UVA CS 4774: Machine Learning

Lecture 13: Supervised Image Classification and Convolutional Neural Networks

Dr. Yanjun Qi

University of Virginia

Department of Computer Science

Last: Basic Neural Network Models

Data: X

Task: y

Representation: : x, f()

Score Function: L()

Search/Optimization : argmin()

Models, Parameters, metrics

X: Tabular / 2D / 1D / 3D / Graph /…

Classification / Regression

Multilayer Network topology

neg Log-likelihood , Cross-Entropy / MSE / Many more

SGD / Backprop

Weights and biases in NN Layers / Accuracy / F1

Early History

- In 1950 English mathematician Alan Turing wrote a landmark paper titled "Computing Machinery and Intelligence" that asked the question: "Can machines think?"
- Further work came out of a 1956 workshop at Dartmouth sponsored by John McCarthy. In the proposal for that workshop, he coined the phrase a "study of artificial intelligence"
- 1950s
  - Samuel's checker player : start of machine learning
  - Selfridge's Pandemonium
- 1952-1969: Enthusiasm: Lots of work on neural networks
- 1970s-80s: Expert systems, Knowledge bases to add on rule-based inference, Decision Trees, Bayes Nets
- 1990s : CNN, RNN, ...
- 2000s : SVM, Kernel machines, Structured learning, Graphical models, semi-supervised, matrix factorization, …

Adapted From Prof. Raymond J. Mooney's slides

“Winter of Neural Networks” in ~2000s

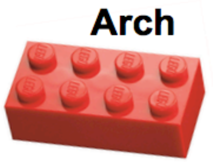
- Non-convex

- Need a lot of tricks to play with
  - How many layers ?
  - How many hidden units per layer ?
  - What topology among layers ? …….

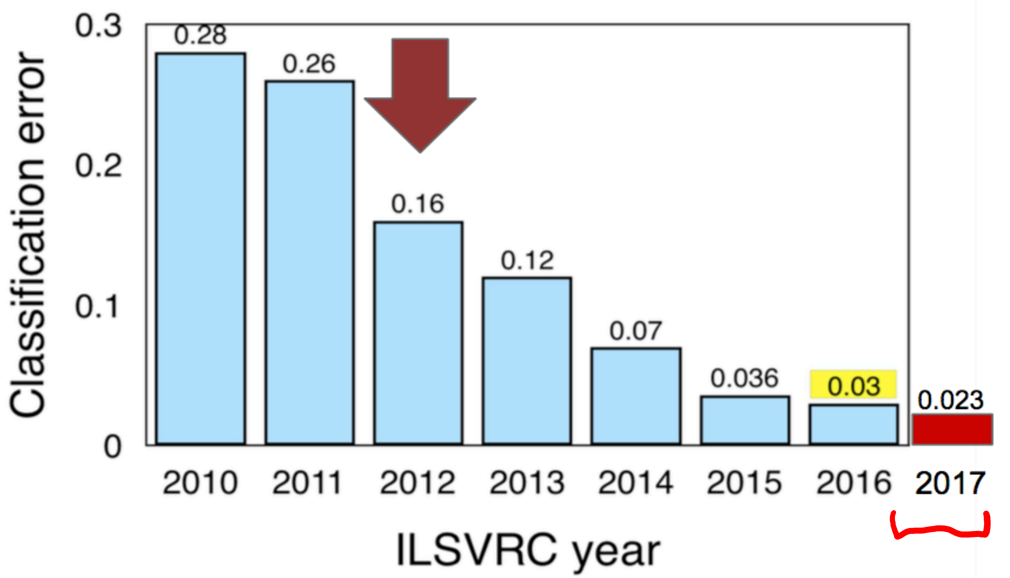
- Hard to perform theoretical analysis

- Large labeled datasets were rare in ~2000s

ImageNet Challenge



- 2010-11: hand-crafted computer vision pipelines
- 2012-2016: ConvNets
  - 2012: AlexNet
    - major deep learning success
  - 2013: ZFNet
    - improvements over AlexNet
  - 2014
    - VGGNet: deeper, simpler
    - InceptionNet: deeper, faster
  - 2015
    - ResNet: even deeper
  - 2016
    - ensembled networks
  - 2017
    - Squeeze and Excitation Network



Adapt from From NIPS 2017 DL Trend Tutorial

Today: Convolutional Network Models on 2D Grid / Image

Data: X

Task: y

Representation: : x, f()

Score Function: L()

Search/Optimization : argmin()

Models, Parameters, metrics

X: 2D Grid Imaging

Classification

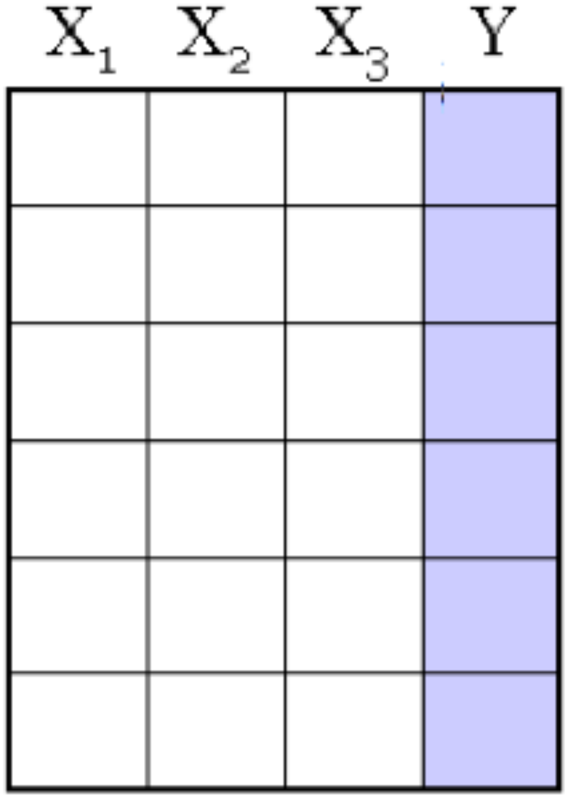
Convolutional Neural Networks

Cross Entropy Loss

SGD / Backprop

weights, bias / architecture /

/ Accuracy / F1



Tabular Dataset for classification

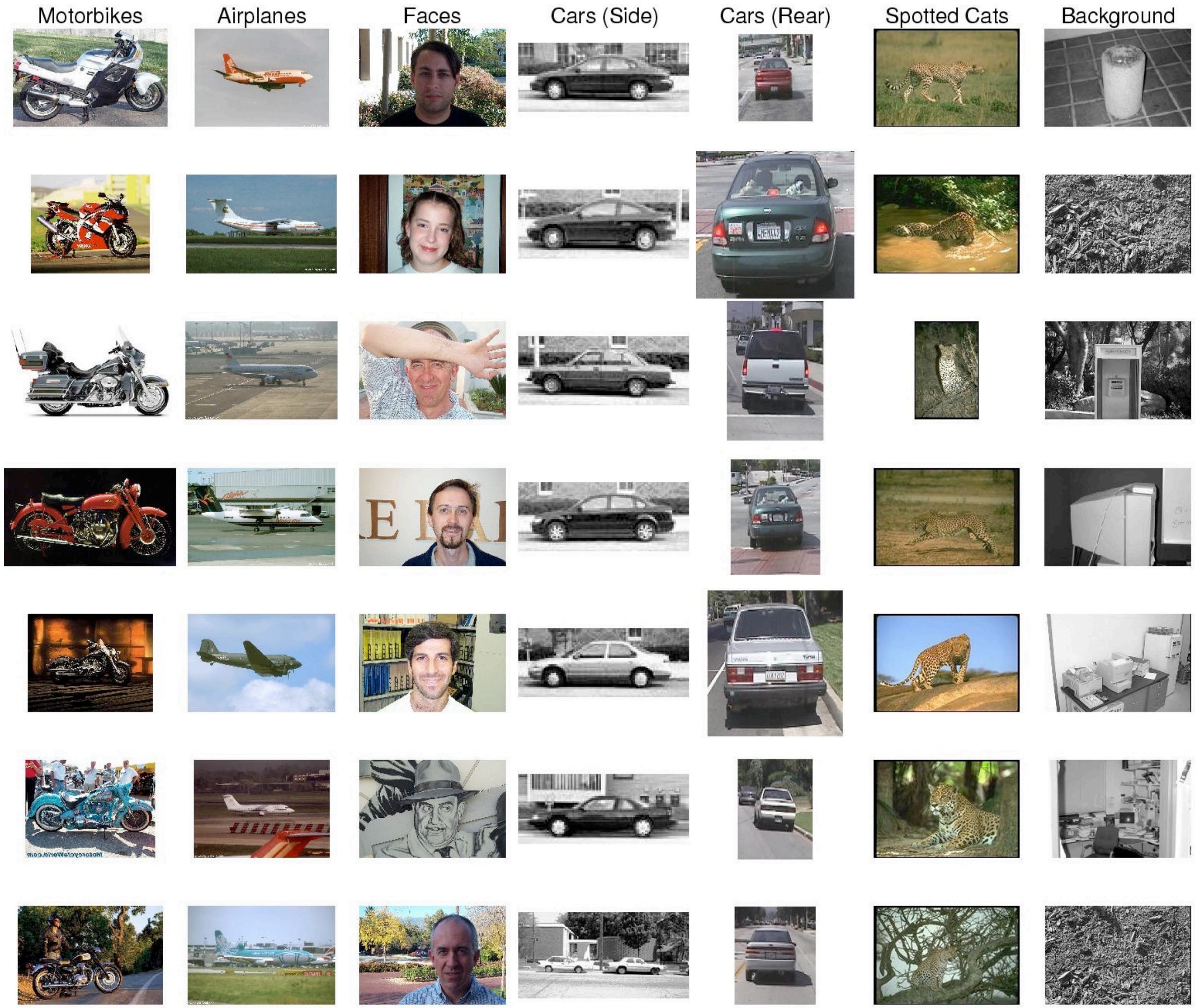
$$f : \boxed{X} \longrightarrow \boxed{Y}$$

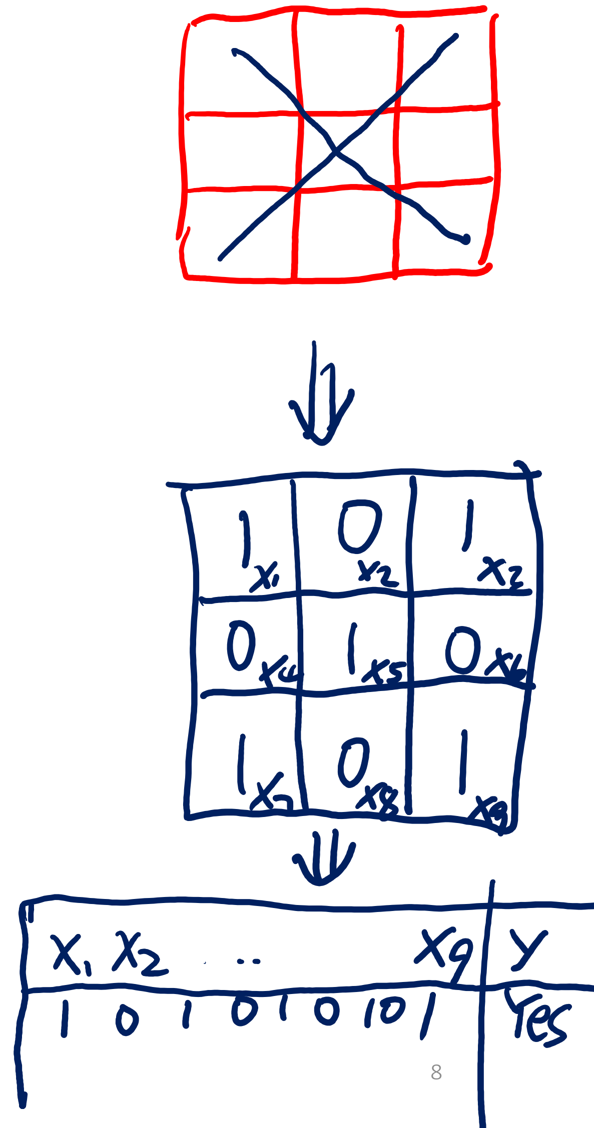
Output Class: categorical variable

- **Data**/*points/instances/examples/samples/records*: [rows]
- **Features**/*attributes/dimensions/independent variables/covariates/predictors/regressors*: [columns, except the last]
- **Target**/*outcome/response/label/dependent variable*: special column to be predicted [last column]

2D Images Dataset for Classification

x





Review: Model Selection and Assessment

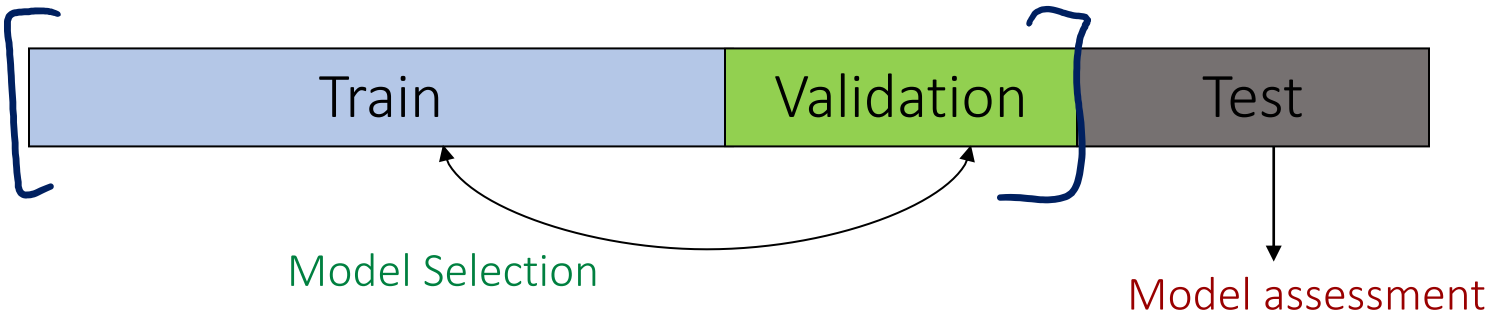
- Model Selection
  - Estimating performances of different models to choose the best one

- Model Assessment
  - Having chosen a model, estimating the prediction error on new data

testing

Model Selection and Assessment

- When Data Rich Scenario: Split the dataset



- When Insufficient data to split into 3 parts
  - Approximate validation step analytically
    - AIC, BIC, MDL, SRM
  - Efficient reuse of samples
    - Cross validation, bootstrap

Model Selection (Hyperparameter Tuning) & Model Assessment Pipelines in HW2

- (1) train / Validation / test →

- (2) k-CV on train to choose hyperparameter / then test → more expensive

Binary Class

$\{T, F\}$

		actual (y)	
		AP $+$	AN $-$
$\hat{y}$	PP	predicted$+$	TP	FP
	PN	predicted$-$	FN	TN

Confusion matrix

- (number of) true positive (TP)
- (number of) true negative (TN)
- (number of) false positive (FP)
- (number of) false negative (FN)

$\hat{y}$	y	
$+$	$+$	TP++
$+$	$-$	

$\hat{y}_1$ vs y_1

$\hat{y}_2$ y_2

$\vdots$

$\hat{y}_m$ vs y_m

Metric	**Formula**	**Interpretation**
Accuracy	$\dfrac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}}$	Overall performance of model
Precision	$\dfrac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$	How accurate the positive predictions are
Recall Sensitivity	$\dfrac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$	Coverage of actual positive sample
Specificity	$\dfrac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}}$	Coverage of actual negative sample
F1 score	$\dfrac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}$	Hybrid metric useful for unbalanced classes

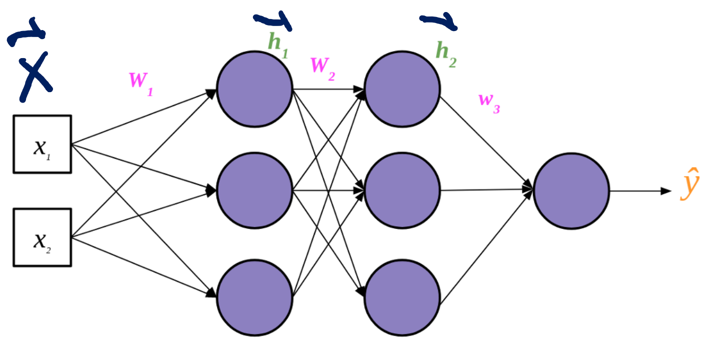
Credit: Stanford Machine Learning course

