EDU

CSDN学院　IT实战派

Python 深度学习

深度学习实战篇：物品分类模型与应用

讲师：彭靖田

本章目录

EDU

CSDN学院　IT实战派

Python 深度学习

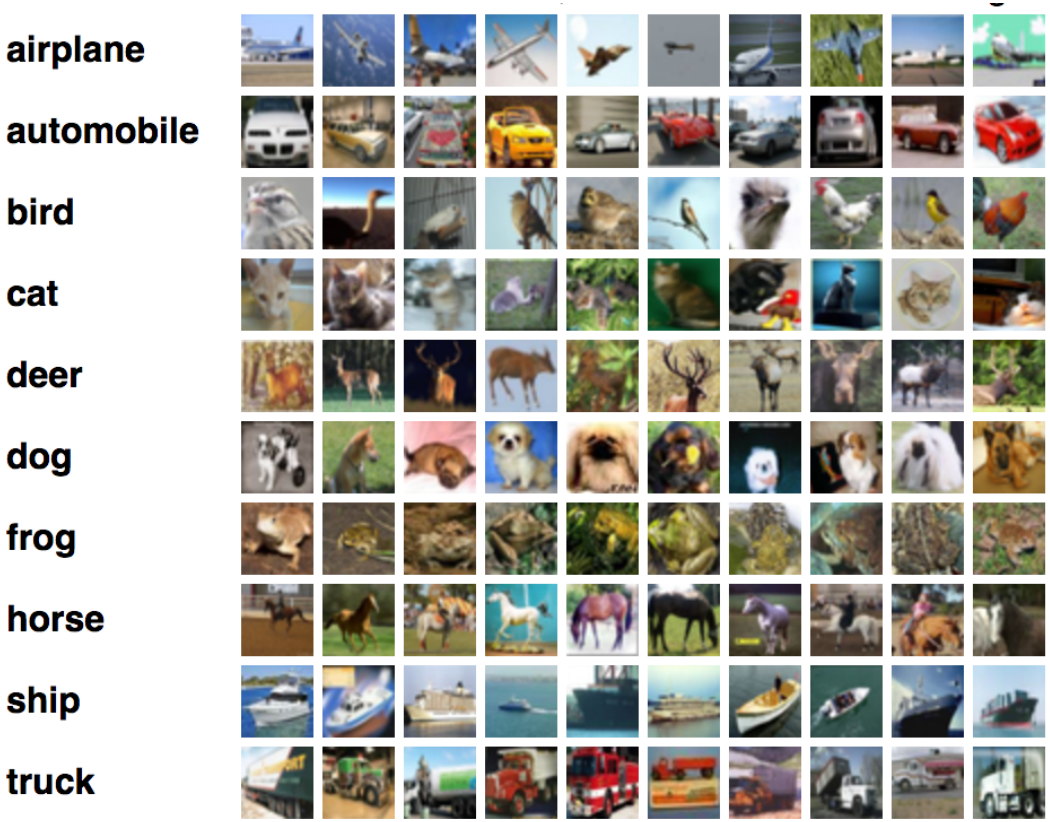
深度学习实战：图像分类问题定义与说明

讲师：彭靖田

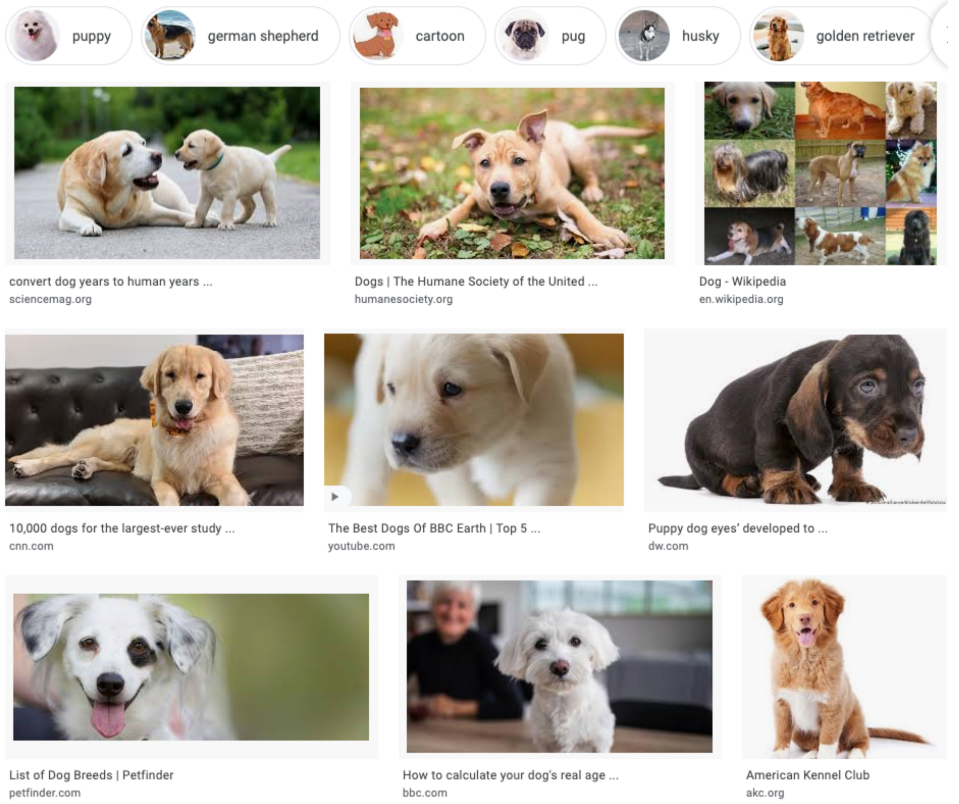
图像分类问题

图像分类问题

语义级分类

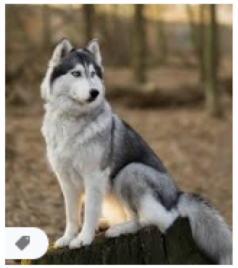


细粒度分类



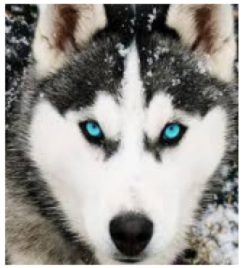
图像分类问题

实例级分类 ——————→ 识别问题

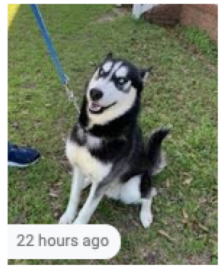

Siberian Husky dog – blog
opsopet.com · In stock


200+ Free Siberian Hus...
pixabay.com


Difference Between an Alaskan Malamute ...
animals.howstuffworks.com

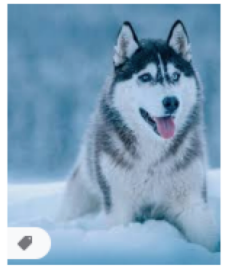

Amazing and Smart Siberi...
pinterest.fr


Siberian Husky Dogs. T...
123rf.com

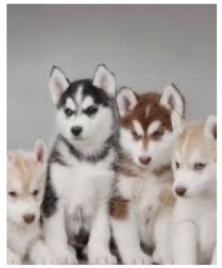


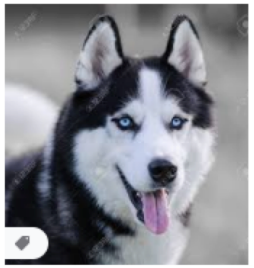
200218 Chief, a Siberia...
petfinder.com


