

+



3099704: AI for Digital Health



Neural Networks & Deep Learning

Prof. Peerapon Vateekul, Ph.D.

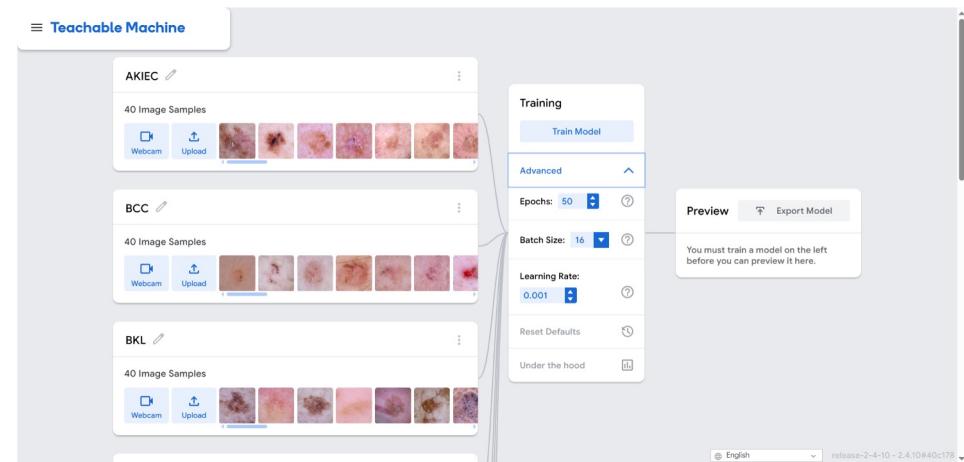
Peerapon.v@chula.ac.th



Outlines

2

- Linear Regression (recap)
- Neural Networks
- Deep Learning
- Image Classification



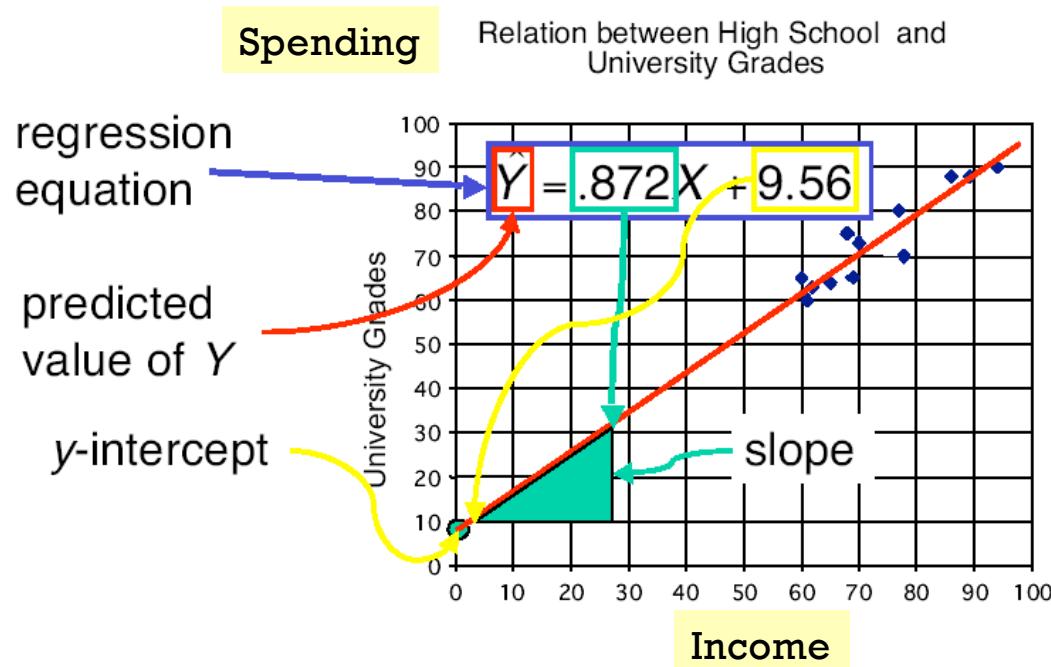
- epoch: [5, 20, 50, 100]
- learning rate: [0.00001, 0.001, 0.1]



Linear Regression (recap)



Linear Regression



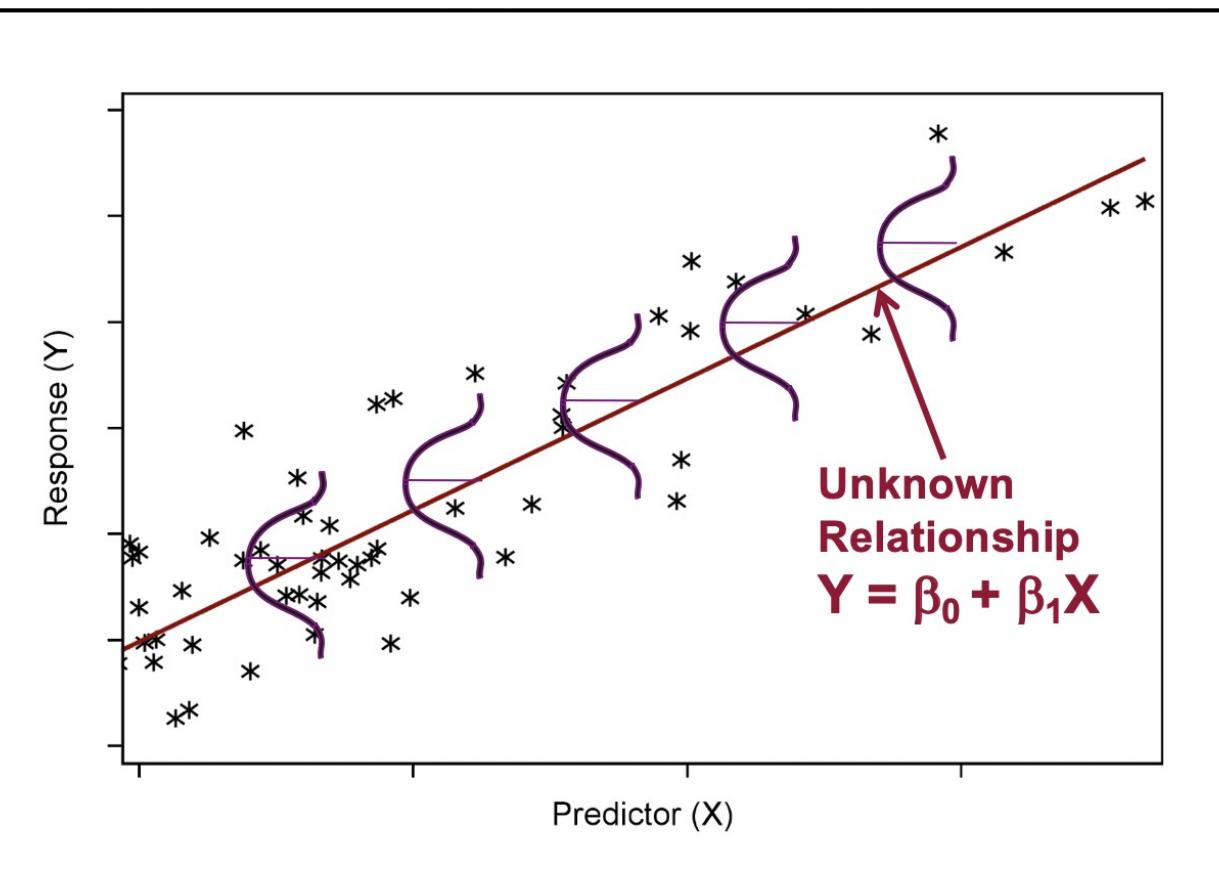
- weight, coefficient
- $\hat{y} = \widehat{w}_0 + \widehat{w}_1 x_1 + \widehat{w}_2 x_2$
- target intercept input
- The least square method aims to minimize the following term

$$\sum_{\text{training data}} (y_i - \hat{y}_i)^2$$



Linear Regression Assumption

$$\text{Spending} = 500 + 10 \times \text{Income10K} + 2 \times \text{Age}$$

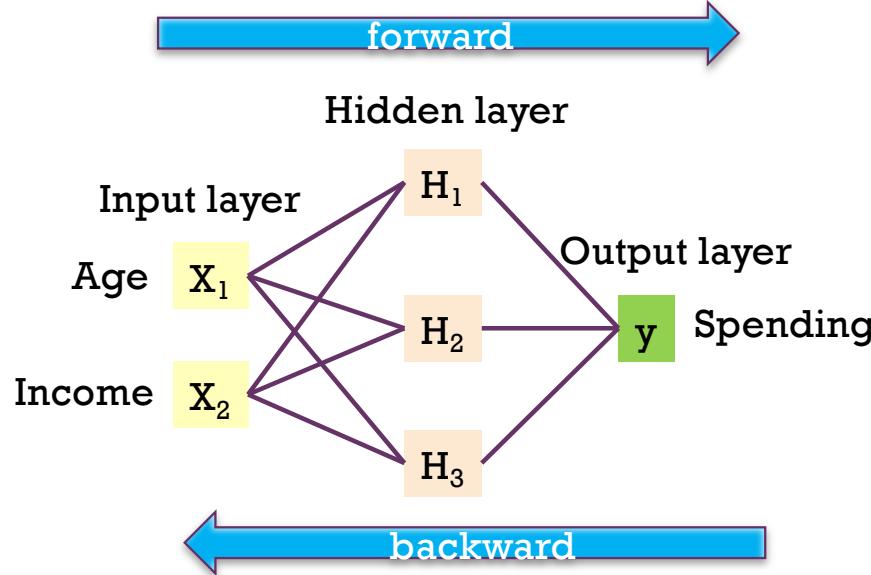


- Linear relationship between (y, x_i) . [Pearson correlation]
- Error is normal distributed. [remove outliers, log-transformation]
- Error has equal variance (homoscedasticity) [remove outliers, log-transformation]
- Errors are independent from each other. [design new data correction]



Neural Networks

Neural Networks (universal approximator)