Deep Learning Frameworks

Chainer
mxnet
Caffe2
Microsoft CNTK
TensorFlow
Keras
GLUON
PYTORCH
theano

fast.ai
Making neural nets uncool again

Pytorch Sample Code




```
import torch.nn as nn
import torch.nn.functional as F

class ThreeLayerNet(torch.nn.Module):
    def __init__(self, d_in, d_hidden, d_out):
        super().__init__()
        self.W1 = nn.Linear(d_in,d_hidden)
        self.W2 = nn.Linear(d_hidden,d_hidden)
        self.w3 = nn.Linear(d_hidden,d_out)
        self.nonlinear = nn.Sigmoid()

    def forward(self, x):
        h1 = self.nonlinear(self.W1(x))
        h2 = self.nonlinear(self.W2(h1))
        y_hat = self.nonlinear(self.w3(h2))
        return y_hat

model = ThreeLayerNet(2,3,1)
```

Demo: Use FastAI and Keras to Classify **CT scans for SARS-CoV-2 (COVID-19) identification**

I will code-run (using FastAI and CNN ResNet34):
https://colab.research.google.com/drive/1mvj9ZB0o-Q49Xq9vYJW4GB09V41u5GeE?usp=sharing

```
#fastai automatically factors the ./train and ./valid folders into seperate datasets
#more details https://docs.fast.ai/vision.data.html#ImageDataLoaders.from_folder
#path = Path('/content/drive/My Drive/Images/SARS-COV-2-Ct-Scan/')
data = ImageDataBunch.from_folder('/content/drive/My Drive/Images/SARS-COV-2-Ct-Scan/', valid_pct=0.2, size=224, num_workers=4, bs=32)
# data = ImageDataBunch.from_folder(path, ds_tfms=get_transforms(do_flip=True, flip_vert=True),
 #                                  valid_pct=0.2, size=size, bs=bs)

#double check the data classes
data.classes
#take a peak at the batch to make sure things were loaded correctly
data.normalize(imagenet_stats)
data.show_batch(rows=5, figsize=(7, 7))
data.show_batch(rows=5, figsize=(7, 7))
```

Another Code using Keras on the same dataset (Using DenseNet121):
https://www.kaggle.com/shawon10/covid-19-diagnosis-from-images-using-densenet121

```
train_data = []
for defects_id, sp in enumerate(disease_types):
    for file in os.listdir(os.path.join(train_dir, sp)):
        train_data.append(['{}/{}'.format(sp, file), defects_id, sp])

train = pd.DataFrame(train_data, columns=['File', 'DiseaseID','Disease T
ype'])
train.head()
```

UVA CS 4774: Machine Learning

Lecture 13: Supervised Image Classification and Convolutional Neural Networks

Dr. Yanjun Qi

University of Virginia

Department of Computer Science

Module II

Today: Convolutional Network Models on 2D Grid / Image

Data: X

↓

Task: y

↓

Representation: :   x, f()

↓

Score Function: L()

↓

Search/Optimization : argmin()

↓

Models, Parameters, metrics

→

X: 2D Grid Imaging

↓

Classification

↓

Convolutional Neural Networks

↓

Cross Entropy Loss

↓

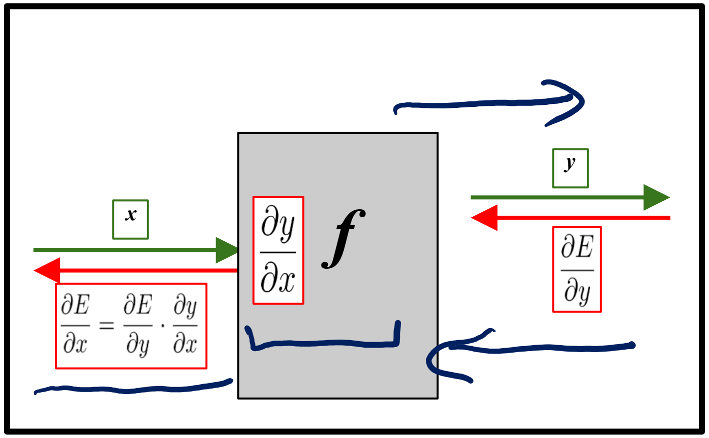
SGD / Backprop

↓

weights, bias / architecture /

/ Accuracy / F1

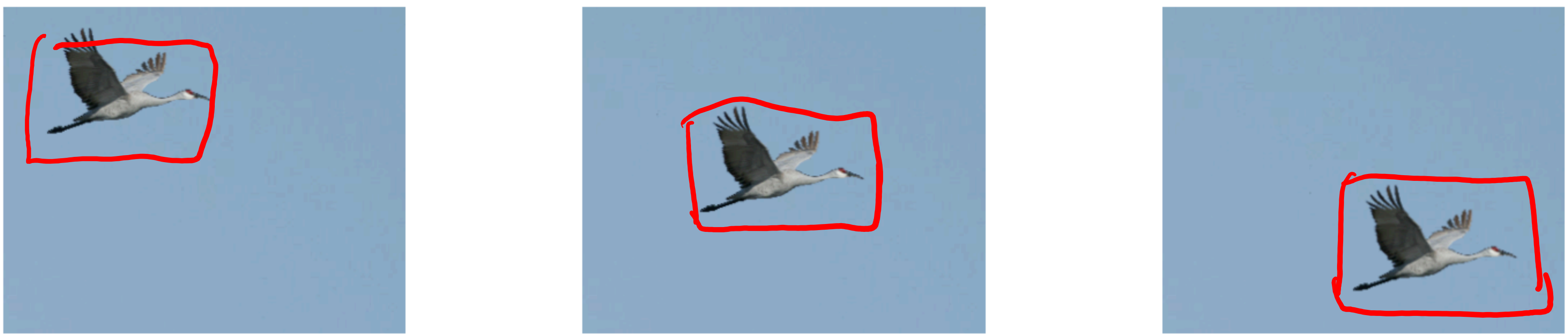
Building Deep Neural Nets



http://cs231n.stanford.edu/slides/winter1516_lecture5.pdf

Important Block: Convolutional Neural Networks (CNN)

- Prof. Yann LeCun invented CNN in 1998
- First NN successfully trained with many layers



The bird occupies a local area and looks the same in different parts of an image.
We should construct neural nets which exploit these properties!

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86(11): 2278–2324, 1998.

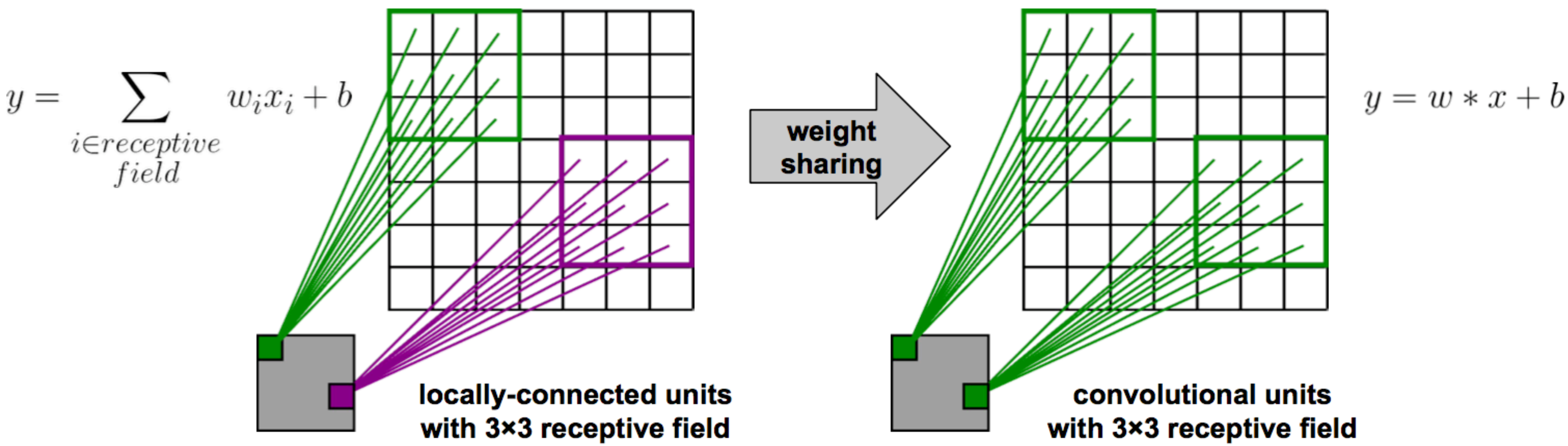
Adapt from From NIPS 2017 DL Trend Tutorial

Locality and Translation Invariance

- **Locality**: objects tend to have a local spatial support
- **Translation invariance**: object appearance is independent of location
- Can define these properties since an image lies on a grid/lattice
  - ConvNet applicable to other data with such properties, e.g. audio/text
  - Lattice: regular spacing or arrangement of geometric points,

CNN models Locality and Translation Invariance

Make fully-connected layer locally-connected and sharing weight

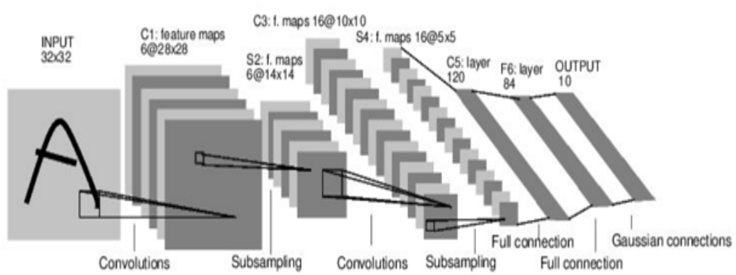


Adapt from From NIPS 2017 DL Trend Tutorial

History of ConvNets

1998

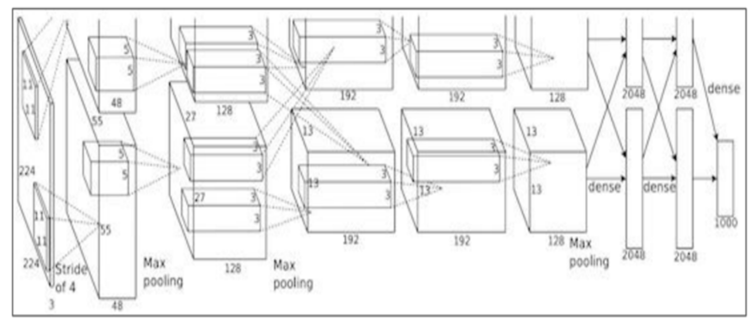
Gradient-based learning applied to document recognition [LeCun, Bottou, Bengio, Haffner]



LeNet-5

2012

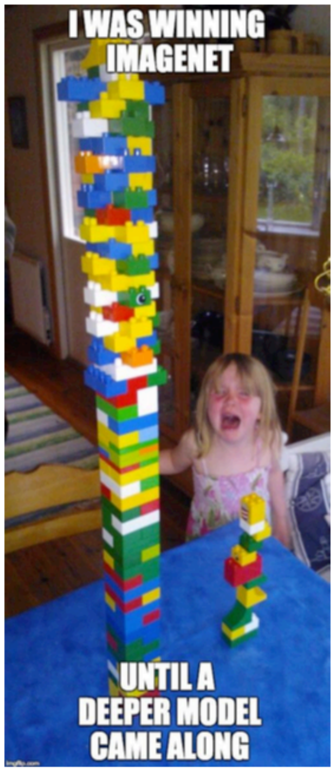
ImageNet Classification with Deep Convolutional Neural Networks [Krizhevsky, Sutskever, Hinton, 2012]

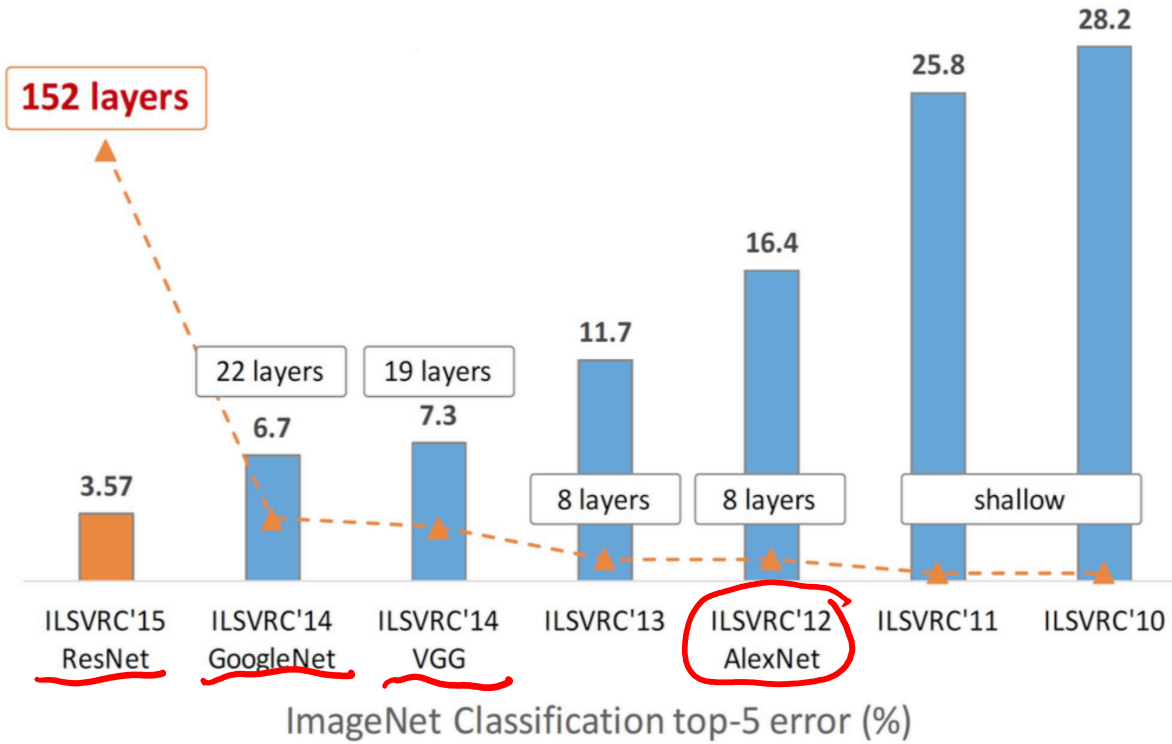


“AlexNet”

Arch

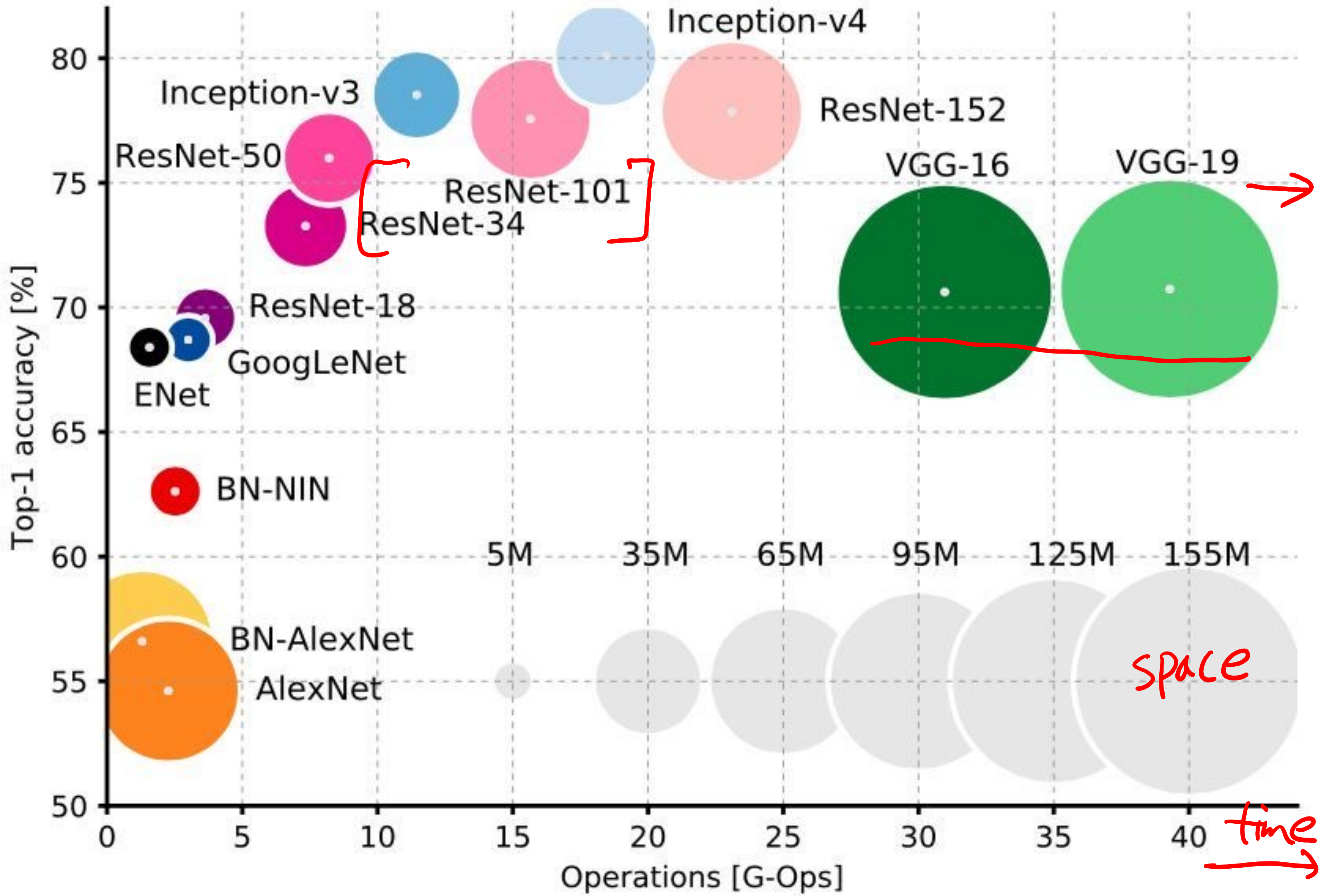
Revolution of Depth





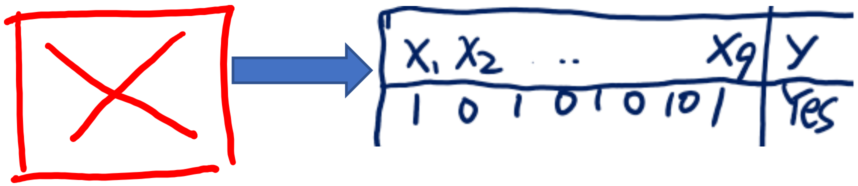
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". CVPR 2016.

