

Lab 2a - Data Preprocessing

Normalization

- Be able to learn the real structures instead of dealing with the scale differences
- Treat all features in the same scale → treat all images in the same way



$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Normalize Data

```
train_images = train_images / 255.0  
test_images = test_images / 255.0
```

Before Normalization:

Image pixel range: **0~255**

After Normalization:

Image pixel range: **0~1**

Lab 2a - Data Preprocessing

One hot encoding

- Label = 9 doesn't mean it is larger and Label = 0 doesn't mean it is smaller
- Don't mathematically imply any ordinal relationship between the classes.

Label	Class	One-hot Encoding
0	T-shirt/top	[1,0,0,0,0,0,0,0,0,0]
1	Trouser	[0,1,0,0,0,0,0,0,0,0]
2	Pullover	[0,0,1,0,0,0,0,0,0,0]
3	Dress	[0,0,0,1,0,0,0,0,0,0]
4	Coat	[0,0,0,0,1,0,0,0,0,0]
5	Sandal	[0,0,0,0,0,1,0,0,0,0]
6	Shirt	[0,0,0,0,0,0,1,0,0,0]
7	Sneaker	[0,0,0,0,0,0,0,1,0,0]
8	Bag	[0,0,0,0,0,0,0,0,1,0]
9	Ankle boot	[0,0,0,0,0,0,0,0,0,1]

```
train_labels = keras.utils.to_categorical(train_labels, num_classes=10)

array([[0., 0., 0., ..., 0., 0., 1.],
       [1., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]], dtype=float32)
```

Lab 2a - training

```
# Clean session first
K.clear_session() Clean graph

# Start building the model
model_sig_sgd_001 = Sequential()
model_sig_sgd_001.add(Flatten(input_shape=(28, 28))) Flatten input image from (28, 28) to (784)
model_sig_sgd_001.add(Dense(128, activation = 'sigmoid'))

##### START CODING HERE #####
# add one dense layer with 64 neurons and with 'sigmoid' activation function (~ 1 line)
model_sig_sgd_001.add(Dense(64, activation = 'sigmoid'))

##### END CODING HERE #####
model_sig_sgd_001.add(Dense(10, activation='softmax')) Create an optimizer
opt = tf.train.GradientDescentOptimizer(learning_rate = 0.001)
model_sig_sgd_001.compile(loss='categorical_crossentropy',
                         optimizer = opt,
                         metrics = [ 'accuracy'])

# Use .summary() to see model details
model_sig_sgd_001.summary()
```

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}$$



Lab 2a - training



```
train_sig_sgd_001 = model_sig_sgd_001.fit(train_images,  
                                         train_labels,  
                                         epochs = 20,  
                                         batch_size = 128,  
                                         validation_split = 0.2,  
                                         shuffle = False)
```

Lab 2-a

Topic : Fashion MNIST classification

Start coding !

Filename	lab2.ipynb
Data	Fashion MNIST
Target	<ul style="list-style-type: none">→ Import dataset→ Data exploration→ Data preprocessing→ Train a classification model
Duration	~ 15 mins

Tips for Deep Learning

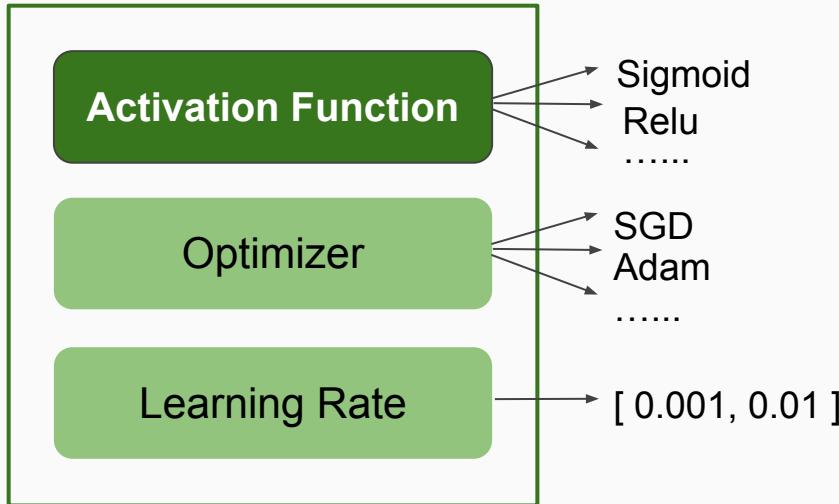
Tips for Deep Learning

Good result on training data?

Yes →

Good result on testing data?

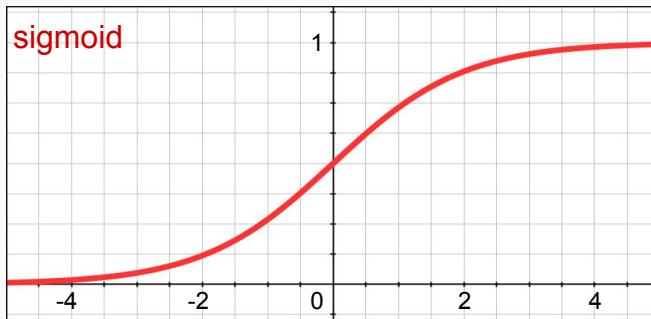
No



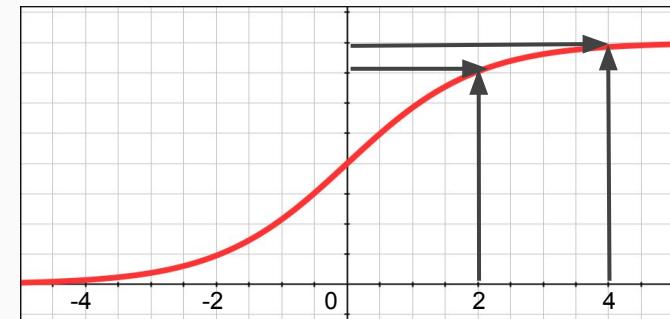
Activation Function

Sigmoid

WX + B

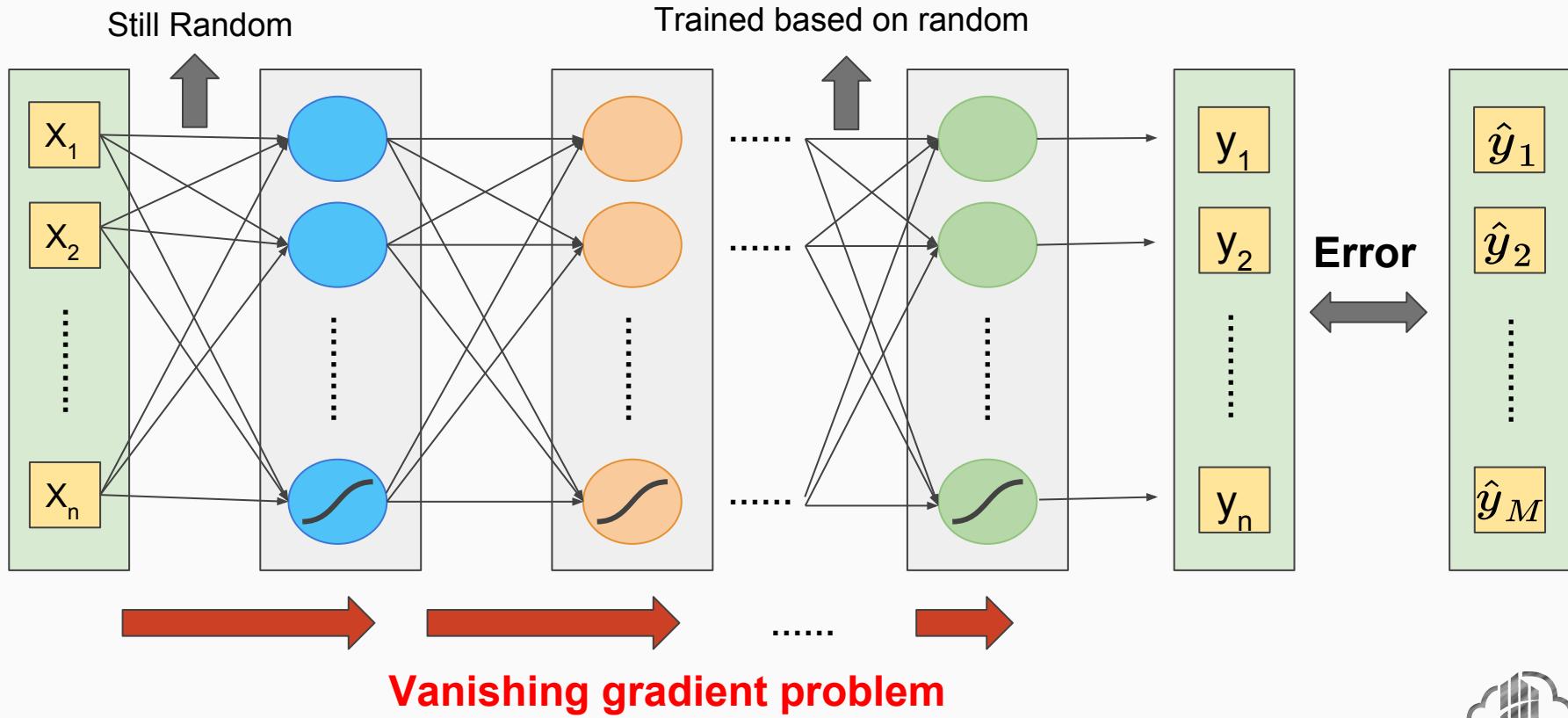


Small output difference

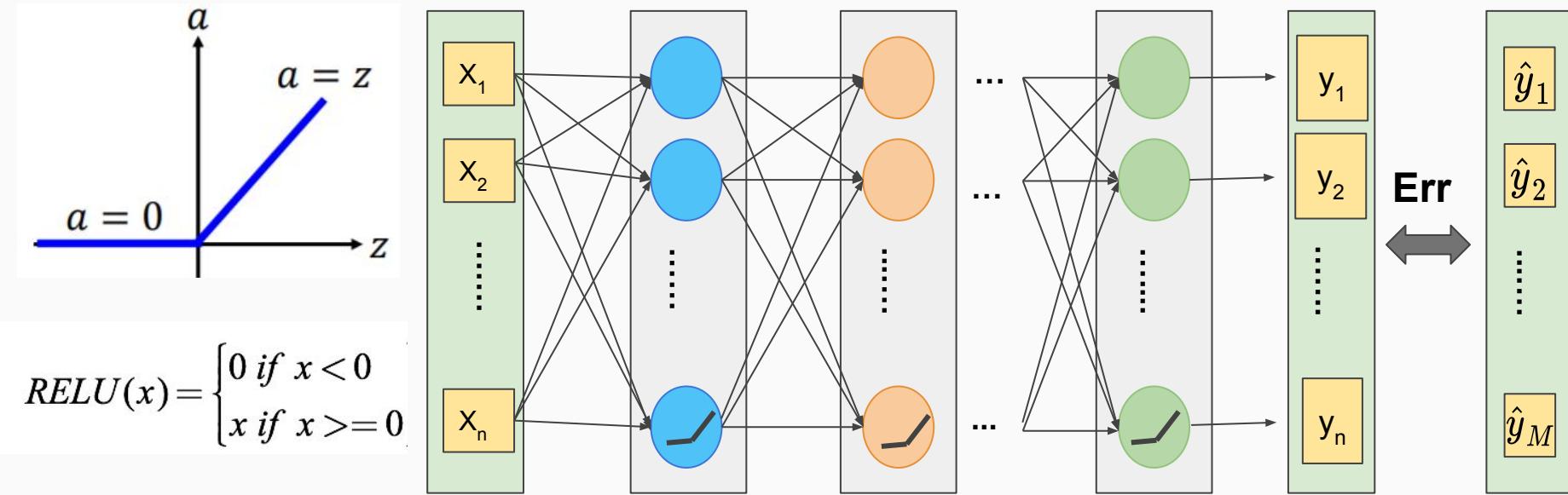


Large input difference

Activation Function



ReLU: Rectified Linear Unit



Lab 2-b

Topic : Fashion MNIST classification with RELU

Start coding !

Filename	lab2.ipynb
Data	Fashion MNIST
Target	<ul style="list-style-type: none">→ Change activation function to RELU→ Train a classification model
Duration	~ 10 mins

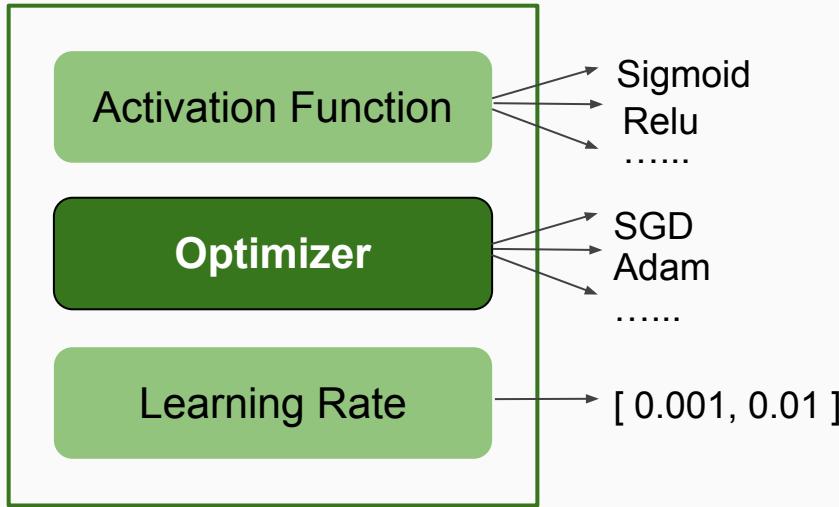
Tips for Deep Learning

Good result on training data?

Yes →

Good result on testing data?

No



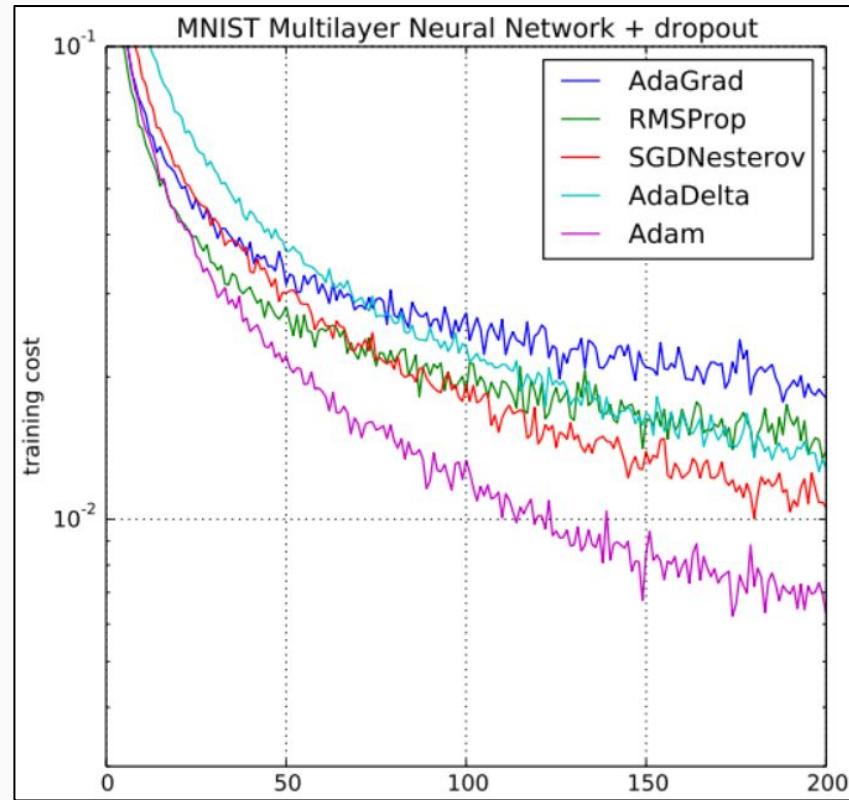
Optimizer

Techniques

- Adaptive learning rate
- Momentum

Optimizer

- Vanilla Gradient Descent
- Adagrad
- RMSprop
- Adam

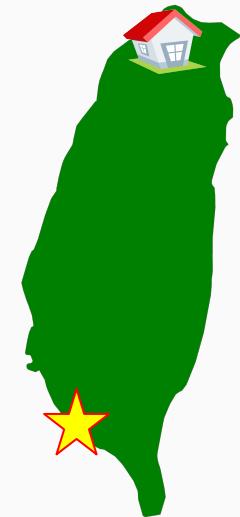


Vanilla Gradient Descent

- Reduce the learning rate by some factor every few epochs.
- At the beginning, we are far from the destination, so we use **larger learning rate**.
- After several epochs, we are close to the destination, so we **reduce the learning rate**.