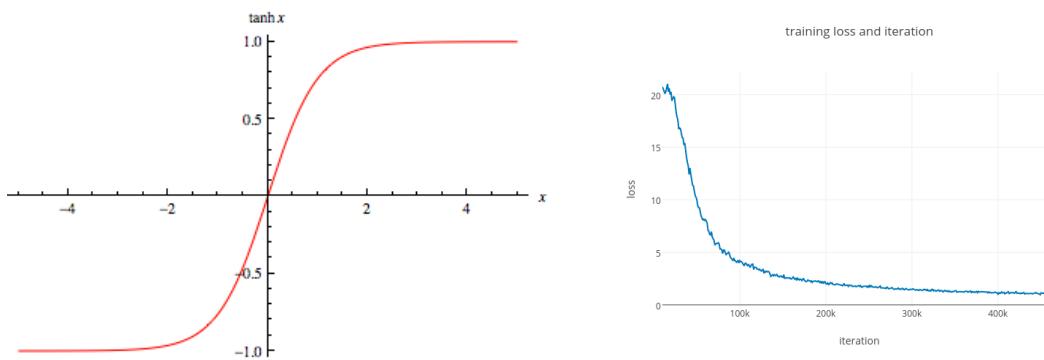


$$\text{Spending} = \hat{w}_0 + \hat{w}_1 H_1 + \hat{w}_2 H_2 + \hat{w}_3 H_3$$

$$H_1 = \tanh(\hat{w}_{10} + \hat{w}_{11}x_1 + \hat{w}_{12}x_2)$$

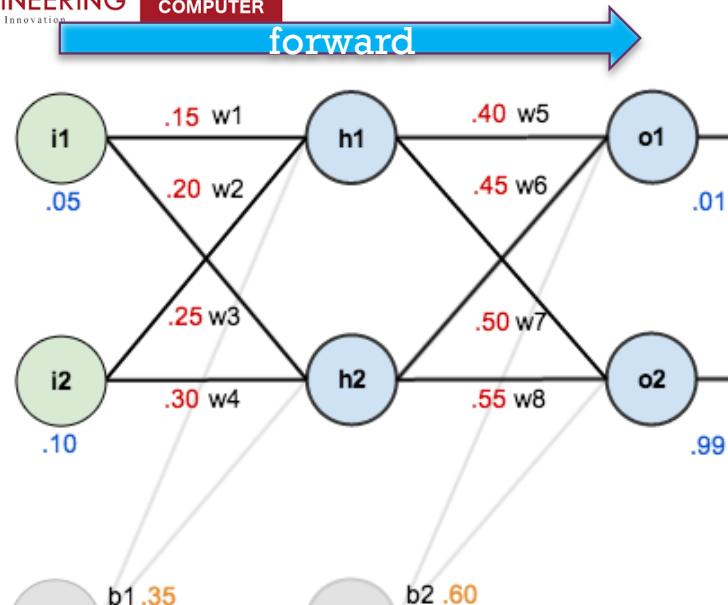
$$H_2 = \tanh(\hat{w}_{20} + \hat{w}_{21}x_1 + \hat{w}_{22}x_2)$$

$$H_3 = \tanh(\hat{w}_{30} + \hat{w}_{31}x_1 + \hat{w}_{32}x_2)$$



How to update weight

<https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/>



$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

$$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$$

Repeating this process for o_2 (remembering that the target is 0.99) we get:

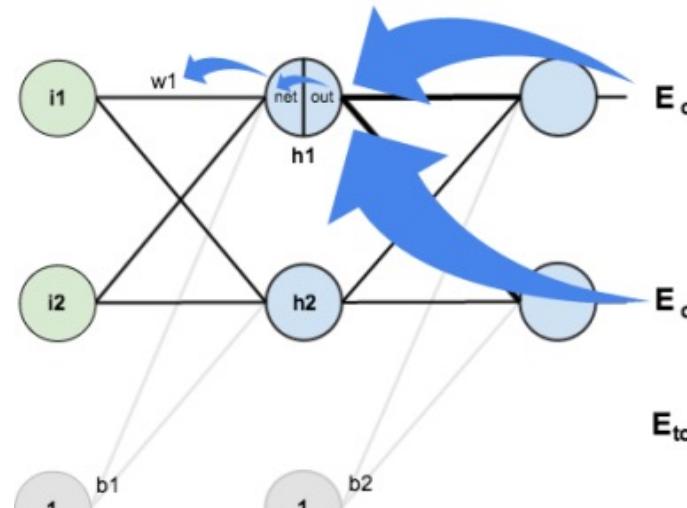
$$E_{o2} = 0.023560026$$

The total error for the neural network is the sum of these errors:

$$E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$$

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$



$$E_{total} = E_{o1} + E_{o2}$$

backward

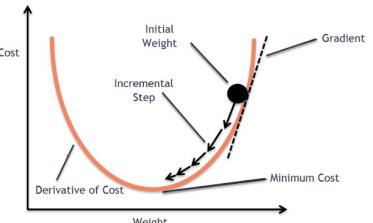
Putting it all together:

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

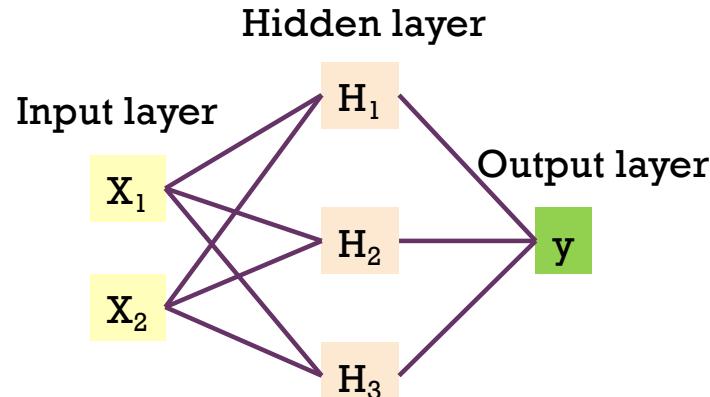
$$\frac{\partial E_{total}}{\partial w_5} = 0.74136507 * 0.186815602 * 0.593269992 = 0.082167041$$

$$w_5^+ = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 * 0.082167041 = 0.35891648$$

Learning rate

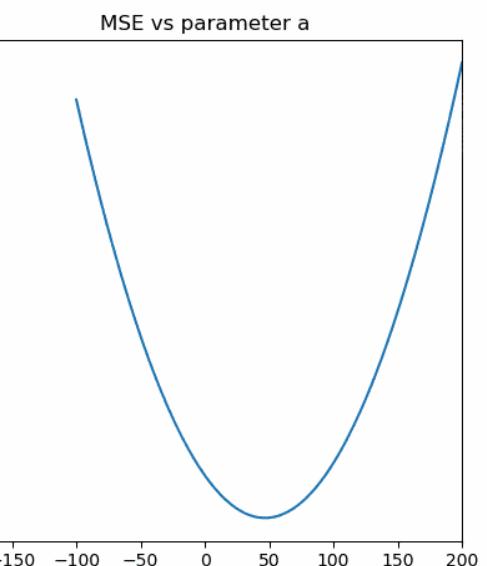
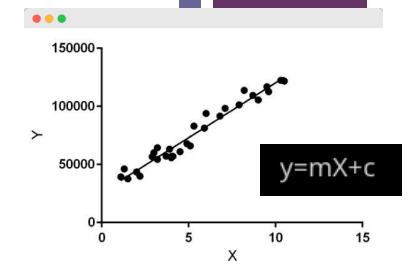
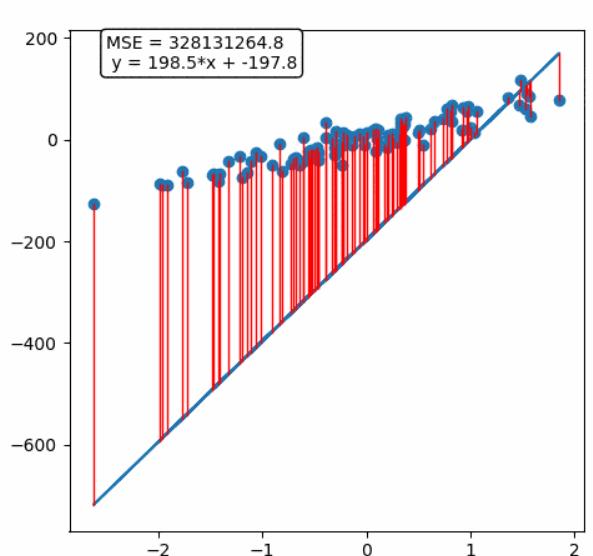
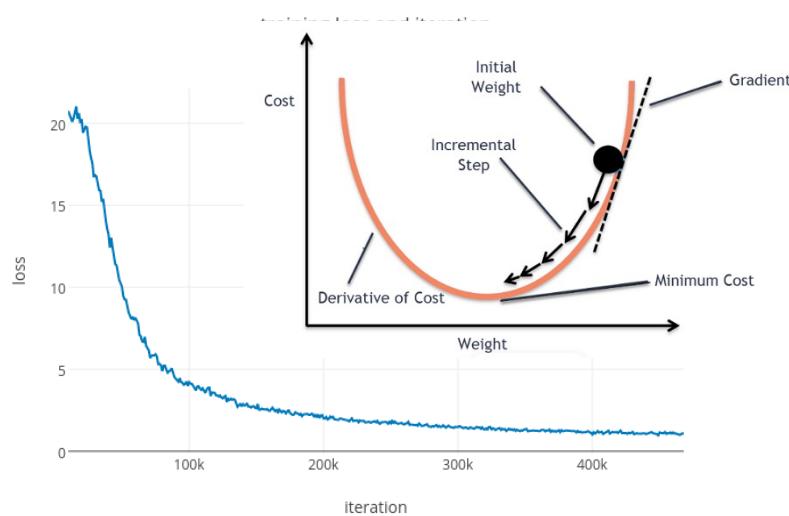


Gradient descent (solver to find weights)

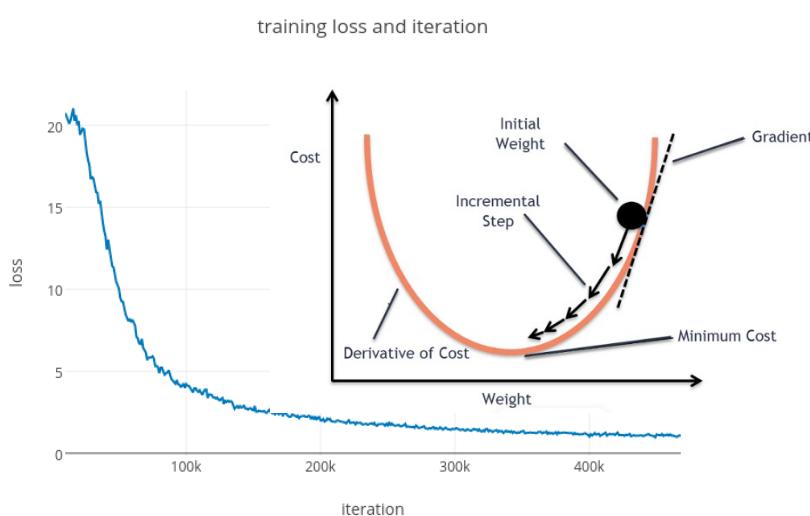
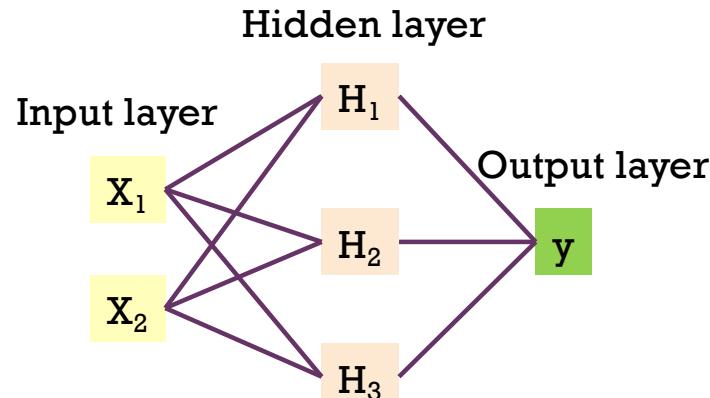


*New weight = weight - Learning rate * Gradient*

$$\text{New weight} = \text{Old weight} - \text{Learning rate} \left(\frac{\partial \text{Error}}{\partial \text{Weight}} \right)$$



Batch, Iteration, Epoch



Stop when?