Adapt from From NIPS 2017 DL Trend Tutorial

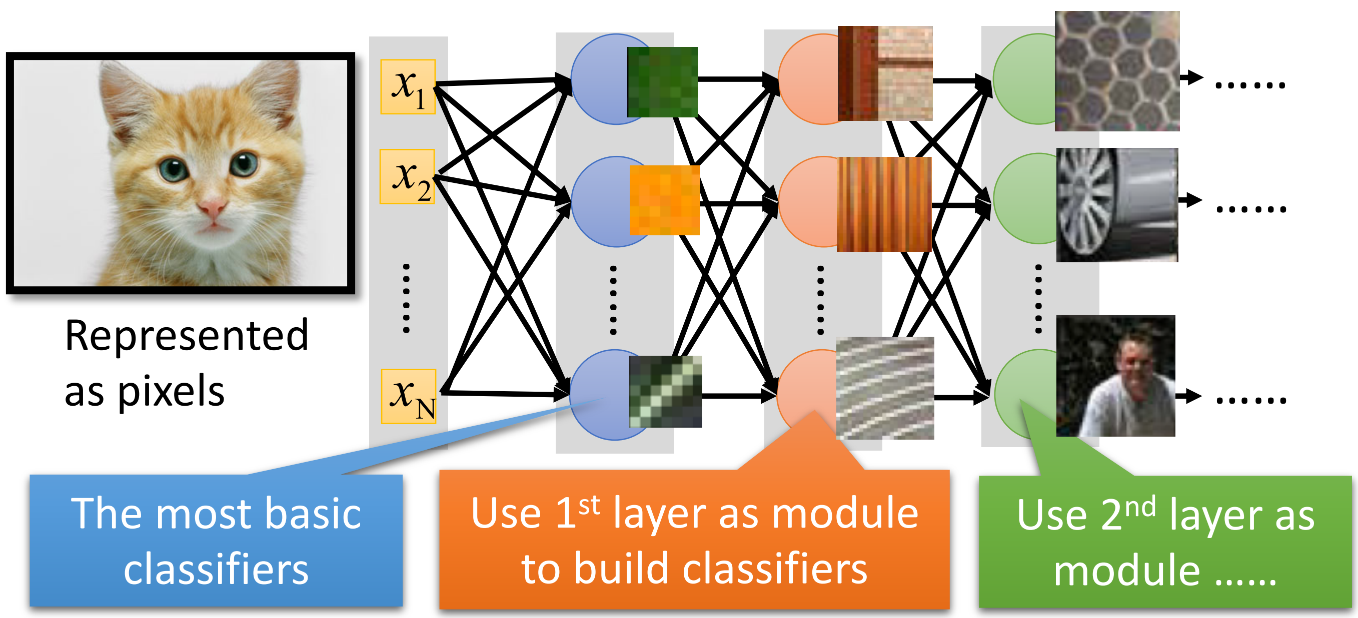


Inception-v4
Inception-v3
ResNet-152
ResNet-50
ResNet-101
ResNet-34
VGG-16
VGG-19
ResNet-18
GoogLeNet
ENet
BN-NIN
5M
35M
65M
95M
125M
155M
BN-AlexNet
AlexNet
space
time
Top-1 accuracy [%]
Operations [G-Ops]

Why CNN for Image?



[Zeiler, M. D., *ECCV 2014*]



Can the MLP network be simplified by considering the properties of images?

Why CNN for Image

- (1) Locality: Some patterns are much smaller than the whole image

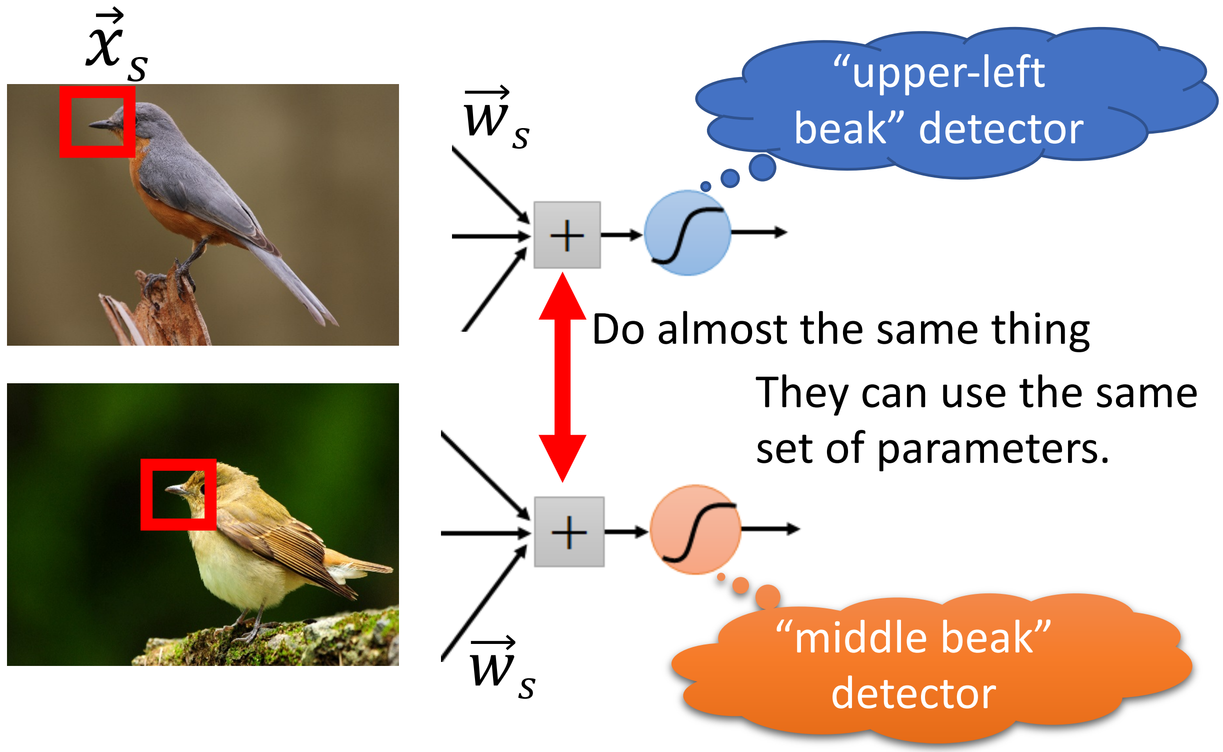
A neuron does not have to see the whole image to discover the pattern.

Connecting to small region with less parameters

Dr. Hung-yi Lee’s CNN slides

Why CNN for Image

- (2) Translation invariance: The same patterns appear in different regions.



Dr. Hung-yi Lee's CNN slides

Why CNN for Image

- (3) Subsampling the pixels will not change the object

bird

subsampling

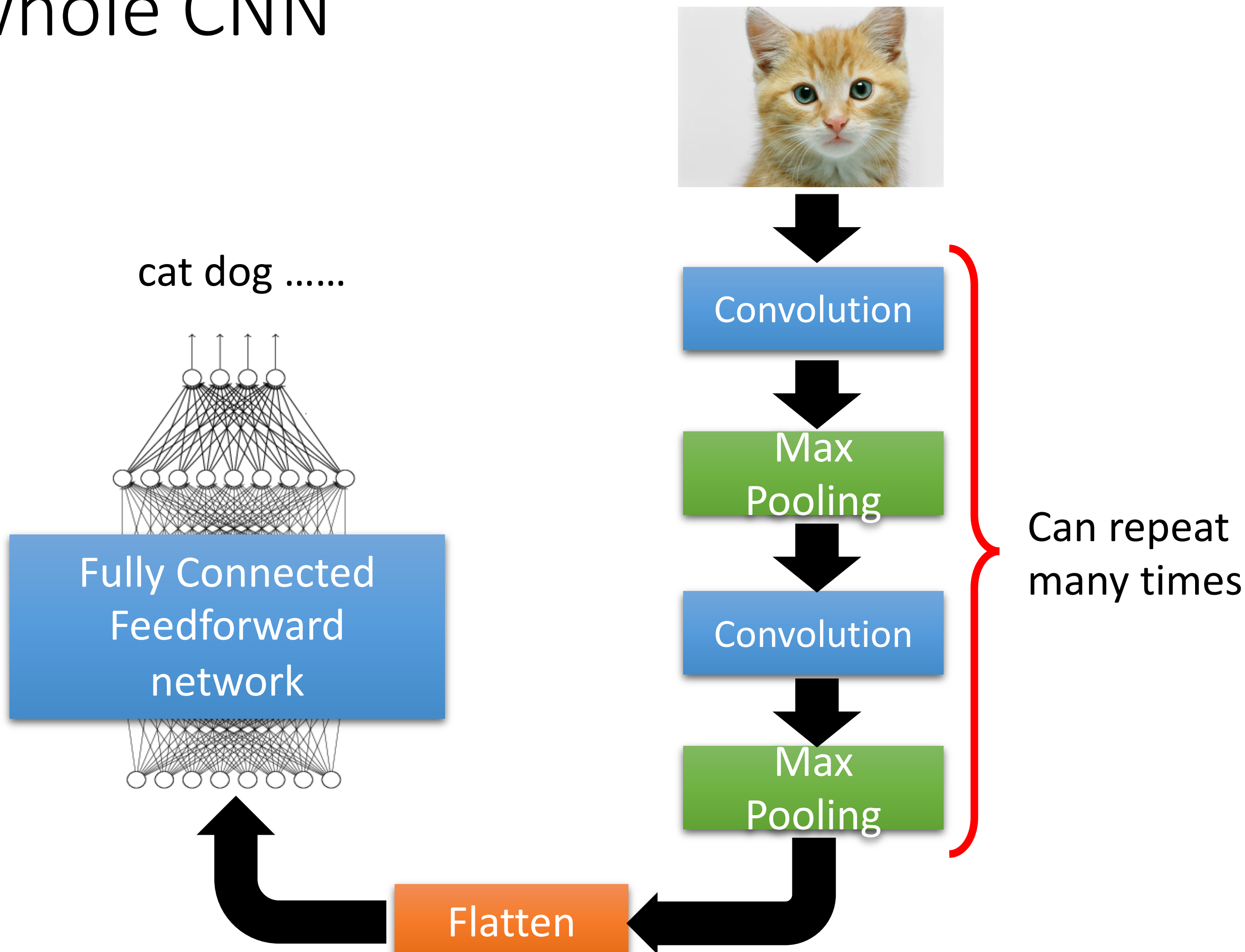
bird

We can subsample the pixels to make image smaller

→ Less parameters for the network to process the image

Dr. Hung-yi Lee’s CNN slides

The whole CNN



Dr. Hung-yi Lee’s CNN slides

The whole CNN

Property 1

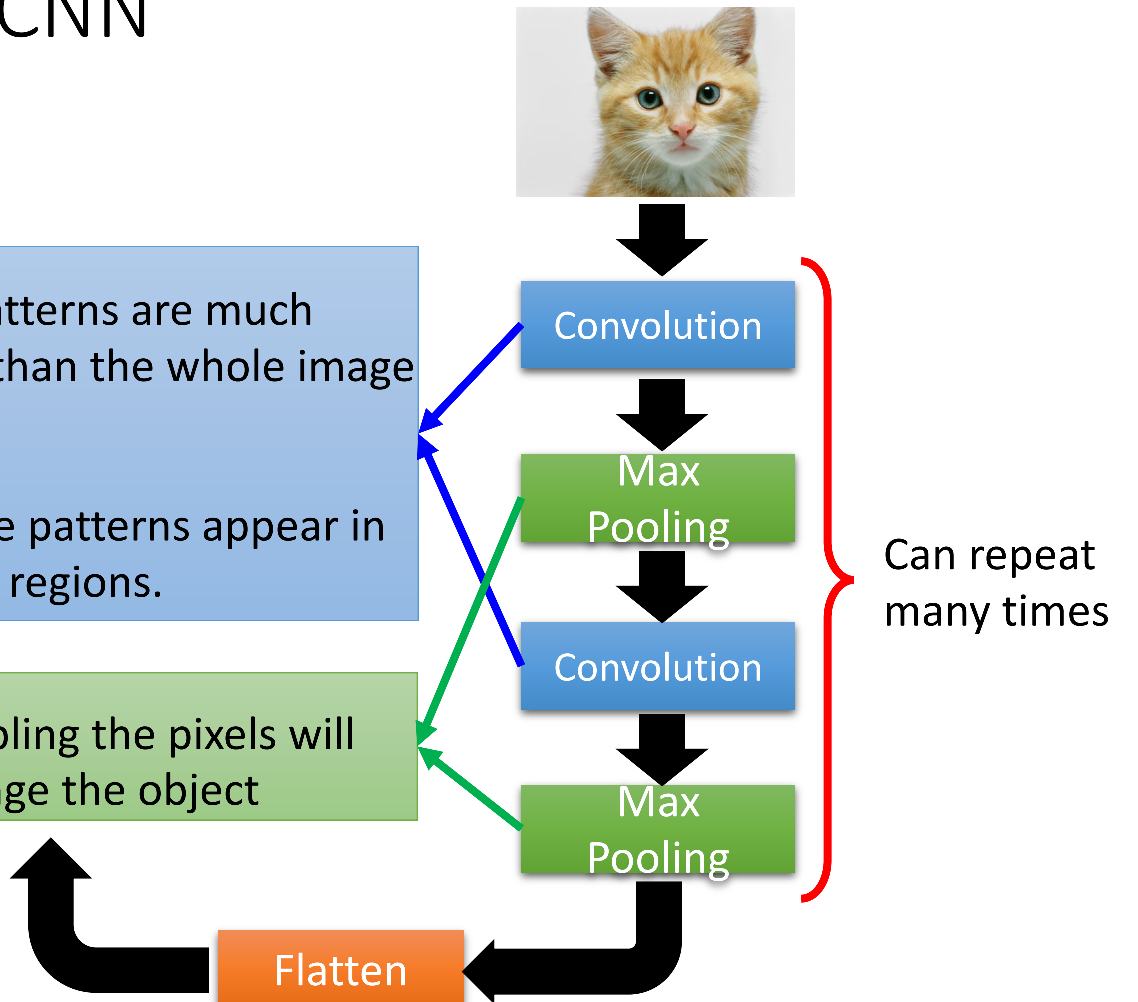
- Some patterns are much smaller than the whole image

Property 2

- The same patterns appear in different regions.