200+ Free Siberian Husky & Husky Image...
pixabay.com


Husky Dog Sledding Fr...
experiencebaltics.com


5 Dog Breeds Who Enjoy Winter ...
pawfeel.com


red husky. | Cute anima...
pinterest.com


Siberian Husky Dog With Bl...
123rf.com

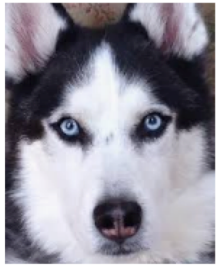

Siberian Husky - Wikipedia
en.wikipedia.org

Husky Snow Sled Images, Stock Photos ...
shutterstock.com


JM Husky Dog Fleece Throw Blank...
amazon.co.uk


Shepsky Mixed Dog Breed Pictures ...
dogtime.com


Australian Shepherd, Siberian...
womansday.com

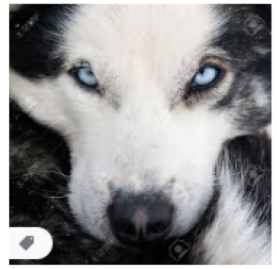

Avalanche, Siberian Hu...
facebook.com


Siberian husky dog. Black and w...
canstockphoto.com


Siberian Husky Dog Breed Information ...
yourshirt.info

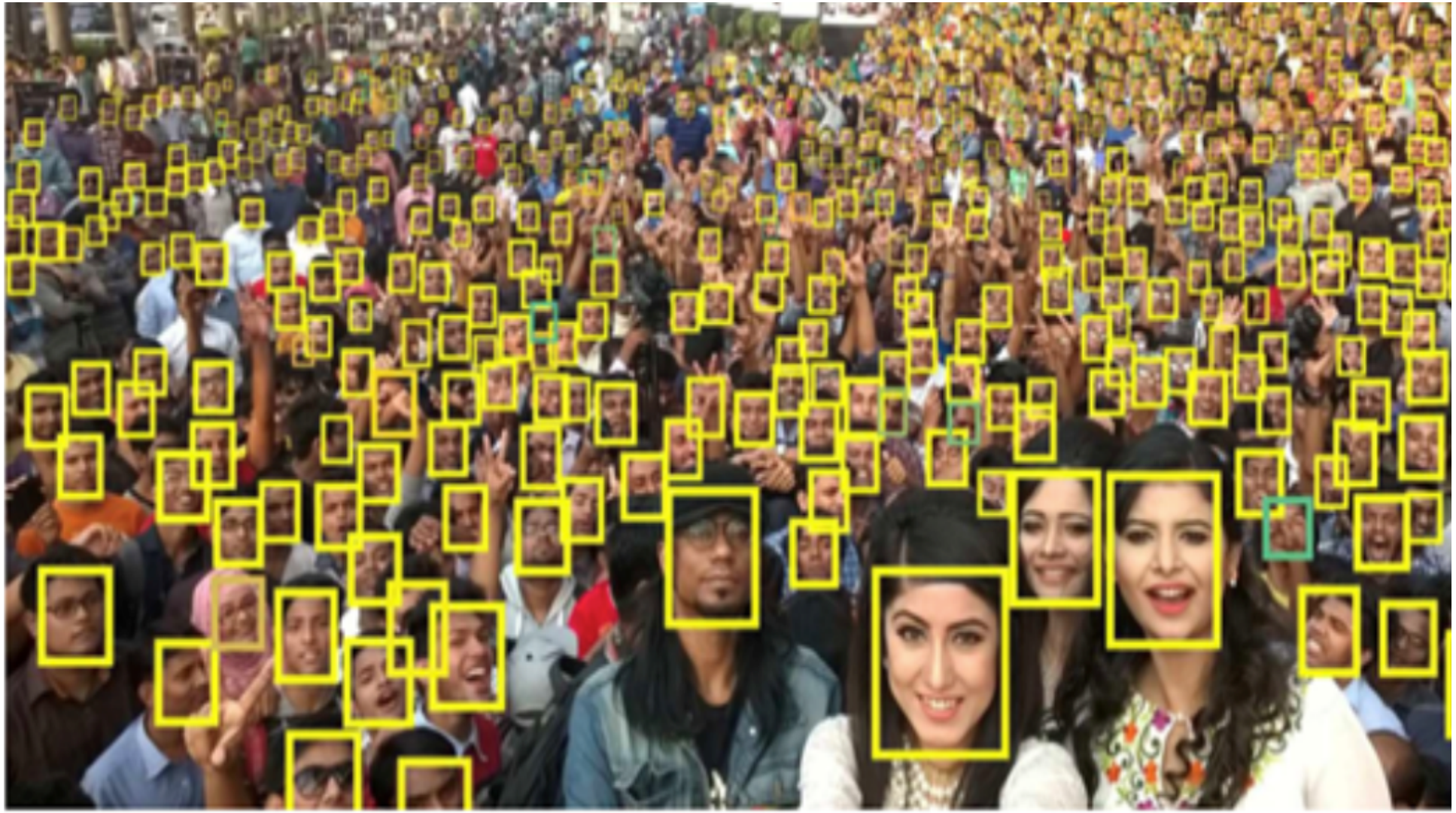

Husky Dog Diamond Painting...
aliexpress.com · In stock


Dog Breeds Siberian Husky - Let Them Fun
doggy.id

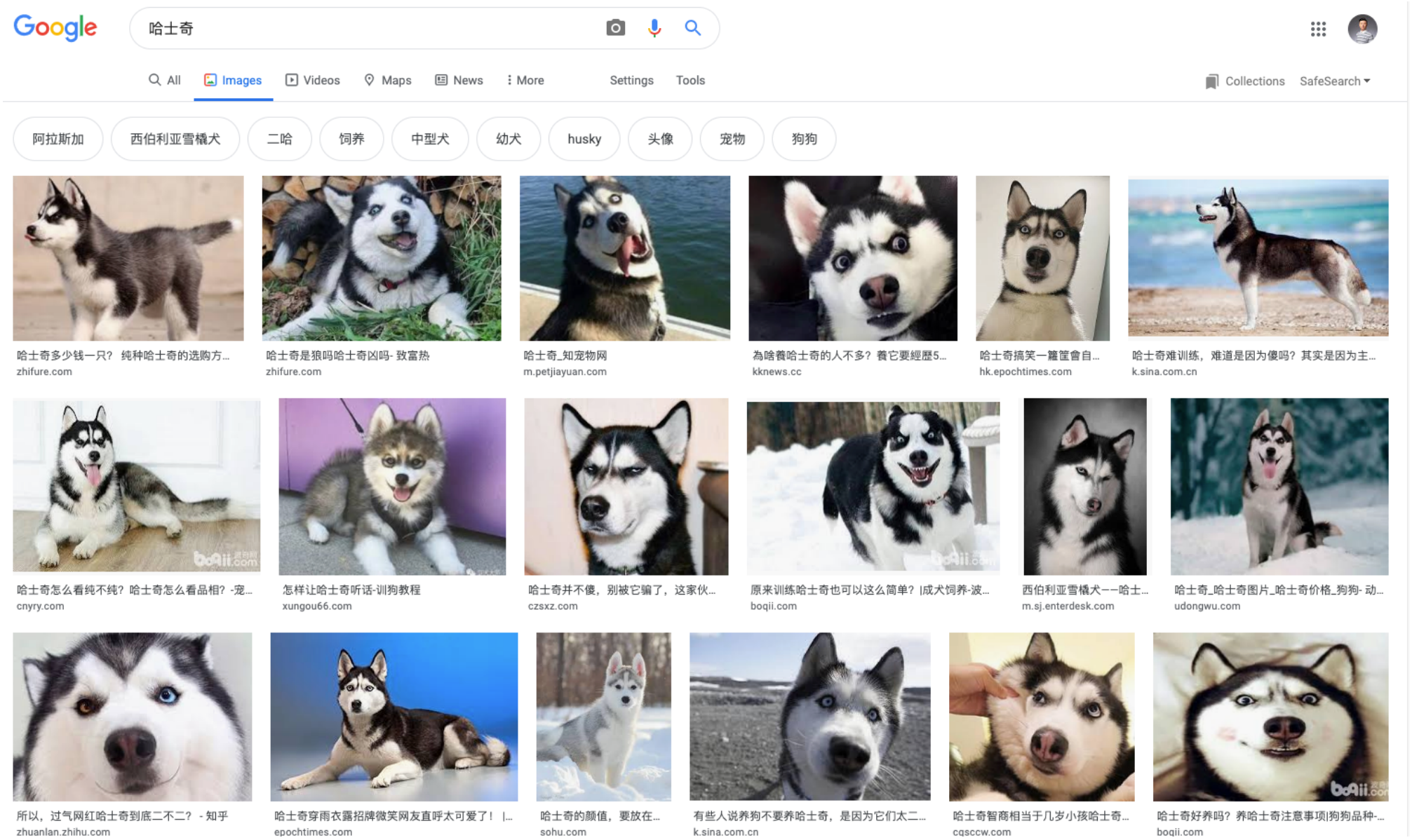

Husky Dog Concept Closeup ...
123rf.com

图像分类问题

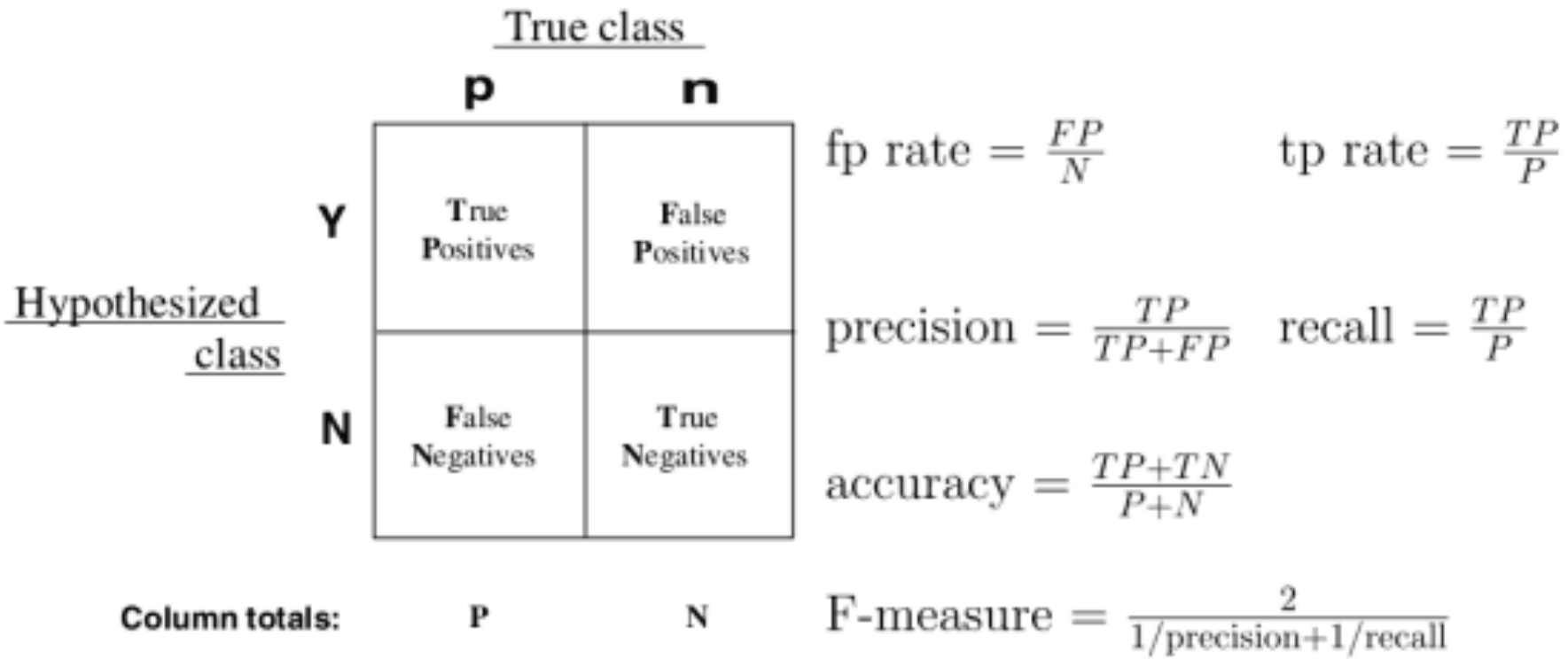
实例级分类 ——————→ 识别问题



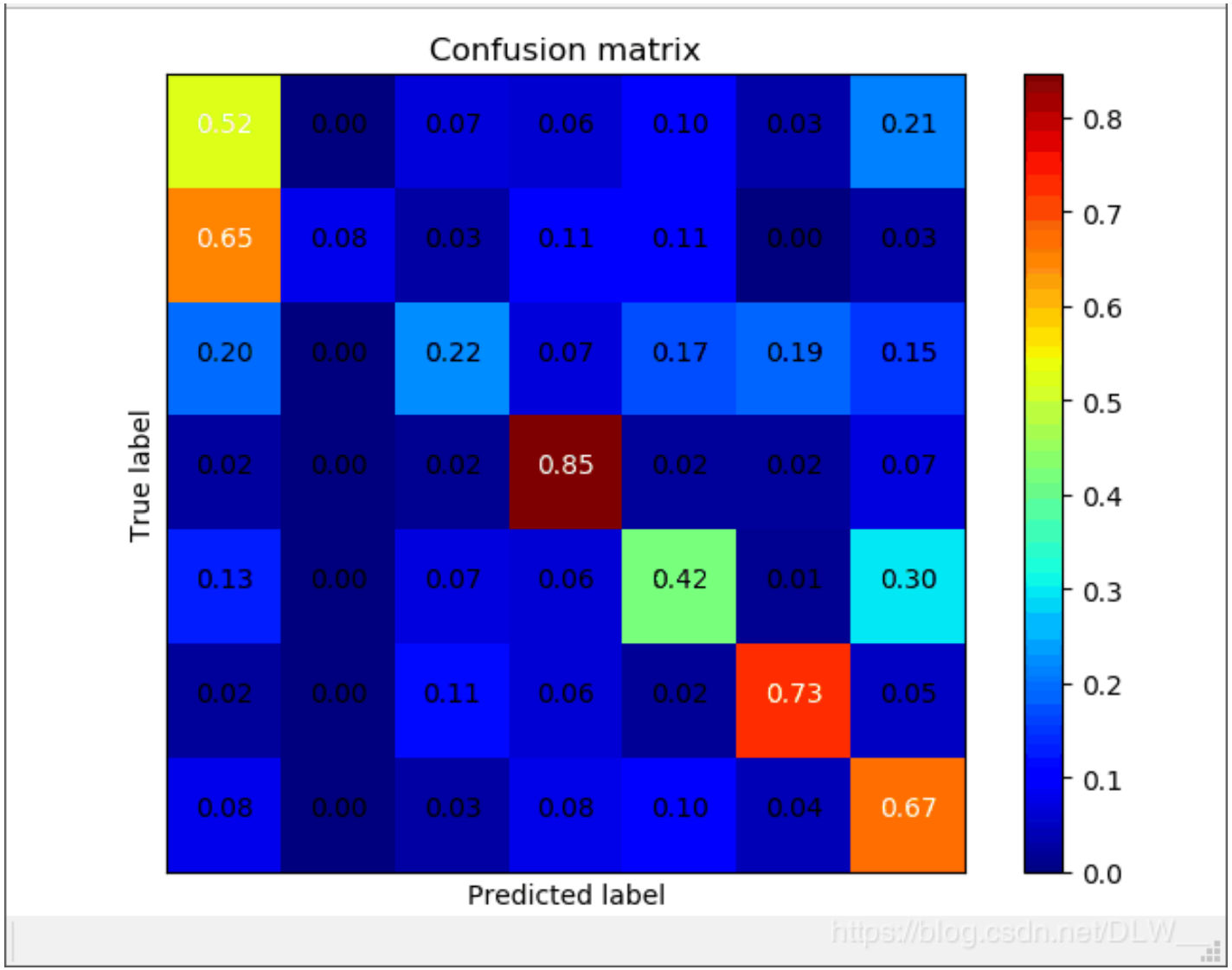
图像分类评估：Top-1 / Top-5 准确率



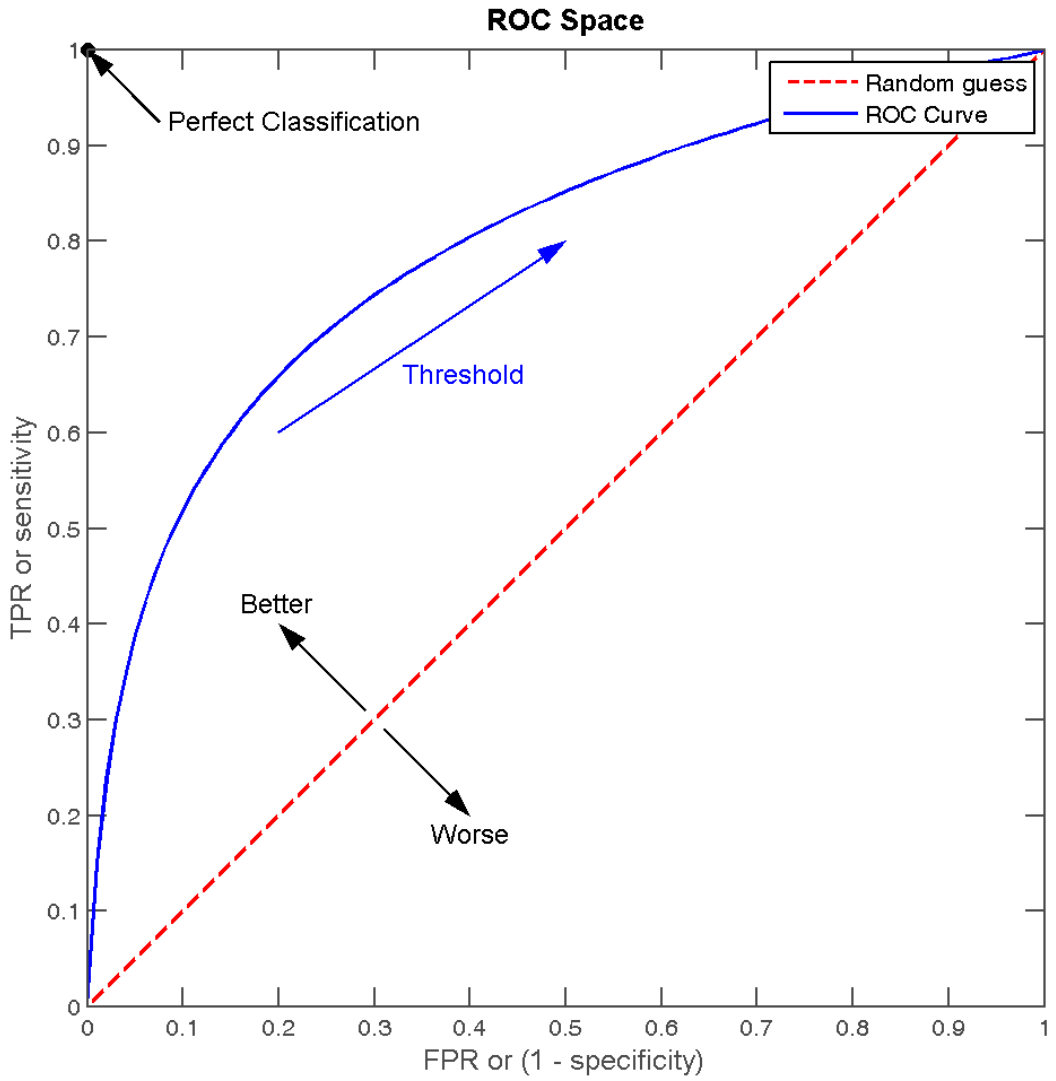
图像分类评估：混淆矩阵


True class
p
n
Hypothesized class
Y
N
True Positives
False Positives
False Negatives
True Negatives
Column totals:
P
N
fp rate = $\frac{FP}{N}$
tp rate = $\frac{TP}{P}$
precision = $\frac{TP}{TP+FP}$
recall = $\frac{TP}{P}$
accuracy = $\frac{TP+TN}{P+N}$
F-measure = $\frac{2}{1/\text{precision}+1/\text{recall}}$

图像分类评估：混淆矩阵


Confusion matrix
0.52 0.00 0.07 0.06 0.10 0.03 0.21
0.65 0.08 0.03 0.11 0.11 0.00 0.03
0.20 0.00 0.22 0.07 0.17 0.19 0.15
0.02 0.00 0.02 0.85 0.02 0.02 0.07
0.13 0.00 0.07 0.06 0.42 0.01 0.30
0.02 0.00 0.11 0.06 0.02 0.73 0.05
0.08 0.00 0.03 0.08 0.10 0.04 0.67
True label
Predicted label
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
https://blog.csdn.net/DLW

图像分类评估：ROC 曲线



EDU

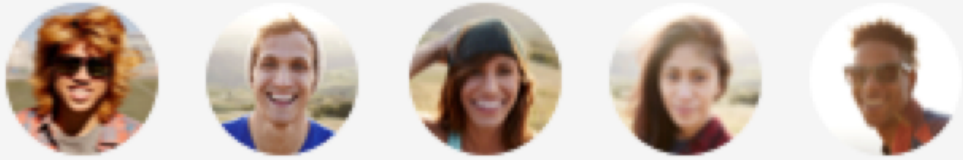
CSDN学院　IT实战派

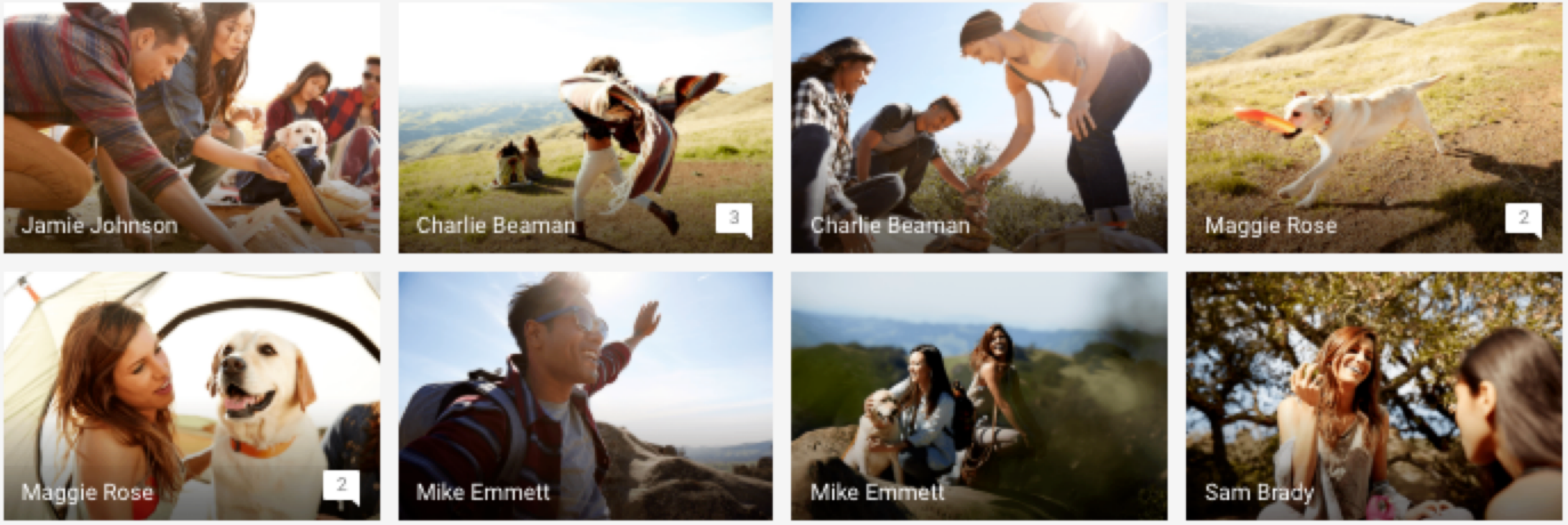
Python 深度学习

深度学习实战：图像分类模型的应用场景

讲师：彭靖田

图像分类应用：智能相册




Jamie Johnson
Charlie Beaman
3
Charlie Beaman
Maggie Rose
2
Maggie Rose
2
Mike Emmett
Mike Emmett
Sam Brady

图像分类应用：以图搜图


Google
1.jpg
二 哈
全部
图片
购物
更多
设置
工具
找到约 25,270,000,000 条结果 （用时 0.92 秒）
图片尺寸:
320 × 240
查找该图片的其他尺寸:
全部尺寸 - 小尺寸 - 中尺寸
可能相关的搜索查询： 二 哈
www.youtube.com › watch 转为简体网页
哈士奇最新搞笑Top10～二哈的十個搞笑鏡頭～ - YouTube
2016年12月2日 - 哈士奇最新搞笑Top10～二哈的十個搞笑鏡頭～. 驚奇大排行. Loading...
Unsubscribe from 驚奇大排行? Cancel Unsubscribe. Working.
hashiqi.goupu.com.cn › news › show-2921
哈士奇和二哈其实是不同的品种，你养的是啥?_哈士奇新闻_狗 ...
2019年10月25日 - 我们都知道"二哈"是我们对于哈士奇的一个"爱称"，这个名字也能非常好的体现哈士奇的二和傻，但是，你们知道么?其实哈士奇跟二哈不是一个"品种" ...
外观类似的图片
Evernote
Sign in to Web Clipper to see Related Results
哈士奇
犬
哈士奇是北方地區雪橇型狗的總稱，他們的快速拉動的風格與其他雪橇犬不同。 他們是由速度最快的狗不斷交配的品種。相比之下，阿拉斯加雪橇犬是「最大和最有力的」雪橇犬，常被用於較重的承載。哈士奇被用於雪橇犬拉車比賽。近幾年來，有些業者一直在為冒險旅行者推銷狗拉雪橇的雪地旅遊徒步旅行。 维基百科
寿命： 西伯利亞哈士奇: 10 – 15 年， 薩摩耶犬: 12 – 13 年， 迷你哈士奇: 12 – 16 年， 美國愛斯基摩犬: 16 年， 銀狐犬: 10 – 16 年
体重： 西伯利亞哈士奇: 20 – 27 公斤， 薩摩耶犬: 20 – 30 公斤， 更多
身高： 西伯利亞哈士奇: 53 – 60 厘米， 薩摩耶犬: 53 – 60 厘米， 更多
代表性品种
查看更多項目 (超過 5 項)
西伯利亞哈士奇
薩摩耶犬
阿拉斯加雪橇犬
迷你哈士奇
馬更些河哈士奇