E.g. 1/t decay: $\eta^t = \eta / \sqrt{t + 1}$

$$w^{t+1} \leftarrow w^t - \boxed{\eta^t} g^t$$



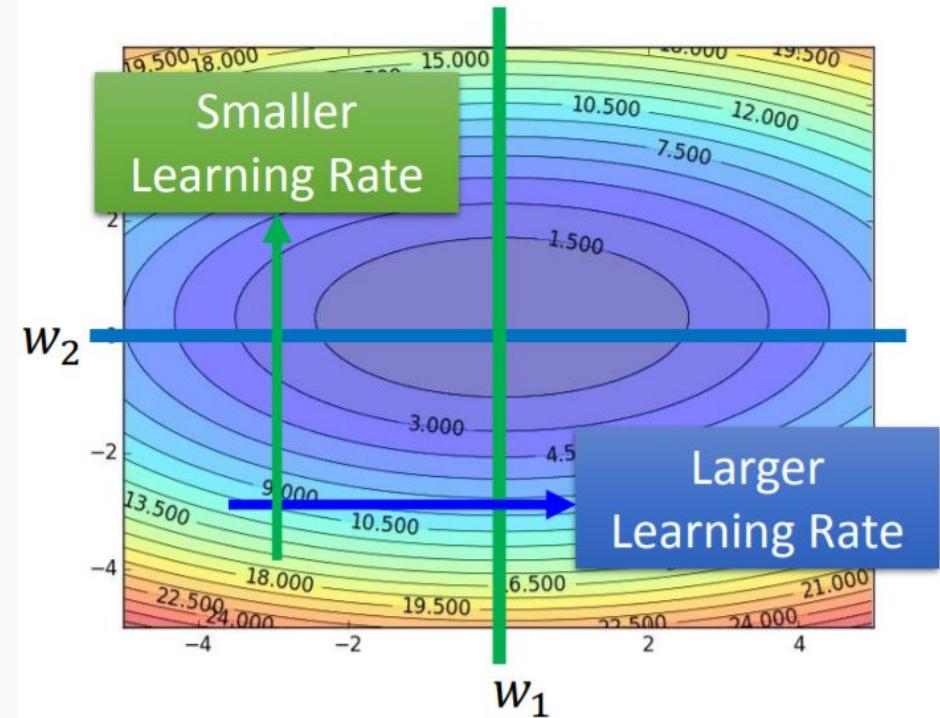
Adagrad

Adagrad

$$\eta^t = \frac{\eta}{\sqrt{t + 1}}$$

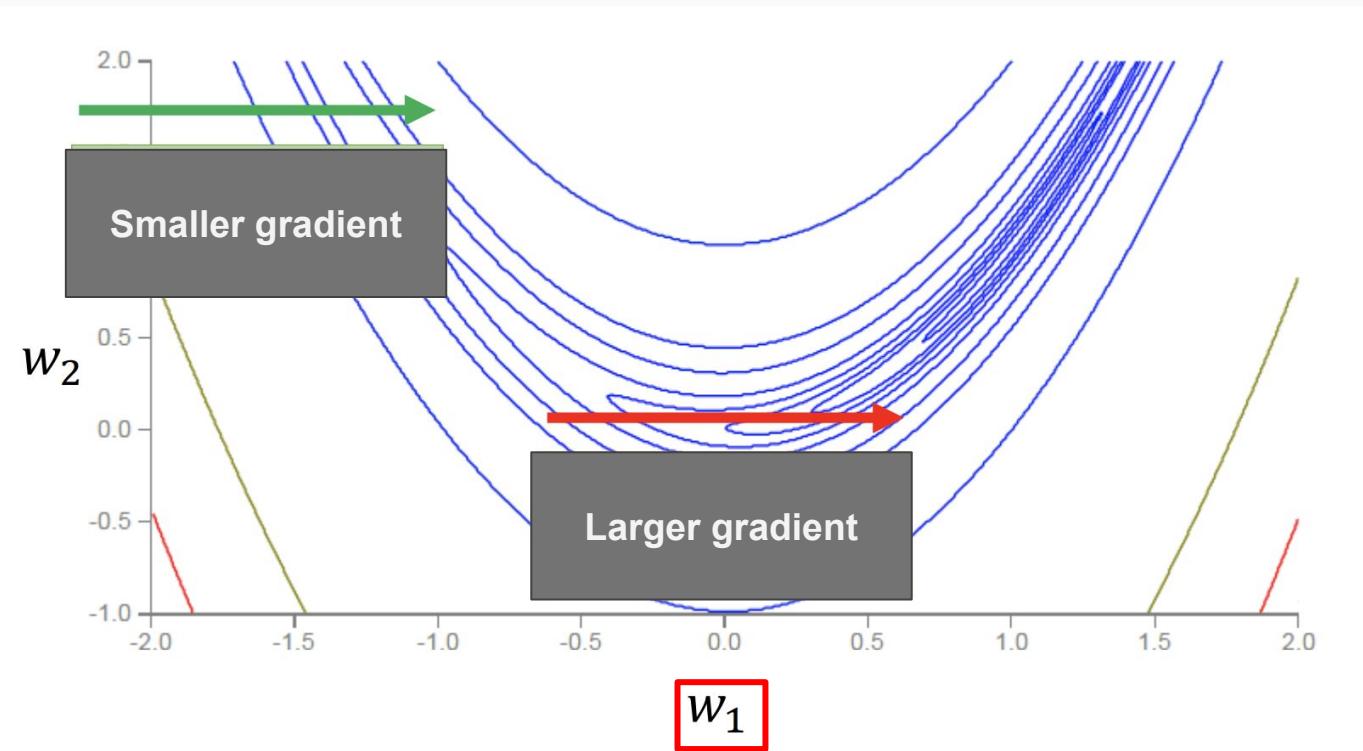
$$\sigma^t = \sqrt{\frac{1}{t + 1} \sum_{i=0}^t (g^i)^2}$$

$$w^{t+1} \leftarrow w^t - \frac{\eta^t}{\sigma^t} g^t$$



Adaptive Learning Rate

RMSProp



Adaptive Learning Rate

RMSProp

$$w^{t+1} = w^t - \frac{\eta}{\sigma^t} g^t$$

$$\sigma^t = \sqrt{\alpha(\sigma^{t-1})^2 + (1 - \alpha)(g^t)^2}$$

$$w^1 \leftarrow w^0 - \frac{\eta}{\sigma^0} g^0 \quad \sigma^0 = g^0$$

$$w^2 \leftarrow w^1 - \frac{\eta}{\sigma^1} g^1 \quad \sigma^1 = \sqrt{\alpha(\sigma^0)^2 + (1 - \alpha)(g^1)^2}$$

$$w^3 \leftarrow w^2 - \frac{\eta}{\sigma^2} g^2 \quad \sigma^2 = \sqrt{\alpha(\sigma^1)^2 + (1 - \alpha)(g^2)^2}$$

⋮

$$w^{t+1} \leftarrow w^t - \frac{\eta}{\sigma^t} g^t \quad \sigma^t = \sqrt{\alpha(\sigma^{t-1})^2 + (1 - \alpha)(g^t)^2}$$

Including g0, g1

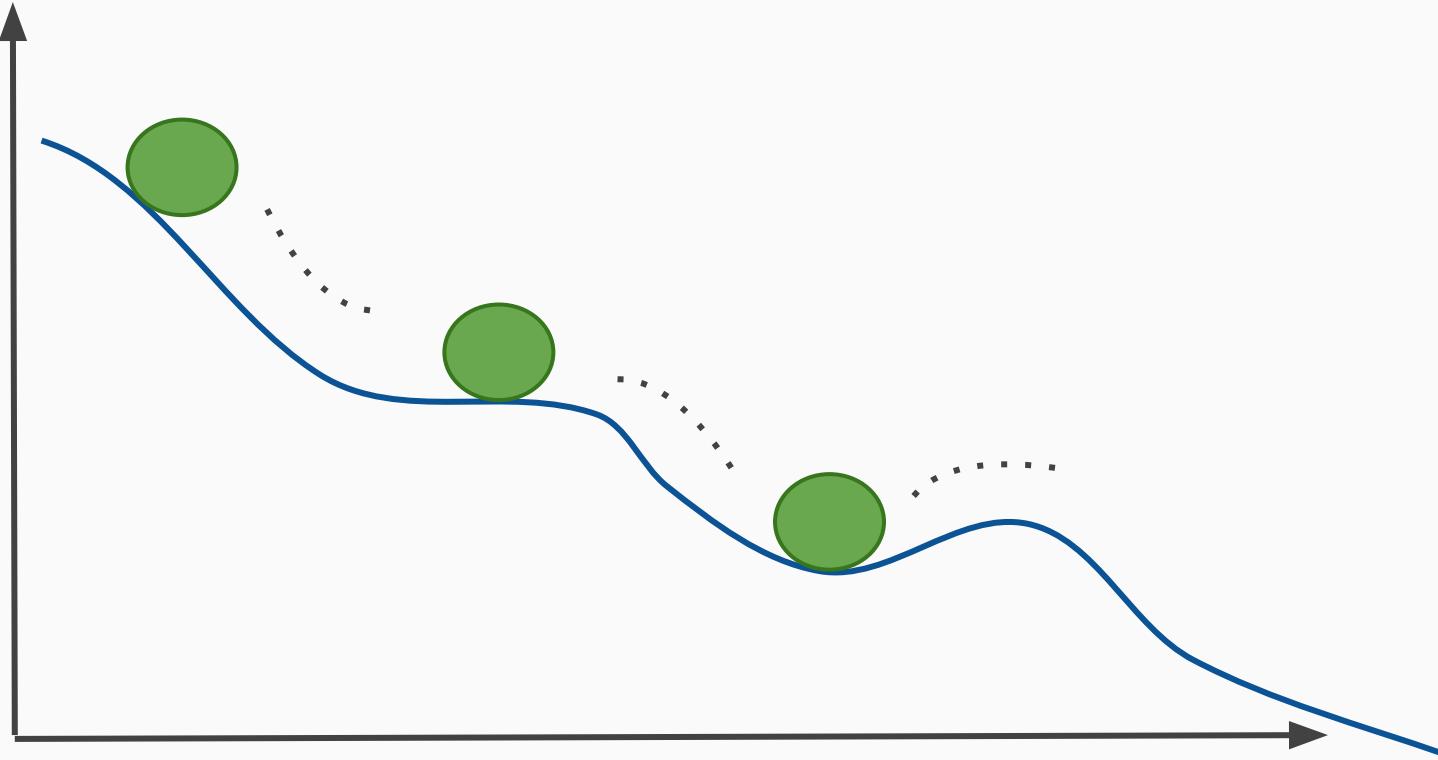
Including g2

Choose your own α

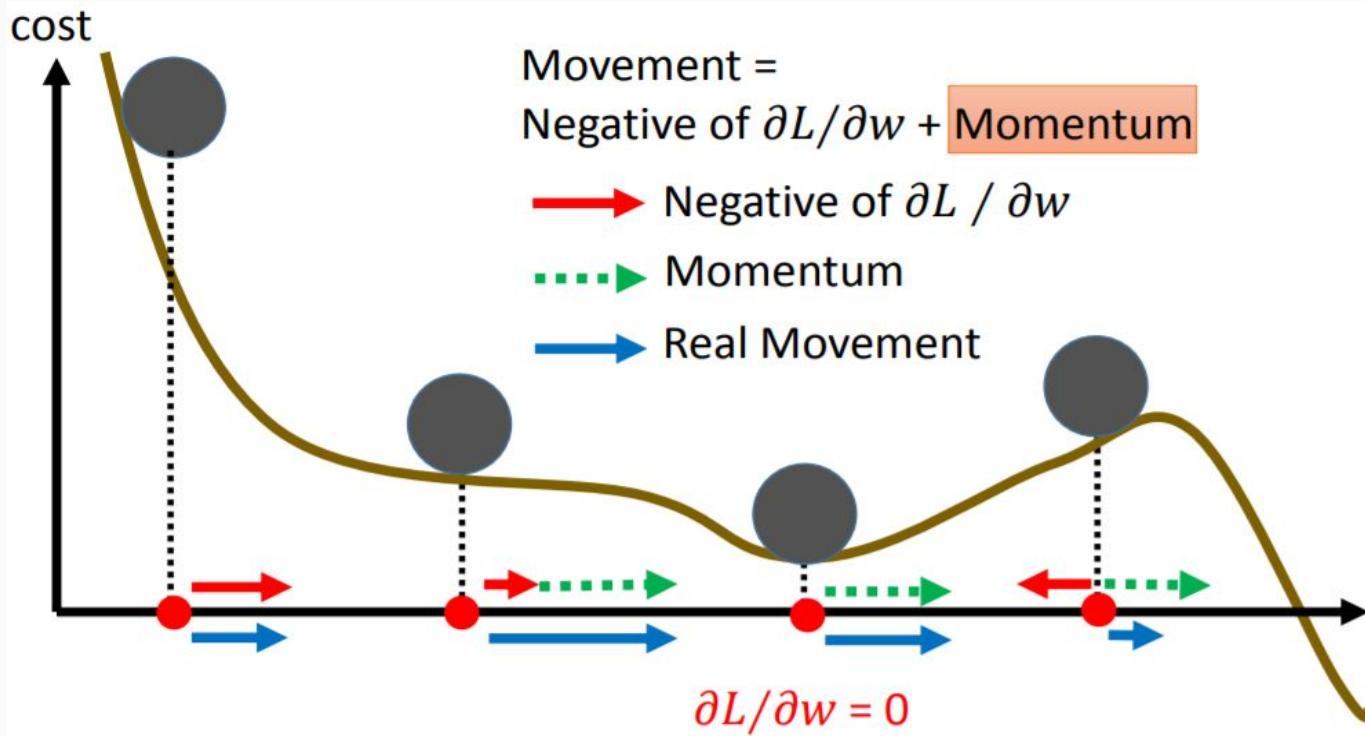
→ **Small α : Depend more on current gradient**

→ **Large α : Depend more on previous gradients**

Momentum



Momentum



Optimizers

RMSProp

$$w^{t+1} = w^t - \frac{\eta}{\sigma^t} g^t \quad \sigma^t = \sqrt{\alpha(\sigma^{t-1})^2 + (1 - \alpha)(g^t)^2}$$

Adam

- Include the advantages from RMSProp and Momentum
- Try with Adam first !

RMSProp + Momentum

Lab 2-c

Topic : Fashion MNIST classification with ReLU + Adam

Start coding !

Filename	lab2.ipynb
Data	Fashion MNIST
Target	<ul style="list-style-type: none">→ Change optimizer to Adam→ Train a classification model
Duration	10 mins

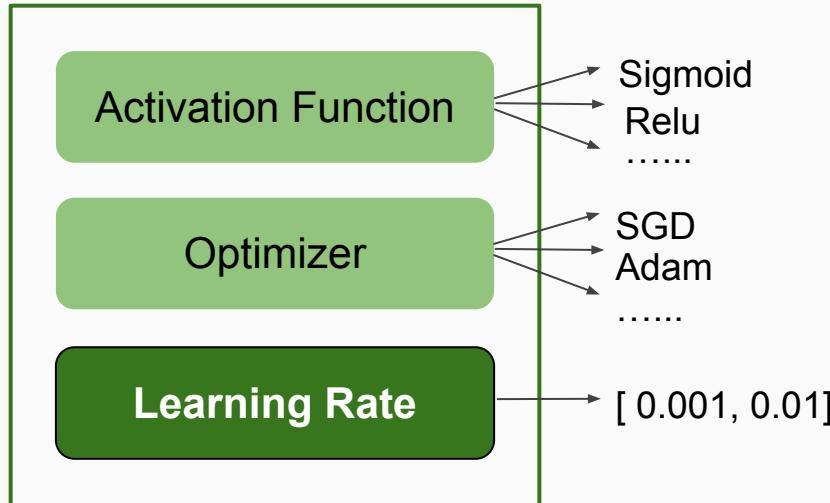
Tips for Deep Learning

Good result on training data?

Yes →

Good result on testing data?

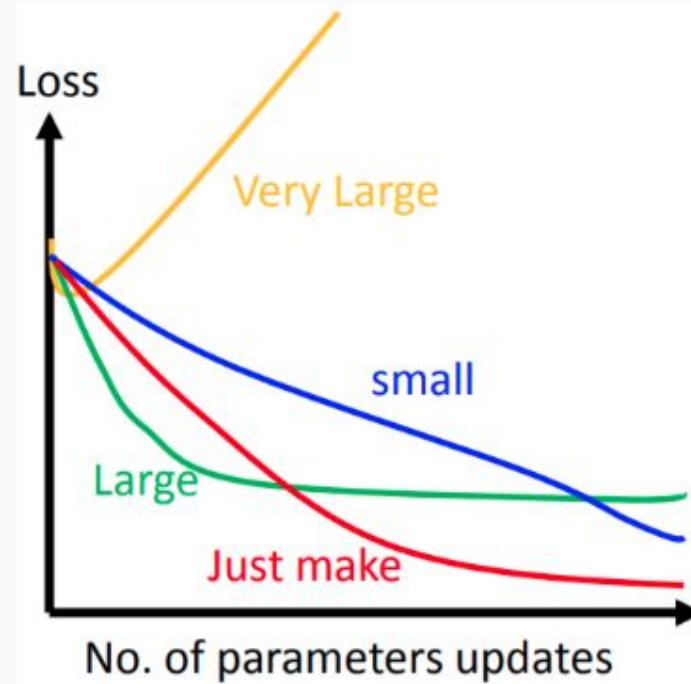
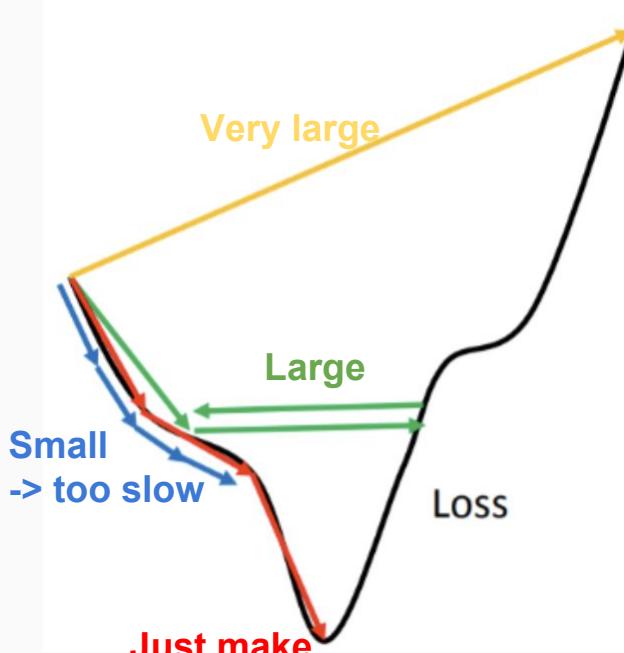
No



Learning rate

- Set learning rate carefully

$$\theta^i = \theta^{i-1} - \eta \nabla L(\theta^{i-1})$$



Topic : Adjust learning rate in Fashion MNIST classification

Filename	lab2.ipynb
Data	Fashion MNIST
Target	<ul style="list-style-type: none">→ Change learning rate from 0.001 to 0.01→ Train a classification model
Duration	~ 10 mins

Start coding !