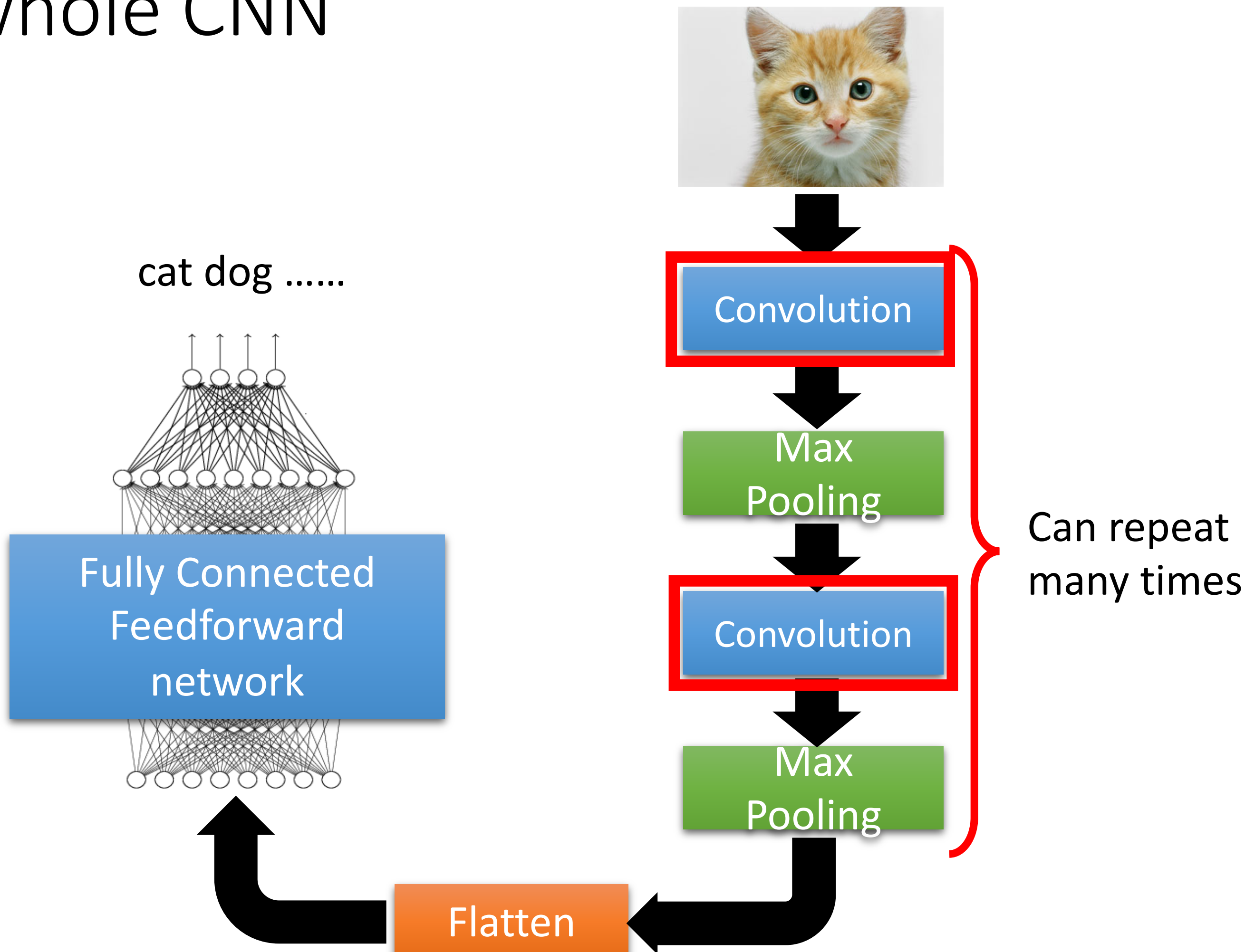
Property 3

- Subsampling the pixels will not change the object



Dr. Hung-yi Lee's CNN slides

The whole CNN



Dr. Hung-yi Lee's CNN slides

CNN – Convolution

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

6 x 6 image

Those are the network parameters to be learned.

"detector 1"

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

Filter 1

Matrix $\vec{w}_{s1}$

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

Filter 2

Matrix $\vec{w}_{s2}$

"detector 2"

⋮ ⋮

Property 1 Each filter detects a small pattern (3 x 3).

Dr. Hung-yi Lee's CNN slides

CNN – Convolution

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

Filter 1

stride=1

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

6 x 6 image

3 -1

Dr. Hung-yi Lee’s CNN slides

CNN – Convolution

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

Filter 1

If stride=2

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

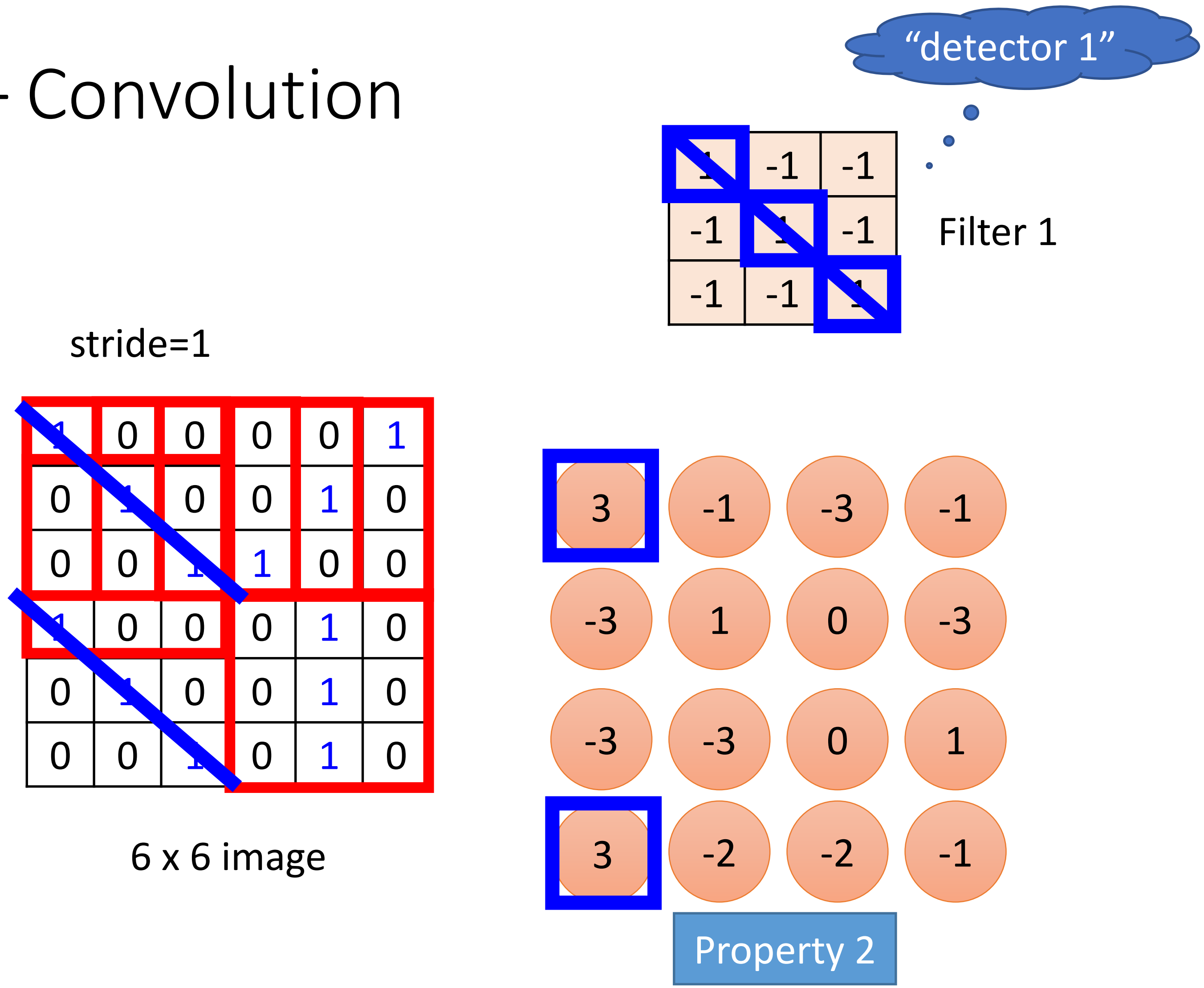
6 x 6 image

3 -3

We set stride=1 below

Dr. Hung-yi Lee's CNN slides

CNN – Convolution



Dr. Hung-yi Lee's CNN slides

CNN – Convolution

“detector 2”

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

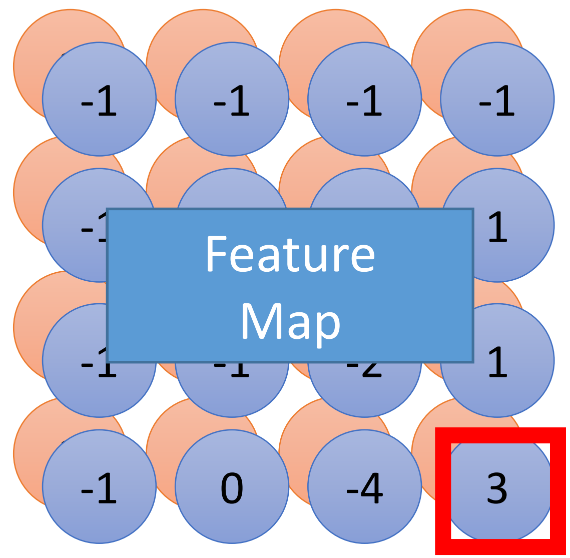
Filter 2

stride=1

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

6 x 6 image

Do the same process for every filter



4 x 4 image

Dr. Hung-yi Lee’s CNN slides

CNN – Convolution

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

Filter 1

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

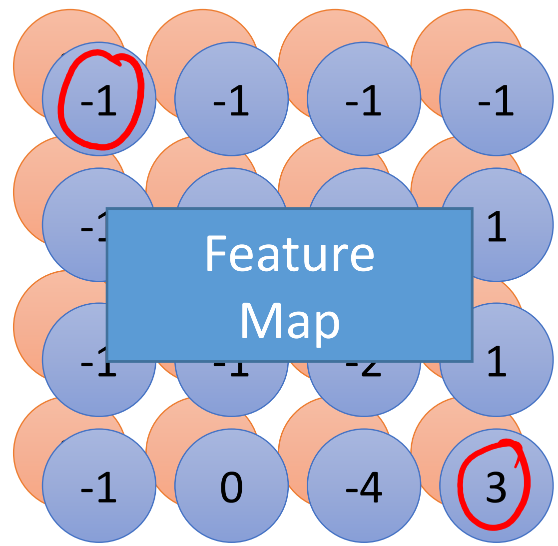
Filter 2

stride=1

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

6 x 6 image

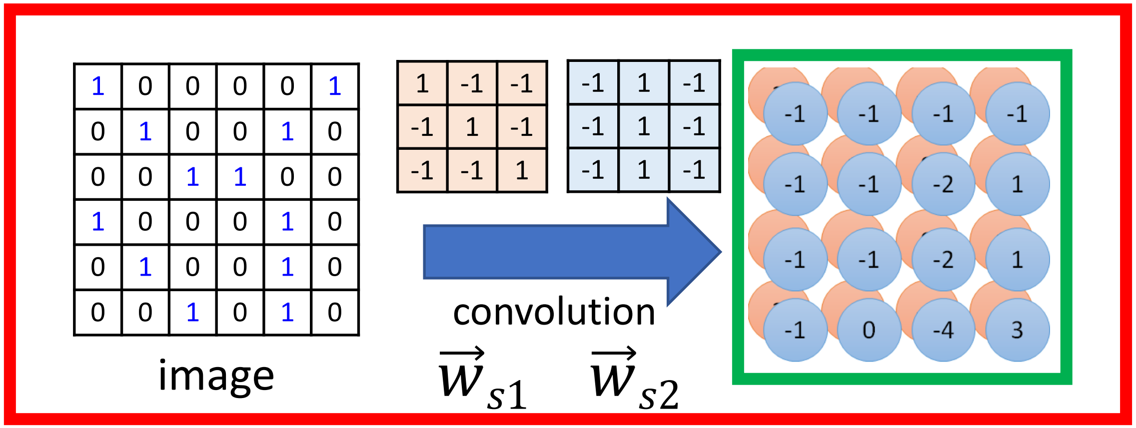
You can do the same process for every filter

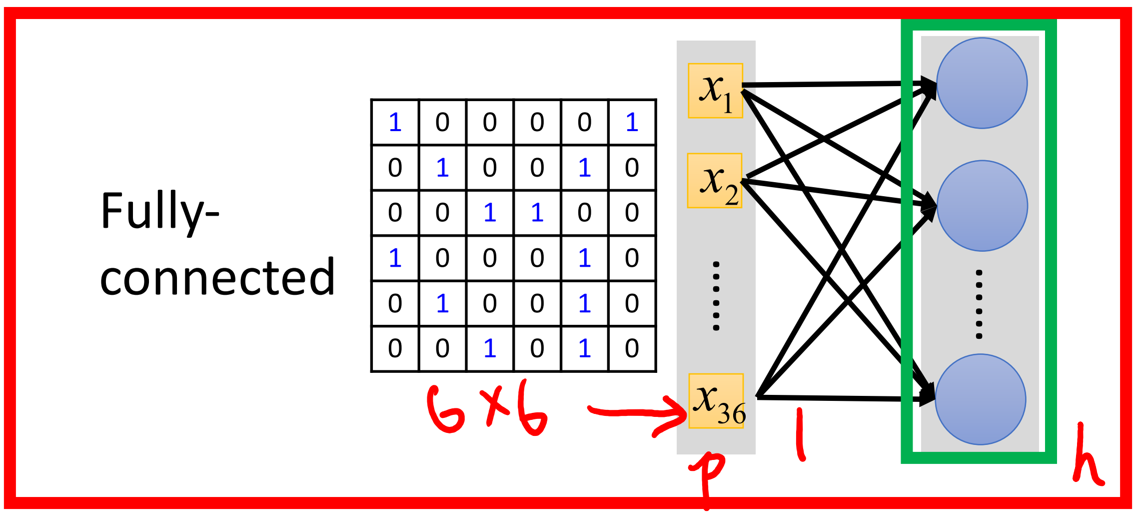


4 x 4 image

Dr. Hung-yi Lee's CNN slides

Convolution v.s. Fully Connected

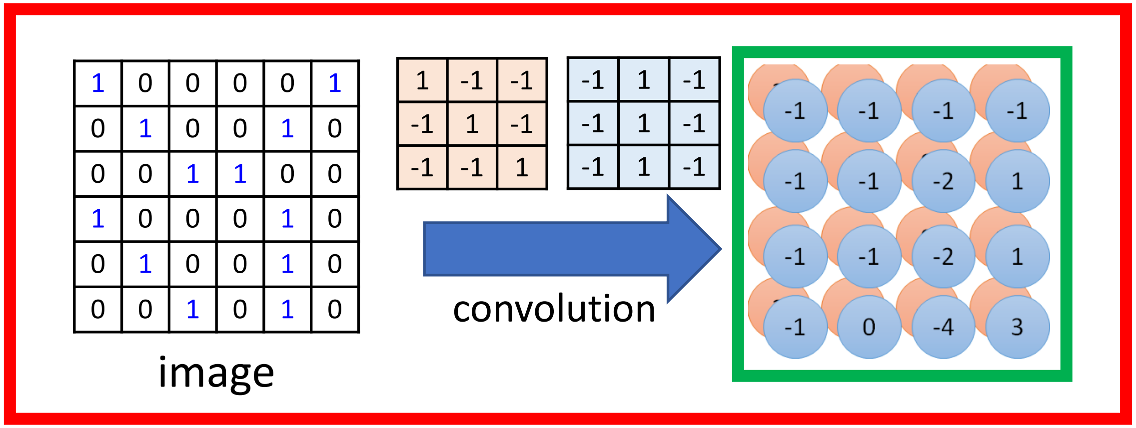


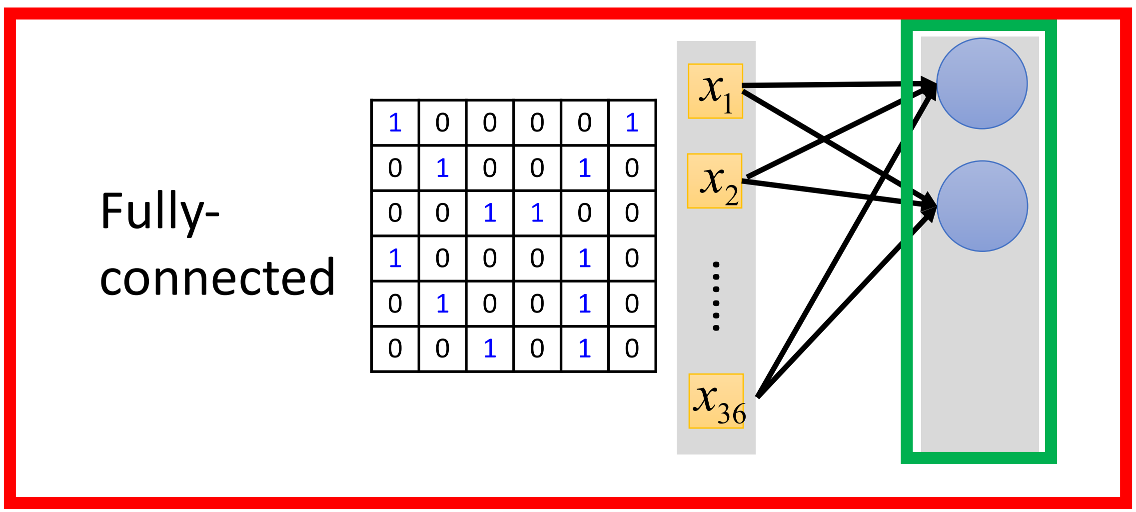


Dr. Hung-yi Lee's CNN slides

When with 2 filters, 3*3*2=18 parameters!