图像分类应用：瑕疵检测

EDU

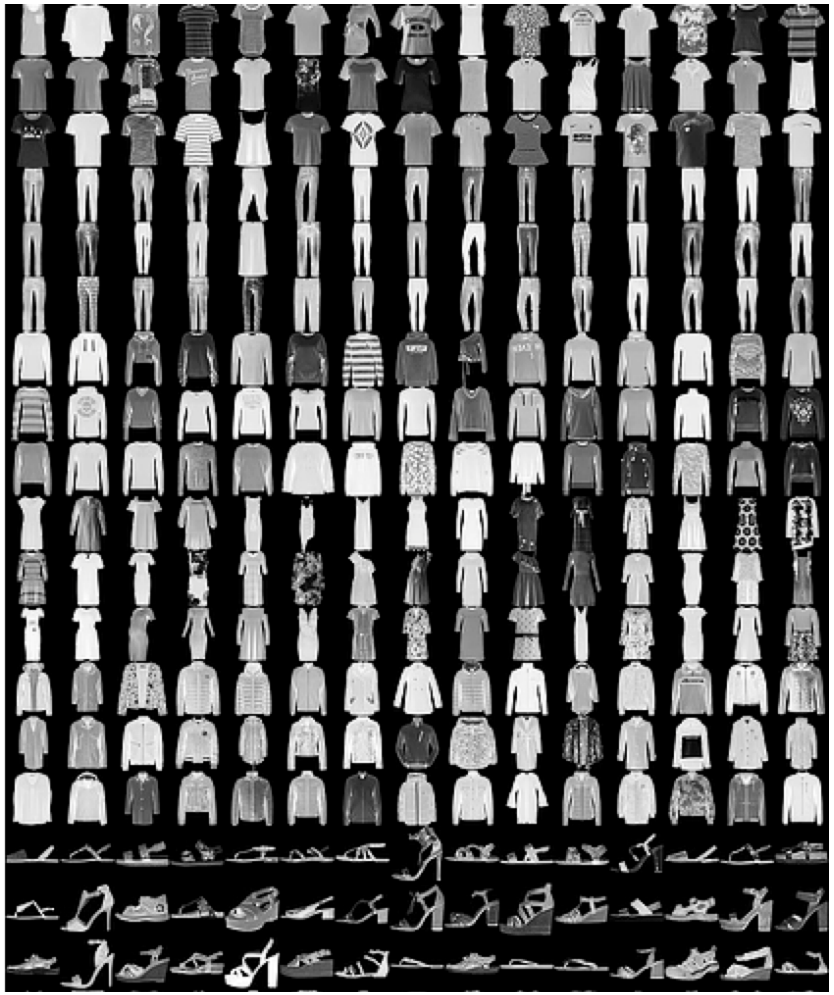
CSDN学院　IT实战派

Python 深度学习

深度学习实战：图像分类常用数据集介绍

讲师：彭靖田

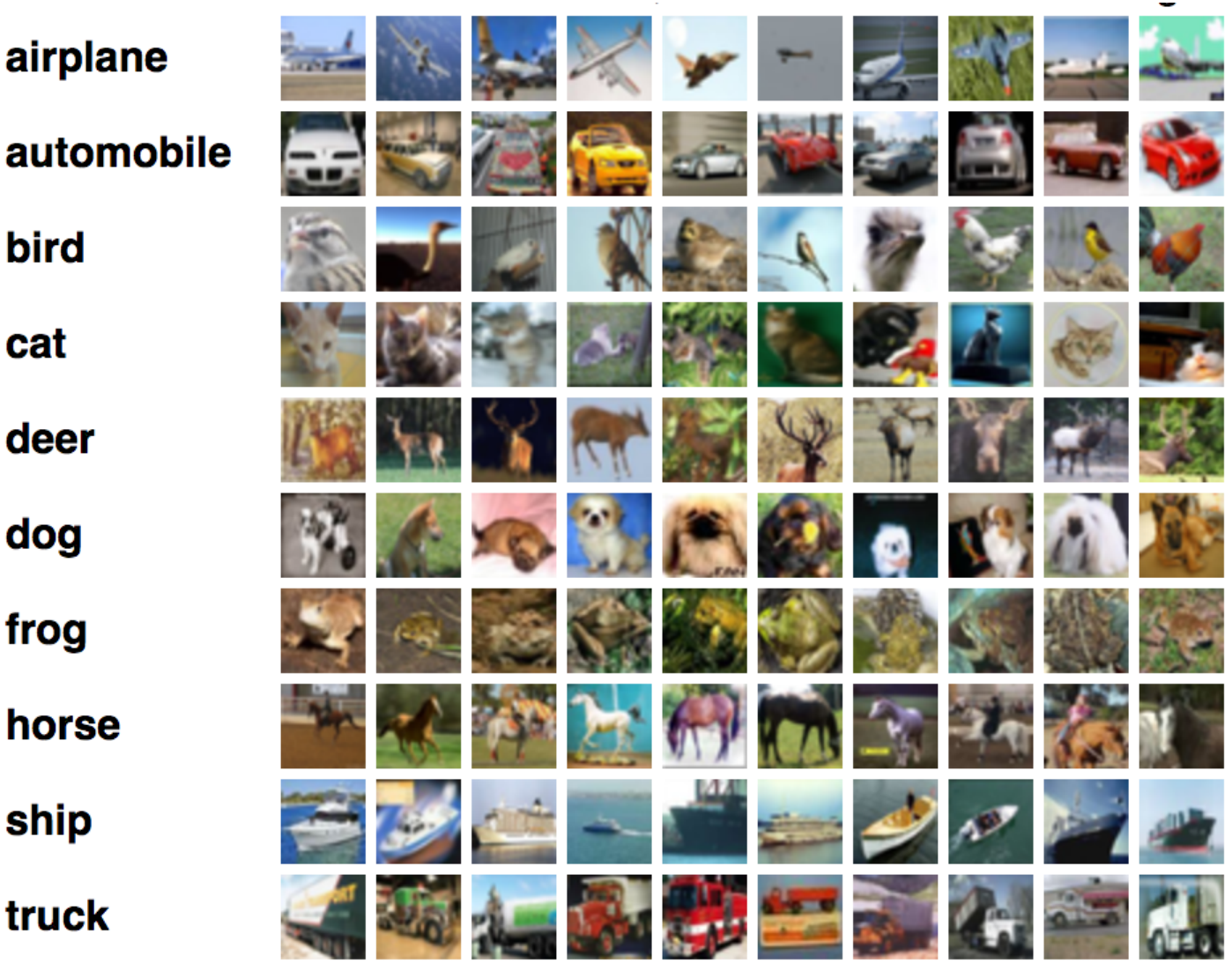
MNIST & Fashion-MNIST



http://yann.lecun.com/exdb/mnist/

https://github.com/zalandoresearch/fashion-mnist

CIFAR-10 & CIFAR-100



https://www.cs.utoronto.ca/~kriz/cifar.html

ImageNet

http://image-net.org/

Caltech 101 & Caltech 256

http://www.vision.caltech.edu/Image_Datasets/Caltech256/

EDU

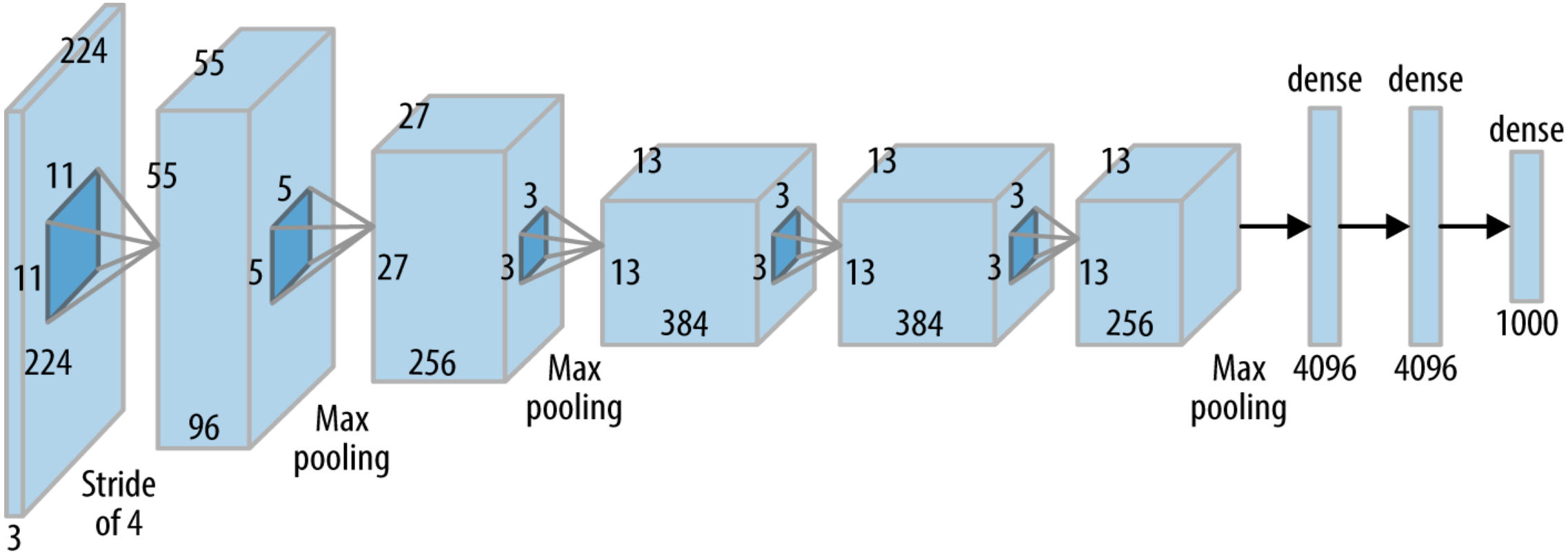
CSDN学院　IT实战派

Python 深度学习

深度学习实战：“重量级”图像分类模型概述

讲师：彭靖田

AlexNet (2012)



AlexNet (2012)

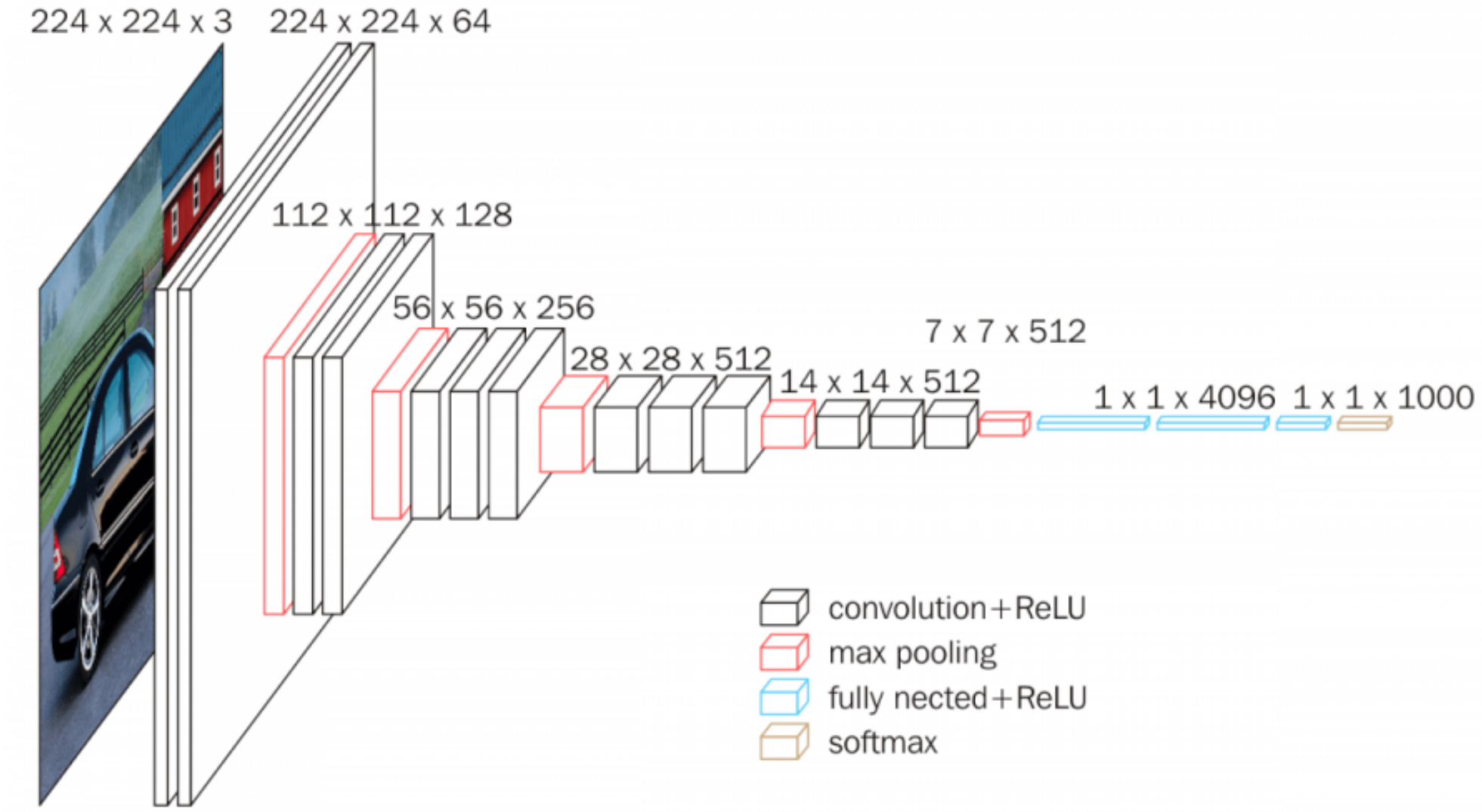
AlexNet Network - Structural Details													
Input			Output			Layer	Stride	Pad	Kernel size		in	out	# of Param
227	227	3	55	55	96	conv1	4	0	11	11	3	96	34944
55	55	96	27	27	96	maxpool1	2	0	3	3	96	96	0
27	27	96	27	27	256	conv2	1	2	5	5	96	256	614656
27	27	256	13	13	256	maxpool2	2	0	3	3	256	256	0
13	13	256	13	13	384	conv3	1	1	3	3	256	384	885120
13	13	384	13	13	384	conv4	1	1	3	3	384	384	1327488
13	13	384	13	13	256	conv5	1	1	3	3	384	256	884992
13	13	256	6	6	256	maxpool5	2	0	3	3	256	256	0
						fc6			1	1	9216	4096	37752832
						fc7			1	1	4096	4096	16781312
						fc8			1	1	4096	1000	4097000
Total													**62,378,344**

AlexNet (2012)

Model	Top-1	Top-5
Sparse coding [2]	*47.1%*	*28.2%*
SIFT + FVs [24]	*45.7%*	*25.7%*
CNN	**37.5%**	**17.0%**

Table 1: Comparison of results on ILSVRC-2010 test set. In *italics* are best results achieved by others.

VGGNet (2014)

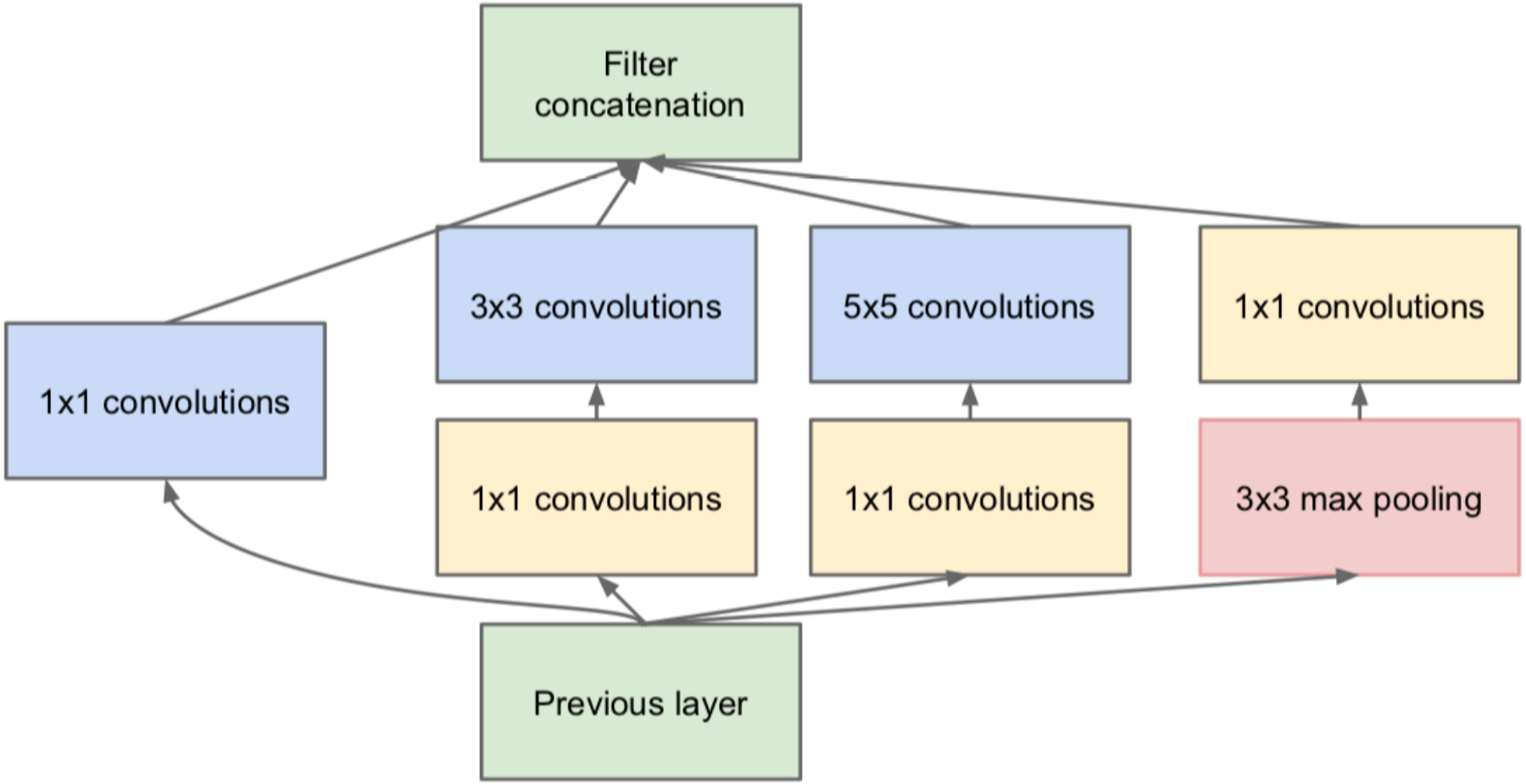

224 x 224 x 3
224 x 224 x 64
112 x 112 x 128
56 x 56 x 256
28 x 28 x 512
14 x 14 x 512
7 x 7 x 512
1 x 1 x 4096
1 x 1 x 1000
convolution+ReLU
max pooling
fully nected+ReLU
softmax

VGGNet (2014)

Table 7: **Comparison with the state of the art in ILSVRC classification**. Our method is denoted as "VGG". Only the results obtained without outside training data are reported.

Method	top-1 val. error (%)	top-5 val. error (%)	top-5 test error (%)
VGG (2 nets, multi-crop & dense eval.)	**23.7**	**6.8**	**6.8**
VGG (1 net, multi-crop & dense eval.)	24.4	7.1	7.0
VGG (ILSVRC submission, 7 nets, dense eval.)	24.7	7.5	7.3
GoogLeNet (Szegedy et al., 2014) (1 net)	-	7.9	
GoogLeNet (Szegedy et al., 2014) (7 nets)	-	**6.7**	
MSRA (He et al., 2014) (11 nets)	-	-	8.1
MSRA (He et al., 2014) (1 net)	27.9	9.1	9.1
Clarifai (Russakovsky et al., 2014) (multiple nets)	-	-	11.7
Clarifai (Russakovsky et al., 2014) (1 net)	-	-	12.5
Zeiler & Fergus (Zeiler & Fergus, 2013) (6 nets)	36.0	14.7	14.8
Zeiler & Fergus (Zeiler & Fergus, 2013) (1 net)	37.5	16.0	16.1
OverFeat (Sermanet et al., 2014) (7 nets)	34.0	13.2	13.6
OverFeat (Sermanet et al., 2014) (1 net)	35.7	14.2	-
Krizhevsky et al. (Krizhevsky et al., 2012) (5 nets)	38.1	16.4	16.4
Krizhevsky et al. (Krizhevsky et al., 2012) (1 net)	40.7	18.2	-

GoogLeNet/Inception (2014)



GoogLeNet/Inception (2014)

Network	Models Evaluated	Crops Evaluated	Top-1 Error	Top-5 Error
VGGNet [18]	2	-	23.7%	6.8%
GoogLeNet [20]	7	144	-	6.67%
PReLU [6]	-	-	-	4.94%
BN-Inception [7]	6	144	20.1%	4.9%
Inception-v3	4	144	**17.2%**	**3.58%** *

Table 5. Ensemble evaluation results comparing multi-model, multi-crop reported results. Our numbers are compared with the best published ensemble inference results on the ILSVRC 2012 classification benchmark. *All results, but the top-5 ensemble result reported are on the validation set. The ensemble yielded 3.46% top-5 error on the validation set.