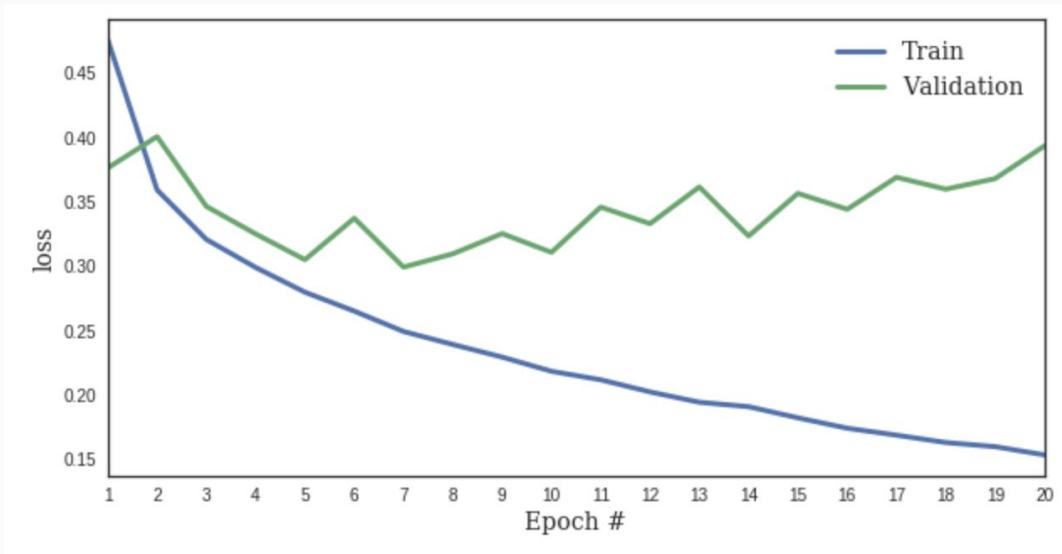
Start coding !

Topic : Change number of hidden layers neurons in Fashion MNIST classification

Filename	lab2.ipynb
Data	Fashion MNIST
Target	<ul style="list-style-type: none">→ Change hidden layers neurons to [2048 , 1024]→ Train a classification model
Duration	~ 10 mins

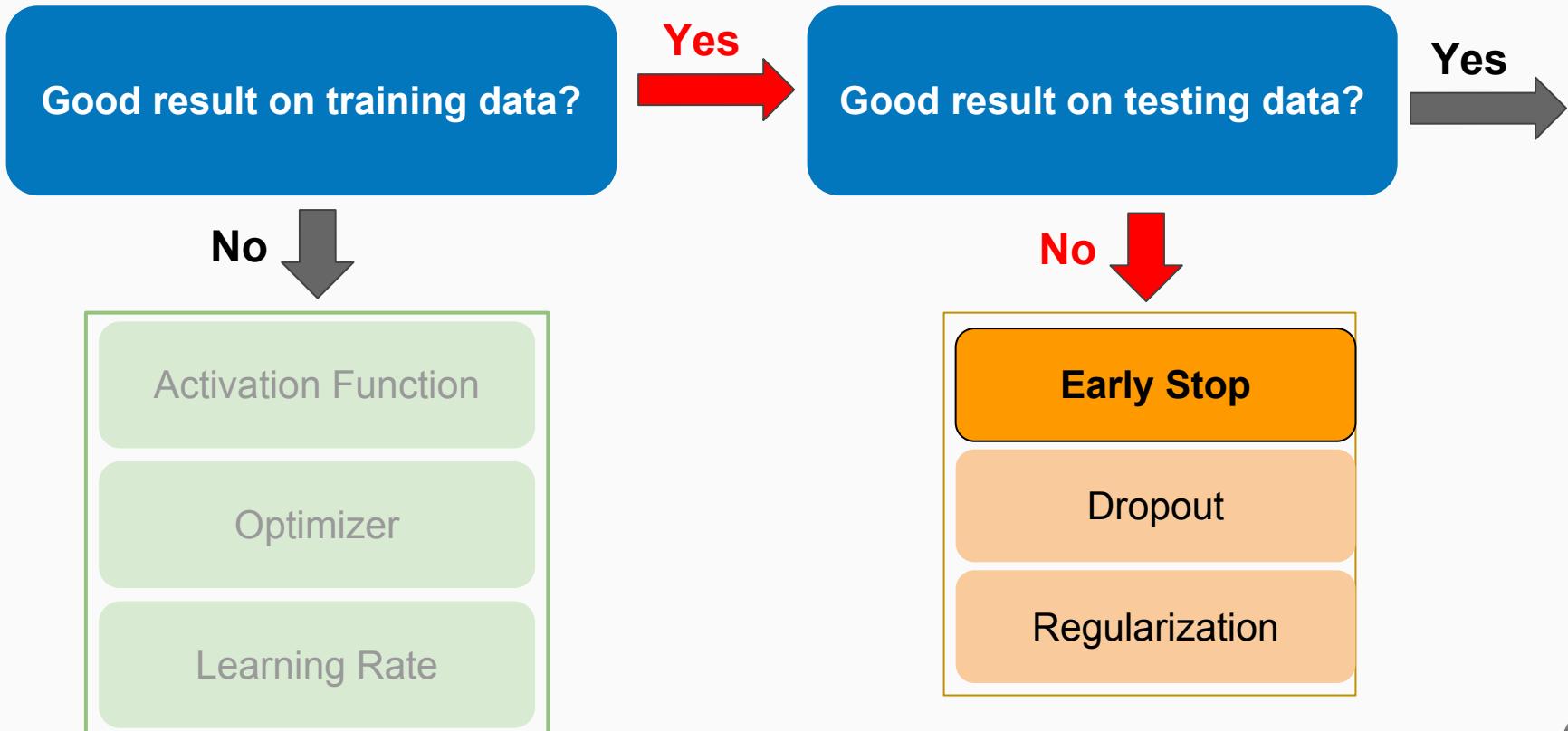
Lab 2-e result

- Training loss is **decreasing**, however, validation loss is **increasing** after a few epochs.
- Train too well → Learn the details and noise in the training data
 - Negatively impacts the model performance on new data

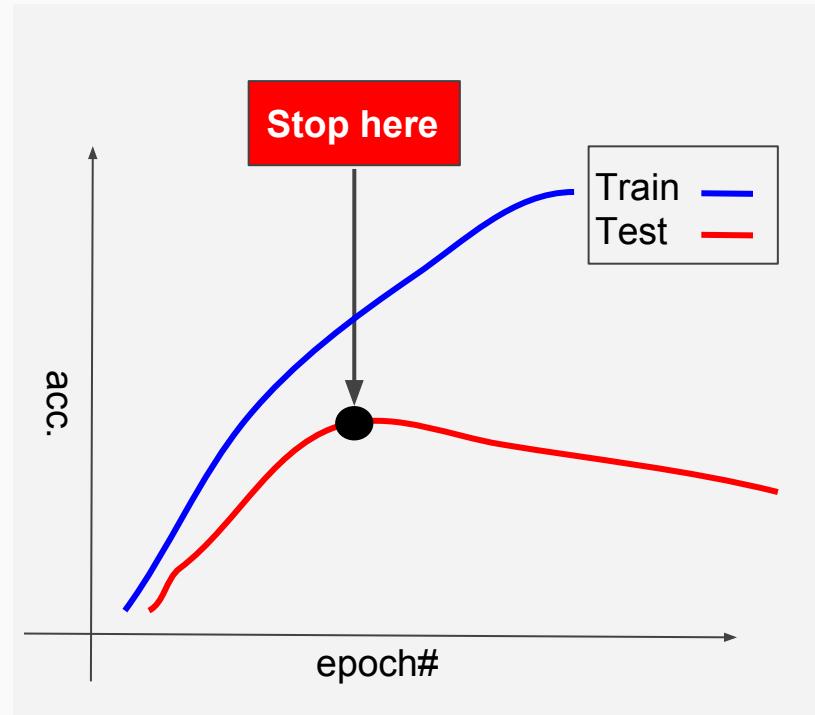
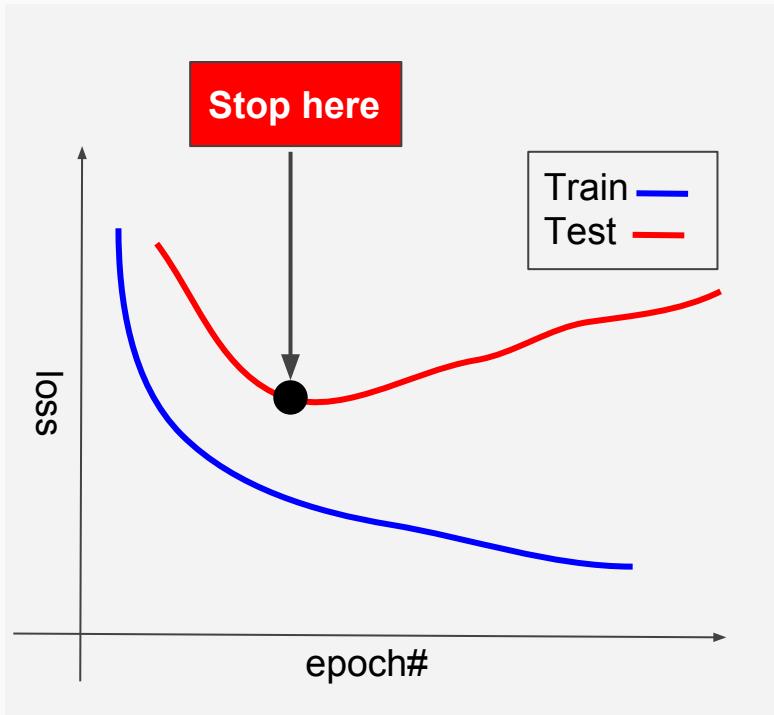


→ **May be overfitting**

Tips for Deep Learning



Early stopping



Keras - early stopping

```
keras.callbacks.EarlyStopping(monitor='val_loss', patience=3)
```

monitor: quantity to be monitored.

patience: number of epochs with no improvement after which training will be stopped.

Example Code

```
EarlyStopping = keras.callbacks.EarlyStopping(  
    monitor='val_loss',  
    patience=5)  
  
callbacks_list = [EarlyStopping]  
  
model_his = model.fit(train_features,  
    train_labels,  
    nb_epoch=100,  
    callbacks=callbacks_list)
```

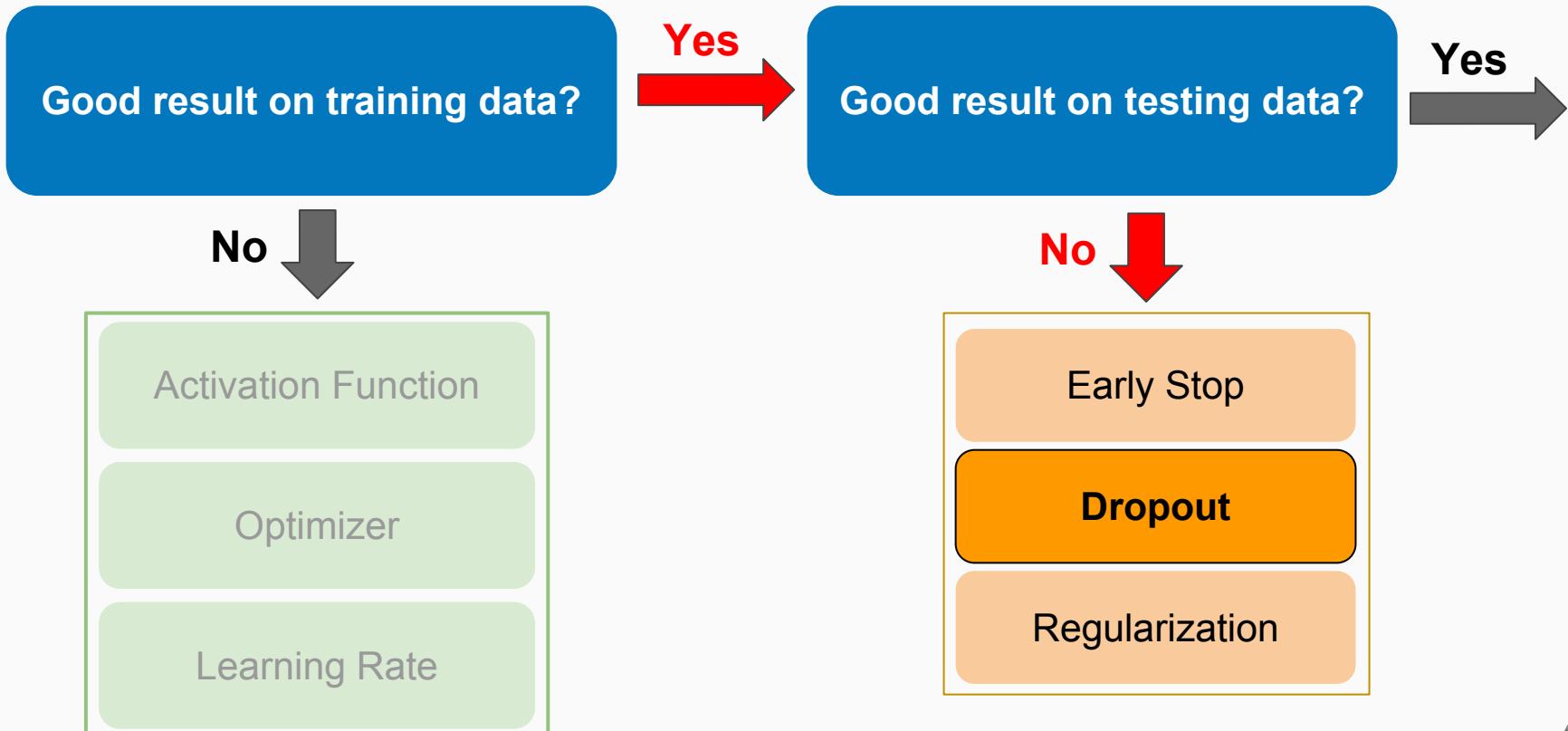
Topic : Early stop to avoid overfitting

Start coding !

Filename	lab2.ipynb
Data	Fashion MNIST
Target	<ul style="list-style-type: none">→ Create early stopping callback to avoid overfitting→ Train a classifier
Duration	~ 10 mins

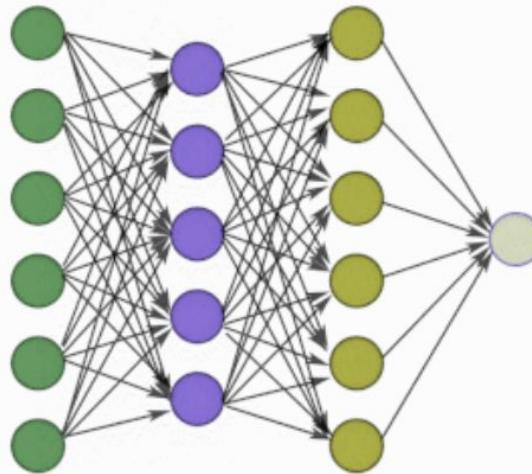
Another solution: **save best model only**