Convolution v.s. Fully Connected

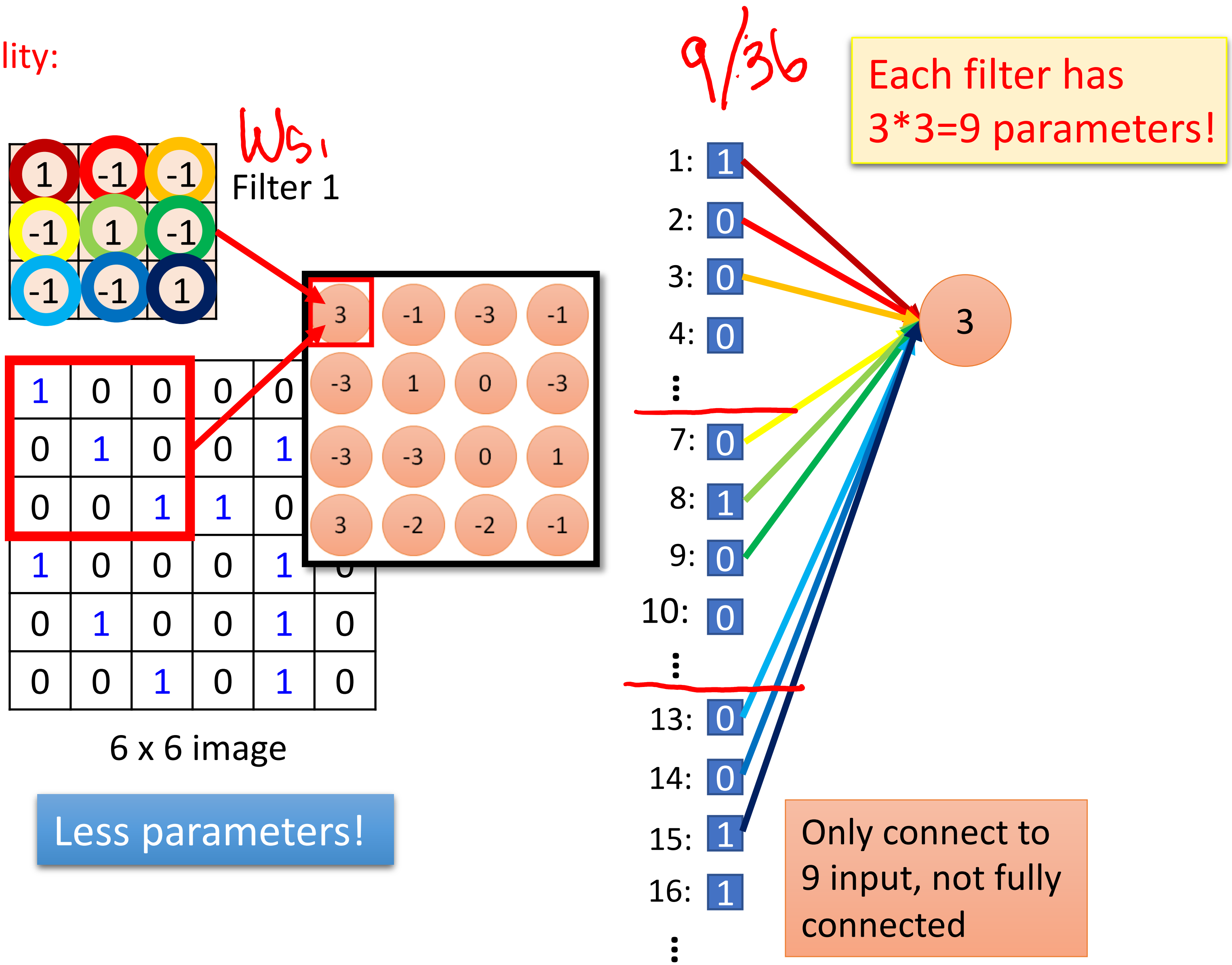




h=2

When 2 filters, 36*2=72 parameters!

(1) Locality:

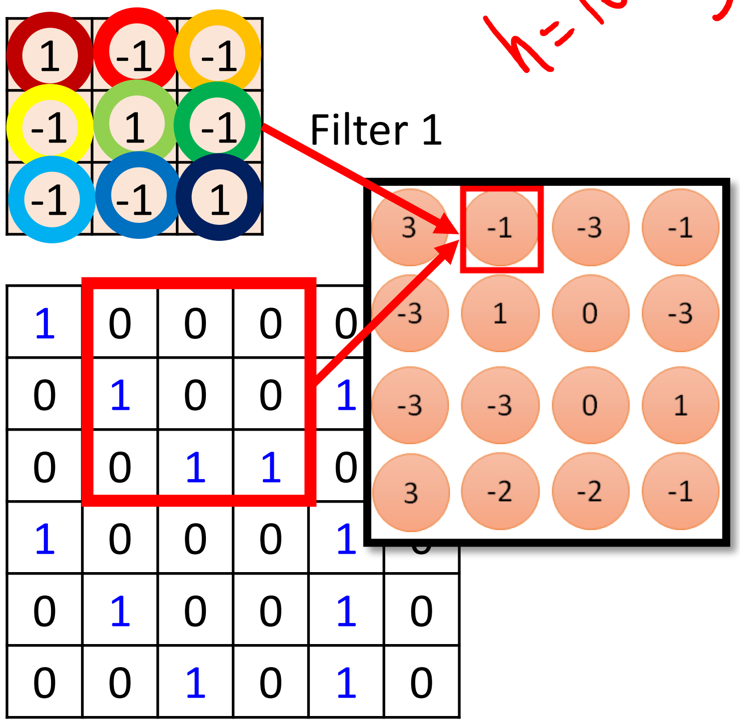


Less parameters!

Dr. Hung-yi Lee's CNN slides

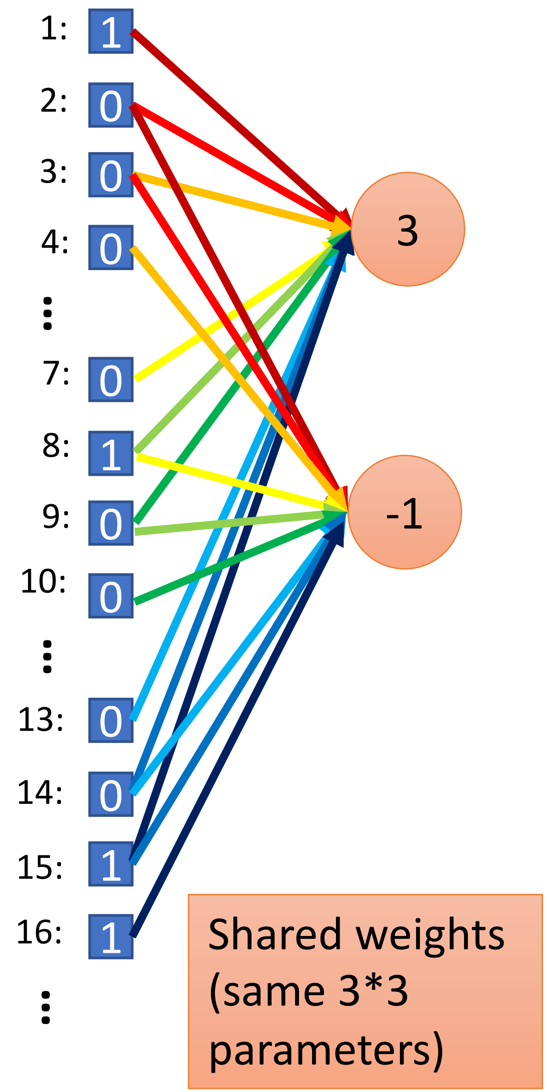
(2) Translation invariance:

$\left.\begin{array}{l}p=36\\ h=16\end{array}\right\} \longrightarrow 3\times 3$



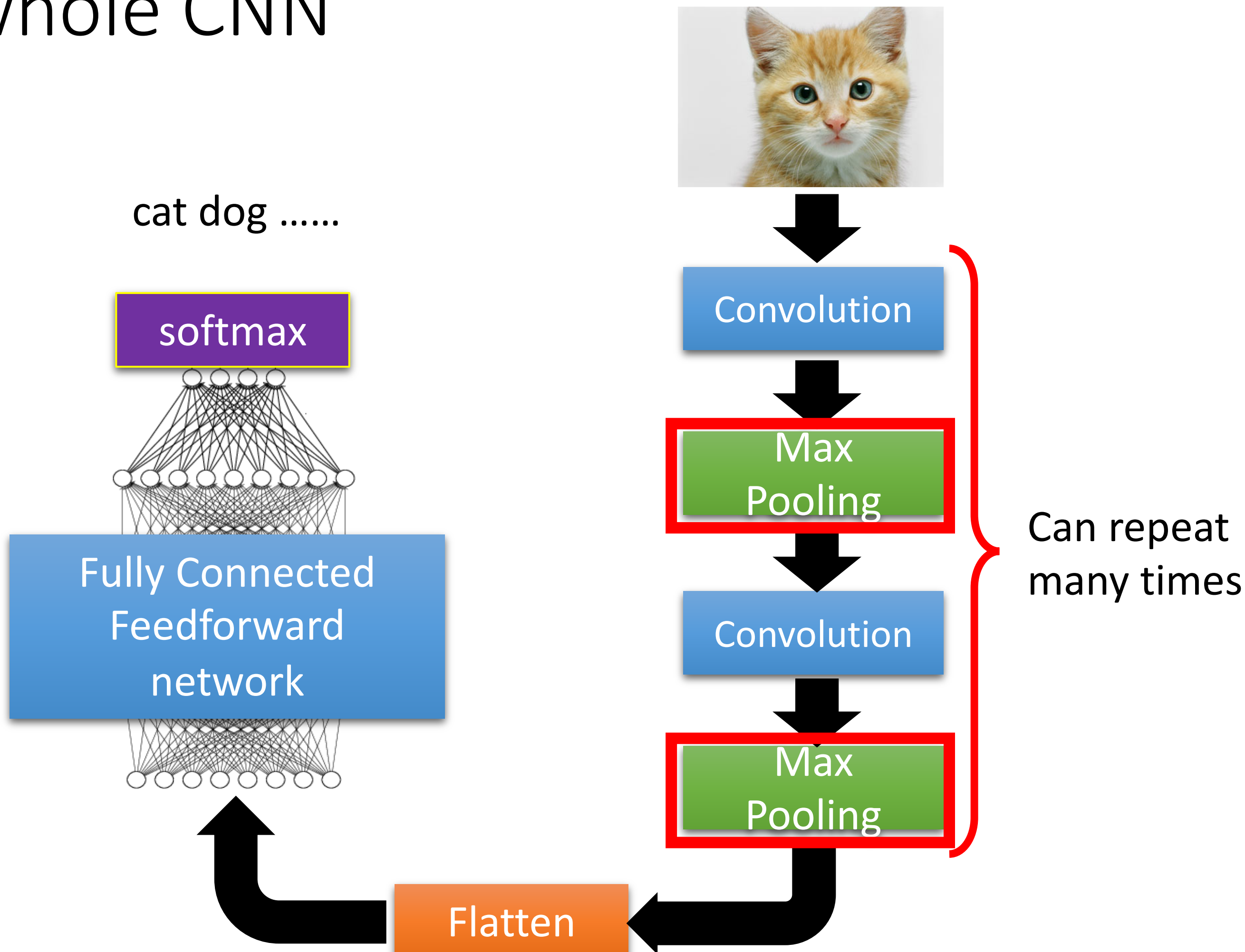
Less parameters!

Even less parameters!
(weight sharing)



Dr. Hung-yi Lee's CNN slides

The whole CNN



Dr. Hung-yi Lee’s CNN slides

(3) Subsampling:

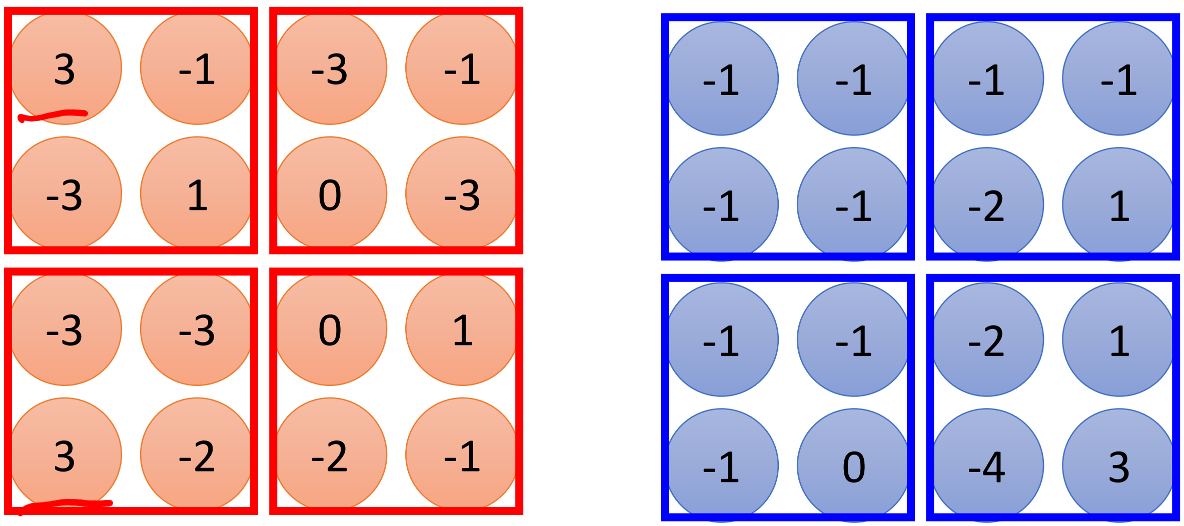
CNN – Max Pooling

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

Filter 1

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

Filter 2



Dr. Hung-yi Lee's CNN slides

(3) Subsampling:

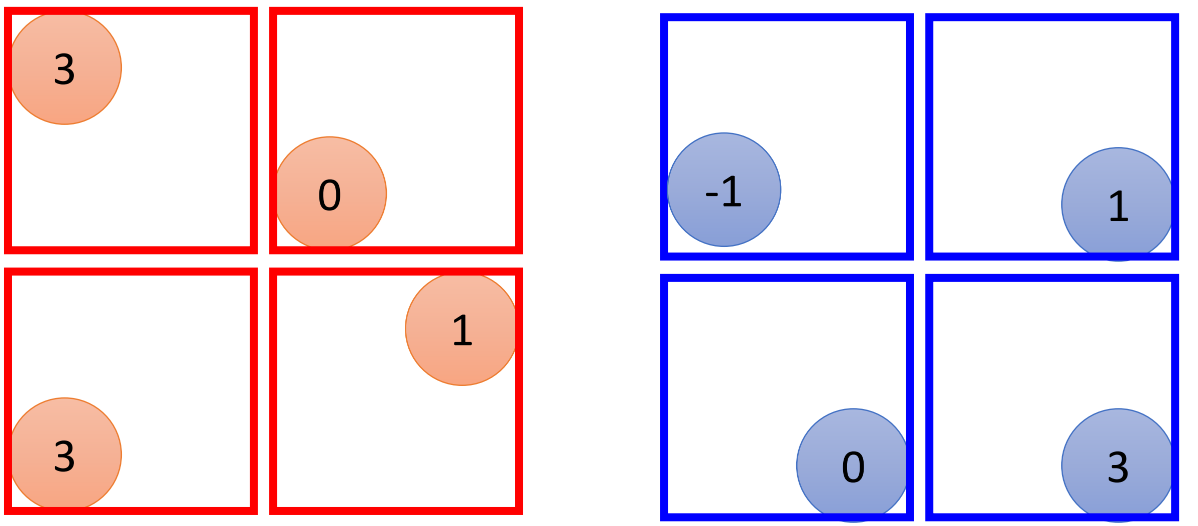
CNN – Max Pooling

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

Filter 1

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

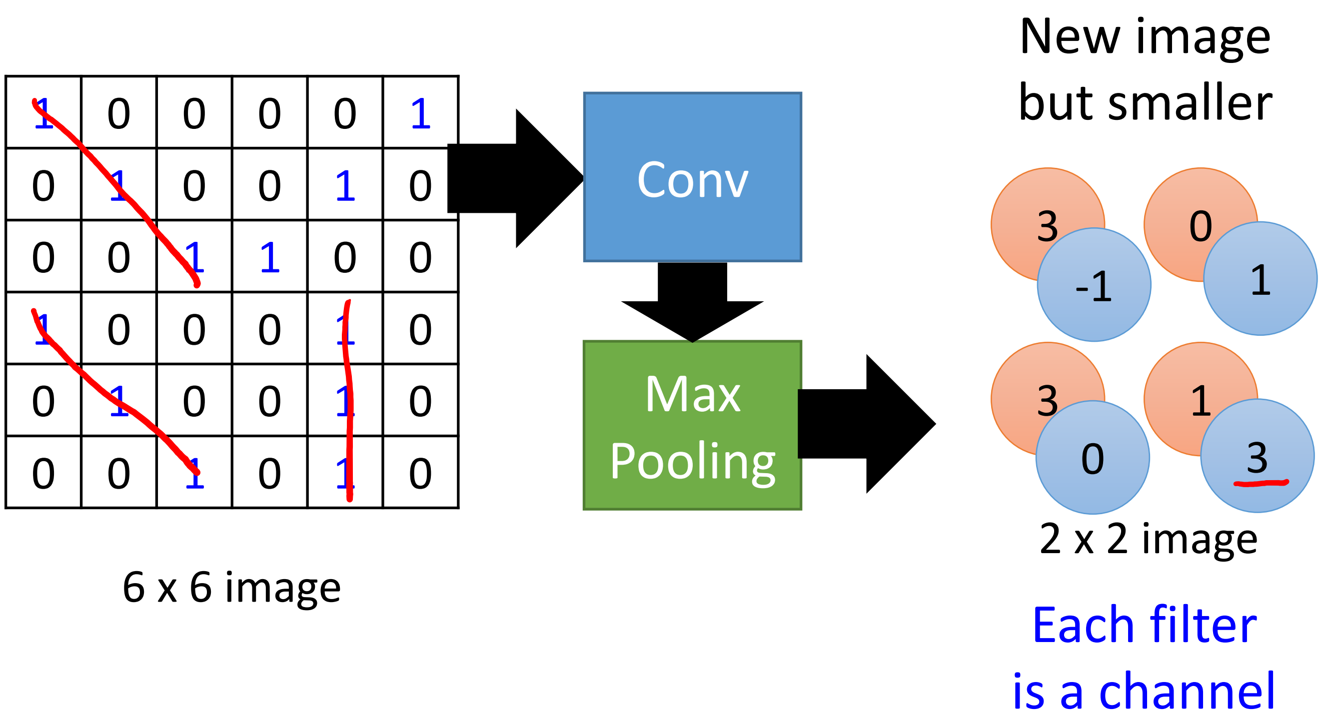
Filter 2



Dr. Hung-yi Lee's CNN slides

(3) Subsampling:

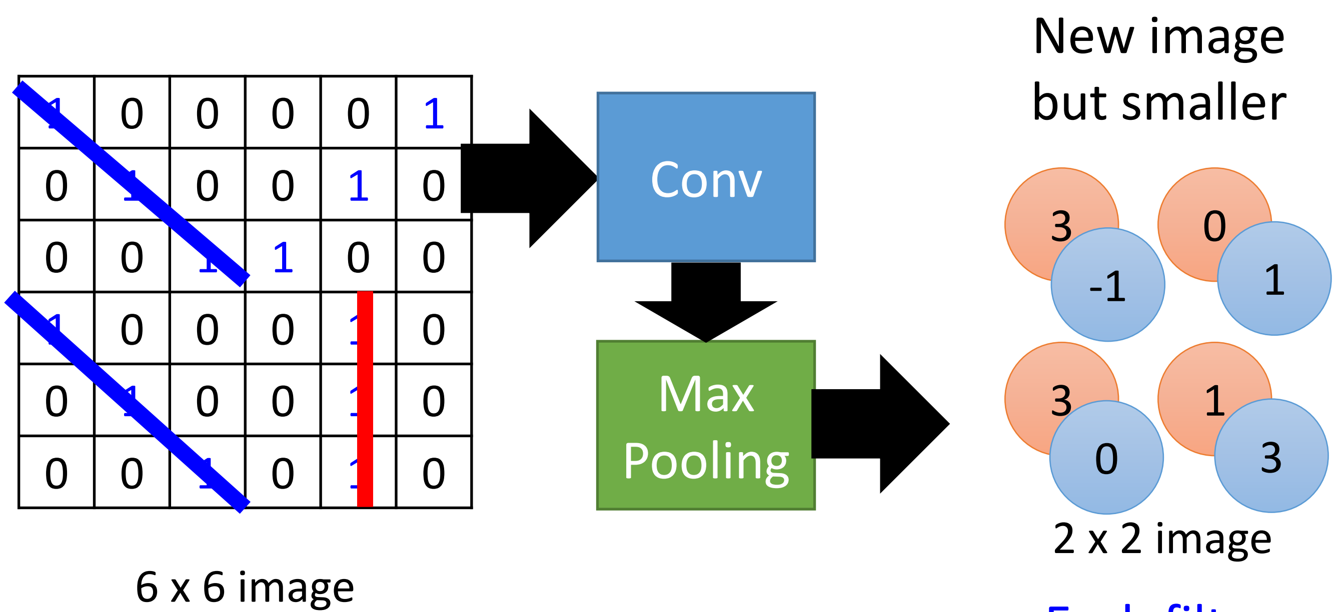
CNN – Max Pooling



Dr. Hung-yi Lee's CNN slides

(3) Subsampling:

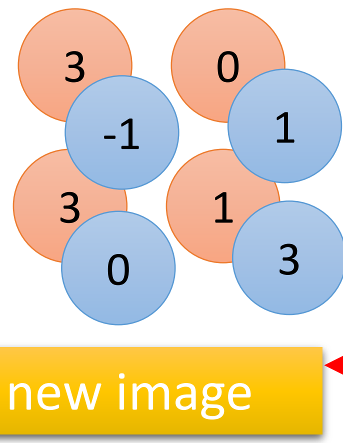
CNN – Max Pooling

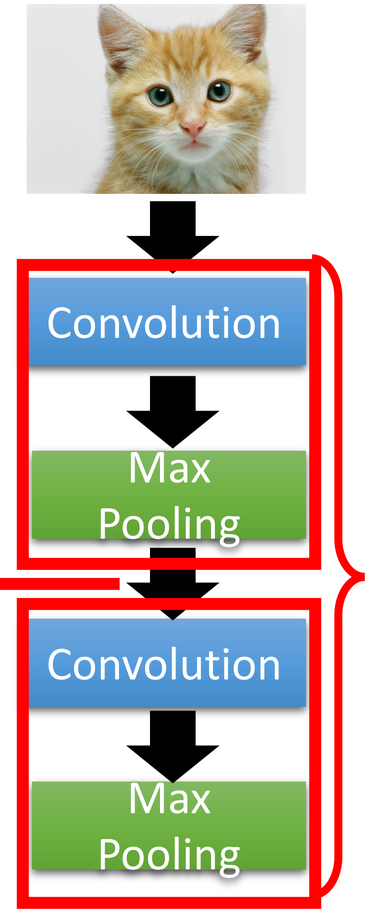


Each filter
is a channel

Dr. Hung-yi Lee's CNN slides

The whole CNN





Can repeat many times

Smaller than the original image

The number of the channel
is the number of filters

Dr. Hung-yi Lee’s CNN slides

CNN – Colorful image (from matrix to tensor)

BW
[W x H
matrix]

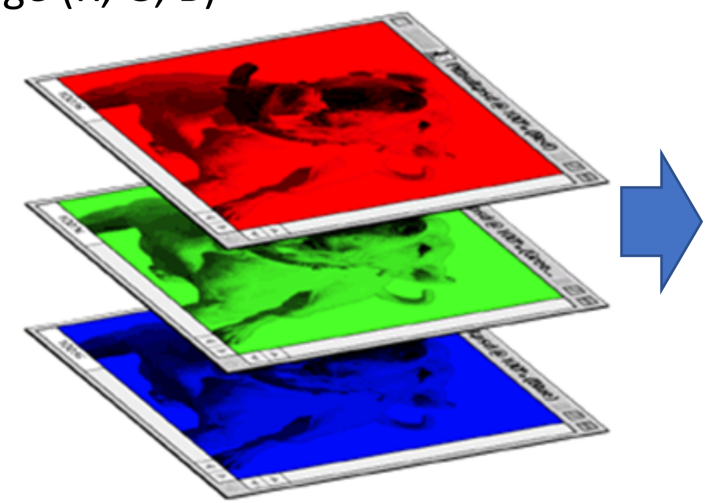
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

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

Filter 2

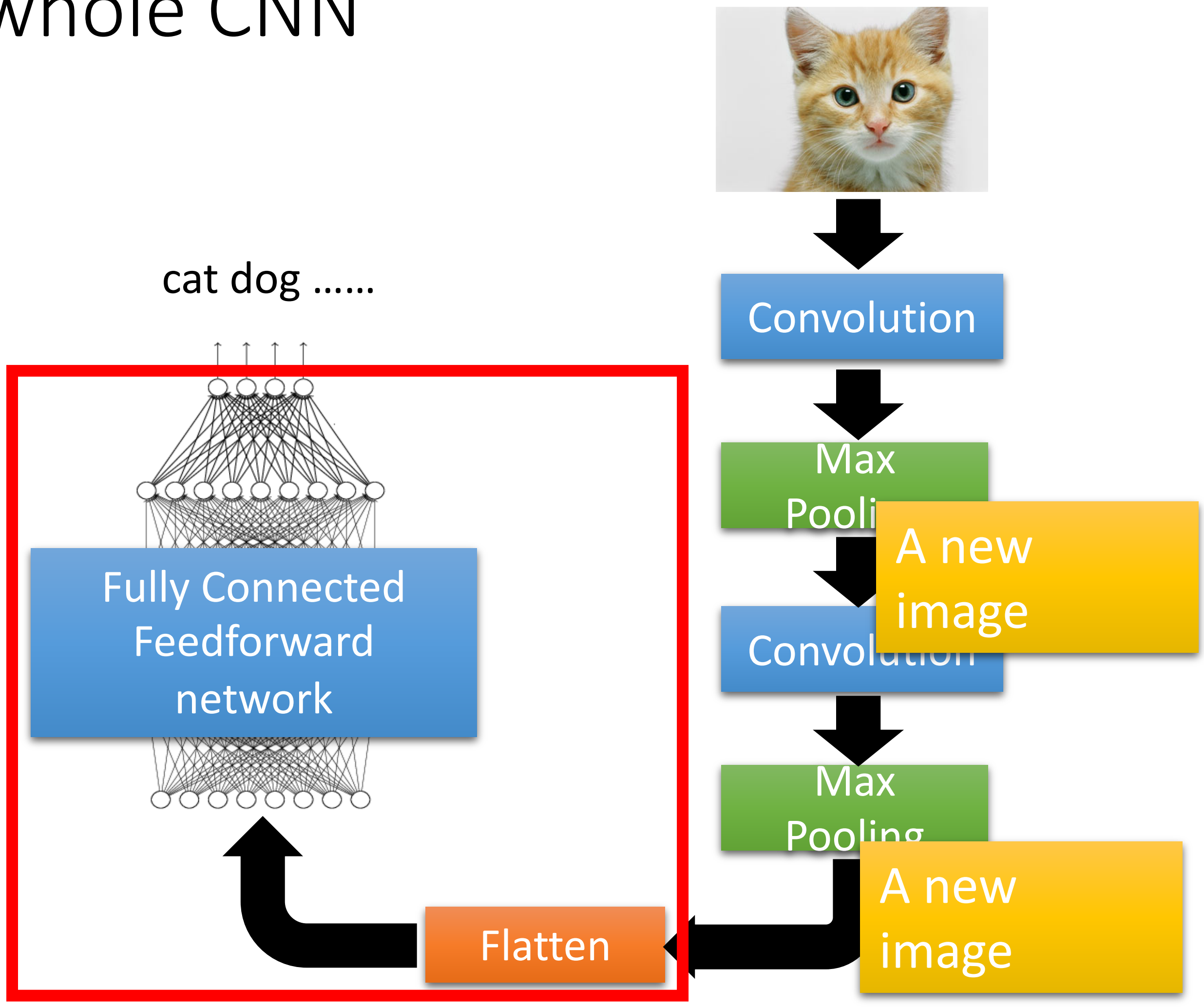
Colorful
image (R, G, B)



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

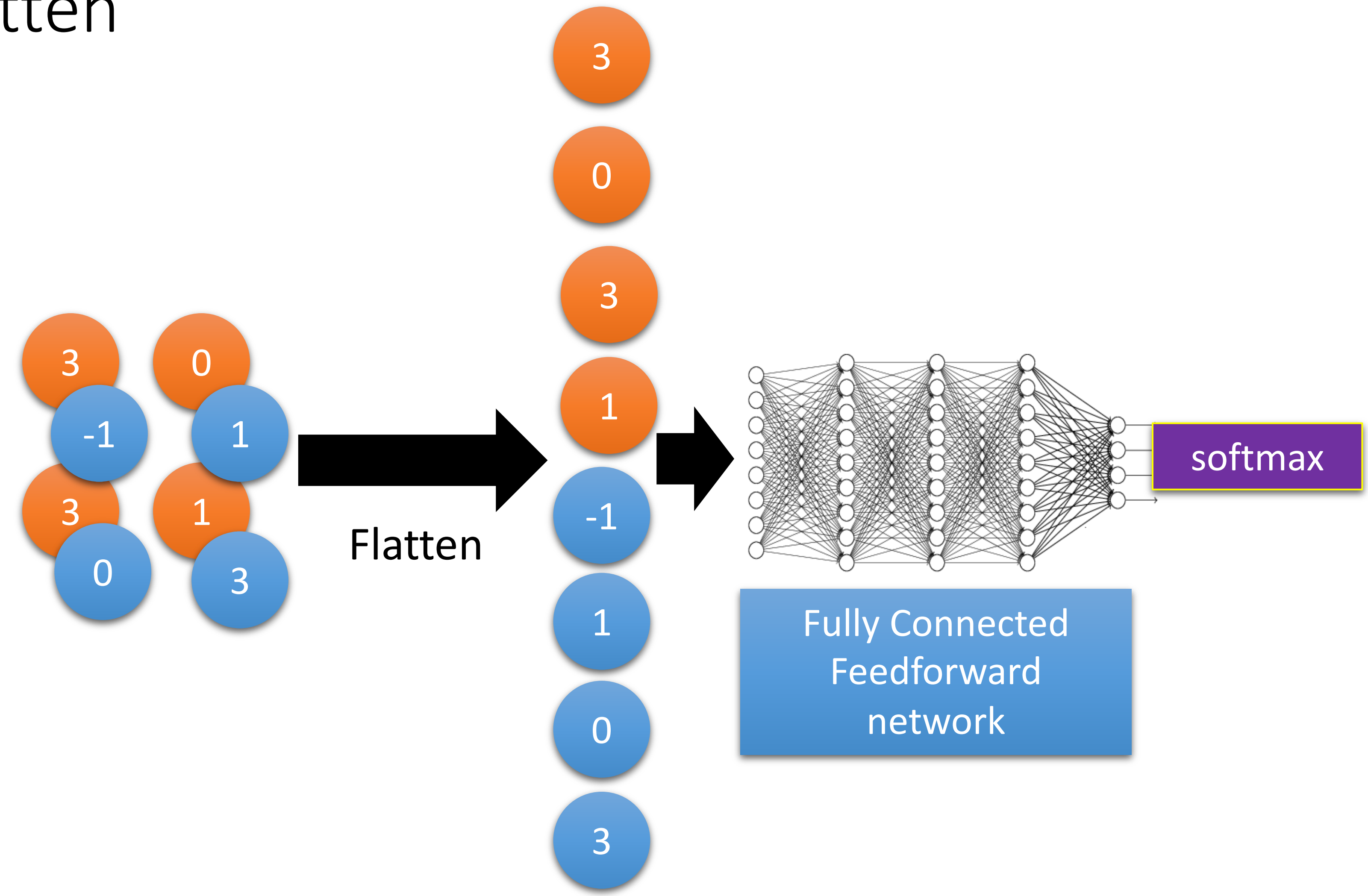
Dr. Hung-yi Lee’s CNN slides

The whole CNN



Dr. Hung-yi Lee's CNN slides

Flatten



Dr. Hung-yi Lee’s CNN slides

How many filters?