- Converge (no change in loss)
- Max epochs

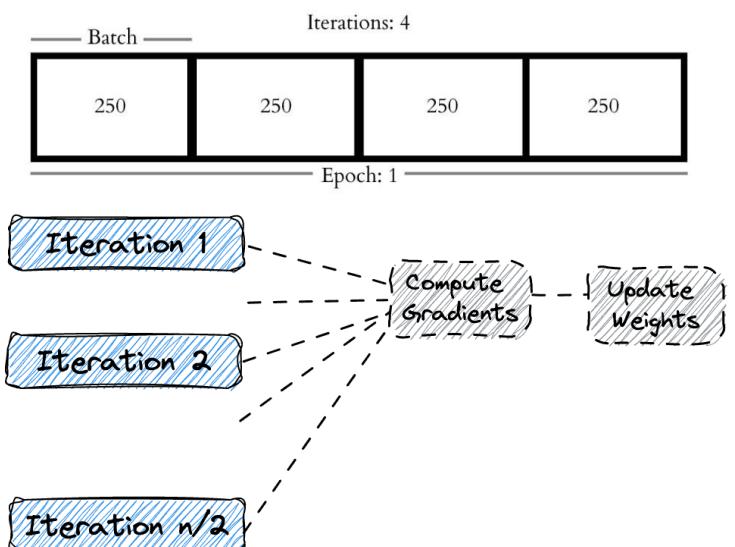
Important Params:

- #hidden units, #hidden layers
- Learning rate, momentum, decay
- Seed number, etc.

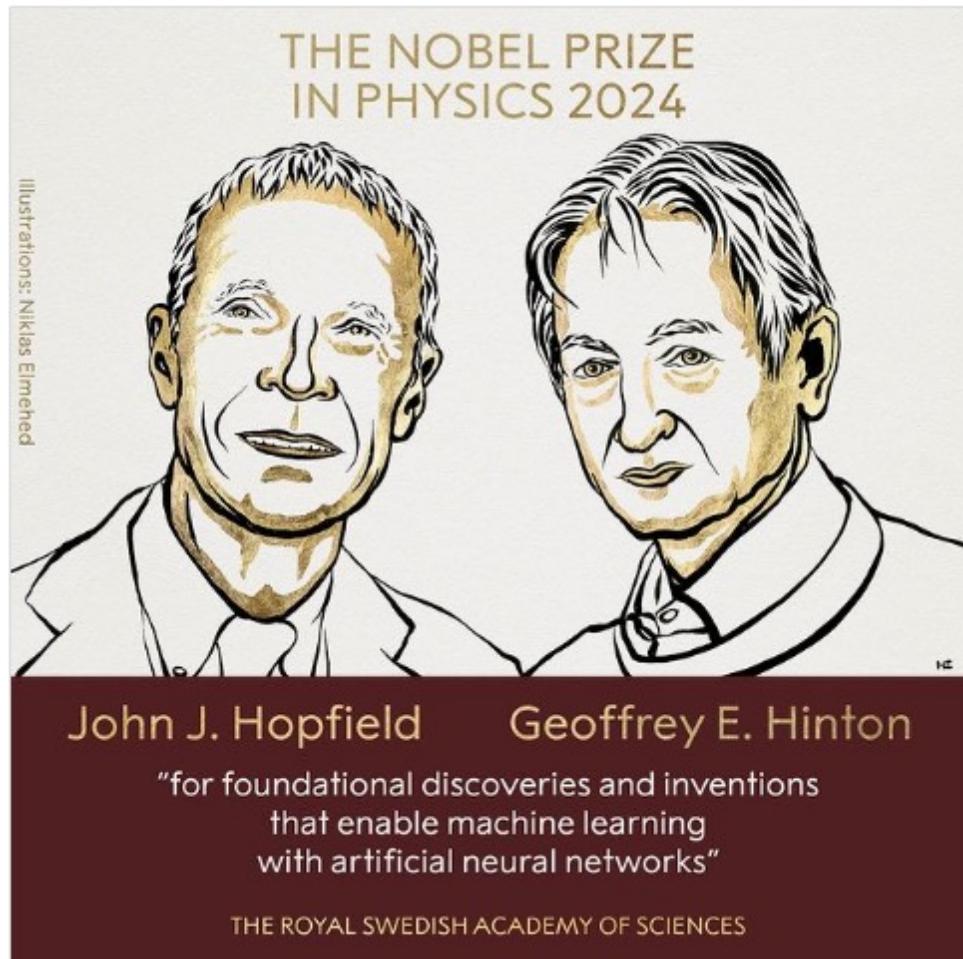
10

Epoch vs Iterations vs Batch

Example Dataset: 1000



Basic research (foundation research)



8 October 2024

The Royal Swedish Academy of Sciences has decided to award the Nobel Prize in Physics 2024 to

John J. Hopfield
Princeton University, NJ, USA

Geoffrey E. Hinton
University of Toronto, Canada

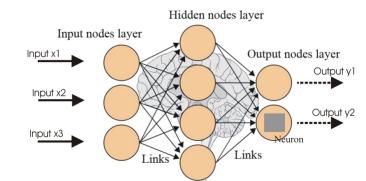
"for foundational discoveries and inventions that enable machine learning with artificial neural networks"

They trained artificial neural networks using physics

John Hopfield invented a network that uses a method for saving and recreating patterns. We can imagine the nodes as pixels.

The *Hopfield network* utilises physics that describes a material's characteristics due to its atomic spin – a property that makes each atom a tiny magnet.

Geoffrey Hinton used the Hopfield network as the foundation for a new network that uses a different method: the *Boltzmann machine*. This can learn to recognise characteristic elements in a given type of data.



Tinker With a Neural Network Right Here in Your Browser.

Don't Worry, You Can't Break It. We Promise.

Epoch 000,000 Learning rate 0.03 Activation Tanh Regularization None Regularization rate 0 Problem type Classification

DATA FEATURES + - 2 HIDDEN LAYERS OUTPUT

Which dataset do you want to use?
Which properties do you want to feed in?

Ratio of training to test data: 50%
Noise: 0
Batch size: 10

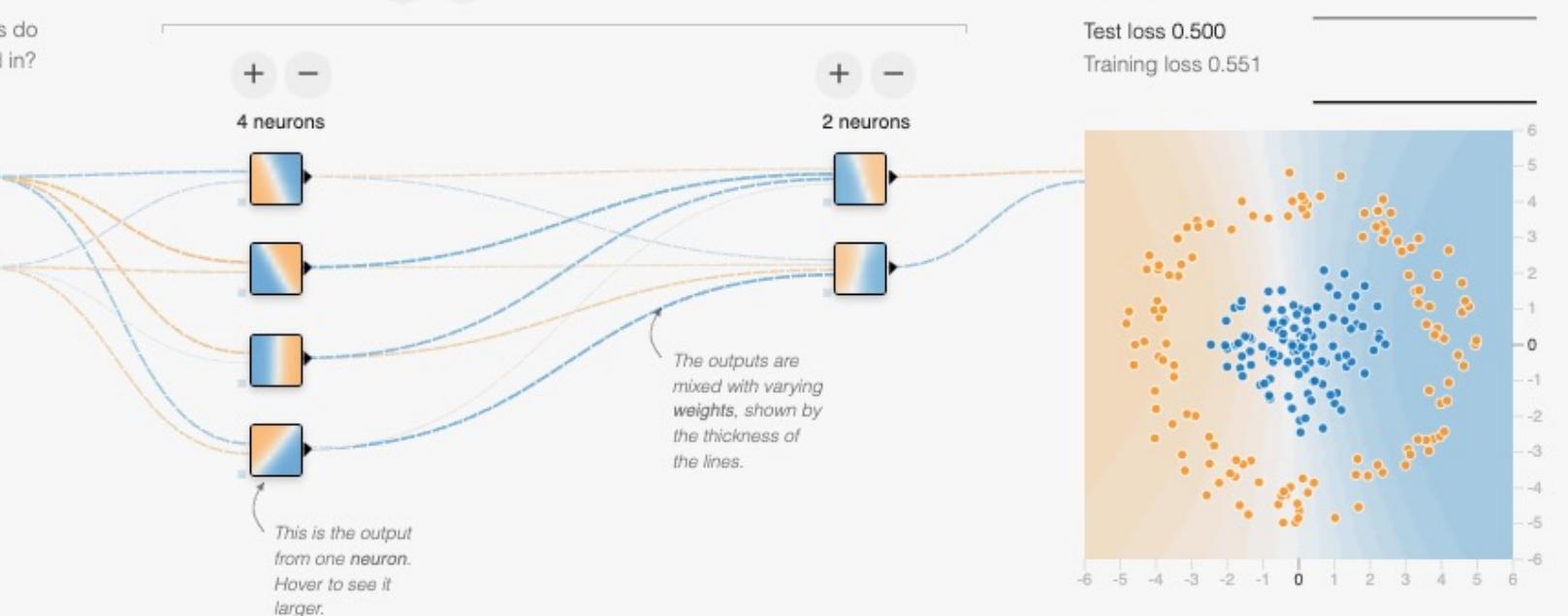
+ - 4 neurons
 X_1 X_2 X_1^2 X_2^2 $X_1 X_2$ $\sin(X_1)$

+ - 2 neurons

The outputs are mixed with varying weights, shown by the thickness of the lines.

This is the output from one neuron. Hover to see it larger.

Test loss 0.500
Training loss 0.551



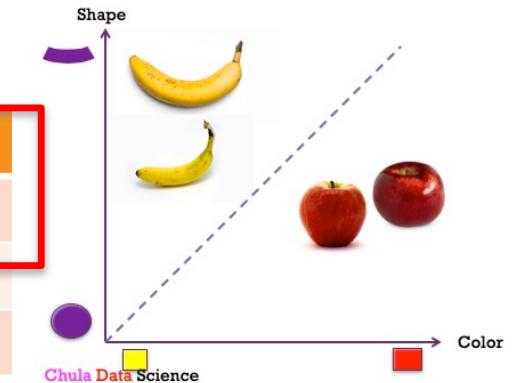
+

Deep Learning



Handcrafted features

Age	Income	Gender	Province	Corona
25	25,000	Female	Bangkok	Yes
35	50,000	Female	Nontaburi	Yes
32	35,000	Male	Bangkok	No



14

Can we still tell the features (columns)?



shutterstock.com - 451802557



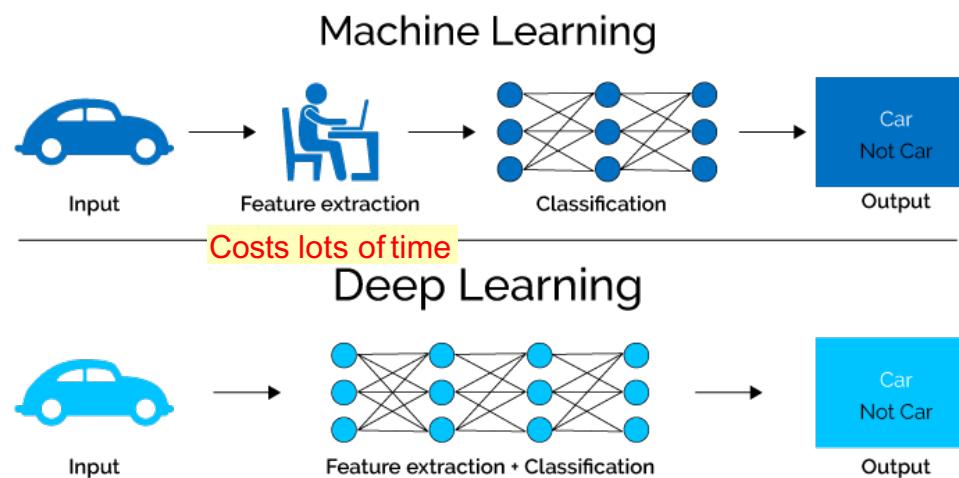
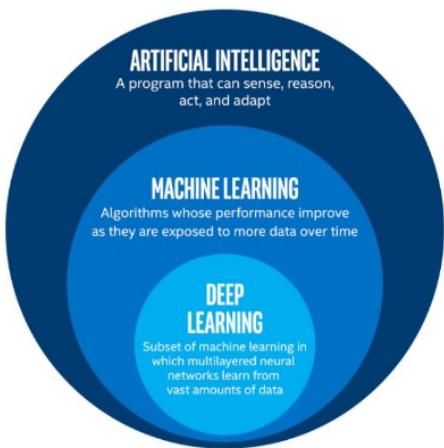
What is Deep Learning (DL)?