ResNet (2015)

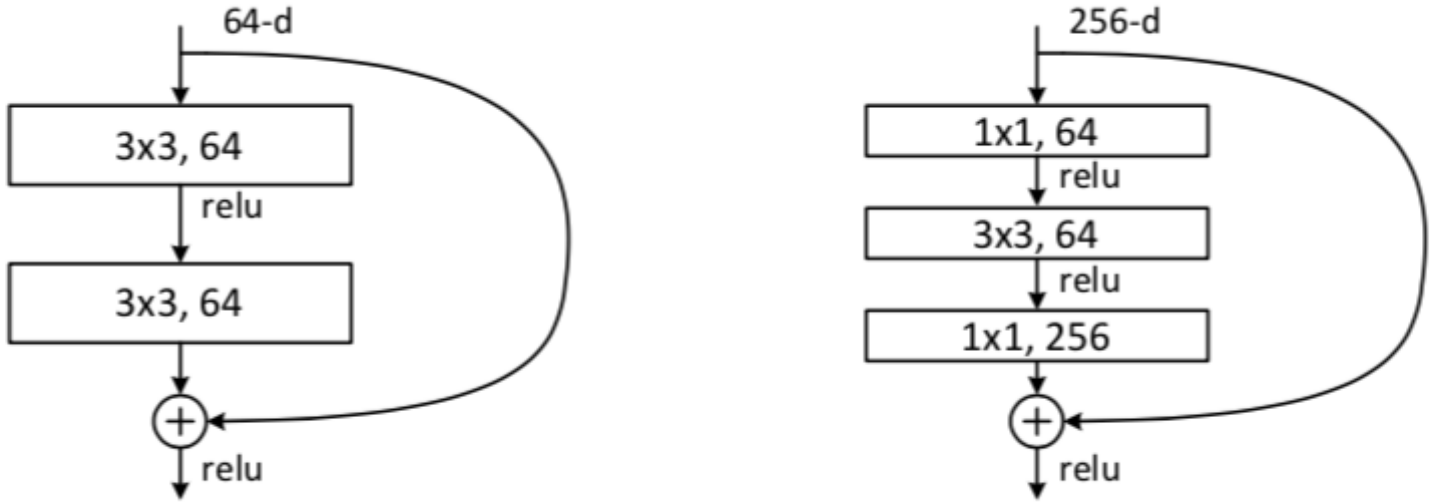


Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.

ResNet (2015)

model	top-1 err.	top-5 err.
VGG-16 [41]	28.07	9.33
GoogLeNet [44]	-	9.15
PReLU-net [13]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

Table 3. Error rates (%, **10-crop** testing) on ImageNet validation. VGG-16 is based on our test. ResNet-50/101/152 are of option B that only uses projections for increasing dimensions.

历年 SOTA 模型对比

Comparison					
Network	Year	Salient Feature	top5 accuracy	Parameters	FLOP
AlexNet	2012	Deeper	84.70%	62M	1.5B
VGGNet	2014	Fixed-size kernels	92.30%	138M	19.6B
Inception	2014	Wider - Parallel kernels	93.30%	6.4M	2B
ResNet-152	2015	Shortcut connections	95.51%	60.3M	11B

EDU

CSDN学院　IT实战派

Python 深度学习

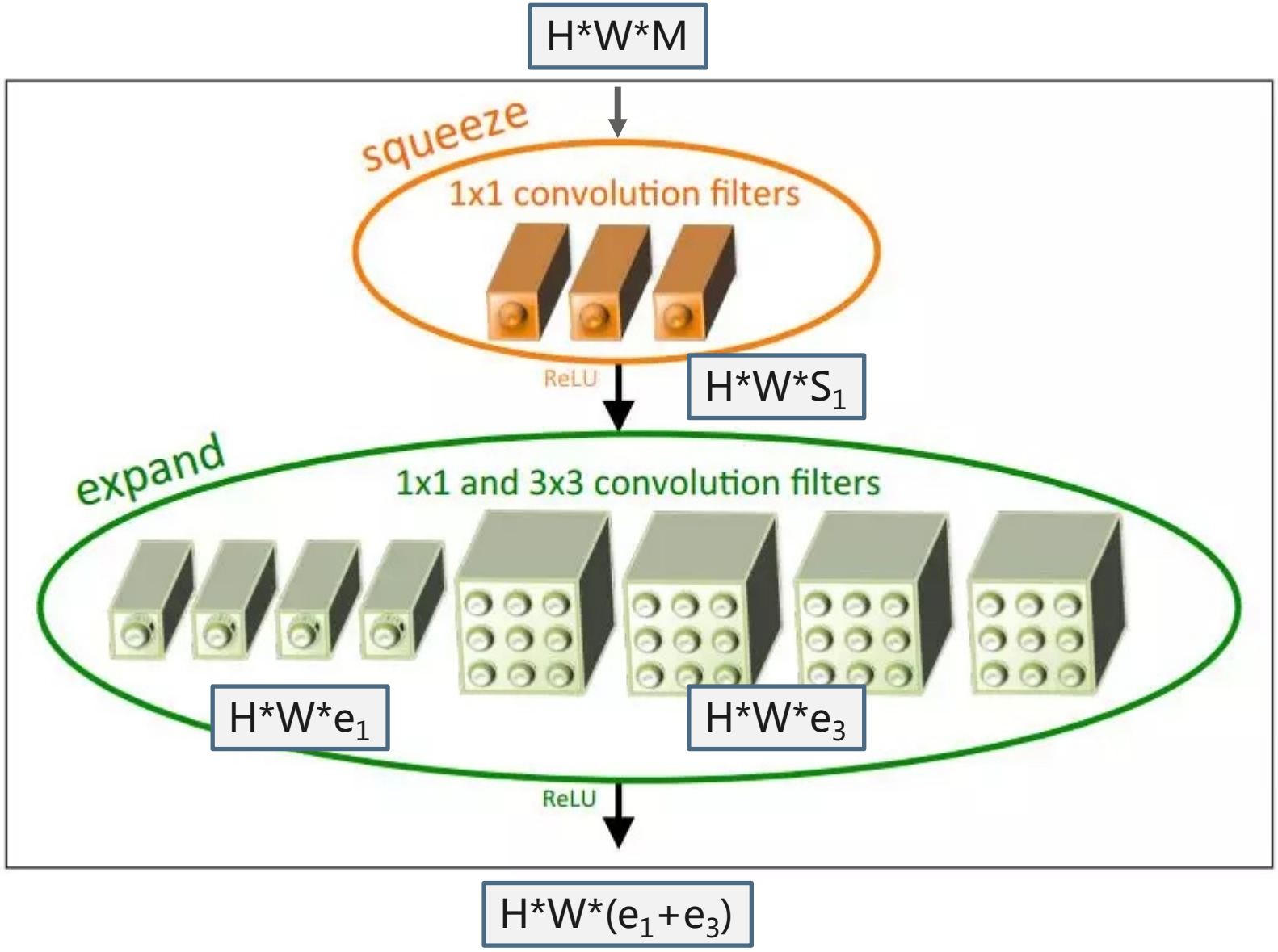
深度学习实战：“轻量级”图像分类模型概述

讲师：彭靖田

轻量级模型对比

网络	最早公开日期	发表情况	作者团队	论文链接
SqueezeNet	2016.02	ICLR-2017	伯克利&斯坦福	https://arxiv.org/abs/1602.07360
MobileNet	2016.04	CVPR-2017	Google	https://arxiv.org/abs/1704.04861
Xception	2016.10	CVPR-2017	Google	https://arxiv.org/abs/1610.02357

SqueezeNet


H*W*M
squeeze
1x1 convolution filters
ReLU
H*W*S1
expand
1x1 and 3x3 convolution filters
H*W*e1
H*W*e3
ReLU
H*W*(e1+e3)

SqueezeNet

Table 1: SqueezeNet architectural dimensions. (The formatting of this table was inspired by the Inception2 paper (Ioffe & Szegedy, 2015).)

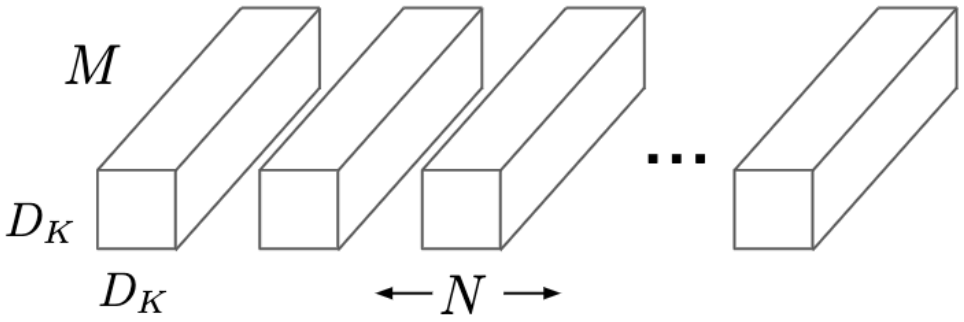
layer name/type	output size	filter size / stride (if not a fire layer)	depth	s_{1x1} (#1x1 squeeze)	e_{1x1} (#1x1 expand)	e_{3x3} (#3x3 expand)	s_{1x1} sparsity	e_{1x1} sparsity	e_{3x3} sparsity	# bits	#parameter before pruning	#parameter after pruning
input image	224x224x3										-	-
conv1	111x111x96	7x7/2 (x96)	1				100% (7x7)			6bit	14,208	14,208
maxpool1	55x55x96	3x3/2	0									
fire2	55x55x128		2	16	64	64	100%	100%	**33%**	6bit	11,920	5,746
fire3	55x55x128		2	16	64	64	100%	100%	**33%**	6bit	12,432	6,258
fire4	55x55x256		2	32	128	128	100%	100%	**33%**	6bit	45,344	20,646
maxpool4	27x27x256	3x3/2	0									
fire5	27x27x256		2	32	128	128	100%	100%	**33%**	6bit	49,440	24,742
fire6	27x27x384		2	48	192	192	100%	**50%**	**33%**	6bit	104,880	44,700
fire7	27x27x384		2	48	192	192	**50%**	100%	**33%**	6bit	111,024	46,236
fire8	27x27x512		2	64	256	256	100%	**50%**	**33%**	6bit	188,992	77,581
maxpool8	13x12x512	3x3/2	0									
fire9	13x13x512		2	64	256	256	**50%**	100%	**30%**	6bit	197,184	77,581
conv10	13x13x1000	1x1/1 (x1000)	1				**20%** (3x3)			6bit	513,000	103,400
avgpool10	1x1x1000	13x13/1	0									
	activations		parameters				compression info				1,248,424 (total)	**421,098** (total)

SqueezeNet

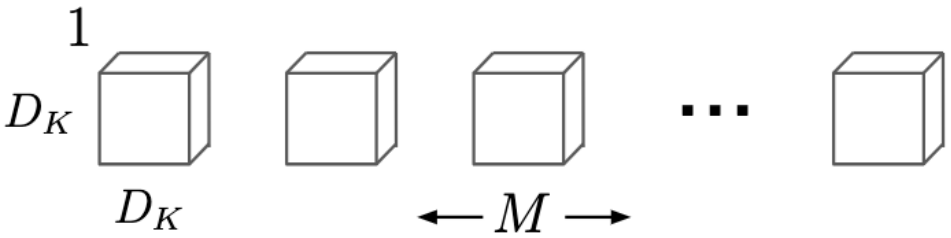
Table 2: Comparing SqueezeNet to model compression approaches. By *model size*, we mean the number of bytes required to store all of the parameters in the trained model.

CNN architecture	Compression Approach	Data Type	Original → Compressed Model Size	Reduction in Model Size vs. AlexNet	Top-1 ImageNet Accuracy	Top-5 ImageNet Accuracy
AlexNet	None (baseline)	32 bit	240MB	1x	57.2%	80.3%
AlexNet	SVD (Denton et al., 2014)	32 bit	240MB → 48MB	5x	56.0%	79.4%
AlexNet	Network Pruning (Han et al., 2015b)	32 bit	240MB → 27MB	9x	57.2%	80.3%
AlexNet	Deep Compression (Han et al., 2015a)	5-8 bit	240MB → 6.9MB	35x	57.2%	80.3%
SqueezeNet (ours)	None	32 bit	4.8MB	**50x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	8 bit	4.8MB → 0.66MB	**363x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	6 bit	4.8MB → 0.47MB	**510x**	57.5%	80.3%

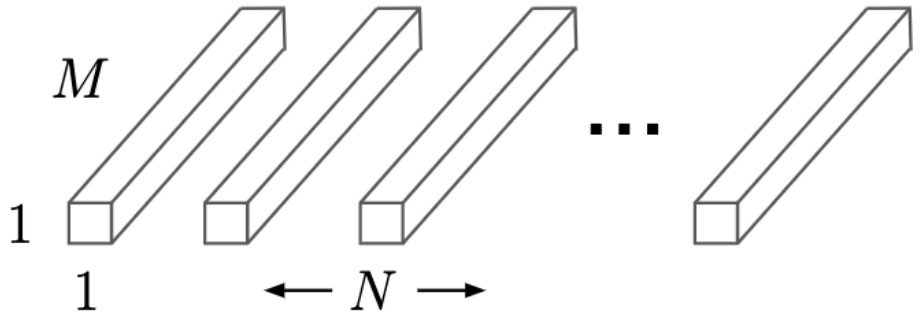
MobileNet



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Figure 2. The standard convolutional filters in (a) are replaced by two layers: depthwise convolution in (b) and pointwise convolution in (c) to build a depthwise separable filter.

$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F} = \frac{1}{N} + \frac{1}{D_K^2}$$

MobileNet

Table 8. MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

Table 9. Smaller MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
0.50 MobileNet-160	60.2%	76	1.32
Squeezenet	57.5%	1700	1.25
AlexNet	57.2%	720	60

Xception

Figure 1. A canonical Inception module (Inception V3).

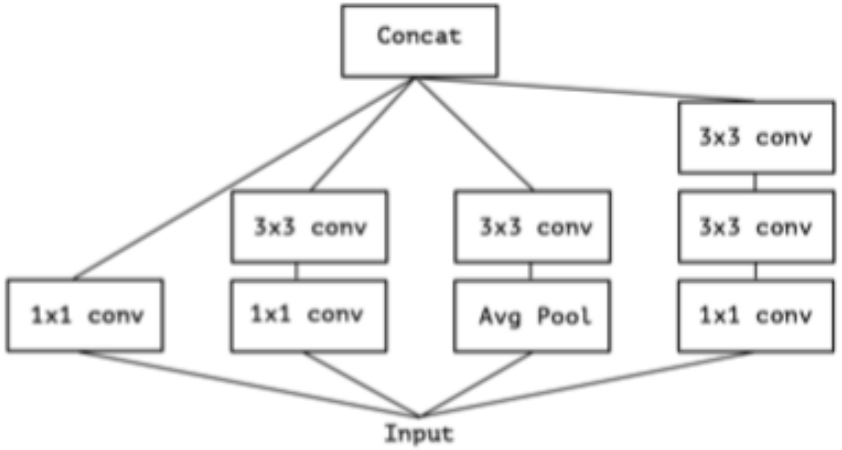


Figure 2. A simplified Inception module.