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

Filter 1

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

Filter 2

stride=1

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

6 x 6 image

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

Filter3

Dr. Hung-yi Lee’s CNN slides

CNN in Keras

network structure and ***input format***
(vector -> 3-D tensor)

```
model2.add( Convolution2D( 25,3,3,
            input_shape=(1,28,28) ) )
```

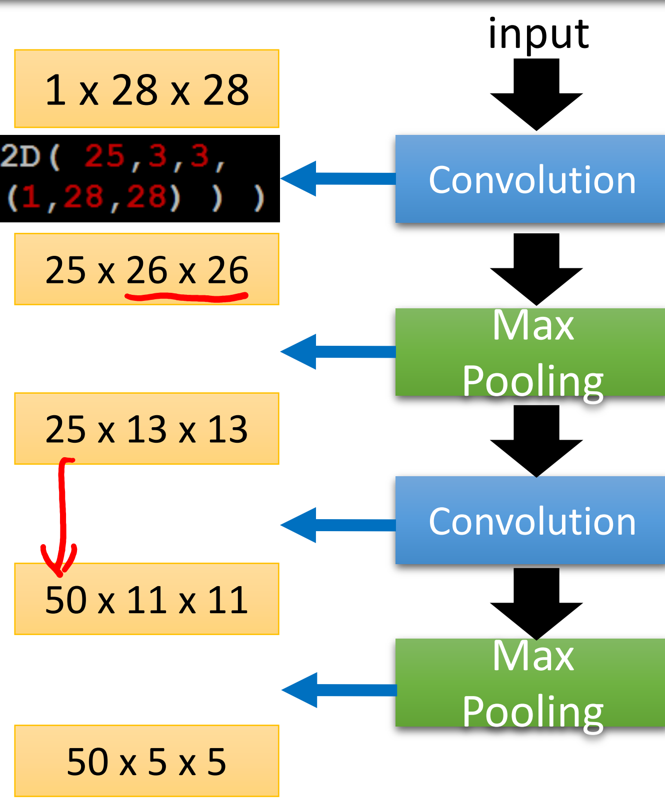
How many parameters for each filter? 9

```
model2.add(MaxPooling2D((2,2)))
```

```
model2.add(Convolution2D(50,3,3))
```

225= 25* 9

```
model2.add(MaxPooling2D((2,2)))
```

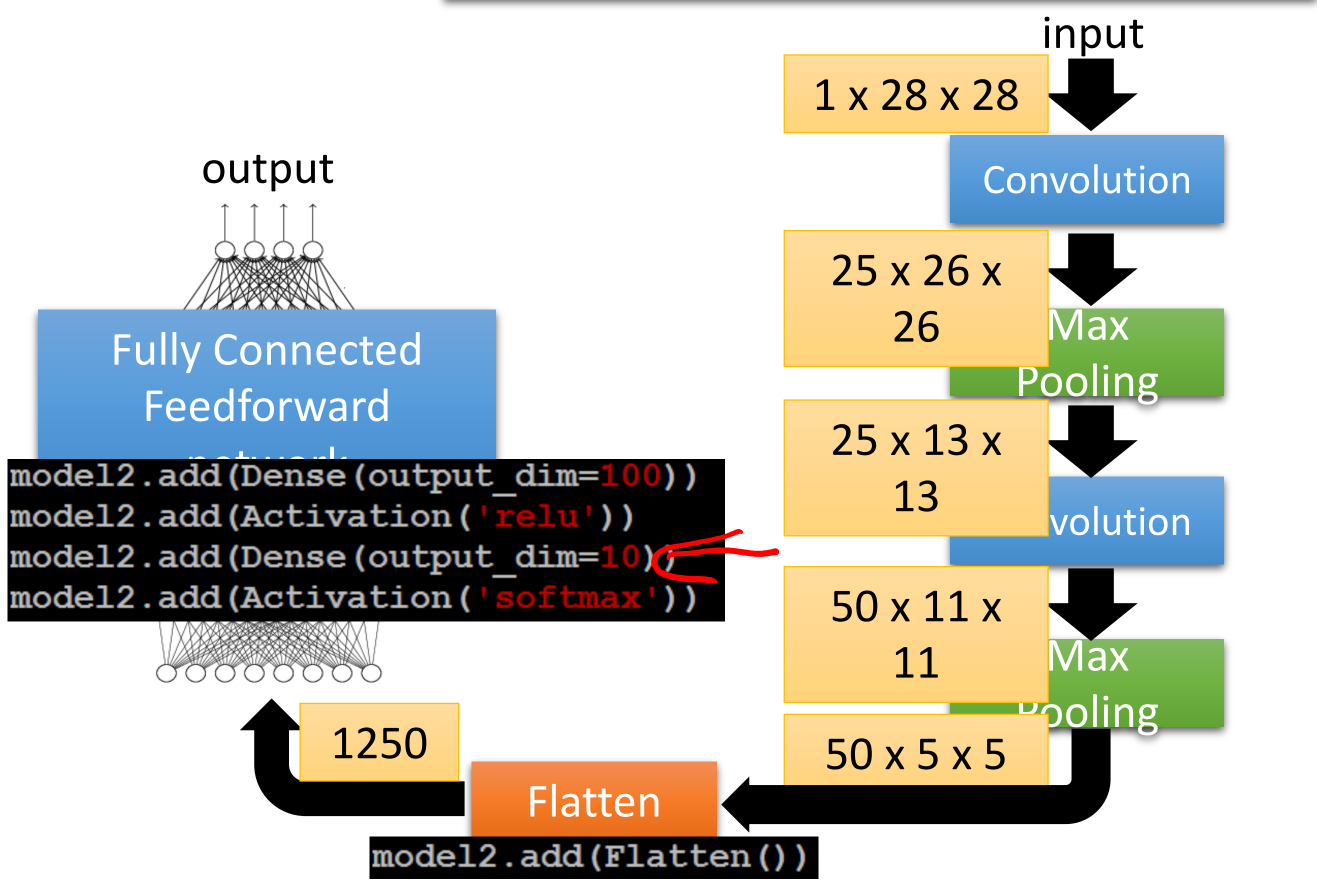


6x6
3x3
4x4
2x2

Dr. Hung-yi Lee's CNN slides

CNN in Keras

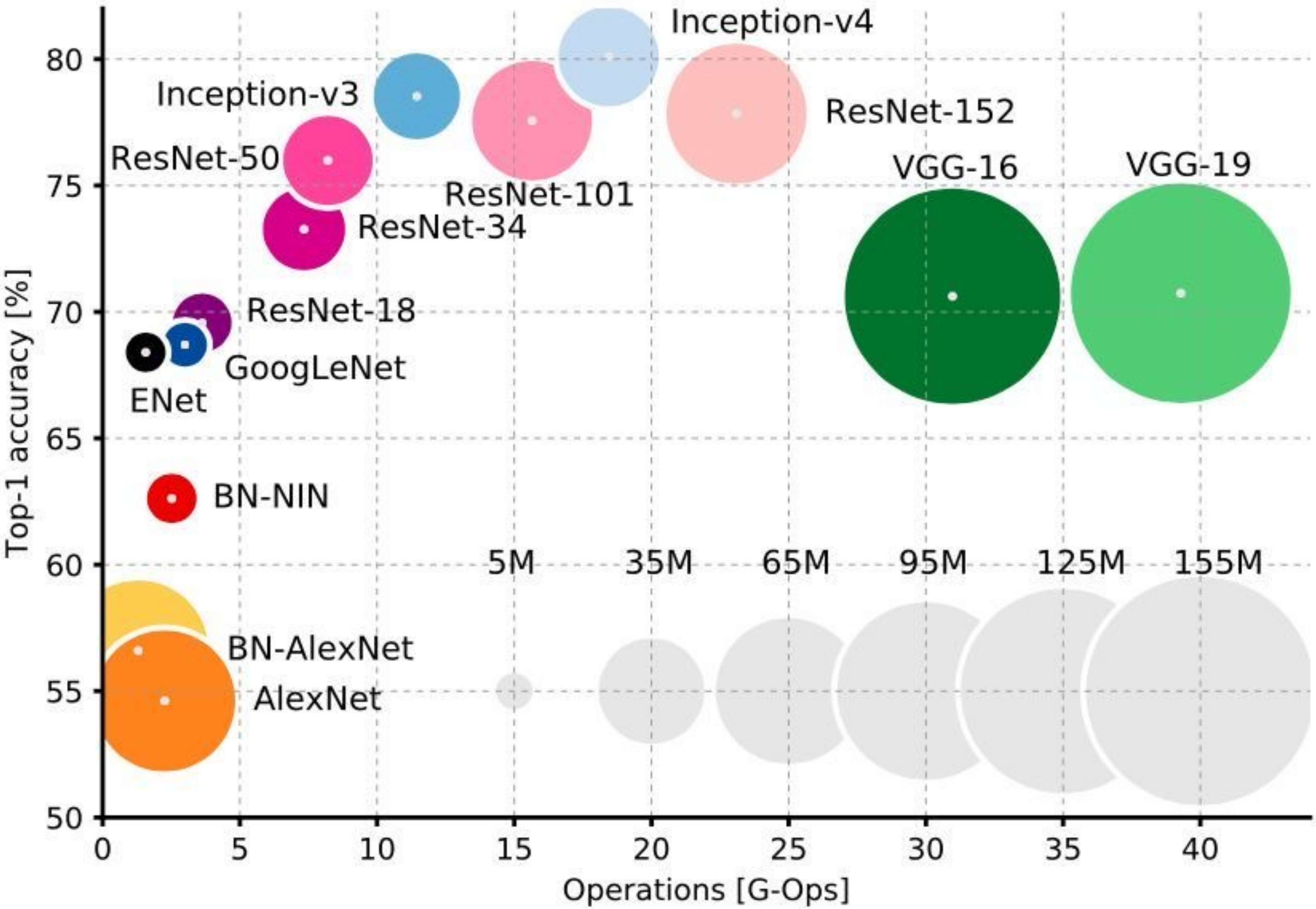
Only modified the ***network structure*** and ***input format (vector -> 3-D tensor)***

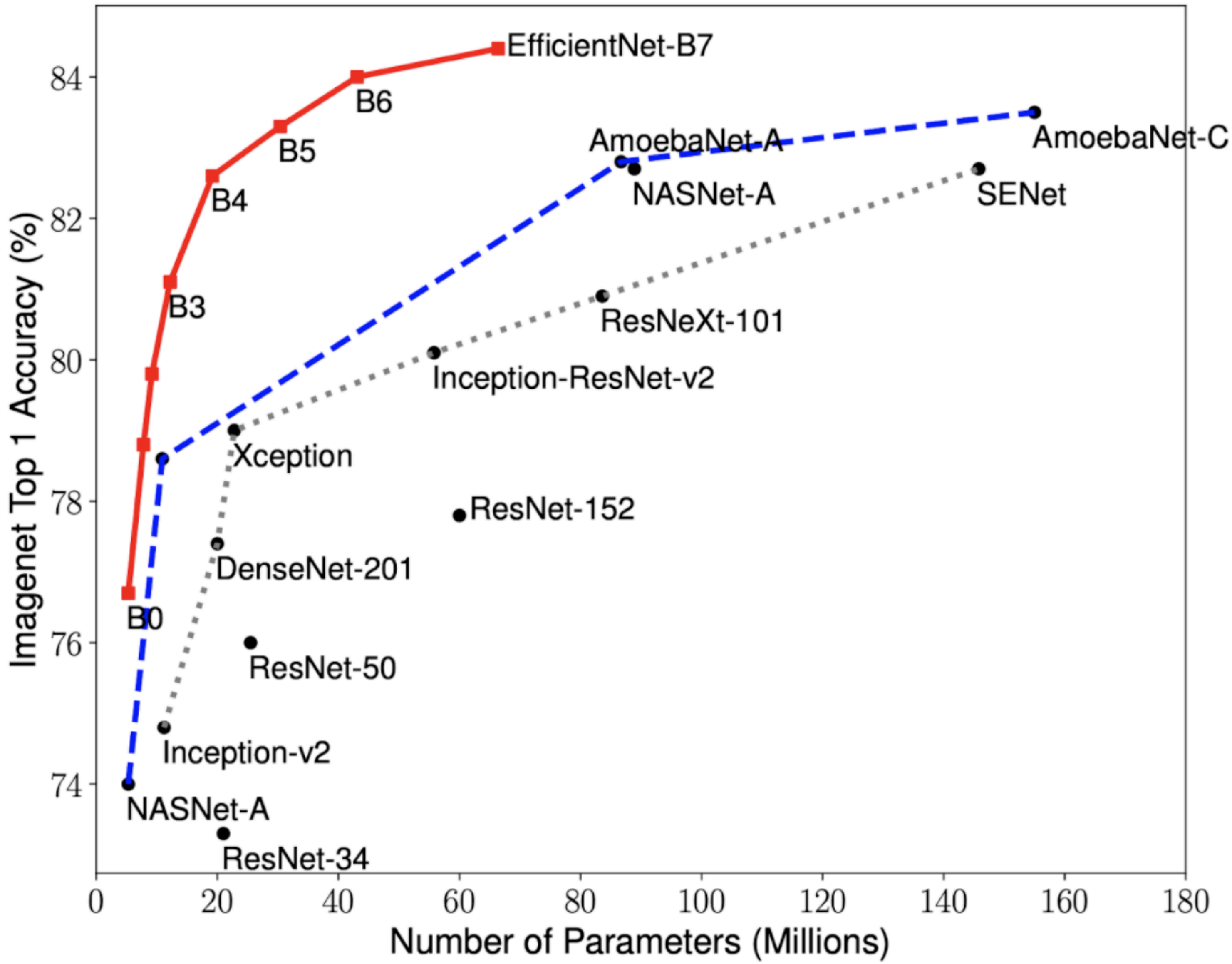



```
model2.add(Dense(output_dim=100))
model2.add(Activation('relu'))
model2.add(Dense(output_dim=10))
model2.add(Activation('softmax'))
```

```
model2.add(Flatten())
```


Dr. Hung-yi Lee’s CNN slides


Inception-v4
Inception-v3
ResNet-152
ResNet-50
ResNet-101
VGG-16
VGG-19
ResNet-34
ResNet-18
GoogLeNet
ENet
BN-NIN
5M
35M
65M
95M
125M
155M
BN-AlexNet
AlexNet
Top-1 accuracy [%]
Operations [G-Ops]
80
75
70
65
60
55
50
0
5
10
15
20
25
30
35
40



https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html

Thank You

References

- Big thanks to Prof. Ziv Bar-Joseph and Prof. Eric Xing @ CMU for allowing me to reuse some of his slides
- Elements of Statistical Learning, by Hastie, Tibshirani and Friedman
- Prof. Andrew Moore @ CMU's slides
- Tutorial slides from Dr. Tie-Yan Liu, MSR Asia