Part of the machine learning field of learning representations of data. Exceptional effective at learning patterns.



Utilizes learning algorithms that derive meaning out of data by using a hierarchy of multiple layers that mimic the neural networks of our brain.





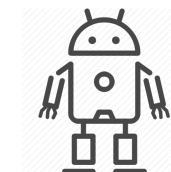
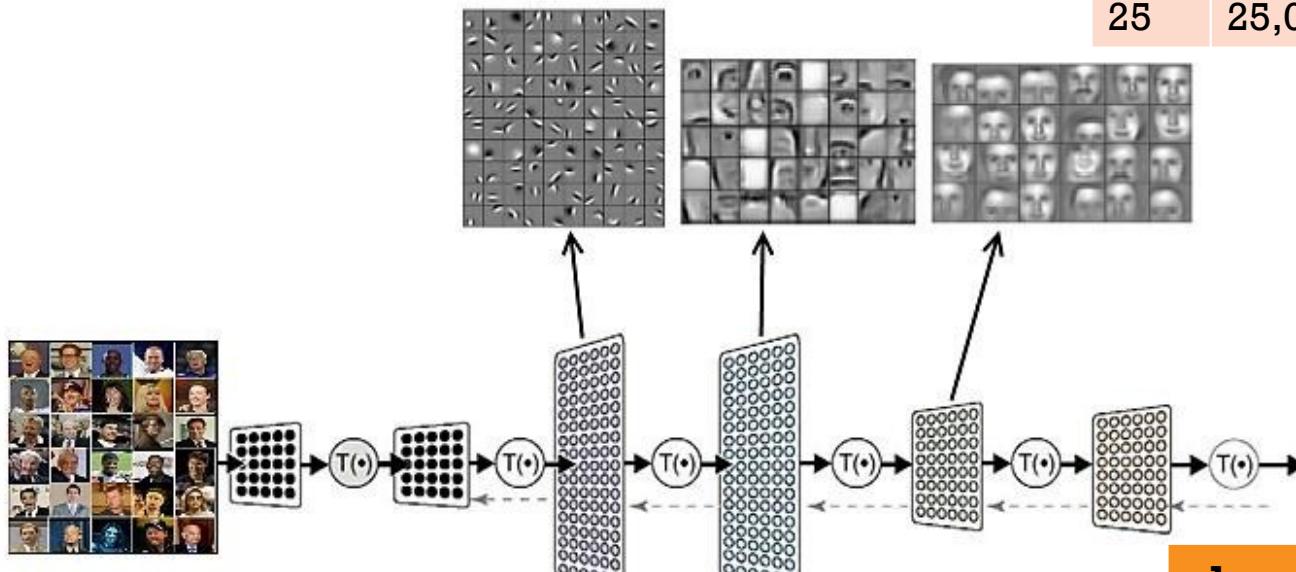
Deep Learning – Basics (cont.)

What did it learn?

A deep neural network consists of a **hierarchy of layers**, whereby each layer **transforms the input data** into more abstract representations (e.g., edge -> nose -> face). The output layer combines those features to make predictions.