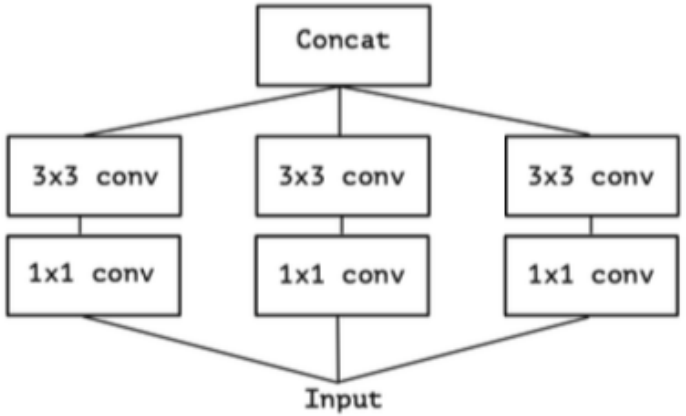
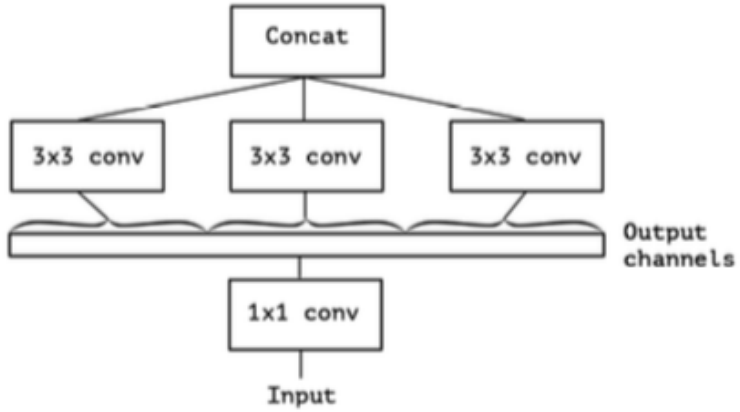
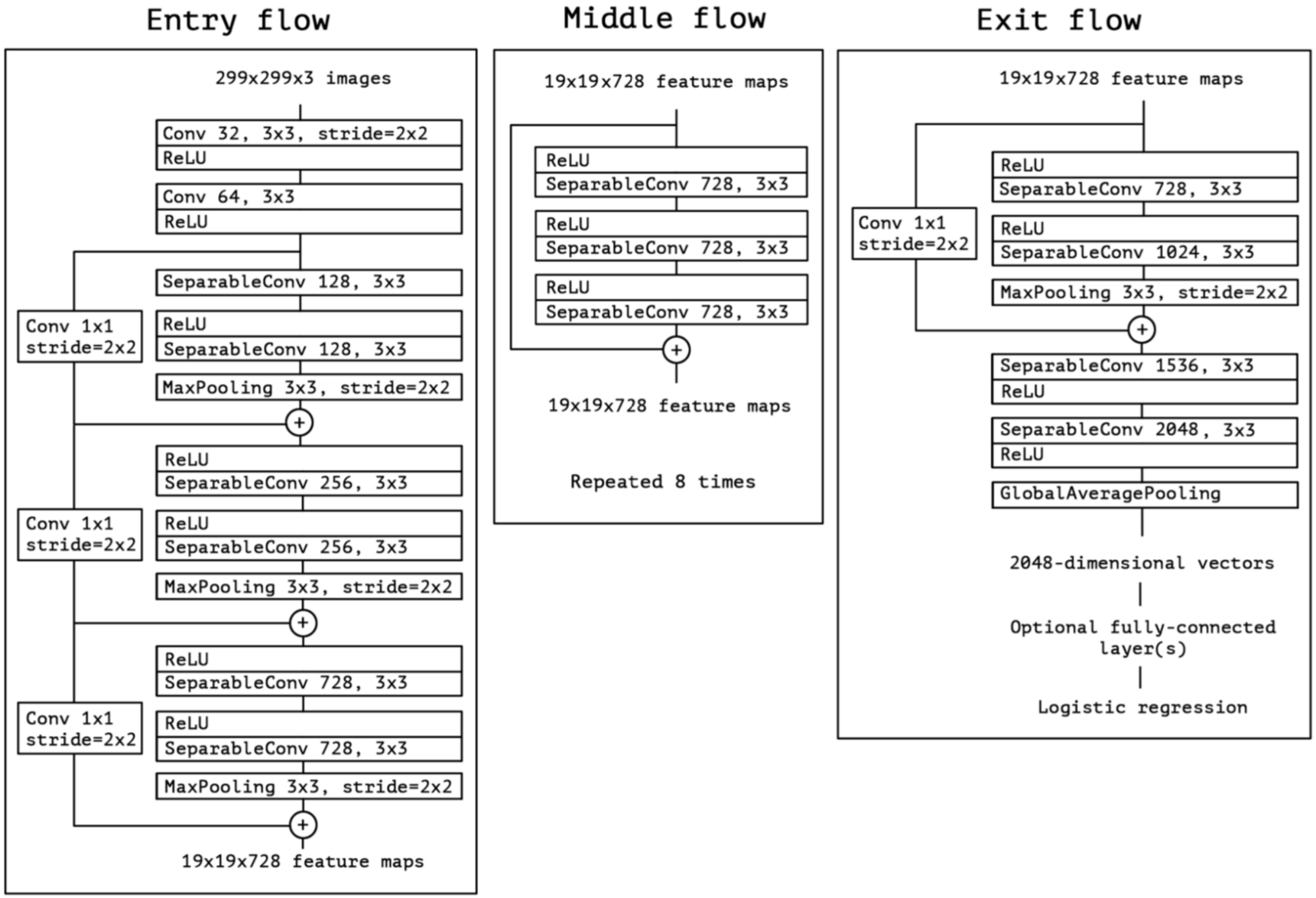


Figure 3. A strictly equivalent reformulation of the simplified Inception module.



Xception


Entry flow
299x299x3 images
Conv 32, 3x3, stride=2x2
ReLU
Conv 64, 3x3
ReLU
SeparableConv 128, 3x3
Conv 1x1 stride=2x2
ReLU
SeparableConv 128, 3x3
MaxPooling 3x3, stride=2x2
ReLU
SeparableConv 256, 3x3
Conv 1x1 stride=2x2
ReLU
SeparableConv 256, 3x3
MaxPooling 3x3, stride=2x2
ReLU
SeparableConv 728, 3x3
Conv 1x1 stride=2x2
ReLU
SeparableConv 728, 3x3
MaxPooling 3x3, stride=2x2
19x19x728 feature maps
Middle flow
19x19x728 feature maps
ReLU
SeparableConv 728, 3x3
ReLU
SeparableConv 728, 3x3
ReLU
SeparableConv 728, 3x3
19x19x728 feature maps
Repeated 8 times
Exit flow
19x19x728 feature maps
Conv 1x1 stride=2x2
ReLU
SeparableConv 728, 3x3
ReLU
SeparableConv 1024, 3x3
MaxPooling 3x3, stride=2x2
SeparableConv 1536, 3x3
ReLU
SeparableConv 2048, 3x3
ReLU
GlobalAveragePooling
2048-dimensional vectors
Optional fully-connected layer(s)
Logistic regression

Xception

Table 1. Classification performance comparison on ImageNet (single crop, single model). VGG-16 and ResNet-152 numbers are only included as a reminder. The version of Inception V3 being benchmarked does not include the auxiliary tower.

	Top-1 accuracy	**Top-5 accuracy**
VGG-16	0.715	0.901
ResNet-152	0.770	0.933
Inception V3	0.782	0.941
Xception	**0.790**	**0.945**

EDU

CSDN学院　IT实战派

Python 深度学习

深度学习实战：图像分类 MobileNets 系列模型发展

讲师：彭靖田

MobileNetV1

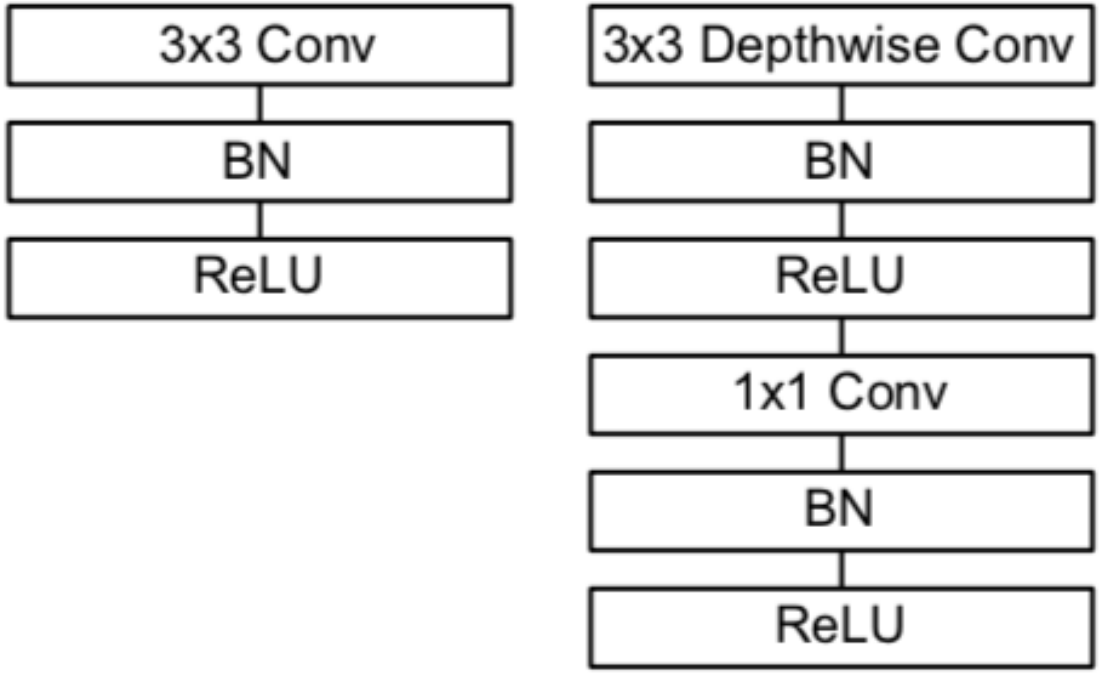


Figure 3. Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.

MobileNetV2

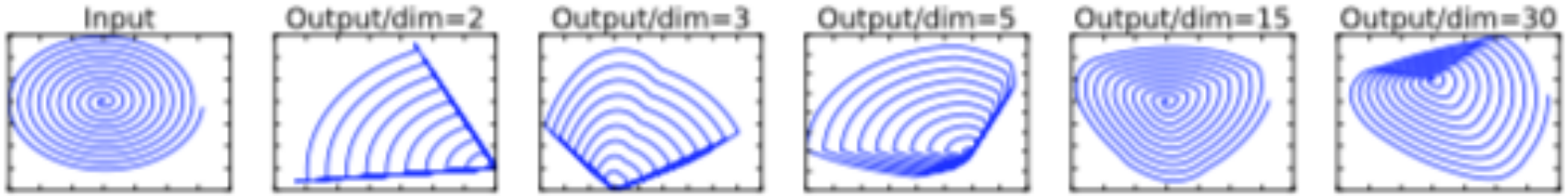
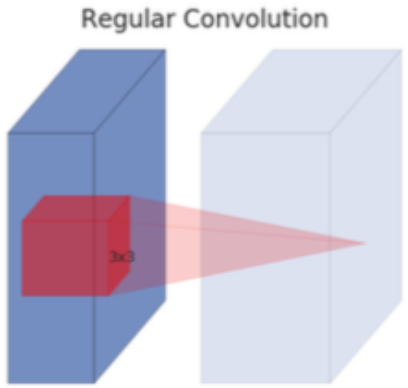


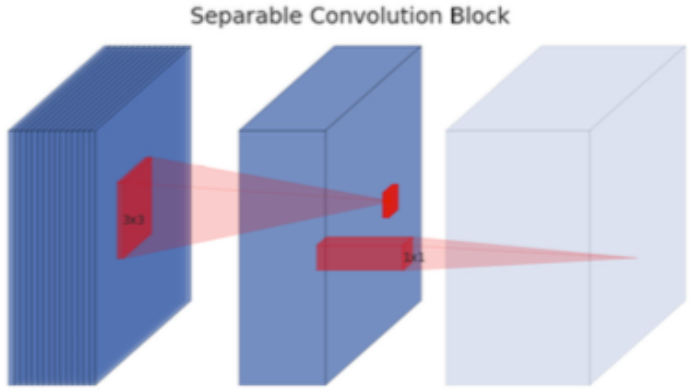
Figure 1: Examples of ReLU transformations of low-dimensional manifolds embedded in higher-dimensional spaces. In these examples the initial spiral is embedded into an n-dimensional space using random matrix T followed by ReLU, and then projected back to the 2D space using T^{-1}. In examples above $n = 2, 3$ result in information loss where certain points of the manifold collapse into each other, while for $n = 15$ to 30 the transformation is highly non-convex.

MobileNetV2

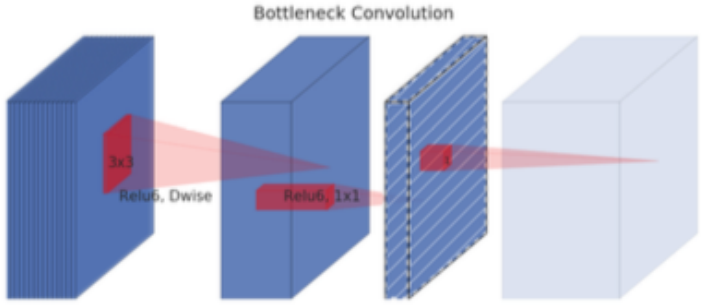
(a) Regular



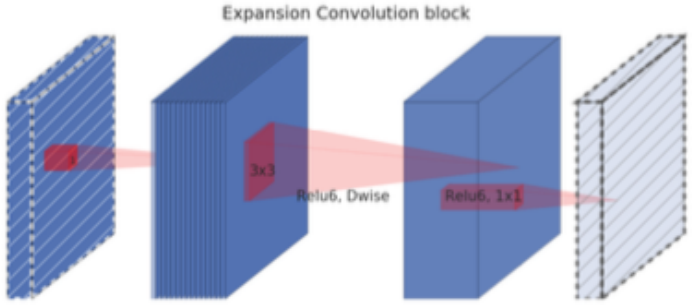
(b) Separable



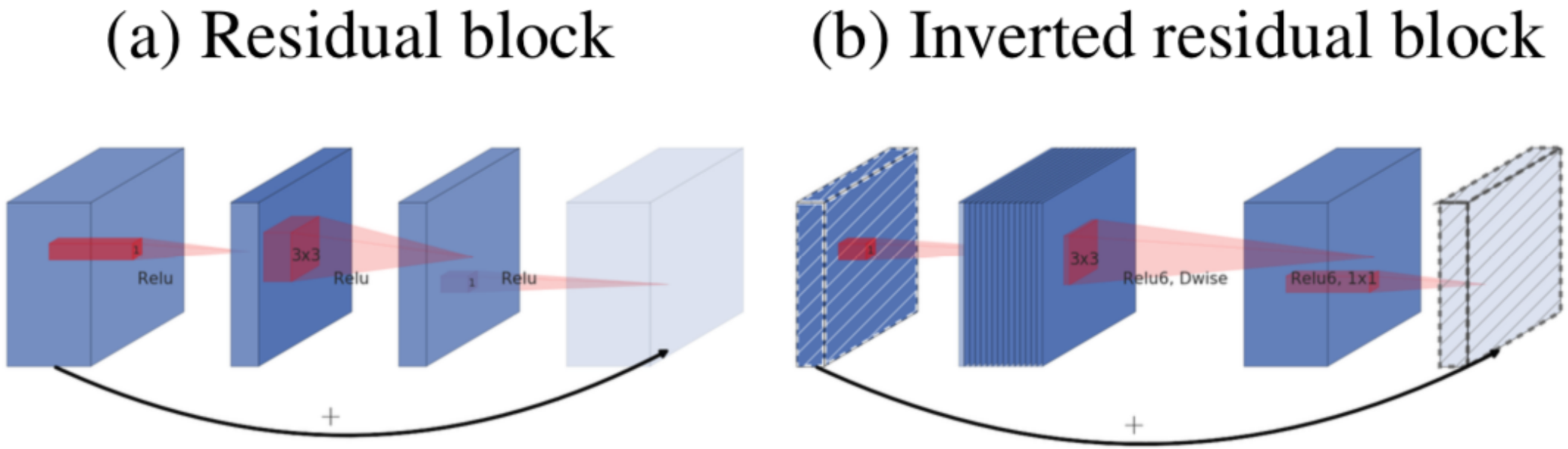
(c) Separable with linear bottleneck



(d) Bottleneck with expansion layer



MobileNetV2

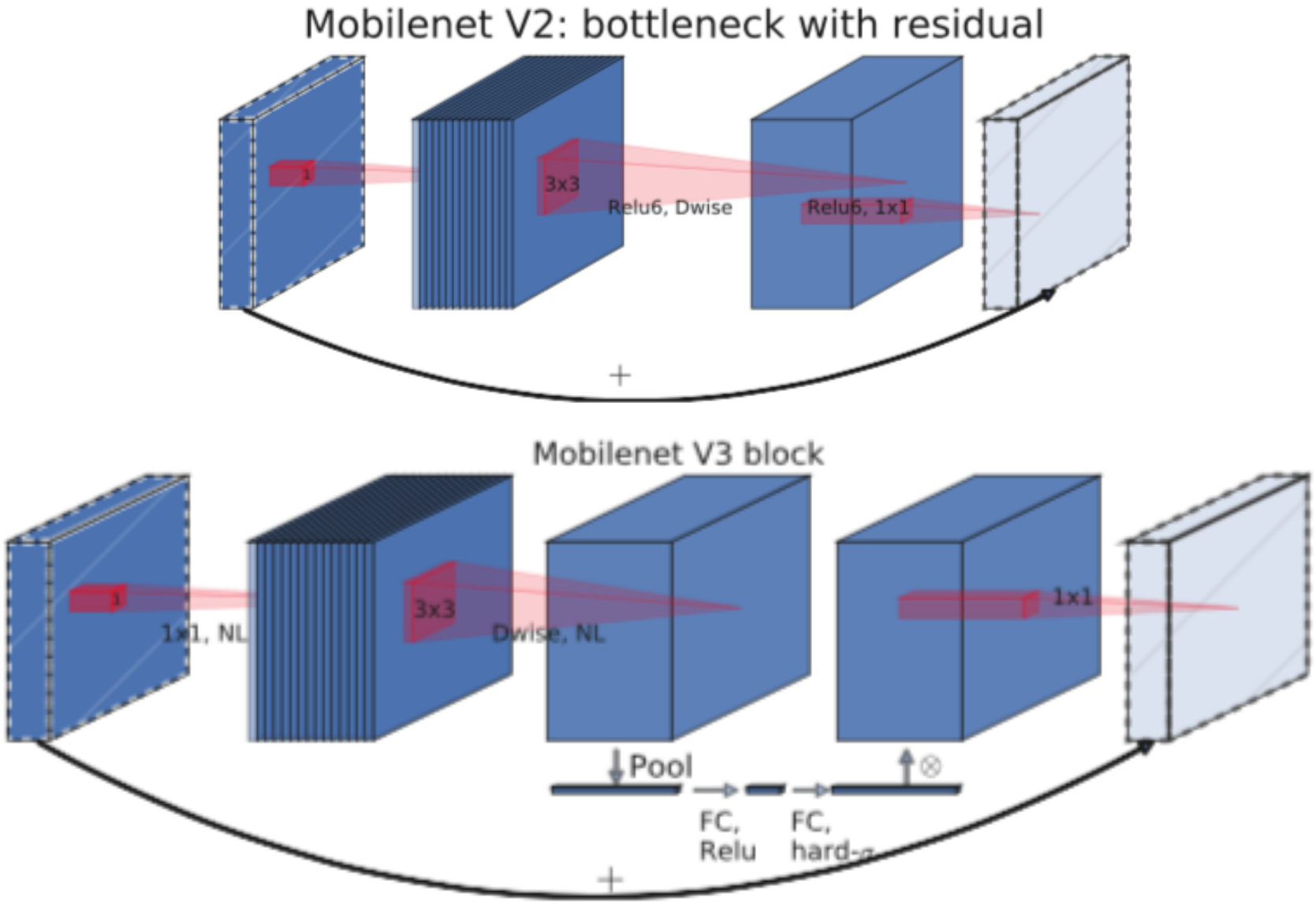


MobileNetV2

Network	Top 1	Params	MAdds	CPU
MobileNetV1	70.6	4.2M	575M	113ms
ShuffleNet (1.5)	71.5	**3.4M**	292M	-
ShuffleNet (x2)	73.7	5.4M	524M	-
NasNet-A	74.0	5.3M	564M	183ms
MobileNetV2	**72.0**	**3.4M**	**300M**	**75ms**
MobileNetV2 (1.4)	**74.7**	6.9M	585M	**143ms**

Table 4: Performance on ImageNet, comparison for different networks. As is common practice for ops, we count the total number of Multiply-Adds. In the last column we report running time in milliseconds (ms) for a single large core of the Google Pixel 1 phone (using TF-Lite). We do not report ShuffleNet numbers as efficient group convolutions and shuffling are not yet supported.