Tips for Deep Learning

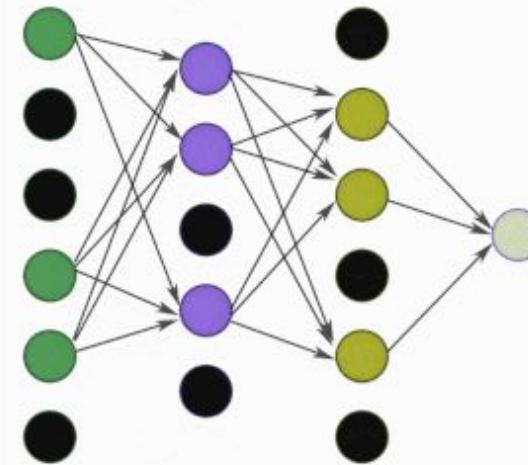


Dropout

Standard Neural Network

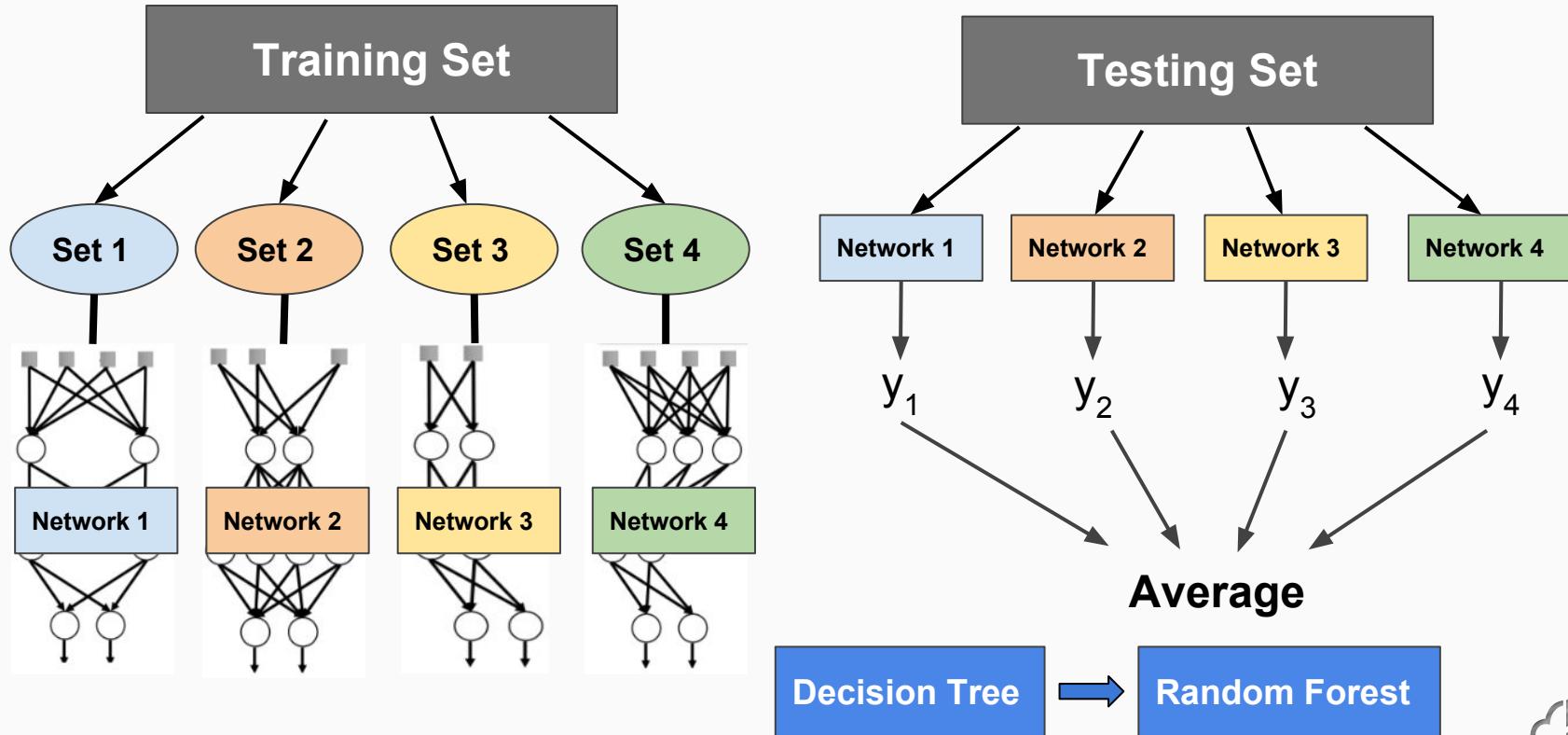


After applying dropout

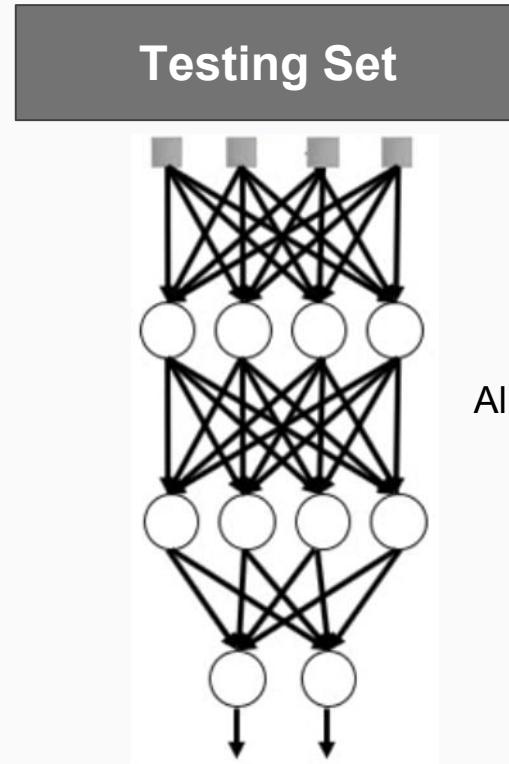
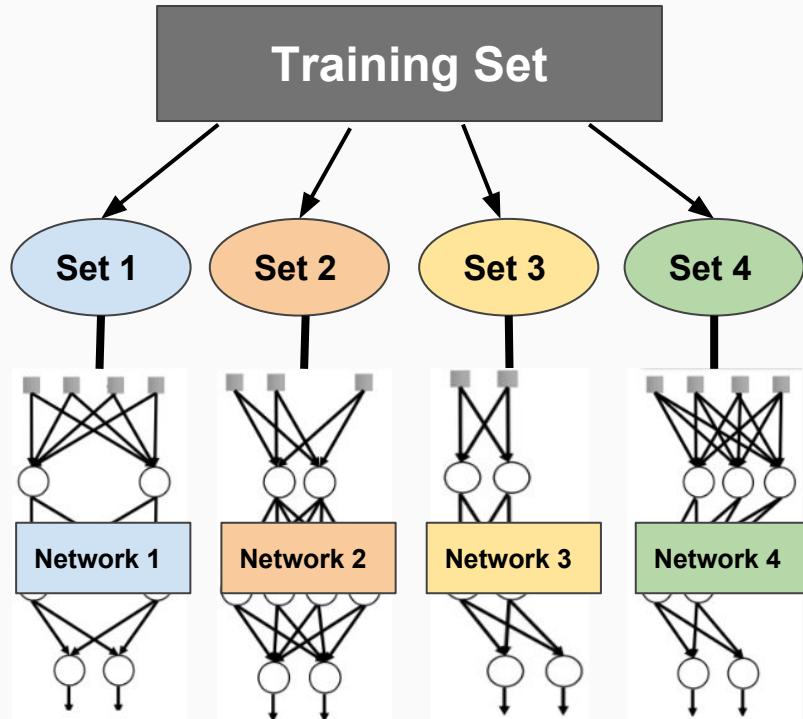


- When training, drop $p\%$ neurons in the batch.
- For each mini batch, we resample the dropout neurons.
- Dropout always work on training time, not on evaluate or prediction.

Dropout



Dropout



All weights multiply $(1-p) \%$

Dropout - Keras

```
keras.layers.Dropout(rate)
```

rate: float between 0 and 1. Fraction of the input units to drop.

Example Code

```
model = Sequential()  
model.add(Dense(60, input_dim=60, activation='relu')  
model.add(Dropout(0.2))  
model.add(Dense(30, activation='relu'))  
model.add(Dropout(0.2))  
model.add(Dense(10, activation='softmax'))
```

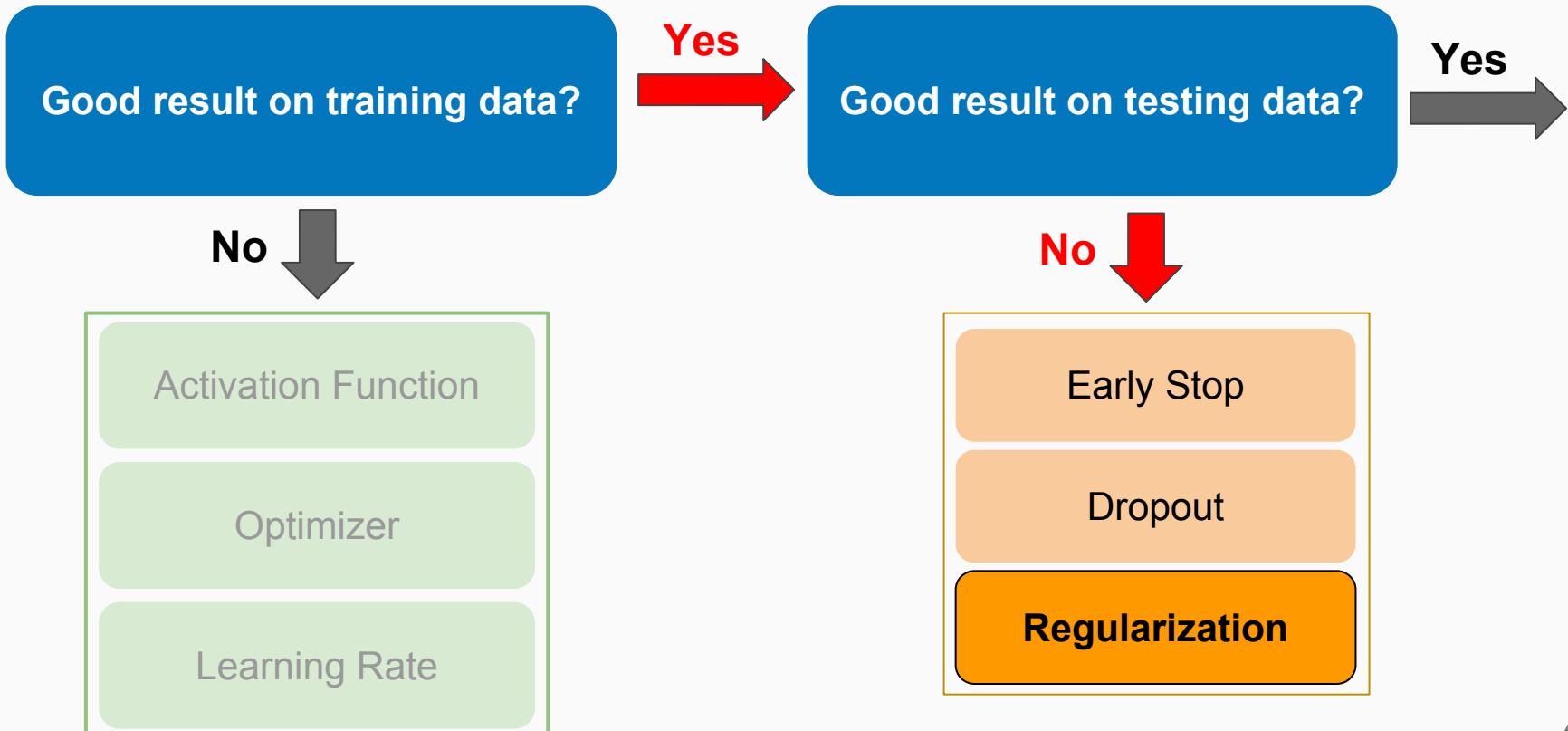
rate

Start coding !

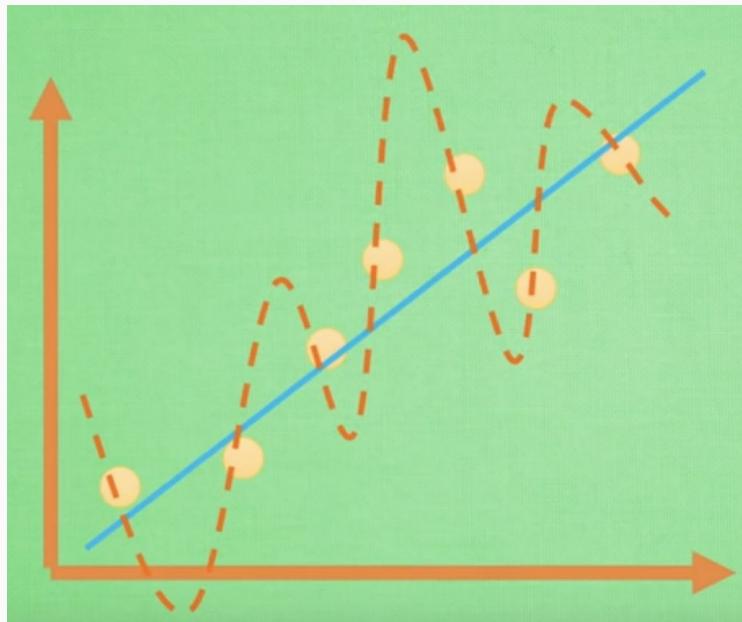
Topic : Use dropout to avoid overfitting

Filename	lab2.ipynb
Data	Fashion MNIST
Target	<ul style="list-style-type: none">→ Create dropout in hidden layers→ Train a classifier
Duration	~ 10 mins

Tips for Deep Learning



Regularization



$$y = a + bx + cx^2 + dx^3$$

$$y = a + bx$$

Regularization

- Add Regularizer into loss

$$L'(\theta) = L(\theta) + \lambda(\text{regularizer})$$

- λ is a parameter (usually below 0.01)
- L1 and L2

$$L_1 = \sum_i |w_i|$$

L1 norm: sum of absolute values

$$L_2 = \sum_i |w_i|^2$$

L2 norm: square sum of absolute values

Discussion

Application

1. Healthcare (Disease Identification , Alert and diagnostics from real-time patient data)
2. Manufacturing(Defection analysis, Demand forecasting, Condition monitoring)
3. Environment (Insects Identification, Earthquake prediction)
4. Financial (Risk Analysis, Customer segmentation)
5. Traveling (Aircraft scheduling, Traffic Patterns and congestion management)

Use Case	Label	Input(s)	Classification or regression

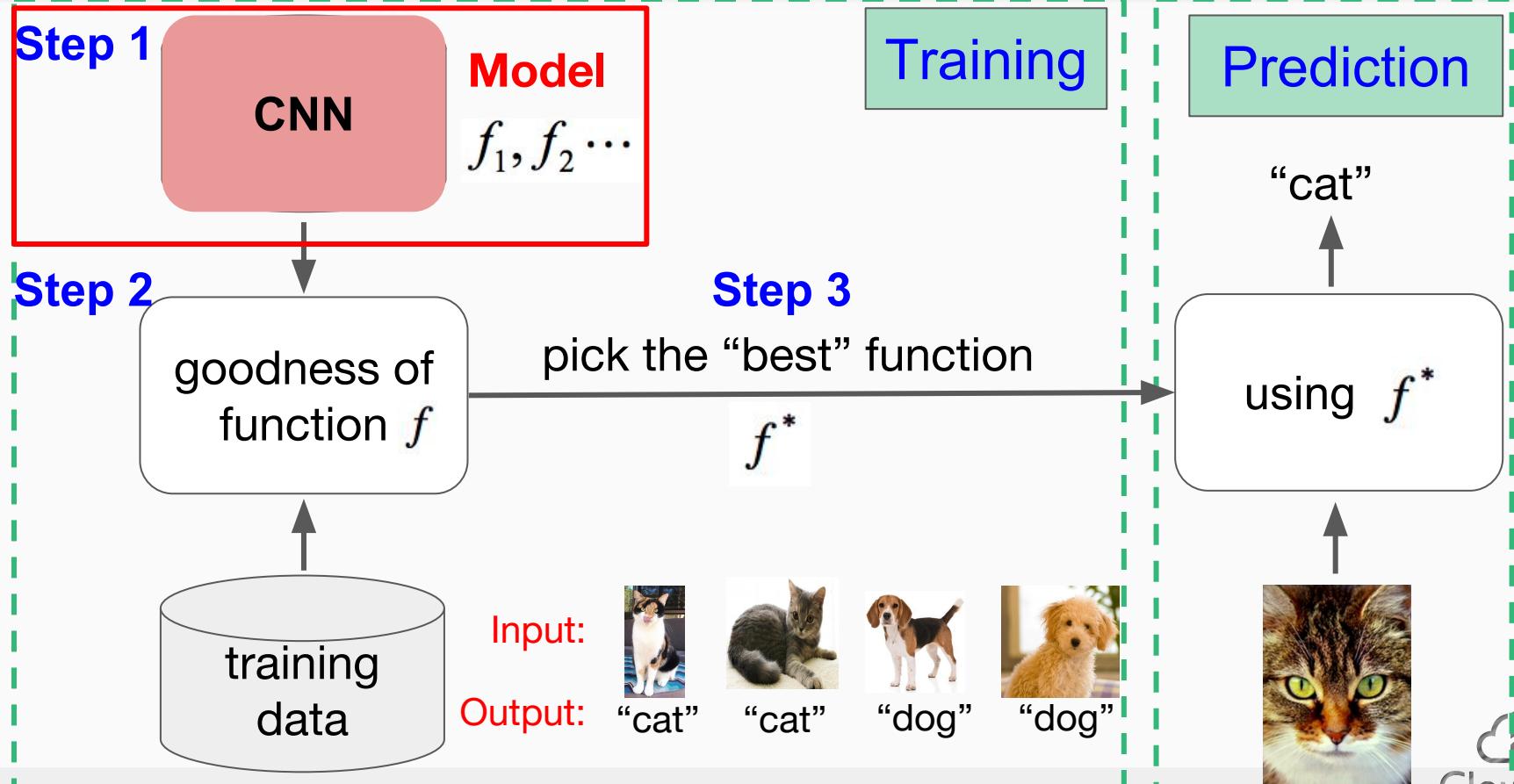
Variants of Neural Networks



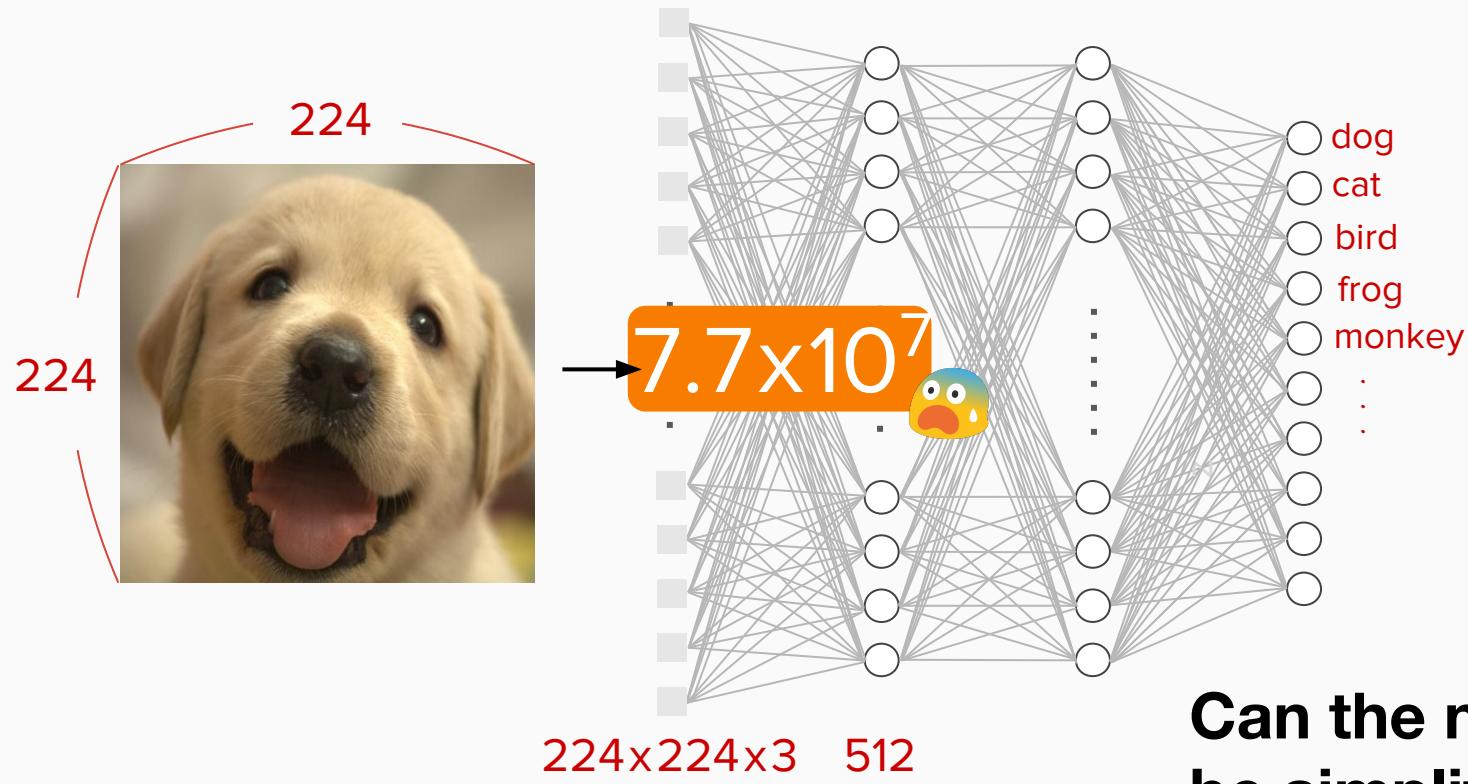
Variants of Neural Networks

- Convolutional neural networks (CNN)
→ good for **images**
- Recurrent neural networks (RNN)
→ good for **sequential** data

Machine Learning Framework



Why CNN for Images?



**Can the network
be simplified?**

Properties of Image Recognition

- **Locality:** objects tend to have a local spatial support.
- **Translation invariance:** object appearance is independent of location.
- **Scale invariance:** subsampling the pixels will not change the object.