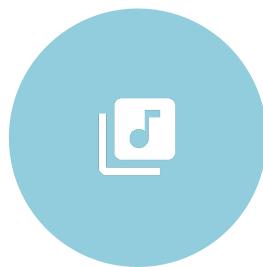


Age	Income	Gender	Province	Corona
25	25,000	Female	Bangkok	Yes



x1	x2	x3	x4	Corona
0.7	0.2	-0.5	-0.1	Yes

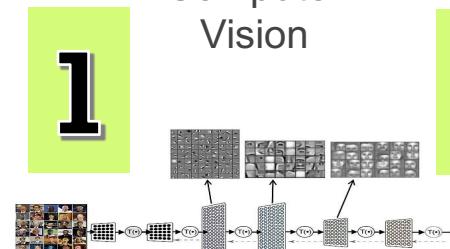
Deep Learning Application



Speech
Recognition



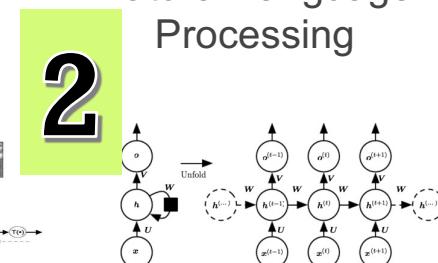
Computer
Vision



CNN

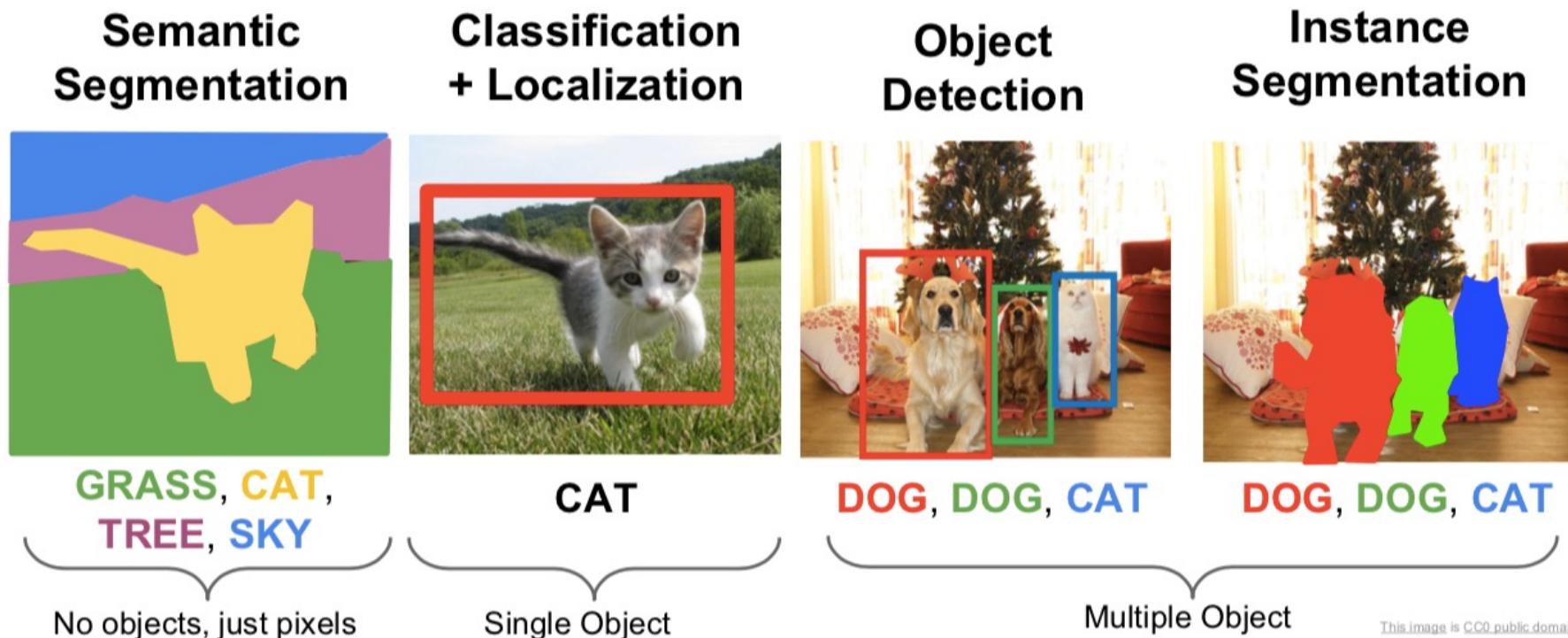


Natural Language
Processing



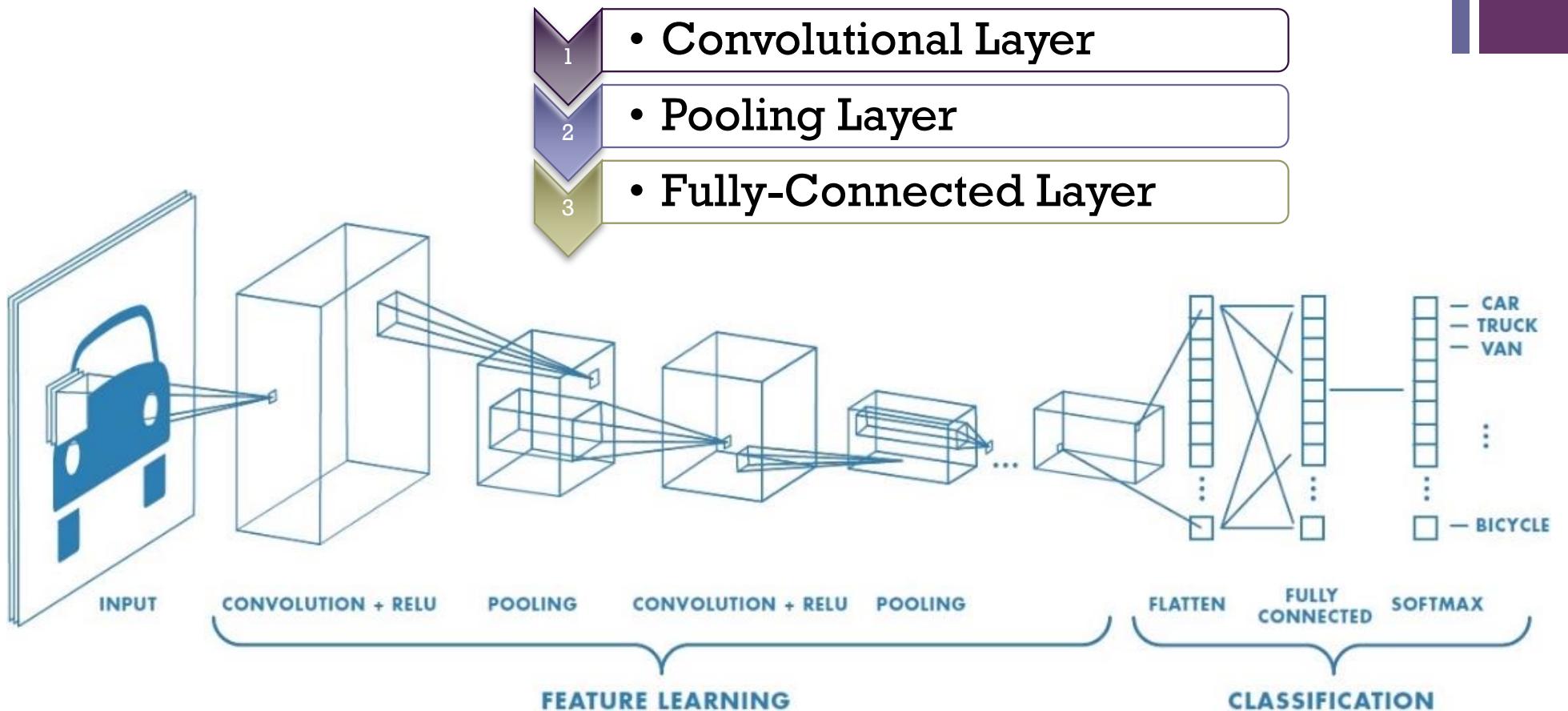
RNN

Type of image tasks

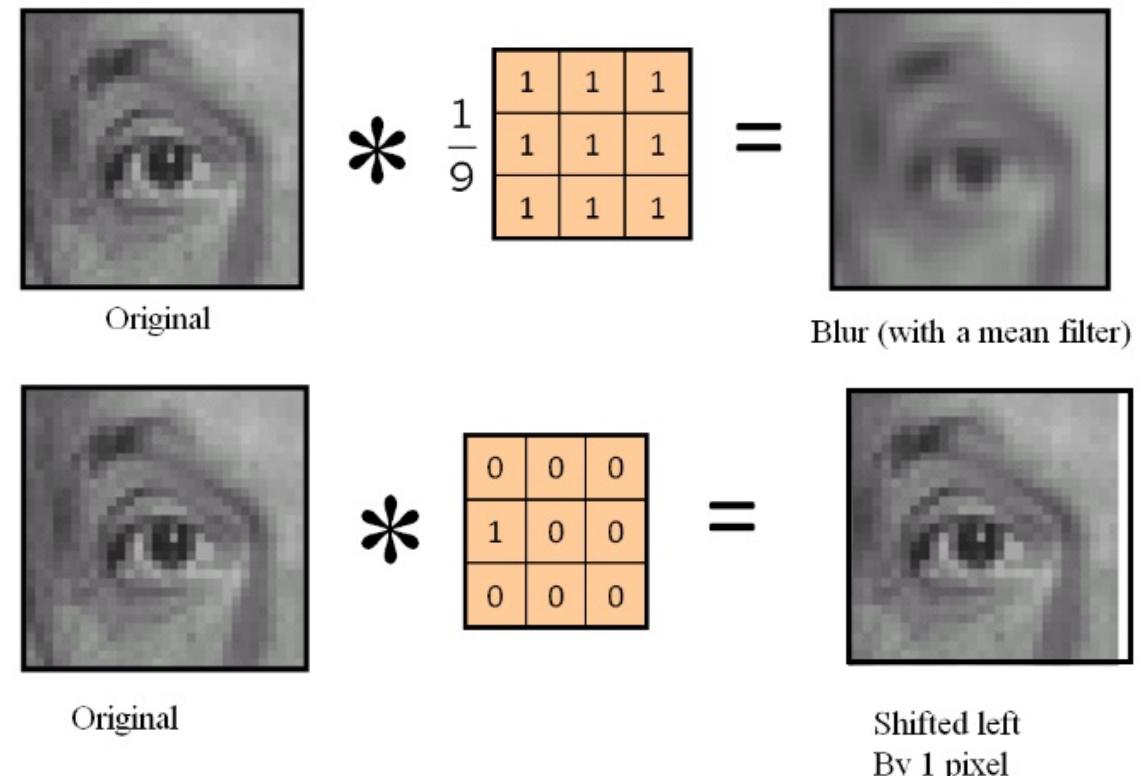
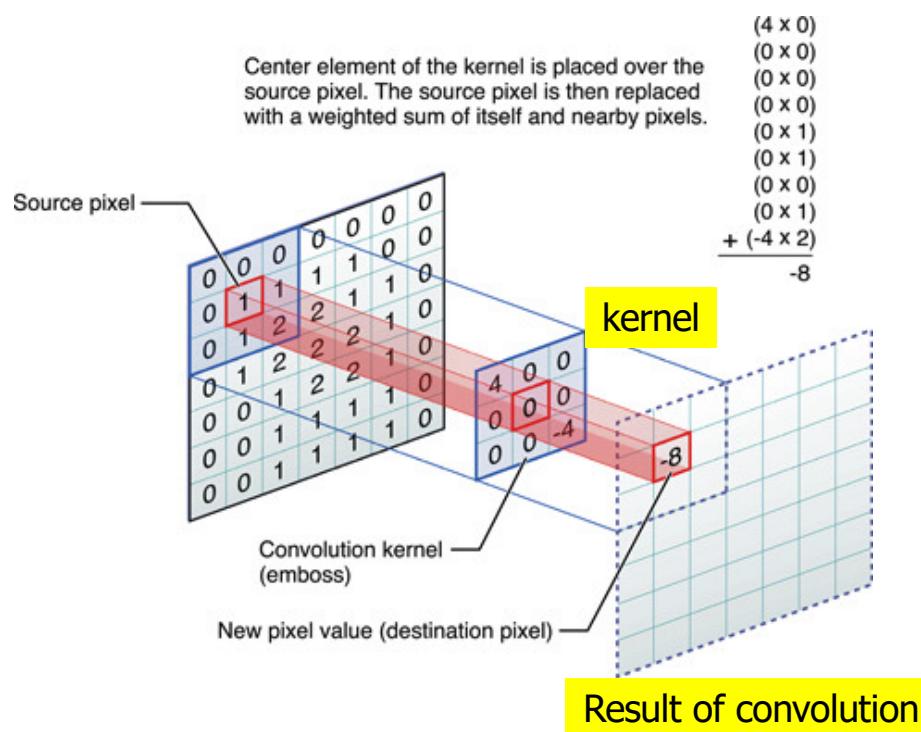




Convolutional Neural Networks (CNN)



Convolution

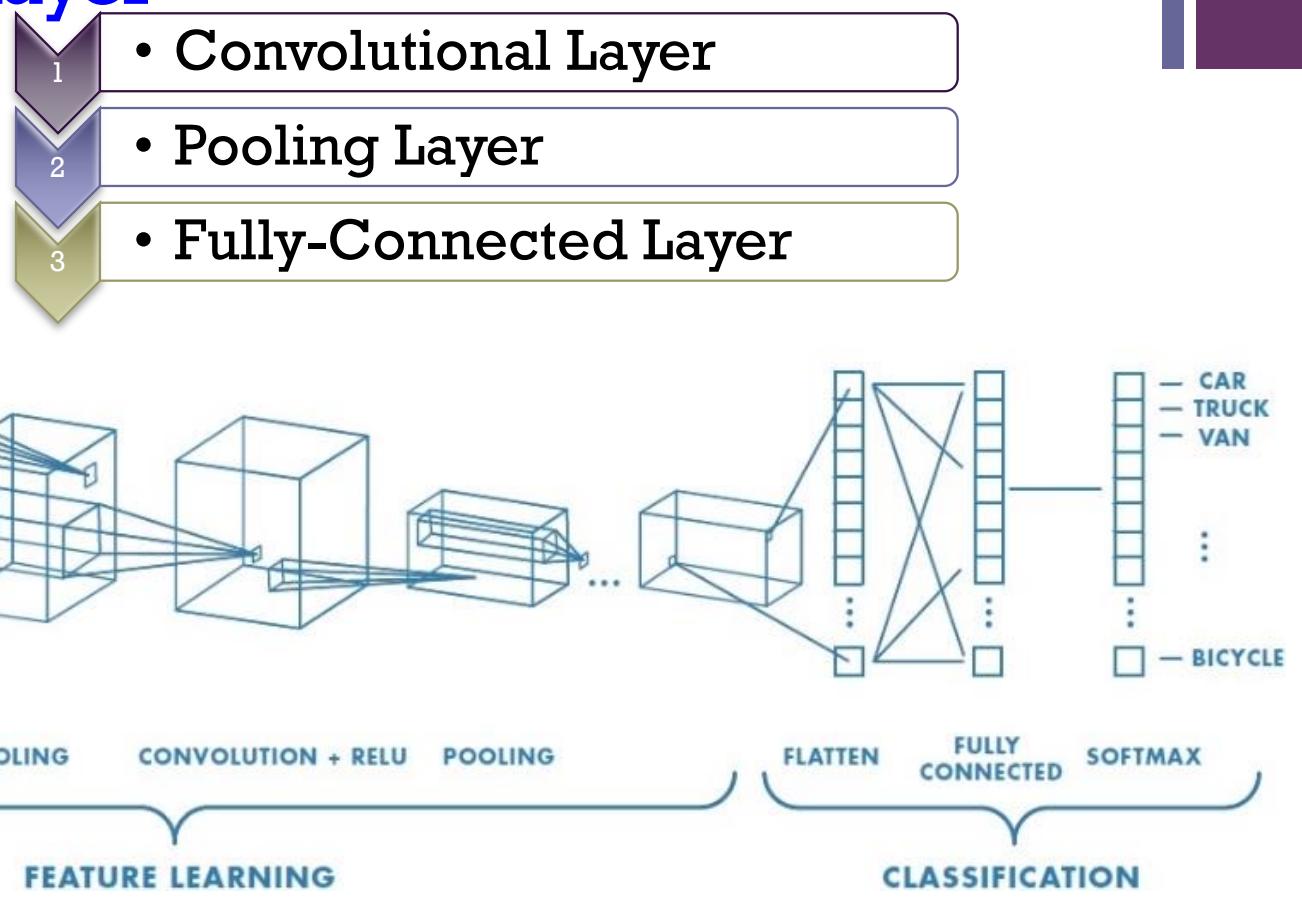


Changing the kernel (filter) results in different image processing outcomes



Convolutional Neural Networks (CNN)

1) Convolutional Layer





Layer1: Convolutional Layer

1 <small>x1</small>	1 <small>x0</small>	1 <small>x1</small>	0	0
0 <small>x0</small>	1 <small>x1</small>	1 <small>x0</small>	1	0
0 <small>x1</small>	0 <small>x0</small>	1 <small>x1</small>	1	1
0	0	1	1	0
0	1	1	0	0

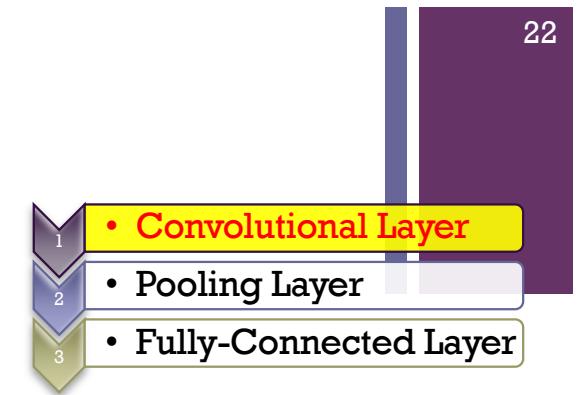
Image

(3*3 filter)