MobileNetV3



MobileNetV3

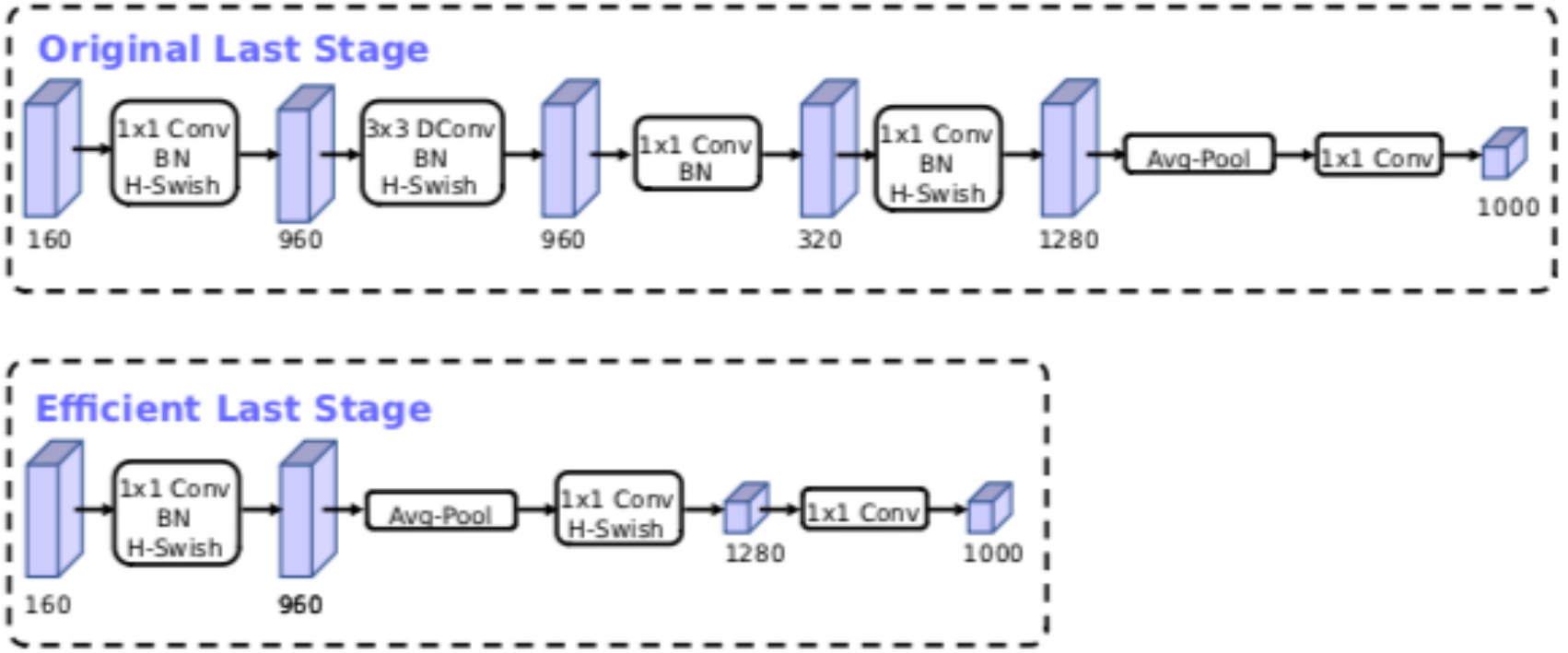


Figure 5. Comparison of original last stage and efficient last stage. This more efficient last stage is able to drop three expensive layers at the end of the network at no loss of accuracy.

MobileNetV3

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Table 2: MobileNetV2 : Each line describes a sequence of 1 or more identical (modulo stride) layers, repeated n times. All layers in the same sequence have the same number c of output channels. The first layer of each sequence has a stride s and all others use stride 1. All spatial convolutions use 3×3 kernels. The expansion factor t is always applied to the input size as described in Table 1.

Input	Operator	exp size	$\#out$	SE	NL	s
$224^2 \times 3$	conv2d, 3x3	-	16	-	HS	2
$112^2 \times 16$	bneck, 3x3	16	16	✓	RE	2
$56^2 \times 16$	bneck, 3x3	72	24	-	RE	2
$28^2 \times 24$	bneck, 3x3	88	24	-	RE	1
$28^2 \times 24$	bneck, 5x5	96	40	✓	HS	2
$14^2 \times 40$	bneck, 5x5	240	40	✓	HS	1
$14^2 \times 40$	bneck, 5x5	240	40	✓	HS	1
$14^2 \times 40$	bneck, 5x5	120	48	✓	HS	1
$14^2 \times 48$	bneck, 5x5	144	48	✓	HS	1
$14^2 \times 48$	bneck, 5x5	288	96	✓	HS	2
$7^2 \times 96$	bneck, 5x5	576	96	✓	HS	1
$7^2 \times 96$	bneck, 5x5	576	96	✓	HS	1
$7^2 \times 96$	conv2d, 1x1	-	576	✓	HS	1
$7^2 \times 576$	pool, 7x7	-	-	-	-	1
$1^2 \times 576$	conv2d 1x1, NBN	-	1024	-	HS	1
$1^2 \times 1024$	conv2d 1x1, NBN	-	k	-	-	1

Table 2. Specification for MobileNetV3-Small. See table 1 for notation.

MobileNetV3

Network	Top-1	MAdds	Params	P-1	P-2	P-3
V3-Large 1.0	**75.2**	**219**	5.4M	**51**	**61**	**44**
V3-Large 0.75	73.3	155	4.0M	39	46	40
MnasNet-A1	75.2	315	3.9M	71	86	61
Proxyless[5]	74.6	320	4.0M	72	84	60
V2 1.0	72.0	300	3.4M	64	76	56
V3-Small 1.0	**67.4**	**56**	2.5M	**15.8**	**19.4**	**14.4**
V3-Small 0.75	65.4	44	2.0M	12.8	15.6	11.7
Mnas-small [43]	64.9	65.1	1.9M	20.3	24.2	17.2
V2 0.35	60.8	59.2	1.6M	16.6	19.6	13.9

Table 3. Floating point performance on the Pixel family of phones (P-n denotes a Pixel-n phone). All latencies are in ms and are measured using a single large core with a batch size of one. Top-1 accuracy is on ImageNet.

EDU

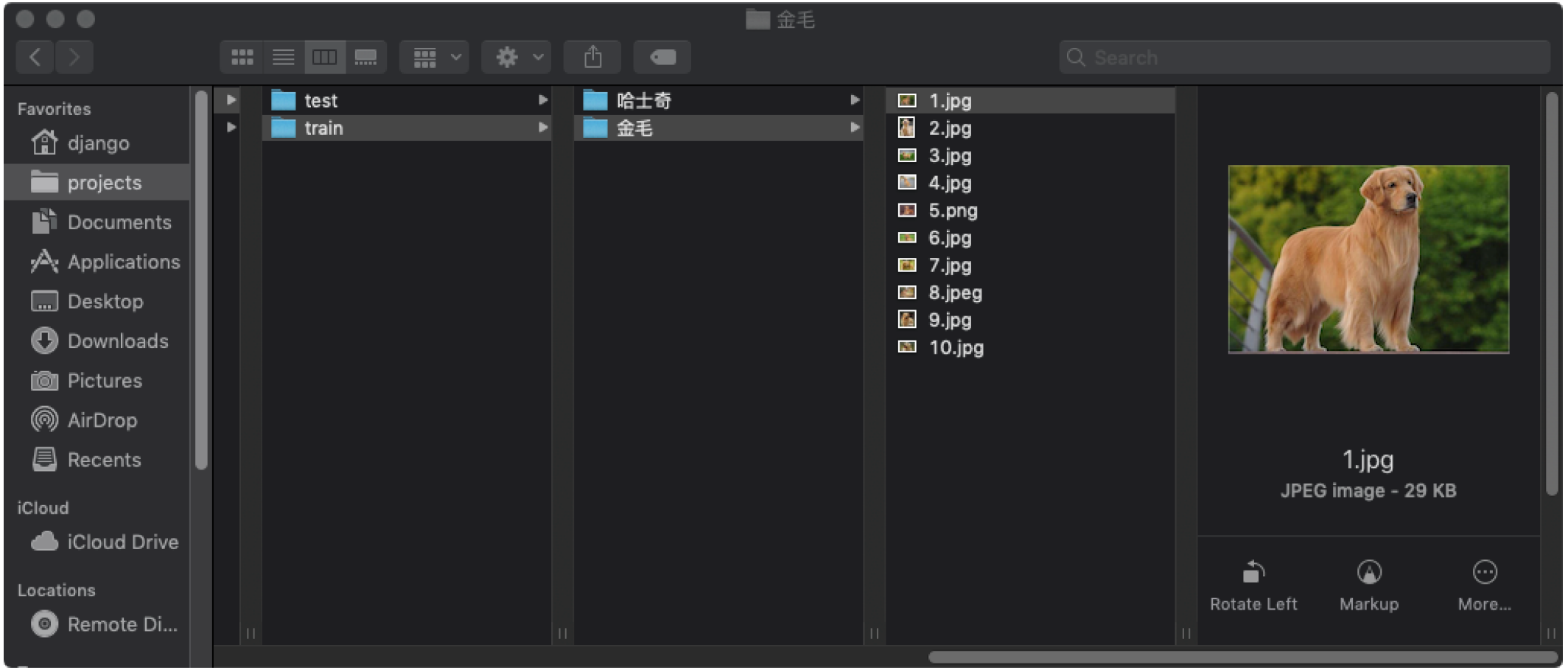
CSDN学院　IT实战派

Python 深度学习

深度学习实战：实战 Keras MobileNet 图像分类

讲师：彭靖田

MobileNet 迁移学习：准备数据



MobileNet 迁移学习：下载预训练模型

```
from keras.layers import Dense, GlobalAveragePooling2D
from keras.applications.mobilenet import MobileNet

base_model = MobileNet(include_top=False, weights='imagenet', classes=1000)
```

```
/anaconda/envs/py35/lib/python3.5/importlib/_bootstrap.py:222: RuntimeWarning: numpy.dtype size changed, may indicate bina
ry incompatibility. Expected 96, got 88
  return f(*args, **kwds)
/anaconda/envs/py35/lib/python3.5/site-packages/h5py/__init__.py:36: FutureWarning: Conversion of the second argument of i
ssubdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).typ
e`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
/anaconda/envs/py35/lib/python3.5/importlib/_bootstrap.py:222: RuntimeWarning: numpy.dtype size changed, may indicate bina
ry incompatibility. Expected 96, got 88
  return f(*args, **kwds)
/anaconda/envs/py35/lib/python3.5/importlib/_bootstrap.py:222: RuntimeWarning: numpy.dtype size changed, may indicate bina
ry incompatibility. Expected 96, got 88
  return f(*args, **kwds)
WARNING:tensorflow:From /anaconda/envs/py35/lib/python3.5/site-packages/tensorflow/python/framework/op_def_library.py:263:
colocate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
/anaconda/envs/py35/lib/python3.5/site-packages/keras_applications/mobilenet.py:208: UserWarning: MobileNet shape is undef
ined. Weights for input shape (224, 224) will be loaded.
  warnings.warn('MobileNet shape is undefined.'
```

MobileNet 迁移学习：新增全连接和Softmax

```
from keras.models import Model

x = base_model.output
x=GlobalAveragePooling2D()(x)
x=Dense(1024,activation='relu')(x)
x=Dense(1024,activation='relu')(x)
x=Dense(512,activation='relu')(x)
preds=Dense(2,activation='softmax')(x)

model=Model(inputs=base_model.input, outputs=preds)
```

MobileNet 迁移学习：构造训练数据生成器

```
from keras.preprocessing.image import ImageDataGenerator
from keras.applications.mobilenet import preprocess_input
from keras.preprocessing import image

import numpy as np

train_dir = 'data/train'
class_name = ["哈士奇", "金毛"]

def prepare_image(file):
    img_path = ''
    img = image.load_img(img_path + file, target_size=(224, 224)) # MobileNet 输入图像尺寸
    img_array = image.img_to_array(img)
    img_array_expanded_dims = np.expand_dims(img_array, axis=0)
    return preprocess_input(img_array_expanded_dims)


train_datagen=ImageDataGenerator(preprocessing_function=preprocess_input)

train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(224,224),
        color_mode='rgb',
        batch_size=10,
        class_mode='categorical',
        shuffle=True)
```

```
Found 20 images belonging to 2 classes.
```

MobileNet 迁移学习：Fine-Tune

```
history = model.fit_generator(
        train_generator,
        steps_per_epoch=10,
        epochs=10)
```

```
WARNING:tensorflow:From /anaconda/envs/py35/lib/python3.5/site-packages/tensorflow/python/ops/math_ops.py:3066: to_int32
(from tensorflow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/10
10/10 [==============================] - 33s 3s/step - loss: 0.7843 - acc: 0.7500
Epoch 2/10
10/10 [==============================] - 26s 3s/step - loss: 5.8115e-04 - acc: 1.0000
Epoch 3/10
10/10 [==============================] - 25s 3s/step - loss: 0.0043 - acc: 1.0000
Epoch 4/10
10/10 [==============================] - 25s 2s/step - loss: 0.0378 - acc: 0.9900
Epoch 5/10
10/10 [==============================] - 25s 2s/step - loss: 0.8159 - acc: 0.9400
Epoch 6/10
10/10 [==============================] - 25s 2s/step - loss: 0.1337 - acc: 0.9900
Epoch 7/10
10/10 [==============================] - 25s 3s/step - loss: 0.3819 - acc: 0.9700
Epoch 8/10
10/10 [==============================] - 25s 2s/step - loss: 0.0140 - acc: 1.0000
Epoch 9/10
10/10 [==============================] - 25s 3s/step - loss: 0.0080 - acc: 1.0000
Epoch 10/10
10/10 [==============================] - 25s 2s/step - loss: 1.2496e-04 - acc: 1.0000
```

MobileNet 迁移学习：新模型预测结果

```
from matplotlib import pyplot as plt

def load_image(img_path):
    img = image.load_img(img_path, target_size=(224, 224))
    img_tensor = image.img_to_array(img)
    img_tensor = np.expand_dims(img_tensor, axis=0)
    img_tensor /= 255.

    plt.imshow(img_tensor[0])
    plt.axis('off')
    plt.show()

    return img_tensor
```

```
img_path = 'data/test/1.jpg'
img_tensor = load_image(img_path)

pred = model.predict(img_tensor)

print([[class_name[i], prob] for i, prob in enumerate(pred[0])])
```

```
[['哈士奇', 2.5269136e-14], ['金毛', 1.0]]
```