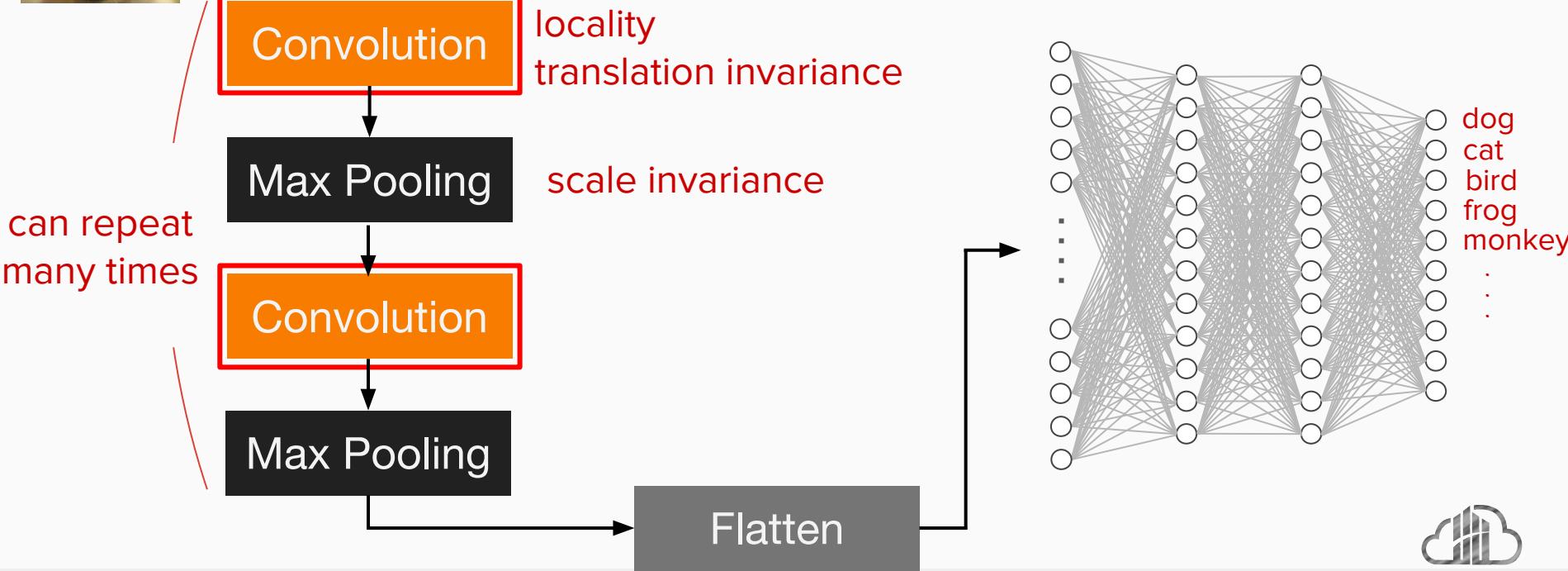


locality
translation invariance



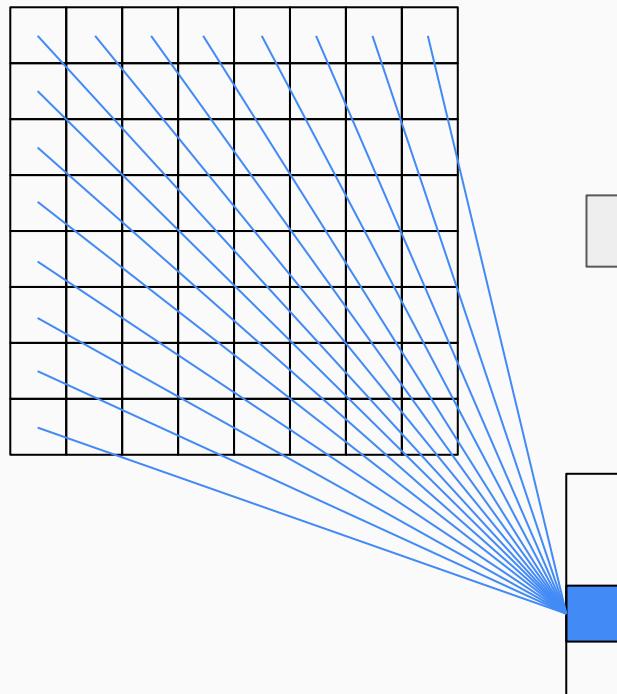
scale invariance

Whole CNN

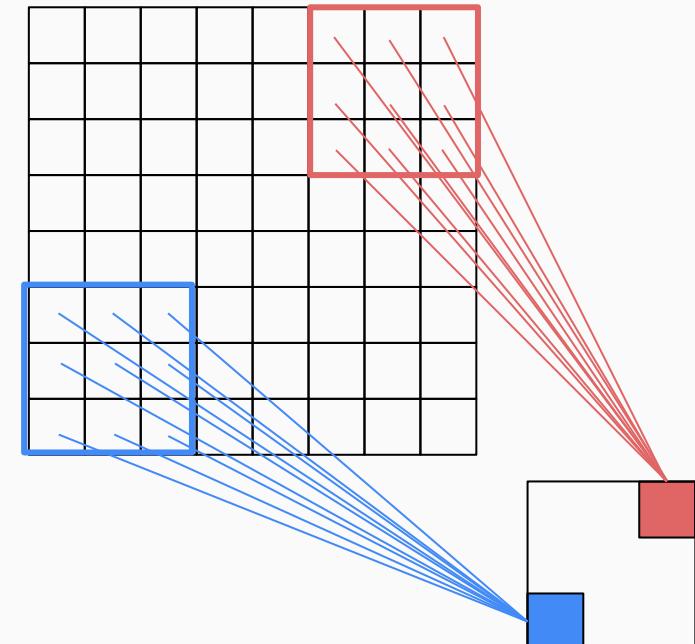


Incorporate Properties: Locality

fully-connected units



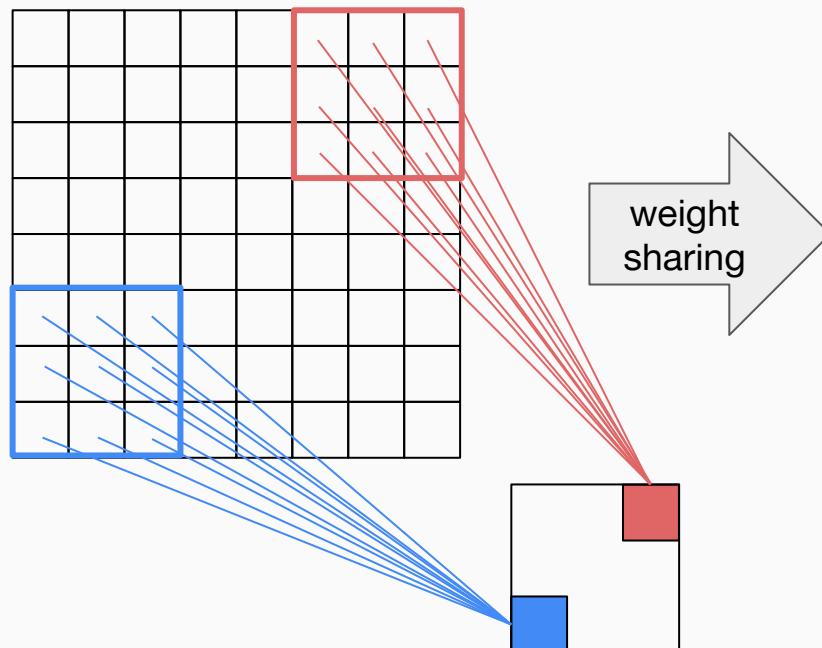
locally-connected units



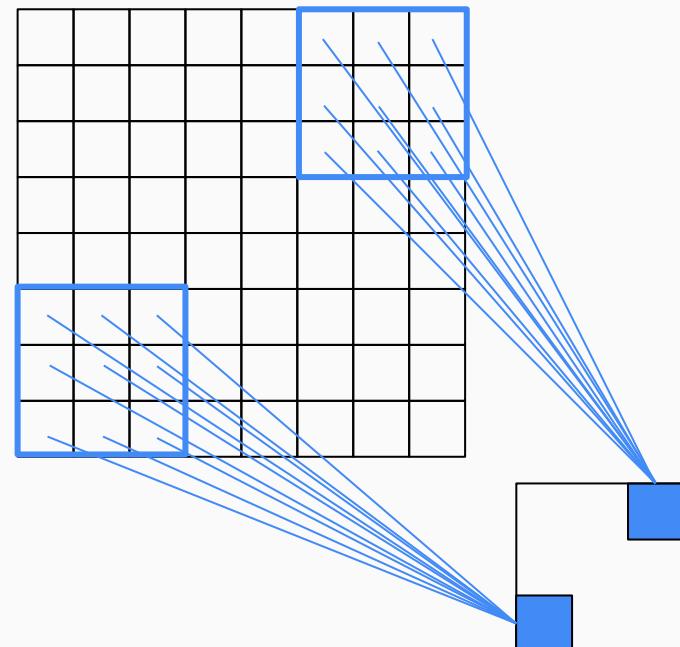
less parameters!

Incorporate Properties: Translation Invariance

locally-connected units

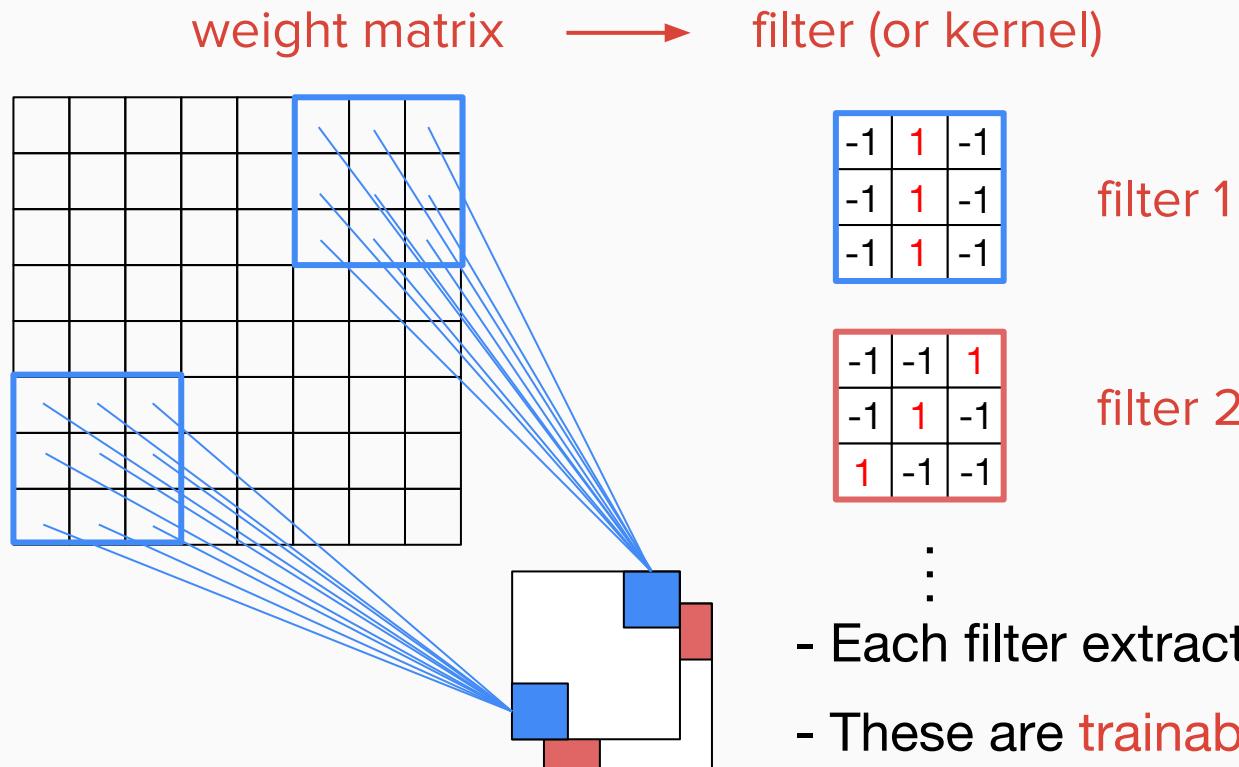


convolutional units

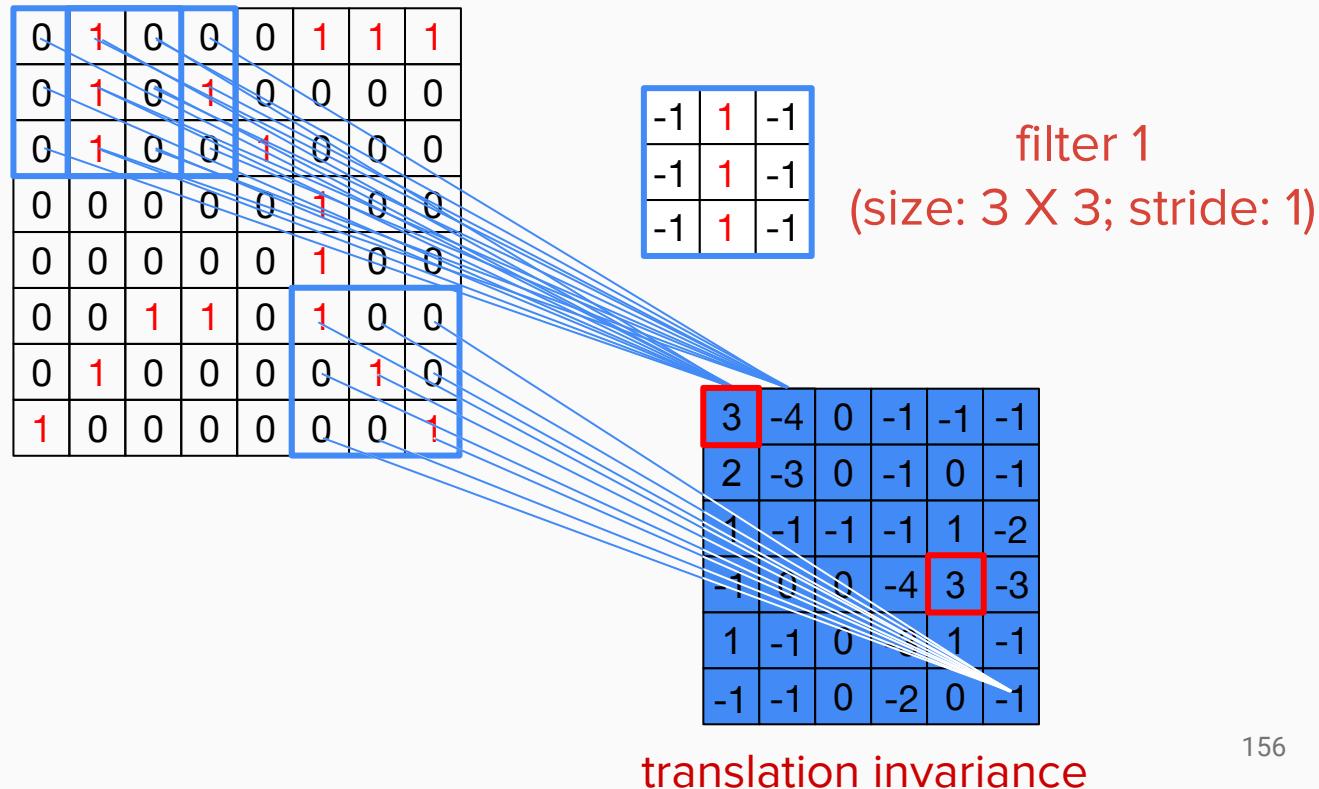


even less parameters!

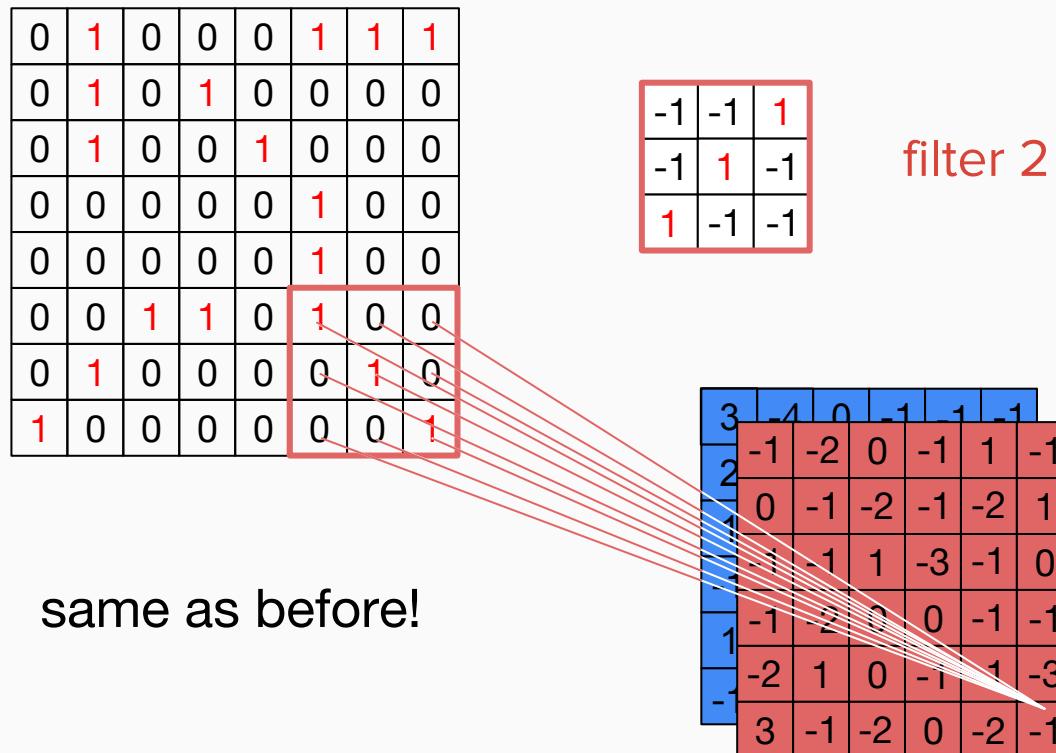
Convolutions



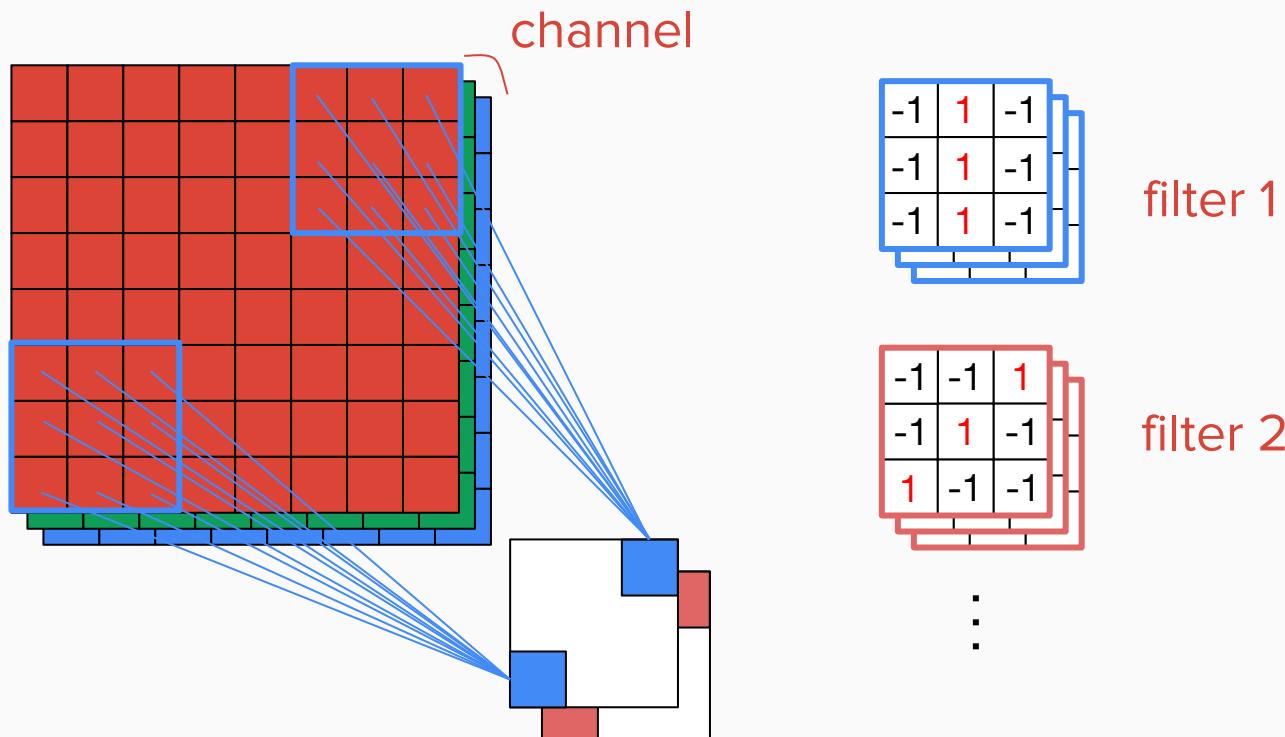
How to Compute the Output Feature?