1	0	1
0	1	0
1	0	1

4		

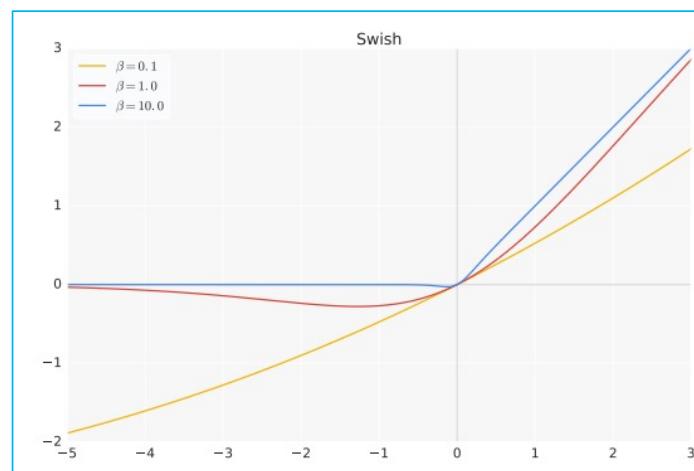
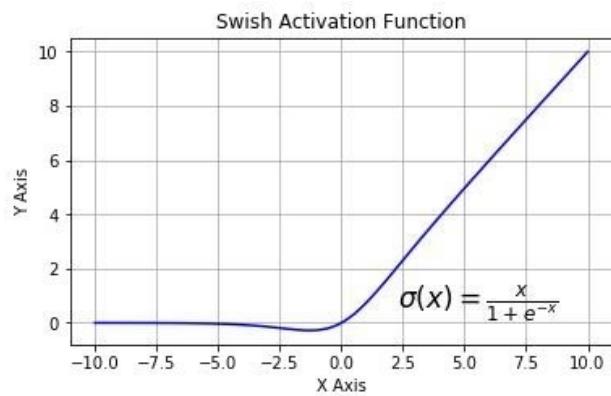
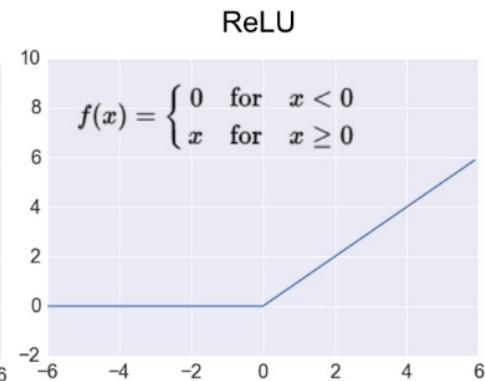
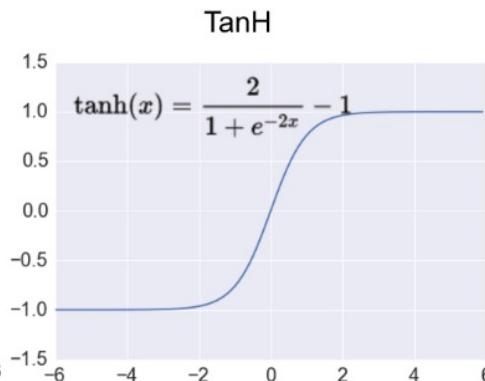
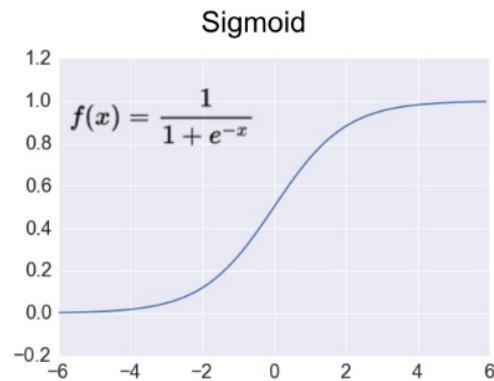
Convolved Feature





CNNs: 1) Convolutional Layer (cont.)

Non-Linear Activation Function

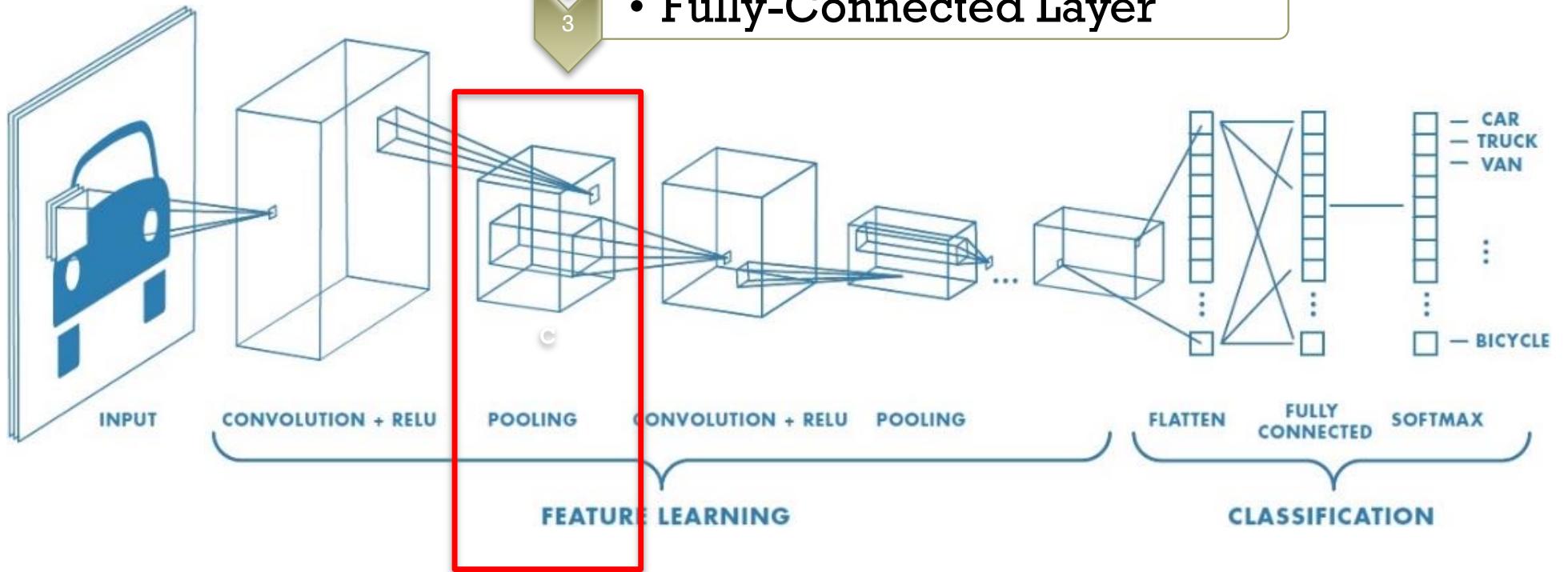




Convolutional Neural Networks (CNN)

2) Pooling Layer

- Convolutional Layer
- Pooling Layer
- Fully-Connected Layer



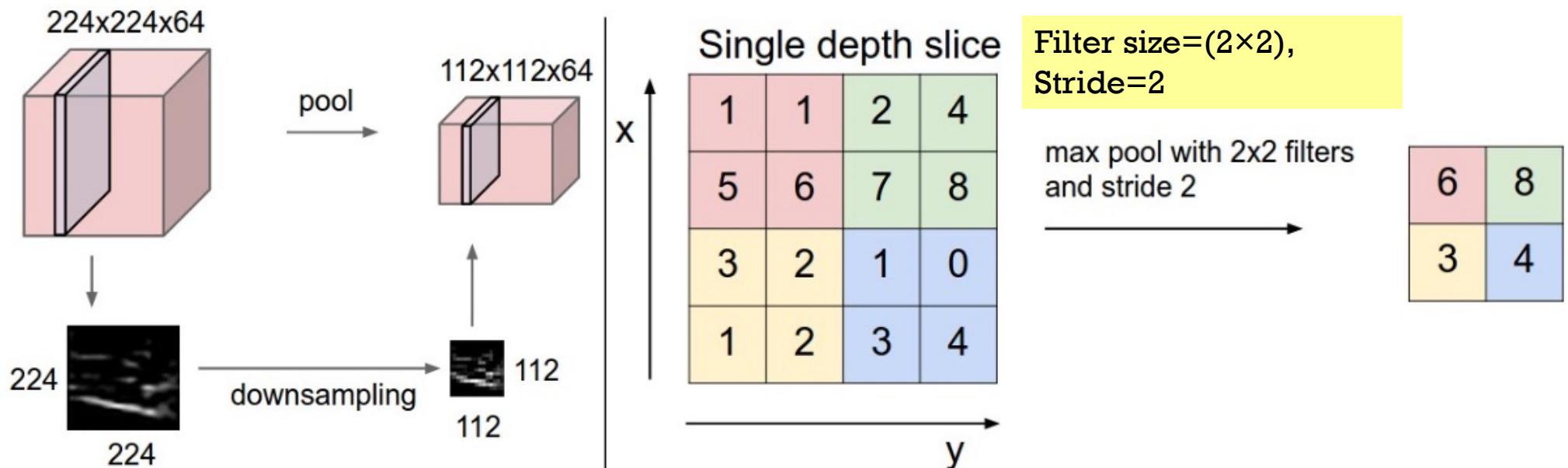
CNNs: 2) Pooling Layer

<http://cs231n.github.io/convolutional-networks/>

Summary: **2 parameters** for pooling layer

- Filter size, Stride

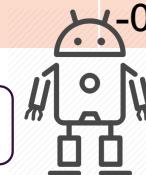
- Feature selection (reduction); downsampling



Convolutional Neural Networks

Embedding Vector

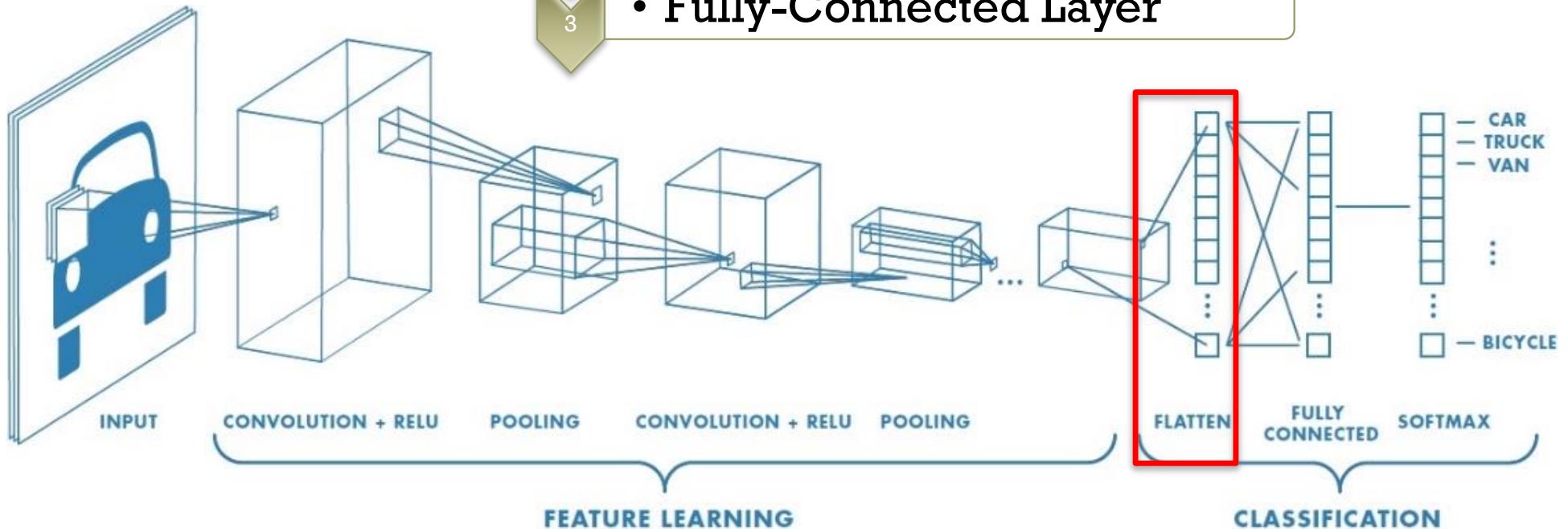
x1	x2	x3	x4	Corona
0.7	0.2	-0.5	-0.1	Yes