(number of) false positive (FP)

eqv. with false alarm, Type I error

(number of) false negative (FN)

eqv. with miss, Type II error

---

sensitivity or true positive rate (TPR)

eqv. with hit rate, recall

$TPR = TP/P = TP/(TP + FN)$ → Actual Positive

specificity (SPC) or true negative rate

$SPC = TN/N = TN/(TN + FP)$

precision or positive predictive value (PPV)

$PPV = TP/(TP + FP)$ PP: predicted positive

negative predictive value (NPV)

$NPV = TN/(TN + FN)$

fall-out or false positive rate (FPR)

$FPR = FP/N = FP/(FP + TN) = 1 - SPC$ → Actual Negative

false negative rate (FNR)

$FNR = FN/(TP + FN) = 1 - TPR$

false discovery rate (FDR)

$FDR = FP/(TP + FP) = 1 - PPV$

---

accuracy (ACC)

$ACC = (TP + TN)/(TP + FP + FN + TN)$

F1 score

is the harmonic mean of precision and sensitivity / Recall

$F1 = 2TP/(2TP + FP + FN)$

From Wiki

When with Unbalanced Issue (binary case)

Acc Bad

- Class imbalance issue

$\# AP << \# AN$

- Balanced accuracy:

	actual +	actual −
predicted+	TP	FP
predicted−	FN	TN

When with Unbalanced Issue (binary case)

- Class imbalance issue

$$\text{num } \underbrace{AP}_{\text{actual positive}} \ll \text{num } \underbrace{AN}_{\text{actual neg.}}$$

- Balanced accuracy:

$$= \frac{1}{2}\left(\frac{TP}{PP} + \frac{TN}{PN}\right)$$

$PP \rightarrow TP+FP$

$PN \rightarrow TN+FN$

	actual +	actual −
predicted+	TP	FP
predicted−	FN	TN

AP vs. $AN = 1 : 99$

① classifier[1]

	y AP	y AN
$\hat{y}$ PP	0	0
$\hat{y}$ PN	1	99

$$ACC = \frac{99}{100} = 99\%$$

$$BACC = \frac{1}{2}\left(\frac{0}{0+\varepsilon} + \frac{99}{100}\right)$$

$$= 49.5\%$$

Low Ratio of Positive Class (binary case)

If $\frac{\text{Actual P}}{\text{AP} + \text{AN}}$ very small (e.g. $< 1\%$)

(1, 99)
pos neg

$\Rightarrow$ a classifier can predict every example as Neg

$\Rightarrow$ 1

	AP 1	AN 99
predict P	0	0
predict N	1	99

$\Rightarrow$ Accuracy $= \frac{99}{100} = 0.99$

$\Rightarrow$ Balanced Acc =

Bad-neg-classifier

① Balanced Acc $= \frac{1}{2}\left(\frac{TP}{P} + \frac{TN}{N}\right)$

$$= \frac{1}{2}\left(\frac{0}{0+\varepsilon} + \frac{99}{100}\right) = 0.495$$

$[0, 1]$

another classifier

②

	AP	AN
PP	1	0
PN	0	99

Balanced Acc $= \frac{1}{2}\left(\frac{1}{1} + \frac{99}{99}\right) = 1$

$$\text{Acc} = \frac{1+99}{1+0+99+0} = 1$$

(pos Ratio 1/100)

③ Third classifier

	AP	AN
PP +	0	1
PN −	1	99

$$ACC = \frac{99}{101} \approx 99\%$$

$$BACC = \frac{1}{2}\left(\frac{0}{1} + \frac{99}{100}\right) \approx 0.495$$

(pos Ratio 2/120)

④ Fourth case

	AP	AN
PP+	1	19
PN−	1	99

$$ACC = \frac{100}{120} \approx 83\%$$

$$BACC = \frac{1}{2}\left(\frac{1}{20} + \frac{99}{100}\right) \approx 0.52$$

When with Unbalanced Issue (binary case)

4th case on previous page — another case: 2 vs. 118 ≈ 1:60

⇒ Balanced Acc cares all classes

⇒ If care more about pos+ class

	AP	AN
PP	1	19
PN	1	99

⇒

- Precision $= \frac{TP}{TP+FP} = \frac{1}{20} = 5\%$

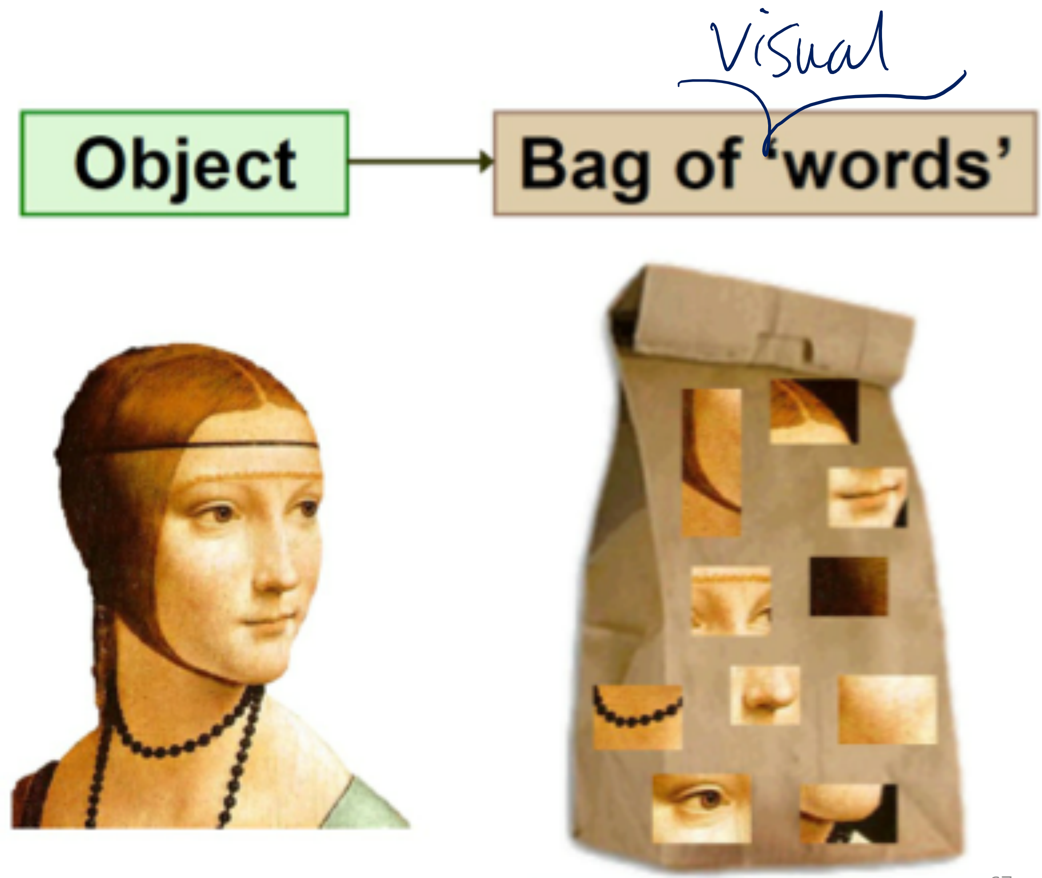
- Recall $= \frac{TP}{TP+FN} = \frac{1}{2} = 50\%$

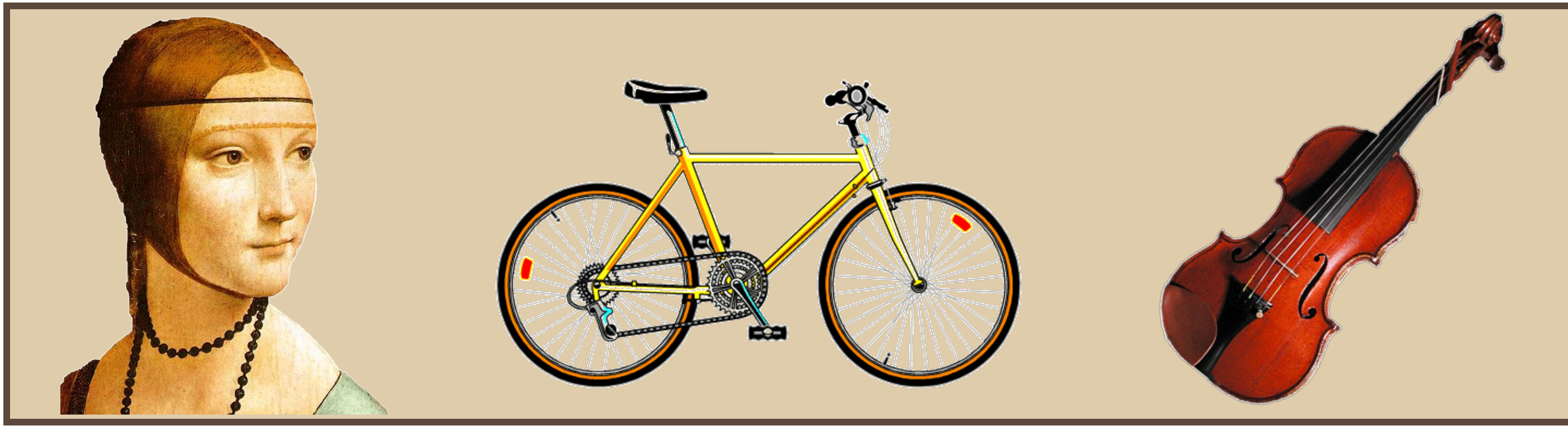
	actual +	actual −
predicted+	TP	FP
predicted−	FN	TN

When not using Deep Learning: Image Representation for – Objective recognition

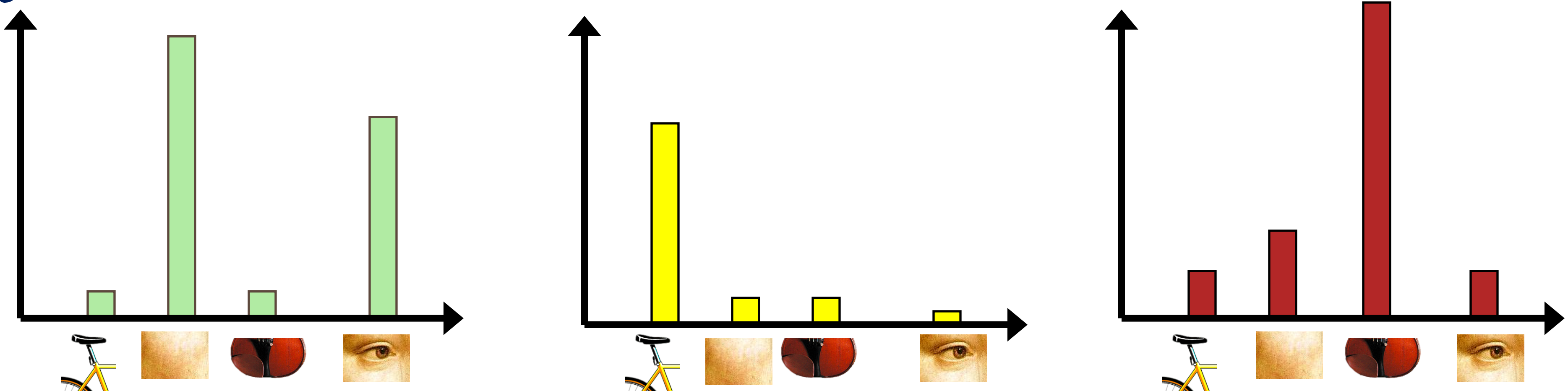
- Image representation ➔ bag of “visual words”

- An object image: histogram of visual vocabulary – a numerical vector of D dimensions.





Occurrence



W_1 W_2 W_3 .. W_D

A study comparing Classifiers

An Empirical Comparison of Supervised Learning Algorithms

Rich Caruana CARUANA@CS.CORNELL.EDU
Alexandru Niculescu-Mizil ALEXN@CS.CORNELL.EDU
Department of Computer Science, Cornell University, Ithaca, NY 14853 USA

Abstract

A number of supervised learning methods have been introduced in the last decade. Unfortunately, the last comprehensive empirical evaluation of supervised learning was the Statlog Project in the early 90's. We present a large-scale empirical comparison between ten supervised learning methods: SVMs, neural nets, logistic regression, naive bayes, memory-based learning, random forests, decision trees, bagged trees, boosted trees, and boosted stumps. We also examine the effect that calibrating the models via Platt Scaling and Isotonic Regression has on their performance. An important aspect of our study is the use of a variety of performance criteria to evaluate the learning methods.

This paper presents results of a large-scale empirical comparison of ten supervised learning algorithms using eight performance criteria. We evaluate the performance of SVMs, neural nets, logistic regression, naive bayes, memory-based learning, random forests, decision trees, bagged trees, boosted trees, and boosted stumps on eleven binary classification problems using a variety of performance metrics: accuracy, F-score, Lift, ROC Area, average precision, precision/recall break-even point, squared error, and cross-entropy. For each algorithm we examine common variations, and thoroughly explore the space of parameters. For example, we compare ten decision tree styles, neural nets of many sizes, SVMs with many kernels, etc.

Because some of the performance metrics we examine interpret model predictions as probabilities and models such as SVMs are not designed to predict probabil-

A study comparing Classifiers
➔ 11 binary classification datasets

Small data

Table 1. Description of problems

PROBLEM	#ATTR	TRAIN SIZE	TEST SIZE	%POZ
ADULT	14/104	5000	35222	25%
BACT	11/170	5000	34262	69%
COD	15/60	5000	14000	50%
CALHOUS	9	5000	14640	52%
COV_TYPE	54	5000	25000	36%
HS	200	5000	4366	24%
LETTER.P1	16	5000	14000	3%
LETTER.P2	16	5000	14000	53%
MEDIS	63	5000	8199	11%
MG	124	5000	12807	17%
SLAC	59	5000	25000	50%

Tree , SVM , NN ,

DeepLearn

A study comparing Classifiers

➔ 11 binary classification problems / 8 metrics

Table 2. Normalized scores for each learning algorithm by metric (average over eleven problems)

MODEL	CAL	ACC	FSC	LFT	ROC	APR	BEP	RMS	MXE	MEAN	OPT-SEL
BST-DT	PLT	.843*	.779	**.939**	**.963**	**.938**	.929*	**.880**	**.896**	**.896**	**.917**
RF	PLT	.872*	.805	.934*	.957	.931	**.930**	.851	.858	.892	.898
BAG-DT	–	.846	.781	.938*	.962*	.937*	.918	.845	.872	.887*	.899
BST-DT	ISO	.826*	.860*	.929*	.952	.921	.925*	.854	.815	.885	.917*
RF	–	**.872**	.790	.934*	.957	.931	**.930**	.829	.830	.884	.890
BAG-DT	PLT	.841	.774	.938*	.962*	.937*	.918	.836	.852	.882	.895
RF	ISO	.861*	**.861**	.923	.946	.910	.925	.836	.776	.880	.895
BAG-DT	ISO	.826	.843*	.933*	.954	.921	.915	.832	.791	.877	.894
SVM	PLT	.824	.760	.895	.938	.898	.913	.831	.836	.862	.880
ANN	–	.803	.762	.910	.936	.892	.899	.811	.821	.854	.885
SVM	ISO	.813	.836*	.892	.925	.882	.911	.814	.744	.852	.882
ANN	PLT	.815	.748	.910	.936	.892	.899	.783	.785	.846	.875
ANN	ISO	.803	.836	.908	.924	.876	.891	.777	.718	.842	.884
BST-DT	–	.834*	.816	**.939**	**.963**	**.938**	.929*	.598	.605	.828	.851
KNN	PLT	.757	.707	.889	.918	.872	.872	.742	.764	.815	.837
KNN	–	.756	.728	.889	.918	.872	.872	.729	.718	.810	.830
KNN	ISO	.755	.758	.882	.907	.854	.869	.738	.706	.809	.844
BST-STMP	PLT	.724	.651	.876	.908	.853	.845	.716	.754	.791	.808
SVM	–	.817	.804	.895	.938	.899	.913	.514	.467	.781	.810
BST-STMP	ISO	.709	.744	.873	.899	.835	.840	.695	.646	.780	.810
BST-STMP	–	.741	.684	.876	.908	.853	.845	.394	.382	.710	.726
DT	ISO	.648	.654	.818	.838	.756	.778	.590	.589	.709	.774