Try it

EDU

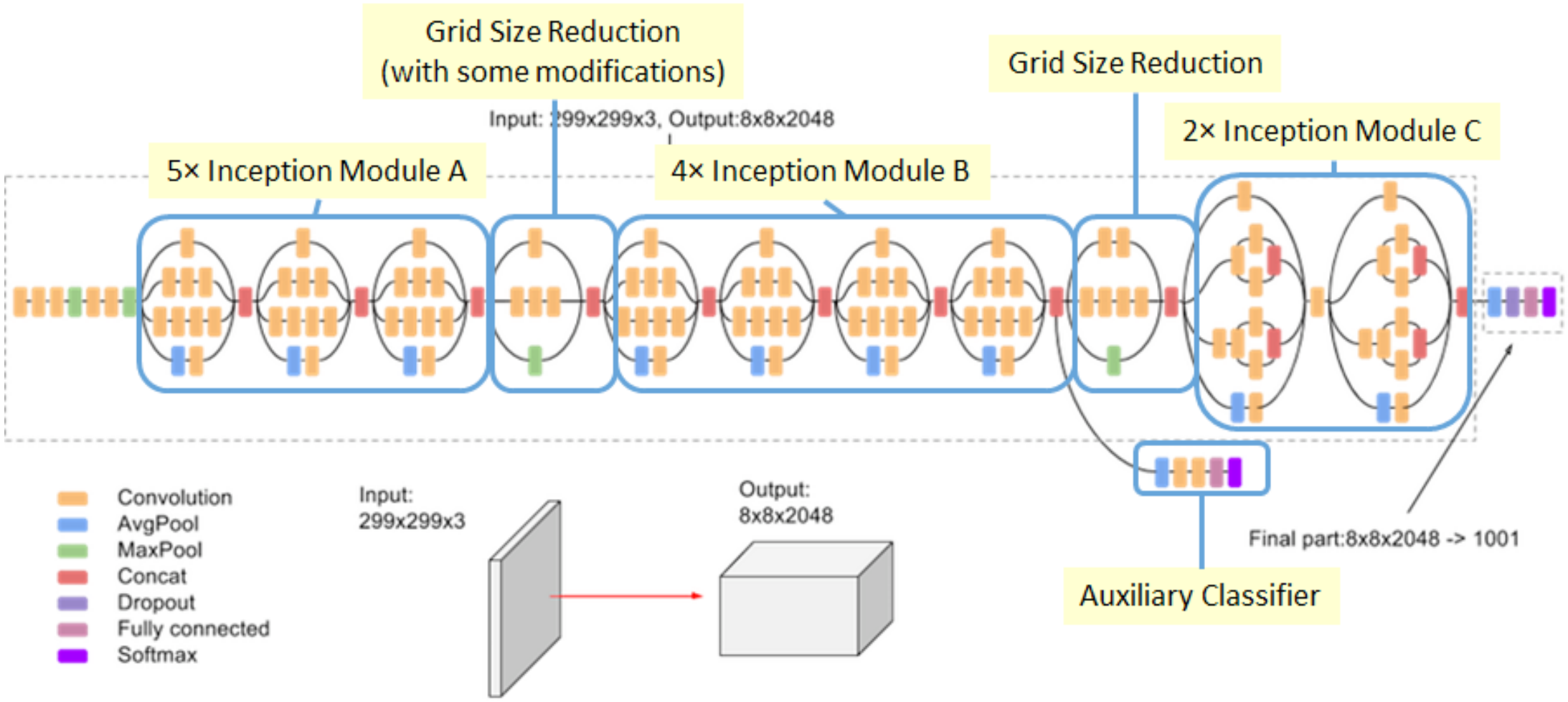
CSDN学院　IT实战派

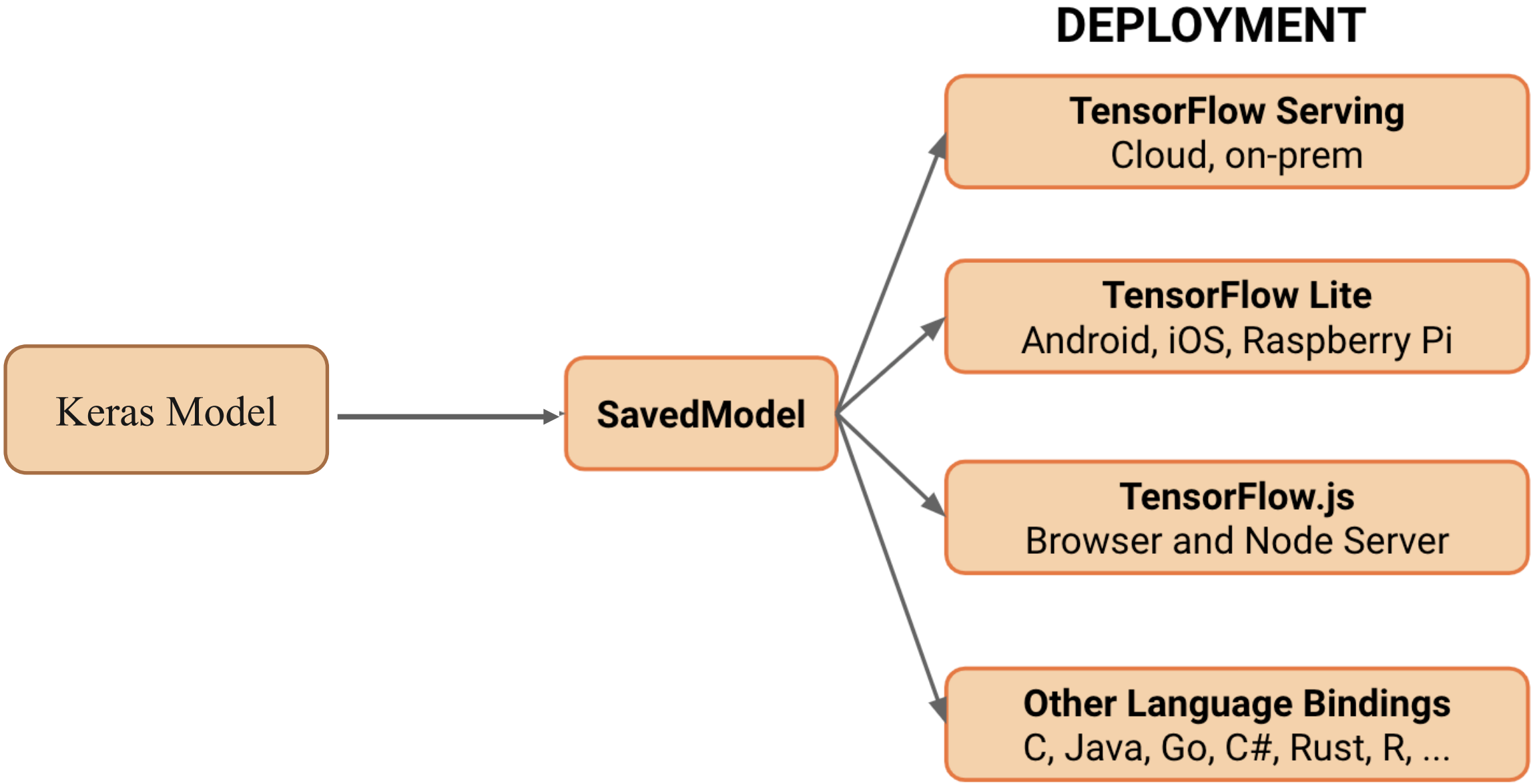
Python 深度学习

深度学习实战：实战 Keras 和 TensorFlow 模型格式转换

讲师：彭靖田

TensorFlow 模型标准 – Saved Model




DEPLOYMENT
Keras Model
SavedModel
TensorFlow Serving
Cloud, on-prem
TensorFlow Lite
Android, iOS, Raspberry Pi
TensorFlow.js
Browser and Node Server
Other Language Bindings
C, Java, Go, C#, Rust, R, ...

Try it

EDU

CSDN学院　IT实战派

Python 深度学习

深度学习实战：实战 TensorFlow Lite 模型格式转换

讲师：彭靖田

TensorFlow 2 转换 SavedModel 模型格式

```
import tensorflow as tf

# Construct a basic model.
root = tf.train.Checkpoint()
root.v1 = tf.Variable(3.)
root.v2 = tf.Variable(2.)
root.f = tf.function(lambda x: root.v1 * root.v2 * x)

# Save the model.
export_dir = "/tmp/test_saved_model"
input_data = tf.constant(1., shape=[1, 1])
to_save = root.f.get_concrete_function(input_data)
tf.saved_model.save(root, export_dir, to_save)

# Convert the model.
converter = tf.lite.TFLiteConverter.from_saved_model(export_dir)
tflite_model = converter.convert()
```

TensorFlow 2 转换 tf.keras 模型格式

```
import tensorflow as tf

# Create a simple Keras model.
x = [-1, 0, 1, 2, 3, 4]
y = [-3, -1, 1, 3, 5, 7]

model = tf.keras.models.Sequential(
    [tf.keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')
model.fit(x, y, epochs=50)

# Convert the model.
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()
```

TensorFlow 2 端到端转换 MobileNetV2 模型

```python
import numpy as np
import tensorflow as tf

# Load the MobileNet tf.keras model.
model = tf.keras.applications.MobileNetV2(
    weights="imagenet", input_shape=(224, 224, 3))

# Convert the model.
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

# Load TFLite model and allocate tensors.
interpreter = tf.lite.Interpreter(model_content=tflite_model)
interpreter.allocate_tensors()

# Get input and output tensors.
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

# Test the TensorFlow Lite model on random input data.
input_shape = input_details[0]['shape']
input_data = np.array(np.random.random_sample(input_shape), dtype=np.float32)
interpreter.set_tensor(input_details[0]['index'], input_data)

interpreter.invoke()

# The function `get_tensor()` returns a copy of the tensor data.
# Use `tensor()` in order to get a pointer to the tensor.
tflite_results = interpreter.get_tensor(output_details[0]['index'])

# Test the TensorFlow model on random input data.
tf_results = model(tf.constant(input_data))

# Compare the result.
for tf_result, tflite_result in zip(tf_results, tflite_results):
  np.testing.assert_almost_equal(tf_result, tflite_result, decimal=5)
```

TensorFlow Lite 使用示例

TensorFlow Lite converter

```
import tensorflow as tf
converter = tf.lite.TFLiteConverter.from_saved_model("/tmp/mobilenet")
tflite_model = converter.convert()
open("/tmp/converted_model.tflite", "wb").write(tflite_model)
```

TensorFlow Lite interpreter

```
try (Interpreter interpreter = new Interpreter("/tmp/converted_model.tflite")) {
  interpreter.run(input, output);
  // For models with multiple inputs or multiple outputs
  interpreter.runForMultipleInputsOutputs(inputs, map_of_indices_to_outputs);
}
```

EDU

CSDN学院　IT实战派

Python 深度学习

深度学习实战：搭建 TensorFlow Lite 模型运行环境

讲师：彭靖田

TensorFlow Lite 官网

https://www.tensorflow.org/lite/examples

Deploy machine learning models on mobile and IoT devices

TensorFlow Lite is an open source deep learning framework for on-device inference.

See the guide

Guides explain the concepts and components of TensorFlow Lite.

See examples

Explore TensorFlow Lite Android and iOS apps.

See models

Easily deploy pre-trained models.

How it works

Pick a model

Pick a new model or retrain an existing one.

Read the developer guide →

Convert

Convert a TensorFlow model into a compressed flat buffer with the TensorFlow Lite Converter.

Read the developer guide →

Deploy

Take the compressed .tflite file and load it into a mobile or embedded device.

Read the developer guide →

Optimize

Quantize by converting 32-bit floats to more efficient 8-bit integers or run on GPU.

Read the developer guide →

TensorFlow Lite Examples

Solutions to common problems

Explore optimized models to help with common mobile and edge use cases.

See all use cases →

Image classification

Identify hundreds of objects, including people, activities, animals, plants, and places.

Object detection

Detect multiple objects with bounding boxes. Yes, dogs and cats too.

Smart reply

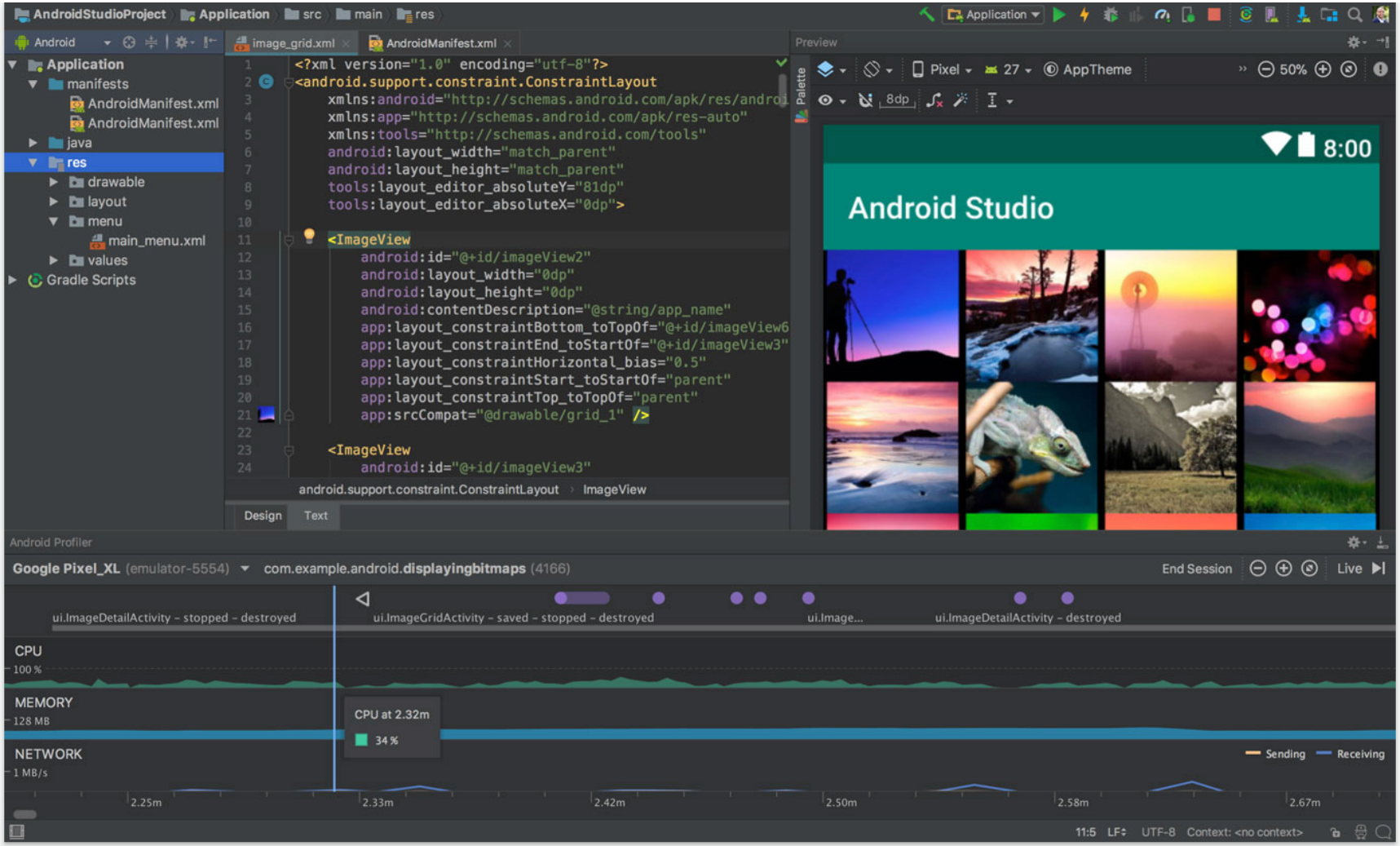
Generate reply suggestions to input conversational chat messages.

搭建 TensorFlow Lite 运行环境 (Android)

android studio

https://developer.android.com/studio

Android Studio provides the fastest tools for building apps on every type of Android device.