- Convolutional Layer

- Pooling Layer

- Fully-Connected Layer

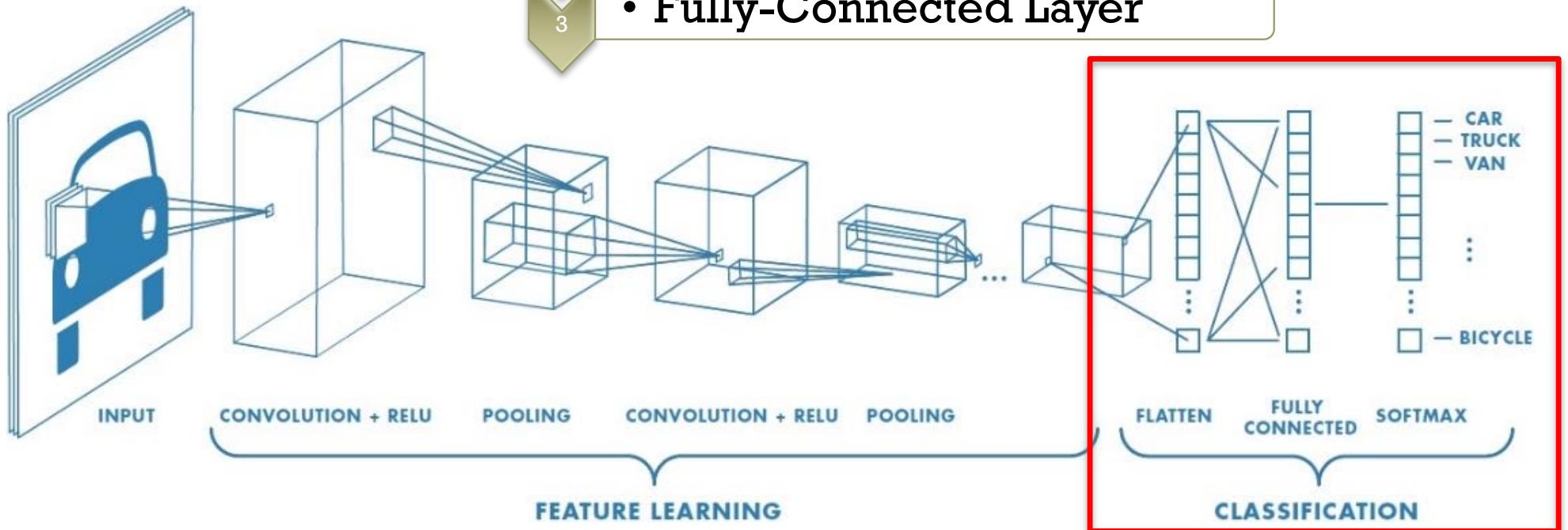




Convolutional Neural Networks (CNN)

3) Fully-Connected Layer (FC) = Neural Networks

- Convolutional Layer
- Pooling Layer
- Fully-Connected Layer



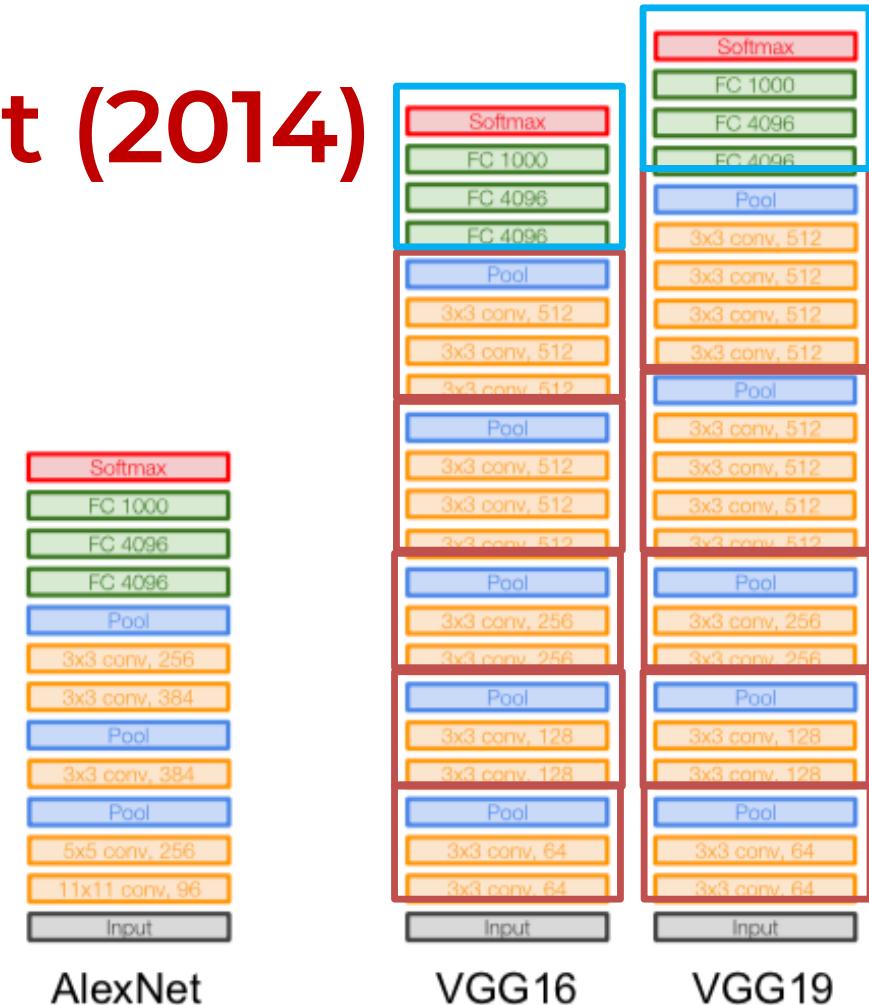
CNN Example: VGGNet (2014)

[Simonyan and Zisserman, 2014]

Small filters, Deeper networks

8 layers (AlexNet) → 16-19 layers (VGG16Net)

Only 3x3 CONV Stride 1, pad 1
And 2x2 MAX POOL stride 2



Network	A	B	C	D	E
Number of parameters (Millions)	133	133	134	138	144



Install Learn API Resources Community Why TensorFlow

Overview
Input
Model
Sequential
activations
applications
Overview
DenseNet121
DenseNet169
DenseNet201
EfficientNetB0
EfficientNetB1
EfficientNetB2
EfficientNetB3
EfficientNetB4
EfficientNetB5
EfficientNetB6
EfficientNetB7
InceptionResNetV2
InceptionV3
MobileNet
MobileNetV2
MobileNetV3Large ↗
MobileNetV3Small ↗
NASNetLarge
NASNetMobile

Modules

`densenet` module: DenseNet models for Keras.

`efficientnet` module: EfficientNet models for Keras.

`imagenet_utils` module: Utilities for ImageNet data preprocessing & prediction decoding.

`inception_resnet_v2` module: Inception-ResNet V2 model for Keras.

`inception_v3` module: Inception V3 model for Keras.

`mobilenet` module: MobileNet v1 models for Keras.

`mobilenet_v2` module: MobileNet v2 models for Keras.

`nasnet` module: NASNet-A models for Keras.

`resnet` module: ResNet models for Keras.

`resnet50` module: Public API for `tf.keras.applications.resnet50` namespace.

`resnet_v2` module: ResNet v2 models for Keras.

`vgg16` module: VGG16 model for Keras.

`vgg19` module: VGG19 model for Keras.

Model 1

- VGG19 (random initialized weights) + 2 Dense layers + Output layer

```
1 base_model = VGG19(weights=None, include_top=False, input_shape=(224, 224, 3))
2
3 for layer in base_model.layers:
4     layer.trainable = True
5
6 x = base_model.output
7 x = Flatten()(x)
8 x = Dense(1024)(x)
9 x = Dropout(0.5)(x)
10 x = Dense(512)(x)
11 x = Dropout(0.5)(x)
12 output = Dense(num_class, activation='softmax')(x)
13
```

TORCHVISION.MODELS

The models subpackage contains definitions of models for addressing different tasks, including: image classification, pixelwise semantic segmentation, object detection, instance segmentation, person keypoint detection and video classification.

Classification

The models subpackage contains definitions for the following model architectures for image classification:

- [AlexNet](#)
- [VGG](#)
- [ResNet](#)
- [SqueezeNet](#)
- [DenseNet](#)
- [Inception v3](#)
- [GoogLeNet](#)
- [ShuffleNet v2](#)
- [MobileNetV2](#)
- [MobileNetV3](#)
- [ResNeXt](#)
- [Wide ResNet](#)
- [MNASNet](#)

We provide pre-trained models, using the PyTorch `torch.utils.model_zoo`. These can be constructed by passing `pretrained=True`:

```
import torchvision.models as models
resnet18 = models.resnet18(pretrained=True)
alexnet = models.alexnet(pretrained=True)
squeezenet = models.squeeze1_0(pretrained=True)
vgg16 = models.vgg16(pretrained=True)
densenet = models.densenet161(pretrained=True)
inception = models.inception_v3(pretrained=True)
googlenet = models.googlenet(pretrained=True)
shufflenet = models.shufflenet_v2_x1_0(pretrained=True)
mobilenet_v2 = models.mobilenet_v2(pretrained=True)
mobilenet_v3_large = models.mobilenet_v3_large(pretrained=True)
mobilenet_v3_small = models.mobilenet_v3_small(pretrained=True)
resnext50_32x4d = models.resnext50_32x4d(pretrained=True)
wide_resnet50_2 = models.wide_resnet50_2(pretrained=True)
mnasnet = models.mnasnet1_0(pretrained=True)
```



Google Teachable Machine



31

- **Teachable Machine** is a web-based tool from Google that allows you to easily create models without writing code or installing software. Simply access it through your web browser at <https://teachablemachine.withgoogle.com/>

New Project

Open an existing project from Drive. Open an existing project from a file.

Image Project
Teach based on images, from files or your webcam.

Audio Project
Teach based on one-second-long sounds, from files or your microphone.

Pose Project
Teach based on images, from files or your webcam.

Teachable Machine

Class 1 edit

Add Image Samples:

Webcam Upload

Class 2 edit

Add Image Samples:

Webcam Upload

Training

Train Model

Advanced

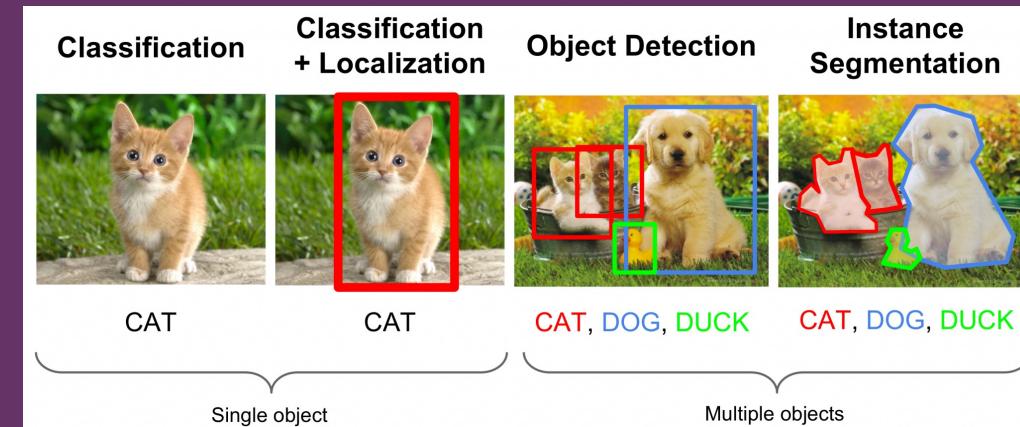
Preview Export Model

You must train a model on the left before you can preview it here.