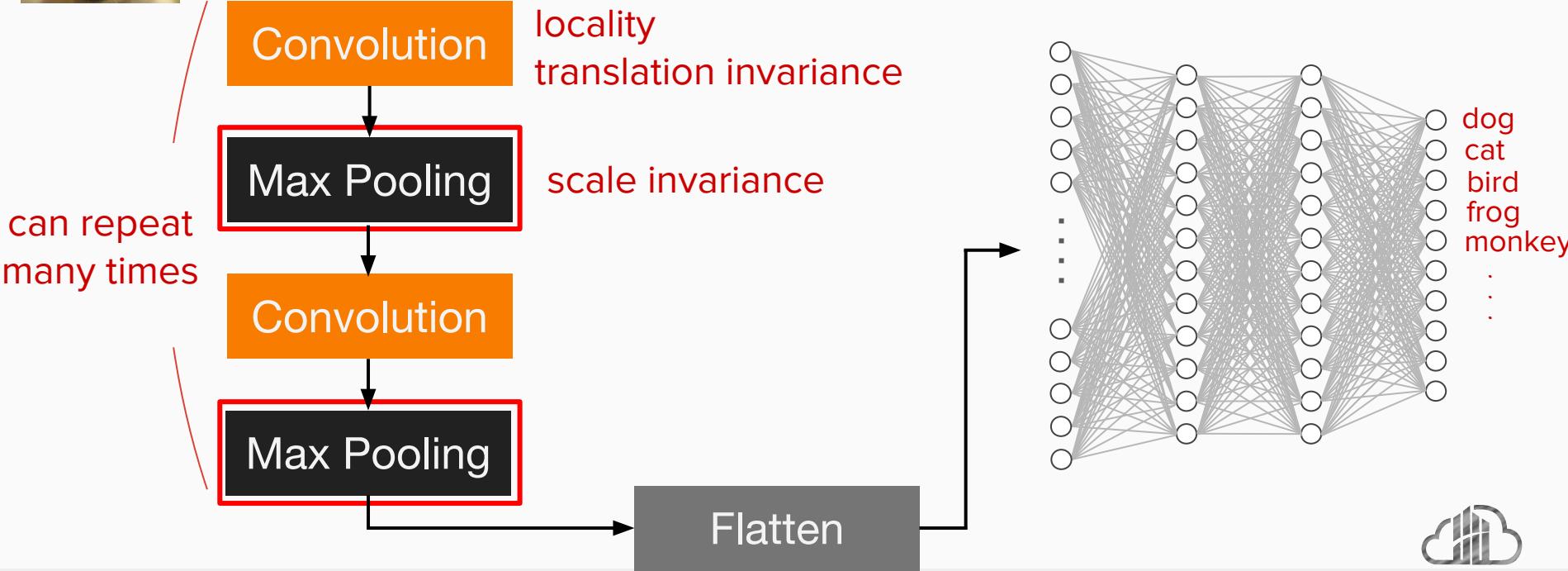
How to Compute the Output Features? (Cont'd)



Convolution for Colorful Images



Whole CNN



Max Pooling

-1	1	-1
-1	1	-1
-1	1	-1

filter 1

-1	-1	1
-1	1	-1
1	-1	-1

filter 2

3	-4	0	-1	-1	-1
2	-3	0	-1	0	-1
1	-1	-1	-1	1	-2
-1	0	0	-4	3	-3
1	-1	0	-3	1	-1
-1	-1	0	-2	0	-1

Max
Pooling

3	0	0
1	0	3
1	0	1

-1	-2	0	-1	1	-1
0	-1	-2	-1	-2	1
-1	-1	1	-3	-1	0
-1	-2	0	0	-1	-1
-2	1	0	-1	-1	-3
3	-1	-2	0	-2	-1

Max
Pooling

0	0	1
-1	1	0
3	0	-1

Max Pooling (Cont'd)

0	1	0	0	0	1	1	1
0	1	0	1	0	0	0	0
0	1	0	0	1	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0
0	0	1	1	0	1	0	0
0	1	0	0	0	0	1	0
1	0	0	0	0	0	0	1

8 x 8 image



Convolution

Max Pooling



3	0	0
1	0	0
1	-1	1

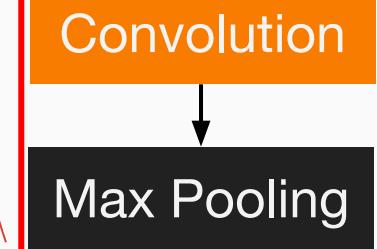
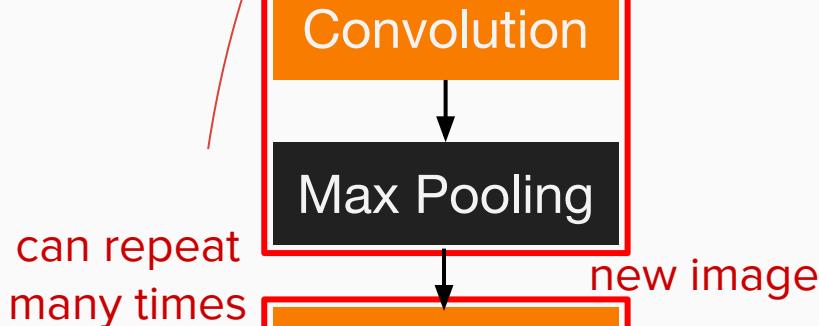
3	0	-1
3	0	-1

3 x 3 image

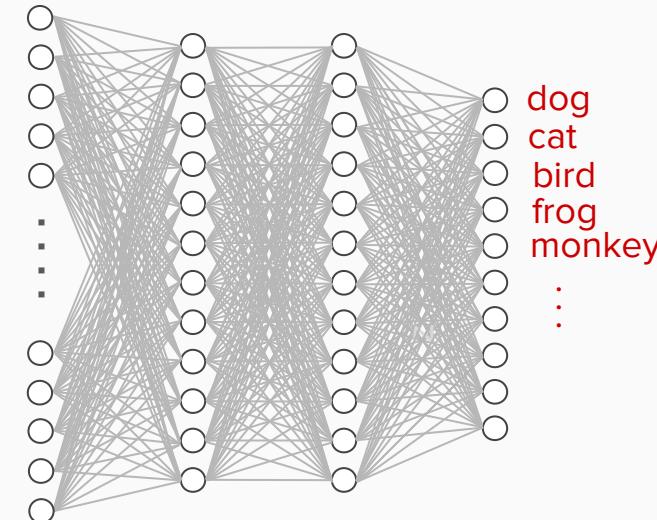
No. of filters is
no. of channels

new image but smaller!
less parameters!

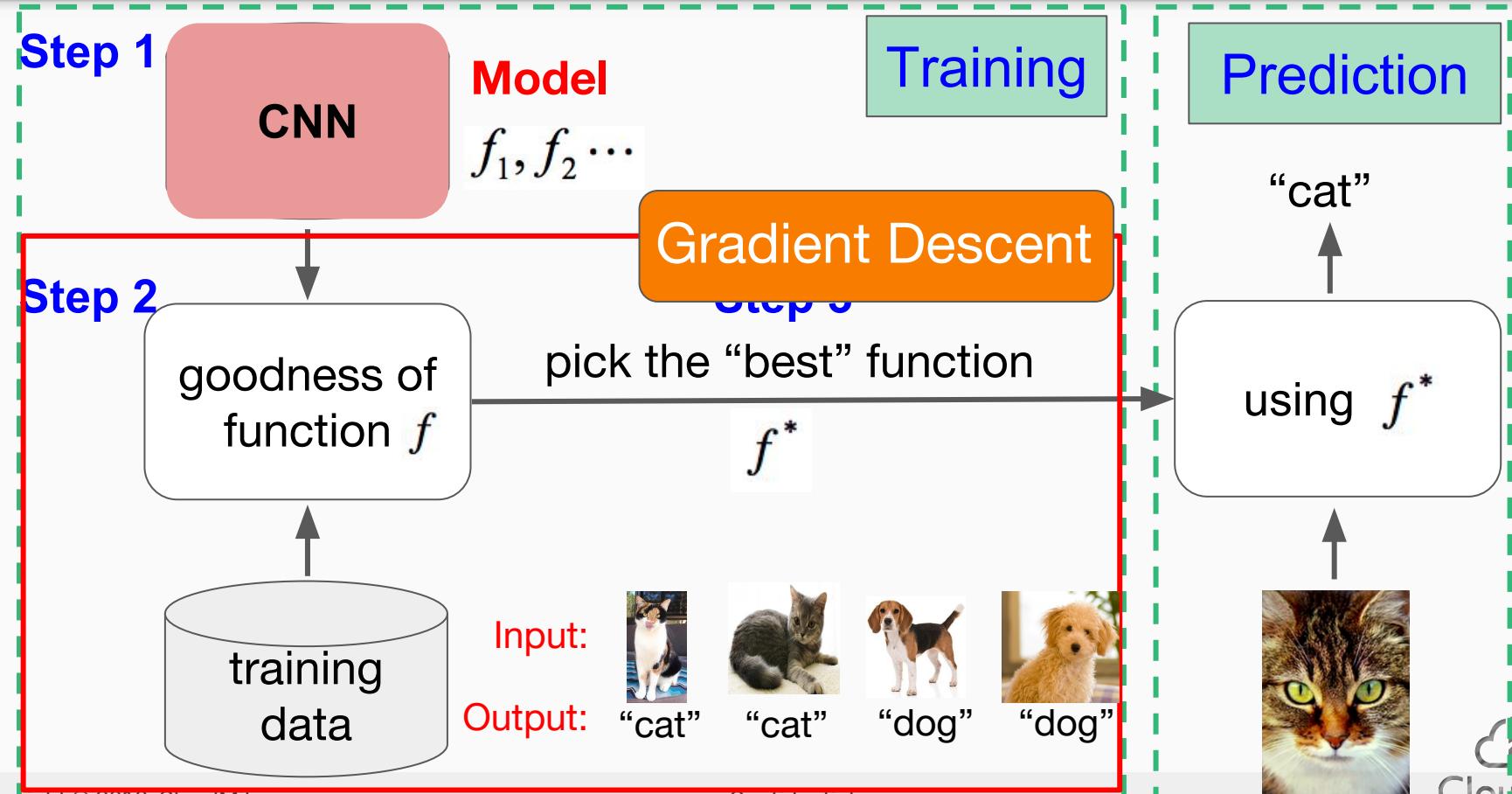
Whole CNN



Flatten



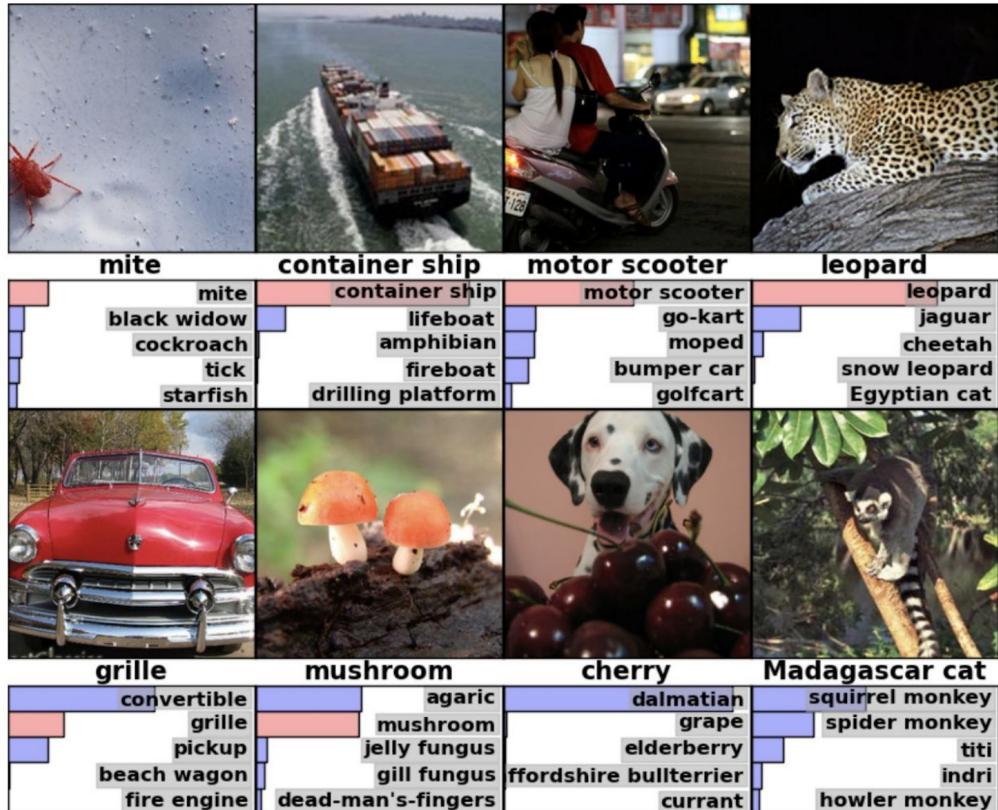
Machine Learning Framework



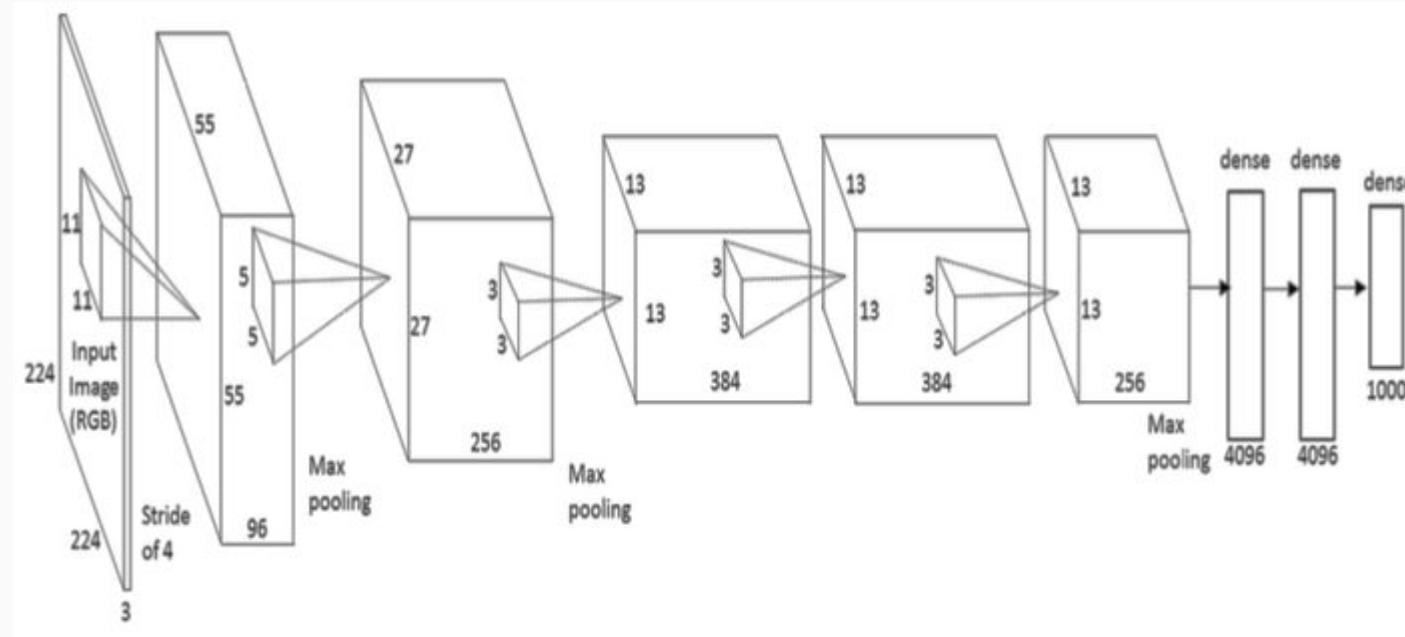
ImageNet Challenge

IMAGENET

- 1,000 object classes (categories).
- Images:
 - 1.2 M train
 - 100k test.