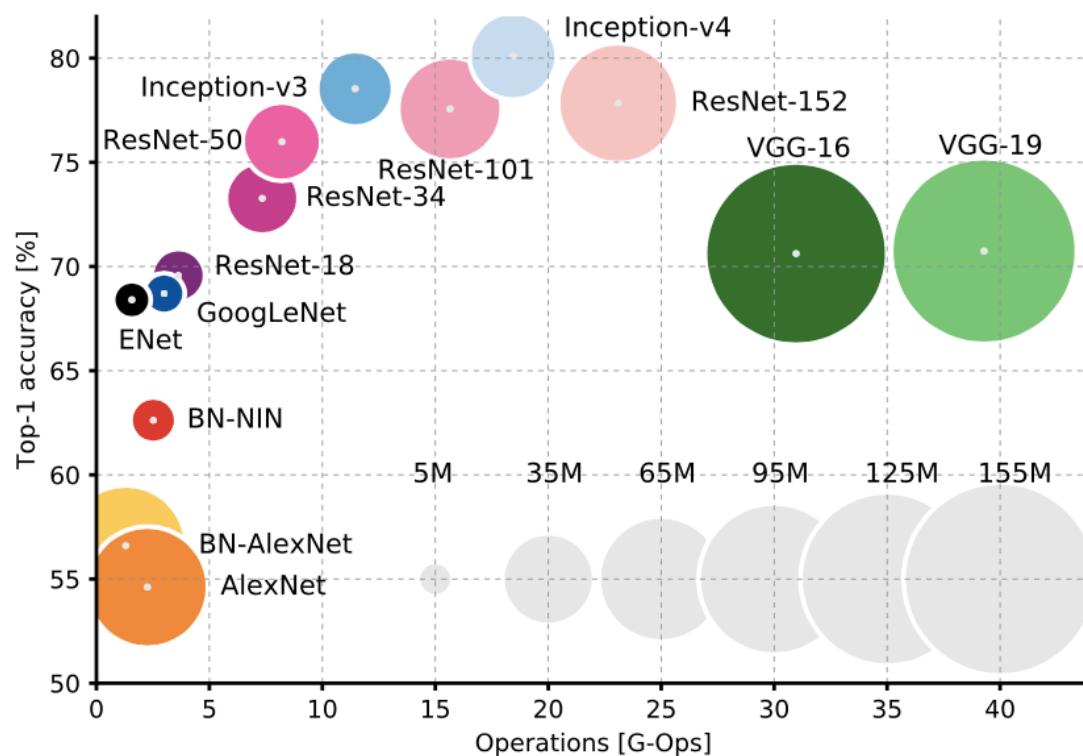
Add a class

31

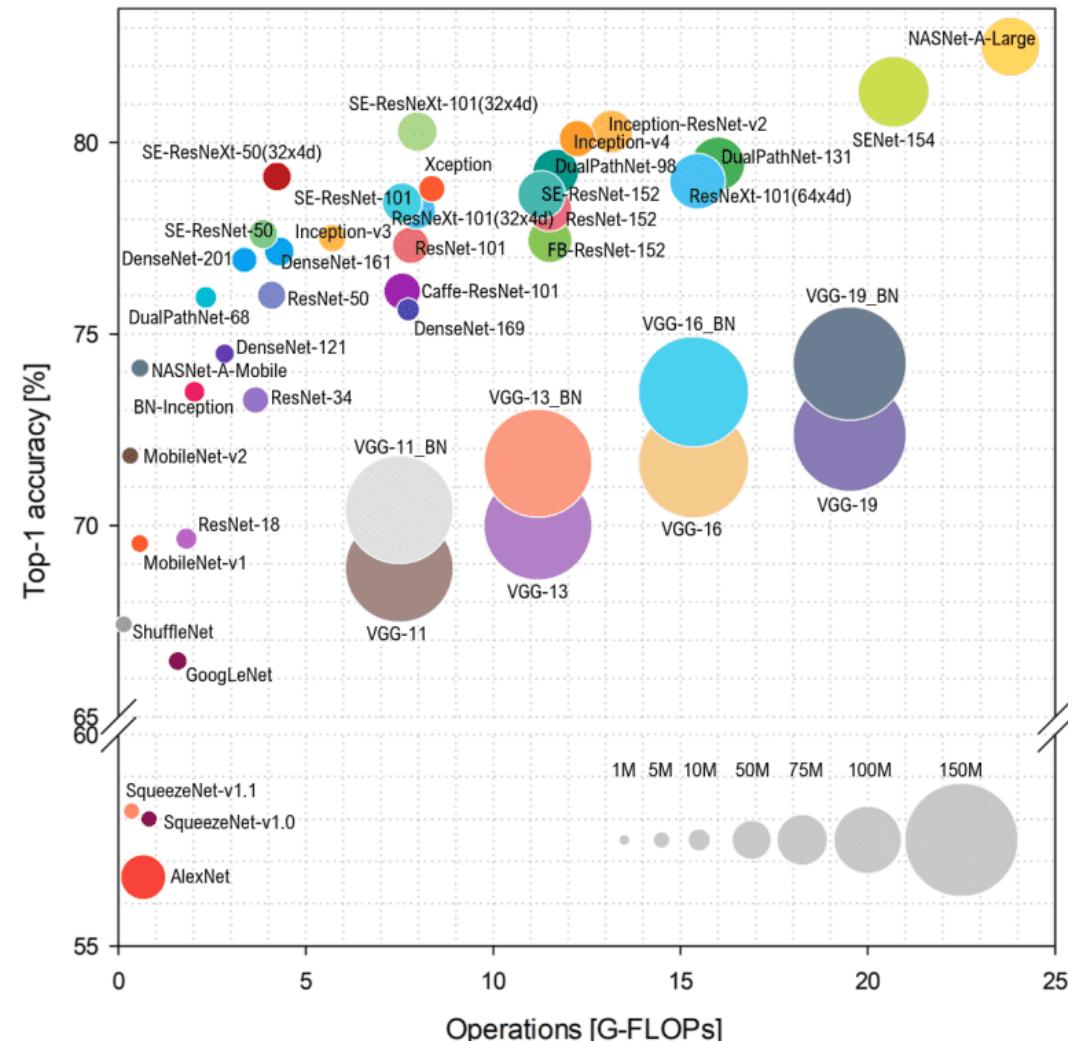
Image Classification



SOTA of Image Classification



https://blog.csdn.net/qq_34216467/article/details/83061692



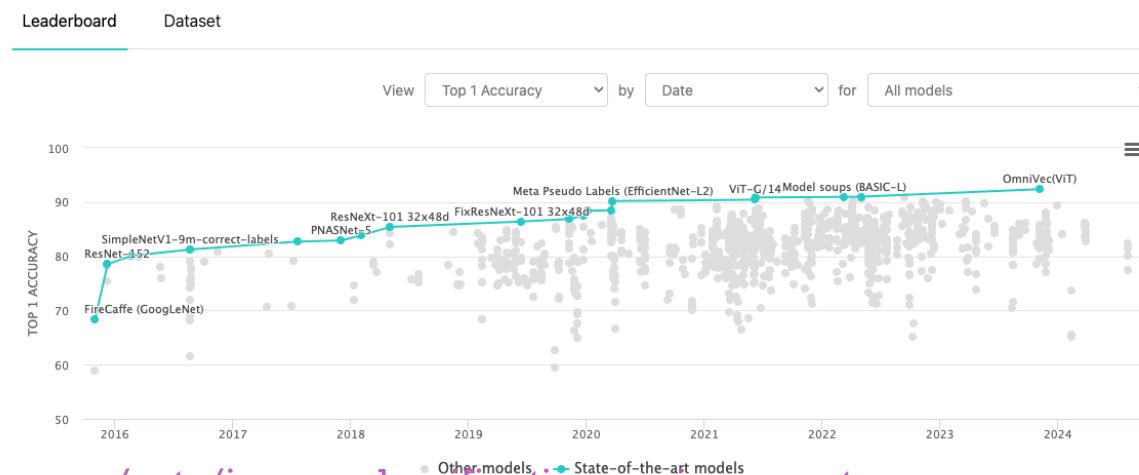
<https://theaisummer.com/cnn-architectures/>



Common Image Classification Models

Model	Year	Team / Organization
ResNet	2015	Microsoft Research
MobileNet	2017	Google
EfficientNet	2019	Google
Vision Transformer (ViT)	2020	Google Research
Swin Transformer	2021	Microsoft Research
ConvNeXt	2022	Meta AI

Image Classification on ImageNet



<https://paperswithcode.com/sota/image-classification-on-imagenet>

EfficientNet (May, 2019) → EffNetV2 (2021)

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

Mingxing Tan¹ Quoc V. Le¹

Abstract

Convolutional Neural Networks (ConvNets) are commonly developed at a fixed resource budget, and then scaled up for better accuracy if more resources are available. In this paper, we systematically study model scaling and identify that carefully balancing network depth, width, and resolution can lead to better performance. Based on this observation, we propose a new scaling method that uniformly scales all dimensions of depth/width/resolution using a simple yet highly effective *compound coefficient*. We demonstrate the effectiveness of this method on scaling up MobileNets and ResNet.

To go even further, we use neural architecture

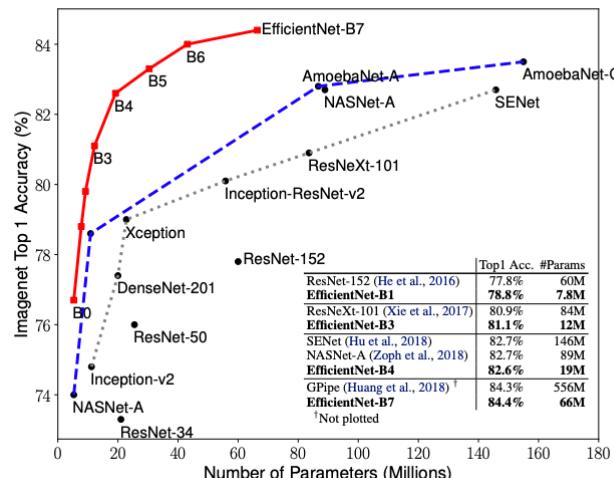
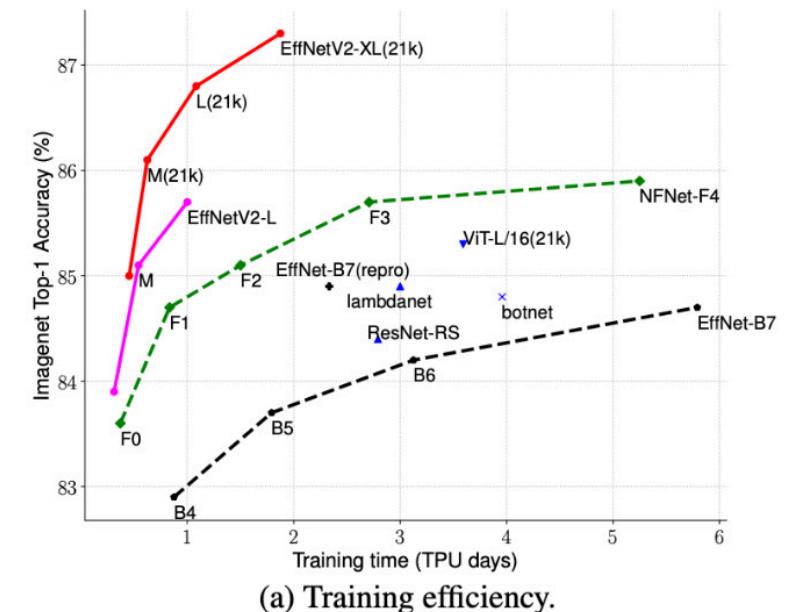


Figure 1. Model Size vs. ImageNet Accuracy. All numbers are



(a) Training efficiency.

	EfficientNet (2019)	ResNet-RS (2021)	DeiT/ViT (2021)	EfficientNetV2 (ours)
Top-1 Acc.	84.3%	84.0%	83.1%	83.9%
Parameters	43M	164M	86M	24M

(b) Parameter efficiency.

<http://proceedings.mlr.press/v97/tan19a/tan19a.pdf>

+

Thank you & any questions