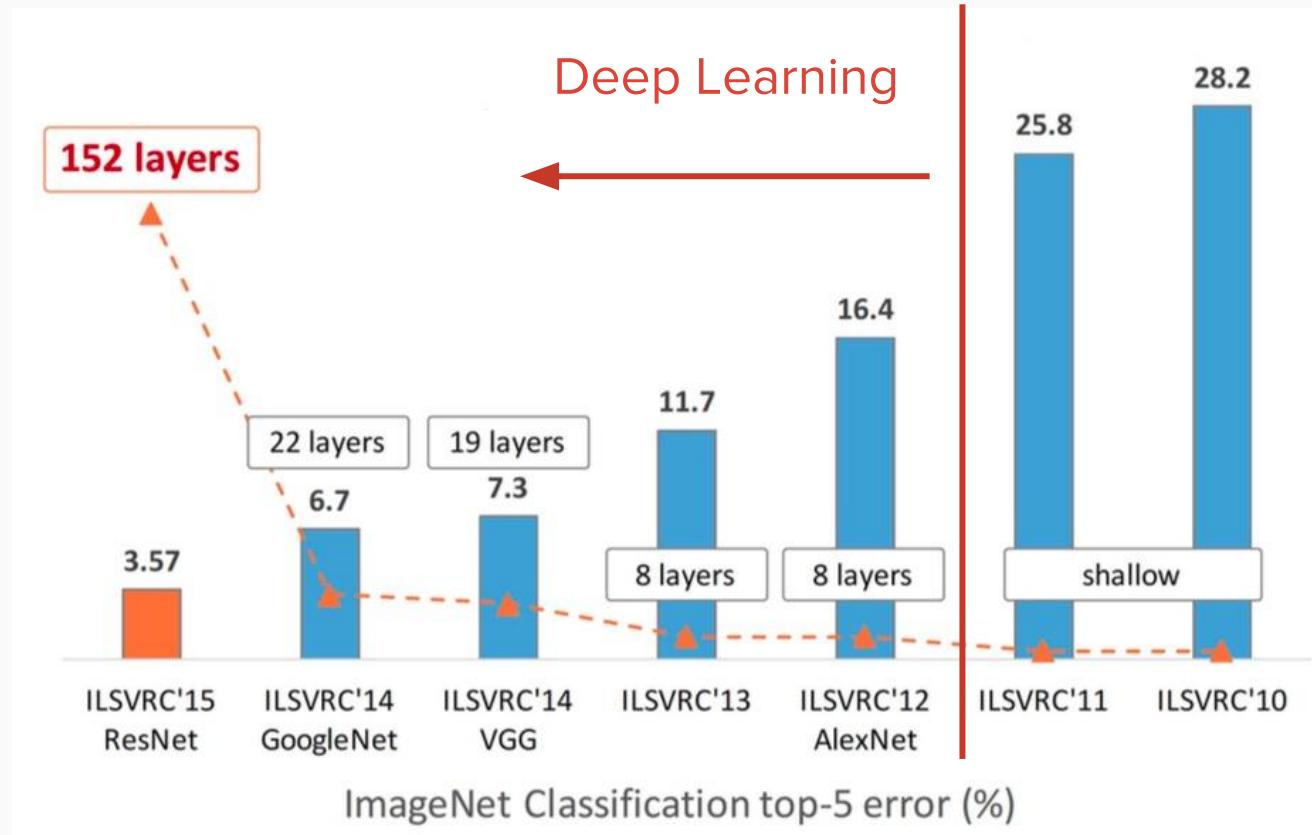


AlexNet (2012)

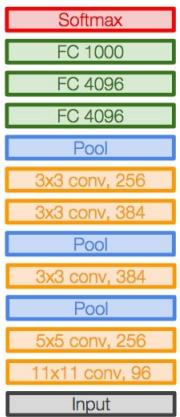


8 layers, ImageNet Classification top-5 error is **16.4%**.

You Need to Go Deeper

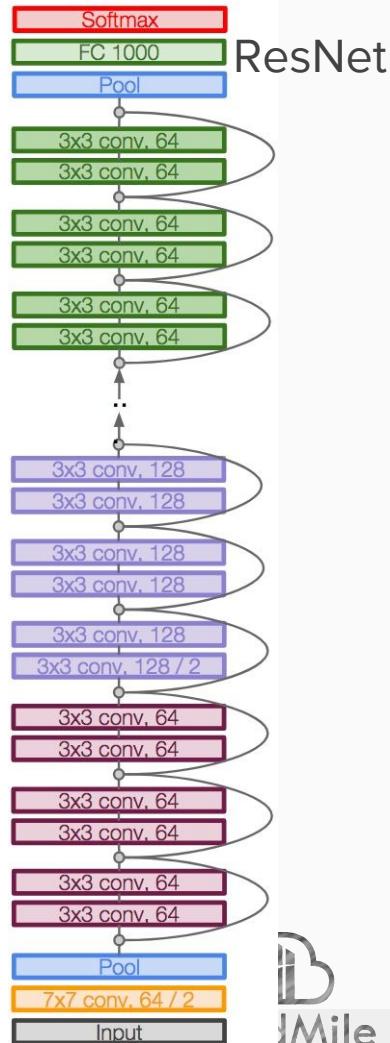
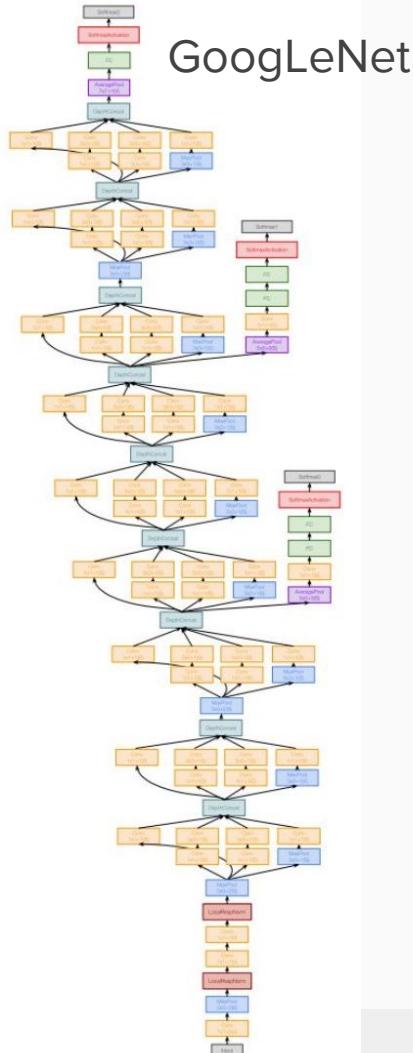
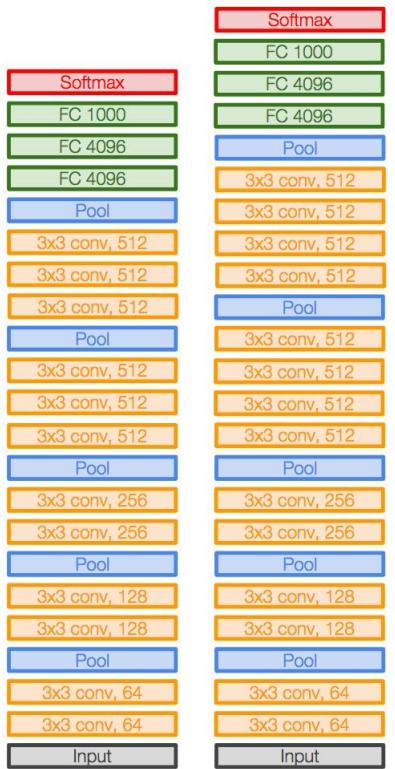


AlexNet

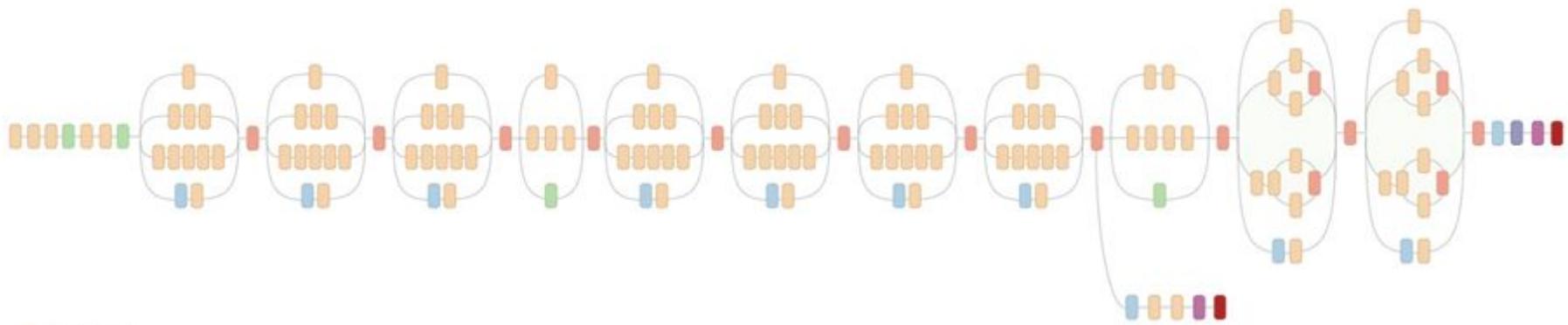


VGG16

VGG19



Inception V3

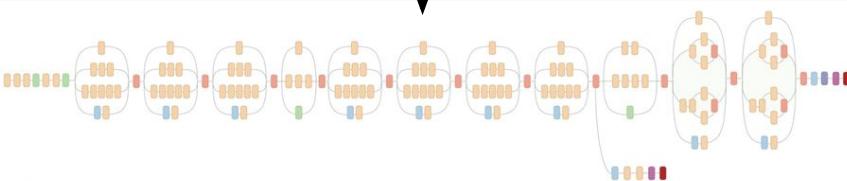


- Convolution
- AvgPool
- MaxPool
- Concat
- Dropout
- Fully connected
- Softmax



CloudMile

Transfer Learning



trained on ImageNet



roses(玫瑰)



sunflowers(向日葵)



daisy(雏菊)



tulips(鬱金香)

flower classifier



CloudMile

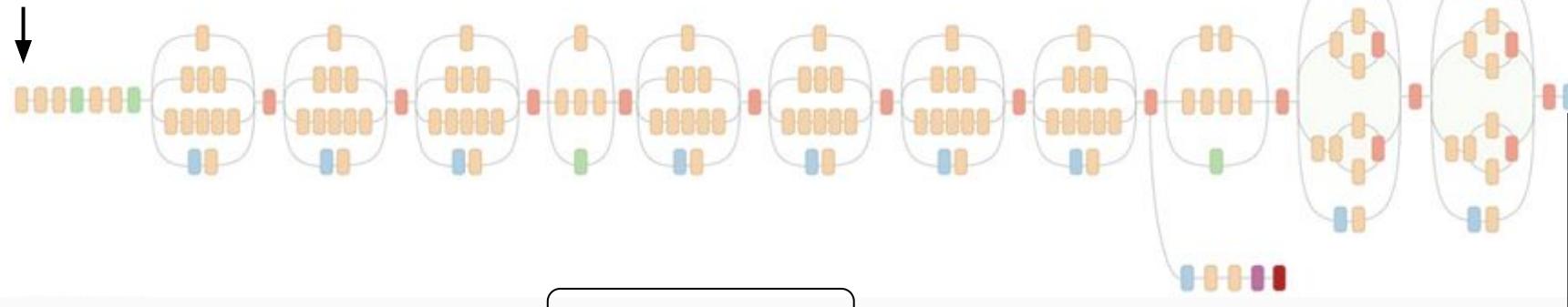
Transfer Learning (Cont'd)



raw images

pretrained on ImageNet

feature extraction part

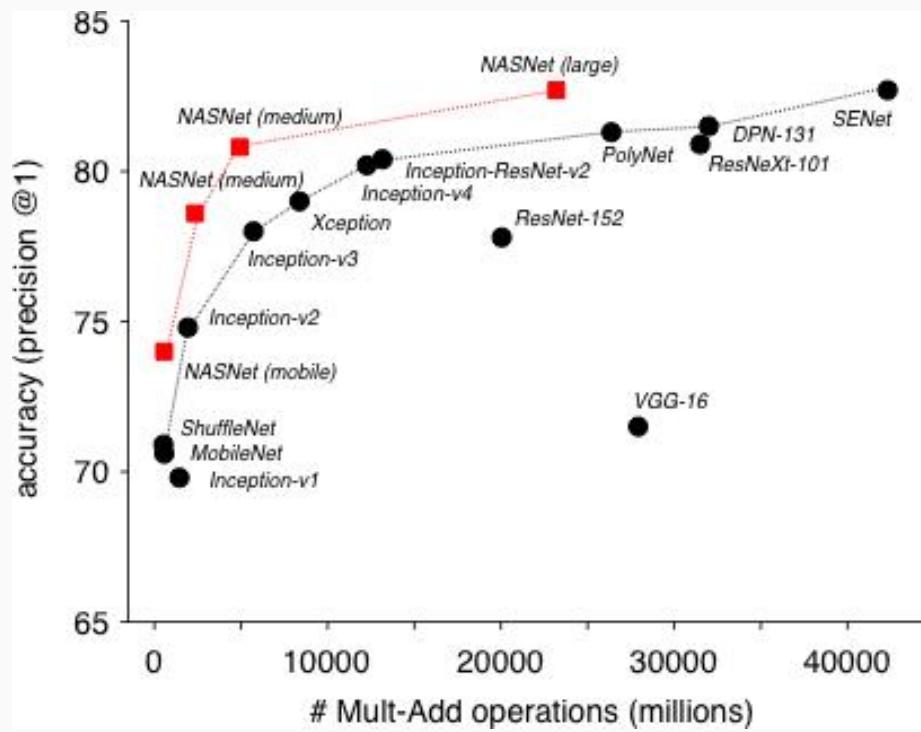


feature vector

only train this part

classification part
Confidential

Choosing a Pretrained Model

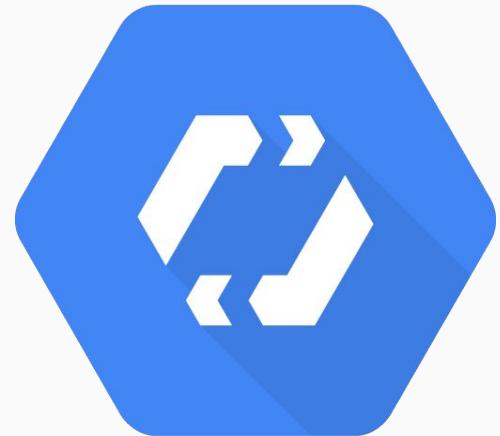


Speed	v.s.	Accuracy
MobileNet		ResNet
NASNet (mobile)		Inception
NASNet (large)		

Cloud AutoML Vision

Simple **transfer learning** with efficient model training for quick demo within minutes

High accuracy model based upon learning to learn within a day



Training an Image Classifier

Train a CNN from scratch

Can be tuned for very specific cases

Requires large amounts of data

Training takes time and resources

Transfer Learning

Works well with small data

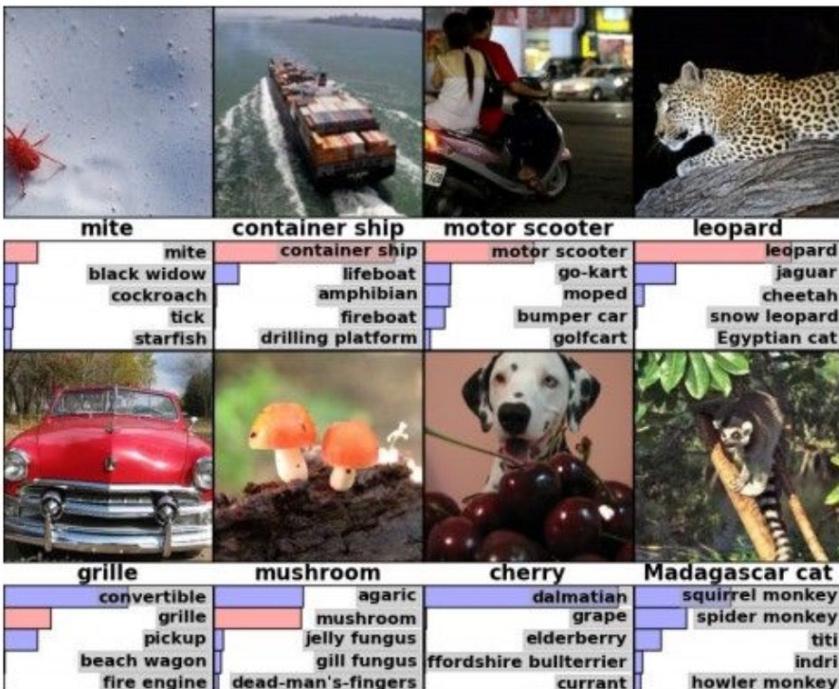
Training is much faster

Less performant on very specific cases

CNNs are everywhere

Slide Credit: stanford.io/2ybZ4Ay

Classification



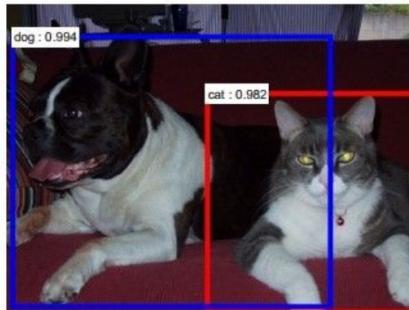
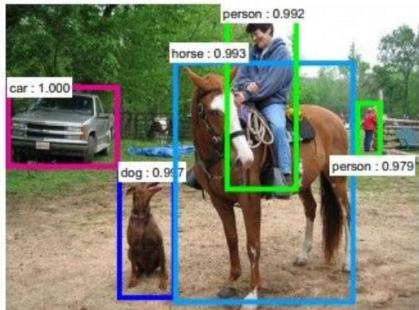
Retrieval



CNNs are everywhere

Slide Credit: stanford.io/2ybZ4Ay

Detection



Segmentation



CNNs are everywhere

Slide Credit: stanford.io/2ybZ4Ay

self-driving car



CNNs are everywhere

Slide Credit: stanford.io/2ybZ4Ay



Images are examples of pose estimation, not actually from Toshev & Szegedy 2014. Copyright Lane McIntosh.

[Toshev, Szegedy 2014]



CNNs are everywhere

Slide Credit: stanford.io/2ybZ4Ay

No errors



A white teddy bear sitting in the grass



A man riding a wave on top of a surfboard

Minor errors



A man in a baseball uniform throwing a ball



A cat sitting on a suitcase on the floor

Somewhat related



A woman is holding a cat in her hand



A woman standing on a beach holding a surfboard

Image Captioning

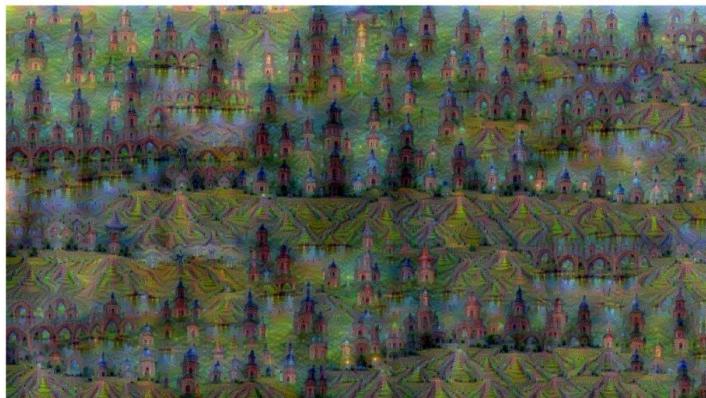
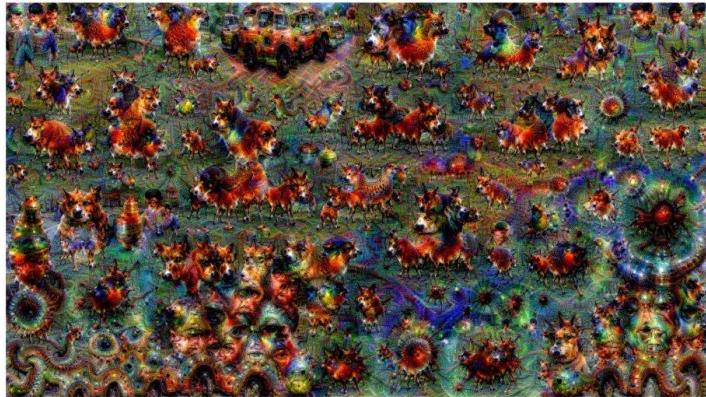
[Vinyals et al., 2015]
[Karpathy and Fei-Fei, 2015]

All images are CC0 Public domain:
<https://pixabay.com/en/luggage-antique-cat-1643010/>
<https://pixabay.com/en/teddy-plush-bears-cute-teddy-bear-1623436/>
<https://pixabay.com/en/surf-wave-summer-sport-litoral-1668716/>
<https://pixabay.com/en/woman-female-model-portrait-adult-983967/>
<https://pixabay.com/en/handstand-lake-meditation-496008/>
<https://pixabay.com/en/baseball-player-shortstop-infield-1045263/>

Captions generated by Justin Johnson using [Neuraltalk2](#)

CNNs are everywhere

Slide Credit: stanford.io/2ybZ4Ay



Original image is CC0 public domain
Starry Night and Tree Roots by Van Gogh are in the public domain
Bokeh image is in the public domain
Stylized images copyright Justin Johnson, 2017;
reproduced with permission

Confidential

Figures copyright Justin Johnson, 2015. Reproduced with permission. Generated using the Inceptionism approach from a [blog post](#) by Google Research.

Gatys et al, "Image Style Transfer using Convolutional Neural Networks", CVPR 2016
Gatys et al, "Controlling Perceptual Factors in Neural Style Transfer", CVPR 2017

Build your first CNN!

Lab 2-i

Topic : Build a basic CNN classification model

Filename	lab2.ipynb
Data	Fashion MNIST
Target	→ Train a CNN classification model
Duration	~ 10 mins

Lab 2-i

Conv2D

```
keras.layers.Conv2D(filters, kernel_size, strides=(1, 1),  
padding='valid', activation=None)
```

Filters: the dimensionality of the output space

Kernel_size: height and width of the 2D convolution window

Strides: the stride of the convolution along the height and width

Padding: one of 'valid' and 'same'

Activation: activation function

MaxPooling2D

```
keras.layers.MaxPooling2D(pool_size=(2, 2), strides=None,  
padding='valid')
```

Pool_size : factors by which to downscale

Strides : Stride values. If none, it will default to pool_size

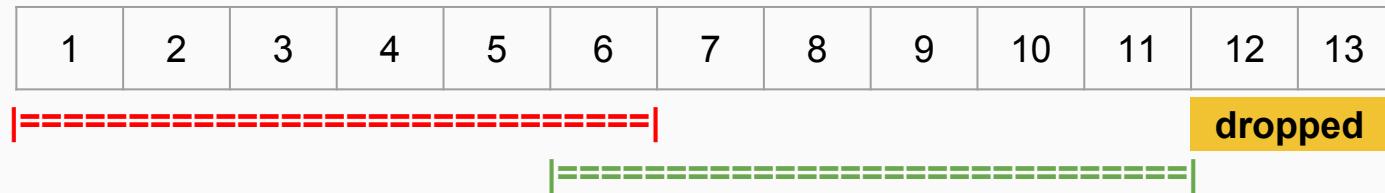
Padding : one of ‘valid’ and ‘same’

What is padding?

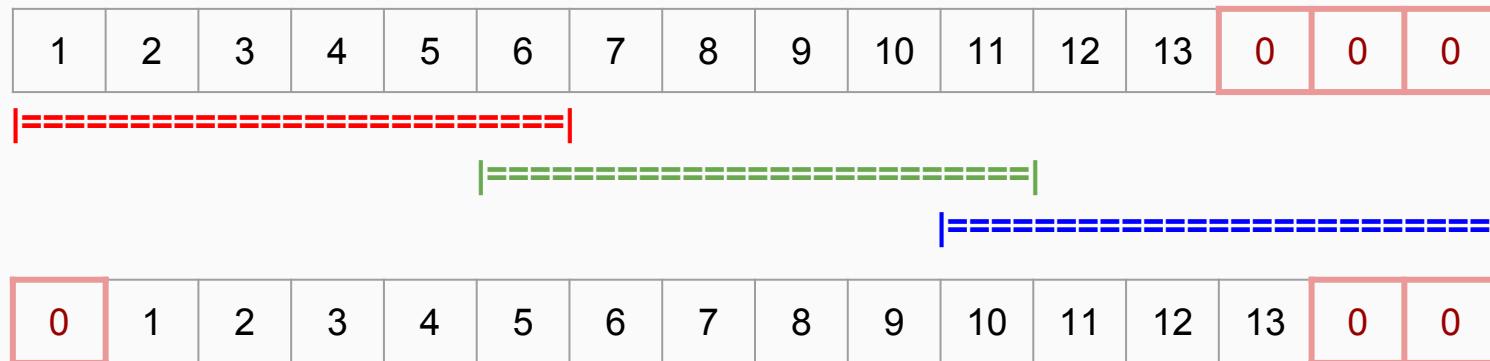
In this example:

- Input width = 13
- Filter width = 6
- Stride = 5

Padding = 'VALID' → Without padding



Padding = 'SAME' → With zero padding



Lab 2-i

```
K.clear_session()

model_basic_cnn = keras.Sequential()

model_basic_cnn.add(Conv2D(filters=32,
                           kernel_size=(3,3),
                           input_shape=(28,28,1),
                           padding='same',
                           activation='relu'))

model_basic_cnn.add(MaxPool2D(pool_size=(2,2),
                             strides=(2,2)))

model_basic_cnn.add(Conv2D(filters=64,
                           kernel_size=(3,3),
                           padding='same',
                           activation='relu'))

model_basic_cnn.add(MaxPool2D(pool_size=(2,2),
                             strides=(2,2)))

model_basic_cnn.add(Conv2D(filters=128,
                           kernel_size=(3,3),
                           padding='same',
                           activation='relu'))

model_basic_cnn.add(MaxPool2D(pool_size=(2,2),
                             strides=(2,2)))

model_basic_cnn.add(Flatten())

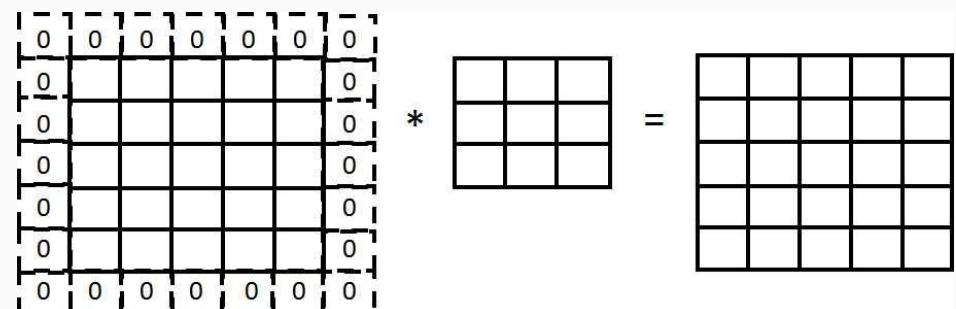
model_basic_cnn.add(Dense(10, activation='softmax'))

model_basic_cnn.summary()
```

Padding → ‘**same**’ : padding the input such that the output has the same length as the original input if the stride = 1
→ ‘**valid**’ : no padding

Stride → specify the stride length of the convolution

Padding(same)



Lab 2-i

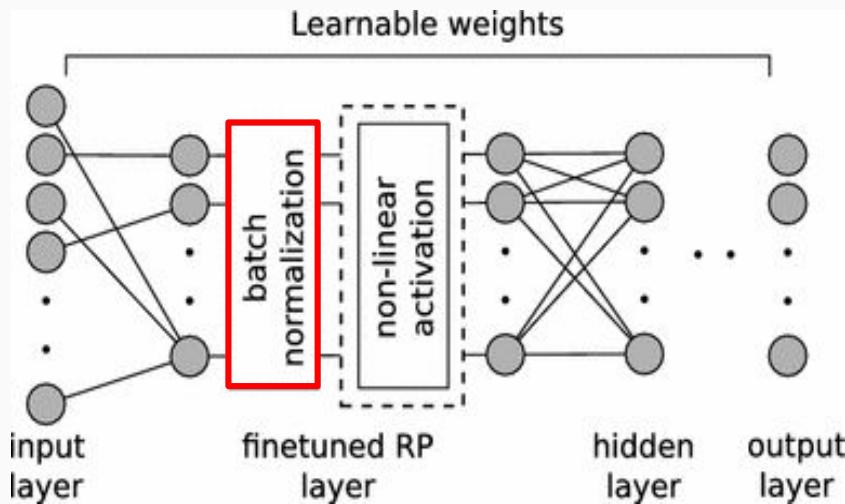
Topic : Build a basic CNN classification model

Start coding !

Filename	lab2.ipynb
Data	Fashion MNIST
Target	→ Train a CNN classification model
Duration	~ 10 mins

Batch normalization

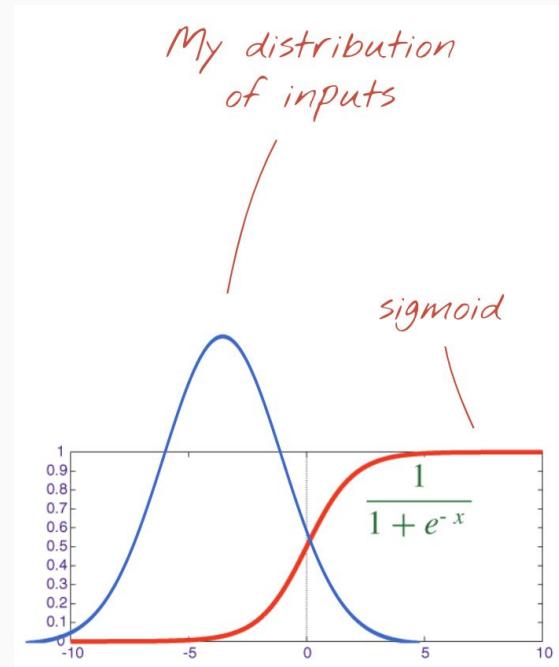
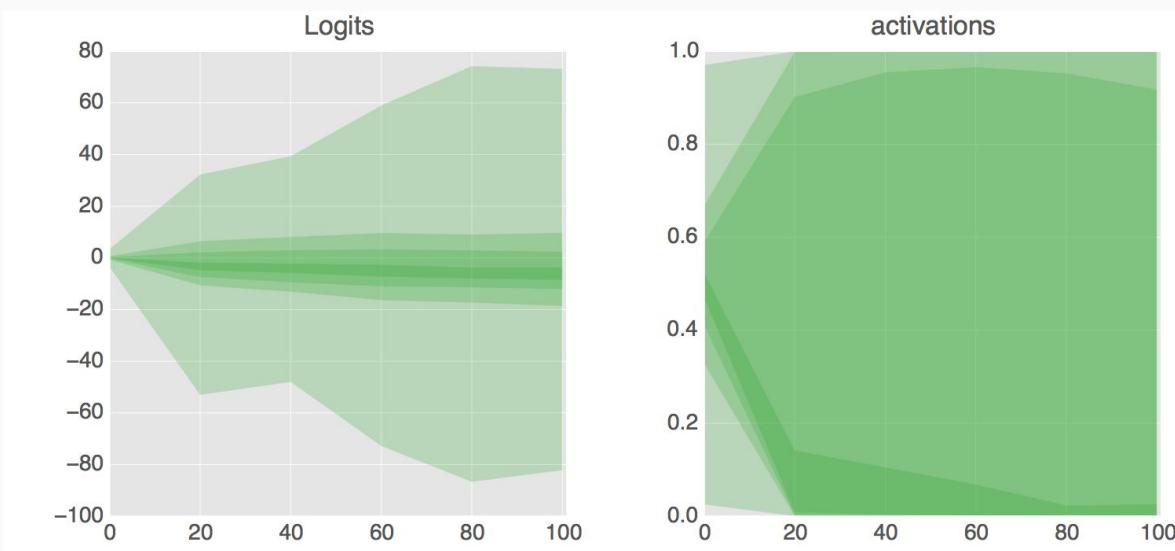
- Before feeding the outputs of one layer into the inputs of the next, we center and rescale them.



$$\hat{x} = \frac{x - avg_{batch}(x)}{stdev_{batch}(x) + \epsilon}$$

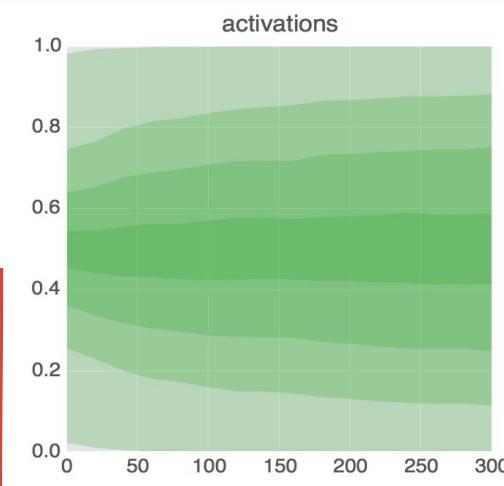
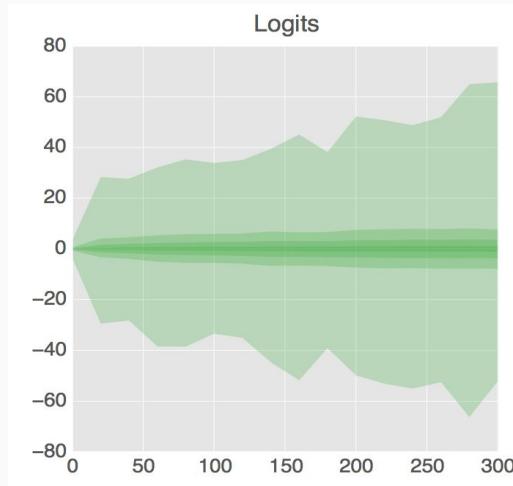
Batch normalization

Before batch normalization



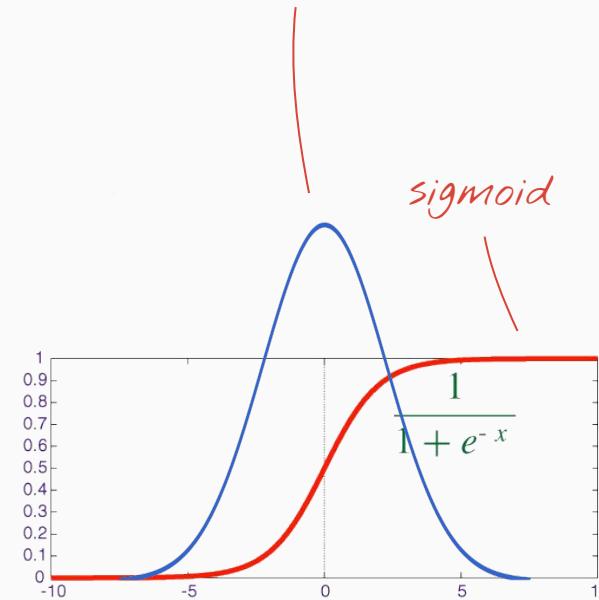
Batch normalization

After batch normalization



Batch norm

*My distribution
of inputs*



Lab 2-e

Topic : Build an advanced CNN classification model

Start coding !

Filename	lab2.ipynb
Data	Fashion MNIST
Target	→ Train a CNN classification model
Duration	~ 10 mins

Why not pretrained model?

- In keras,

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	99 MB	0.749	0.921	25,636,712	168
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-

Keras Functional API

```
from keras.layers import Input, Dense
from keras.models import Model

# This returns a tensor
inputs = Input(shape=(784,))

# a layer instance is callable on a tensor, and returns a tensor
x = Dense(64, activation='relu')(inputs)
x = Dense(64, activation='relu')(x)
predictions = Dense(10, activation='softmax')(x)

# This creates a model that includes
# the Input layer and three Dense layers
model = Model(inputs=inputs, outputs=predictions)
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(data, labels) # starts training
```

Lab 2-f

Topic : MobileNet

Start coding !

Filename	lab2.ipynb
Data	Fashion MNIST
Target	→ Use Mobilenet to classify fashion MNIST
Duration	~ 10 mins

Lab 2-f Result