EDU

CSDN学院　IT实战派

Python 深度学习

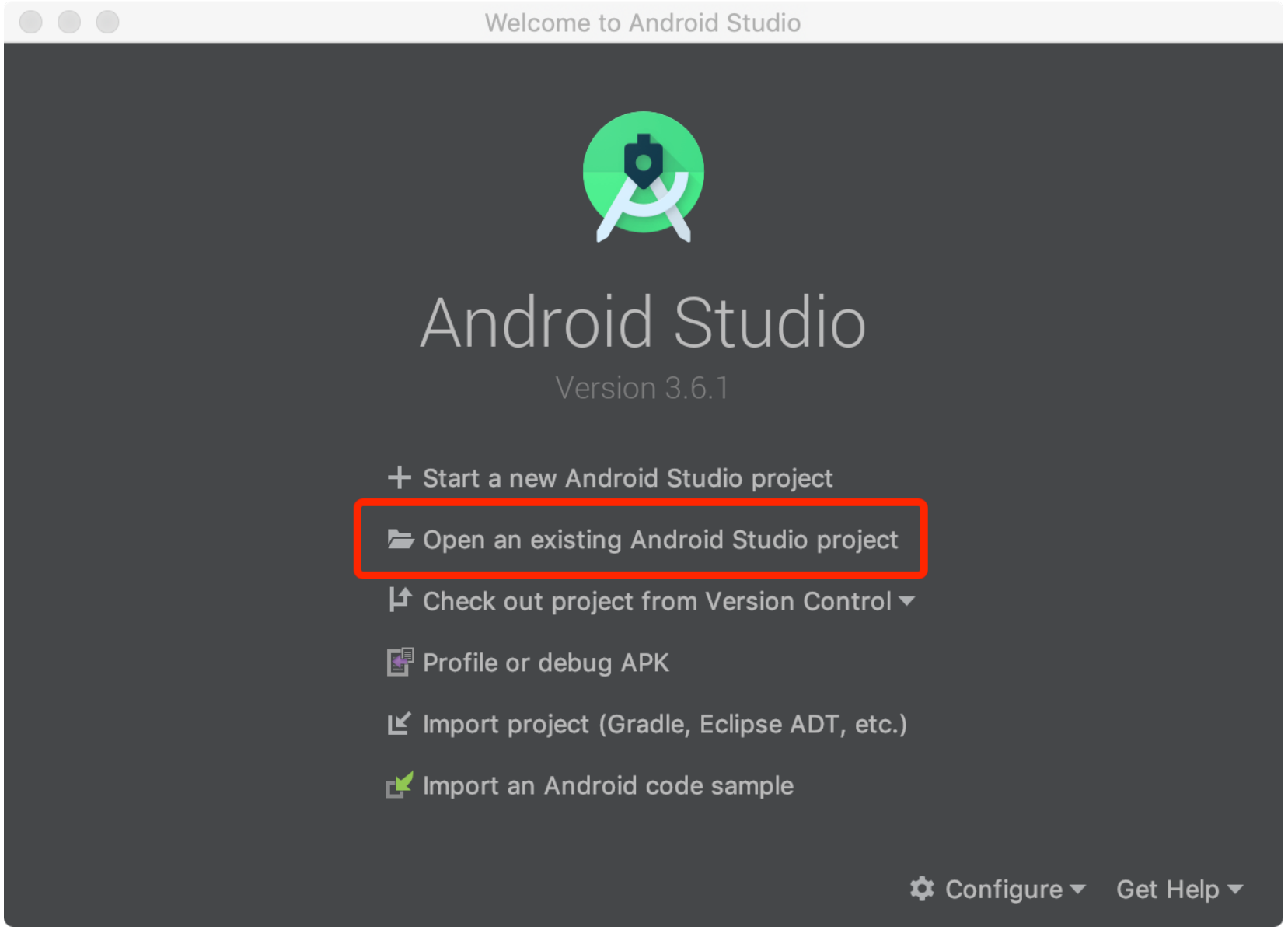
深度学习实战：实现手机上的实时物品分类应用

讲师：彭靖田

Step 1：下载 TensorFlow examples 项目

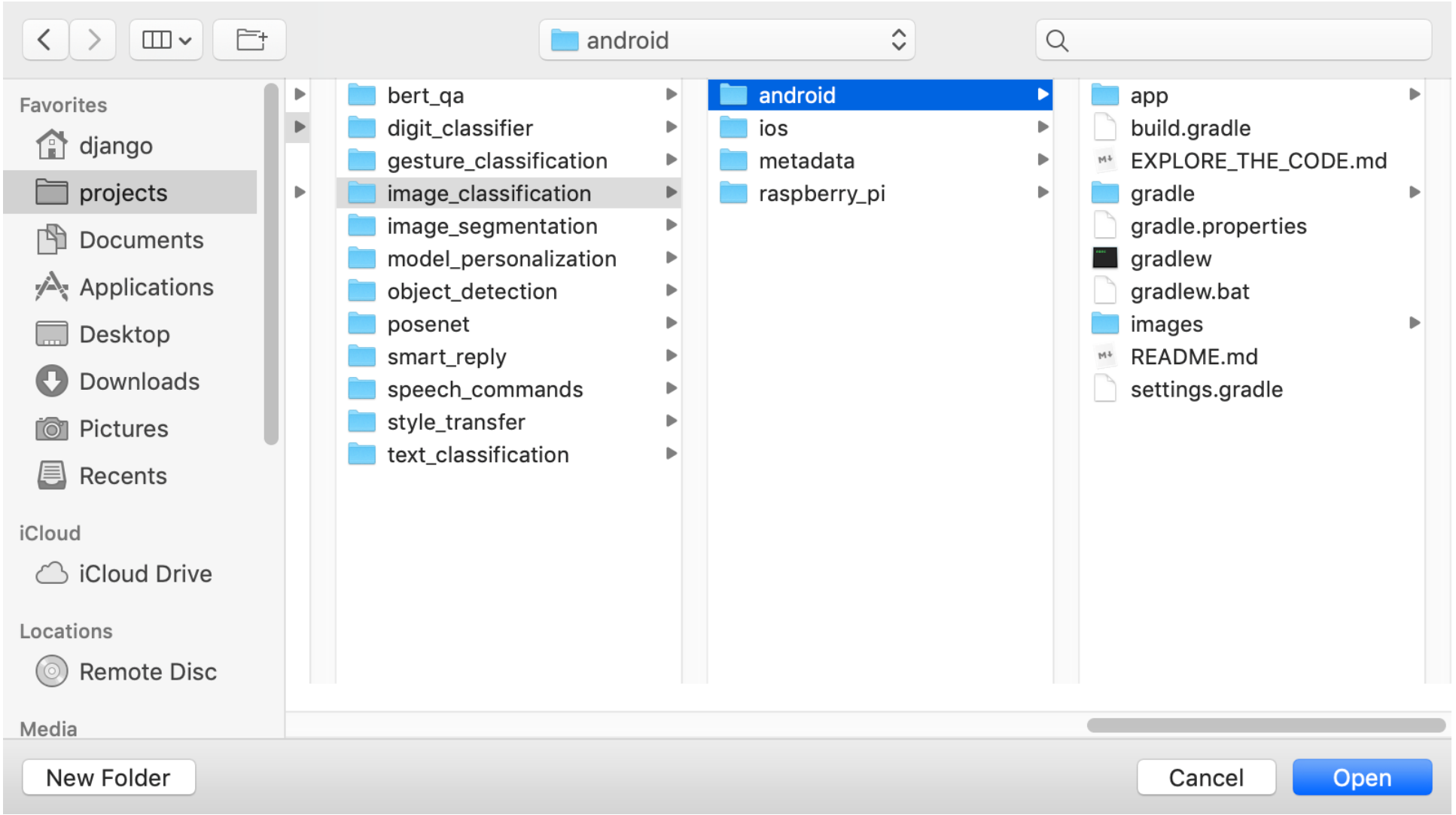
```
$ git clone https://github.com/tensorflow/examples
```

Step 2：在 Android Studio 中加载 examples 项目

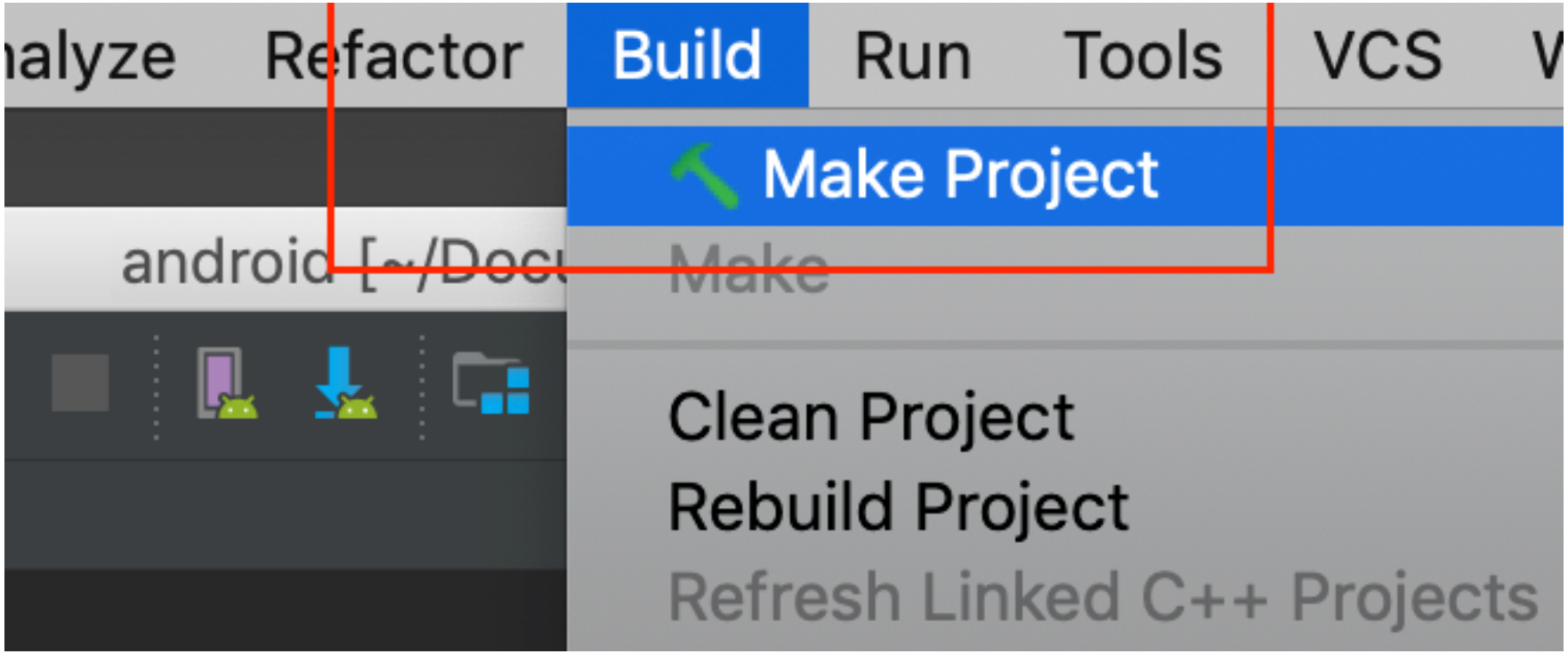


Step 2：在 Android Studio 中加载项目

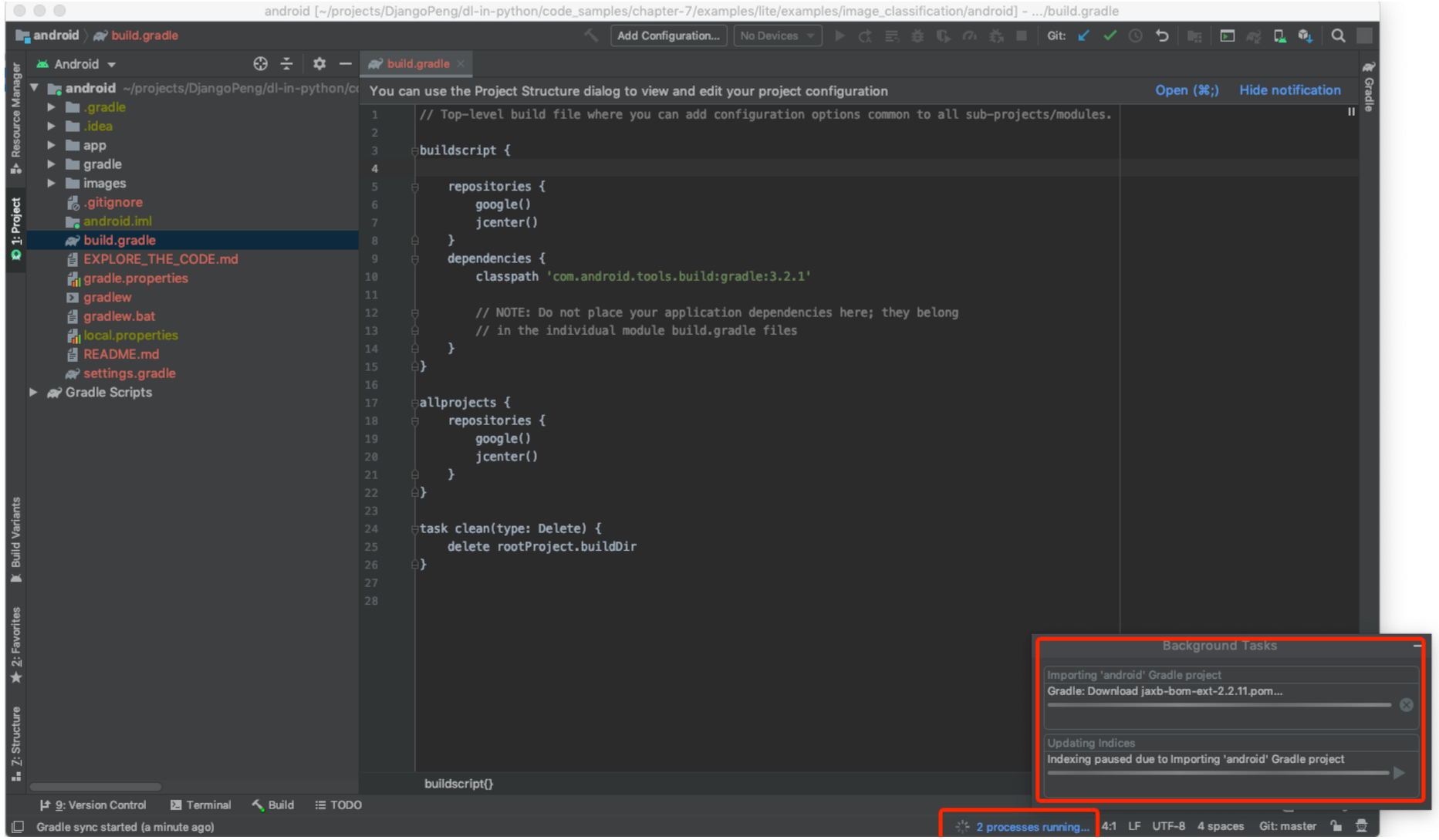
项目路径：examples/lite/examples/image_classification/android



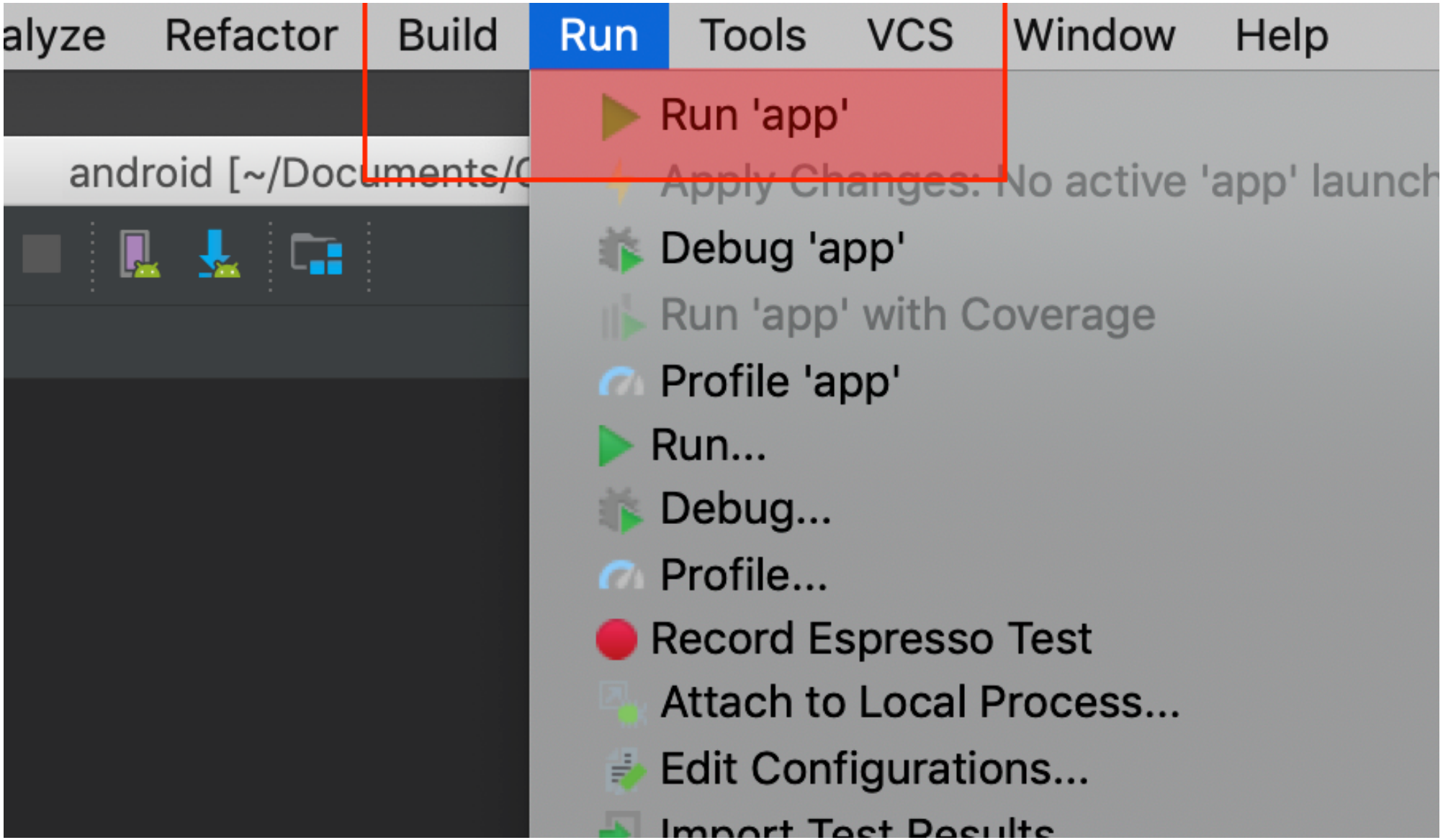
Step 3：在 Android Studio 中编译 examples 项目



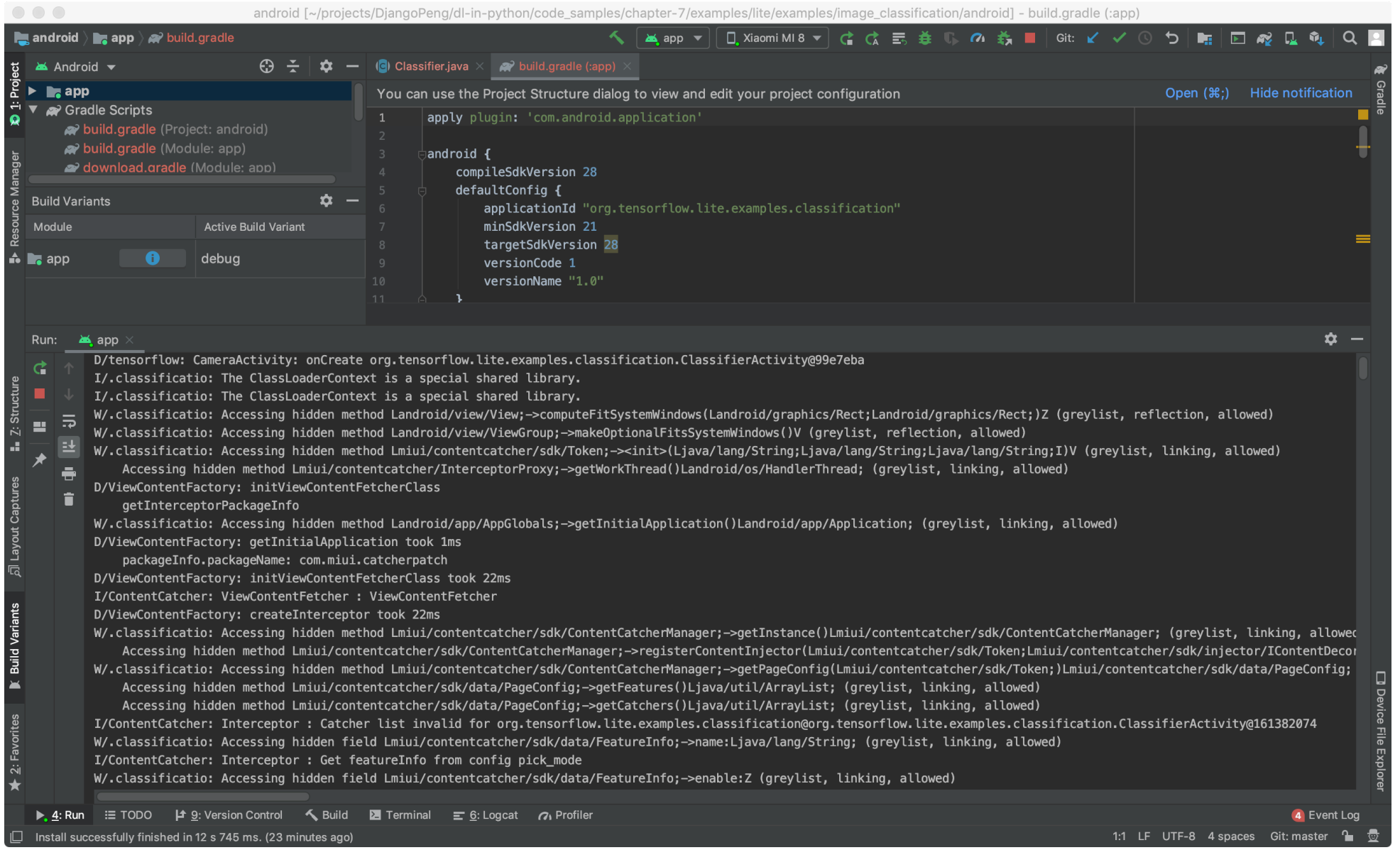
Step 3：在 Android Studio 中编译 examples 项目

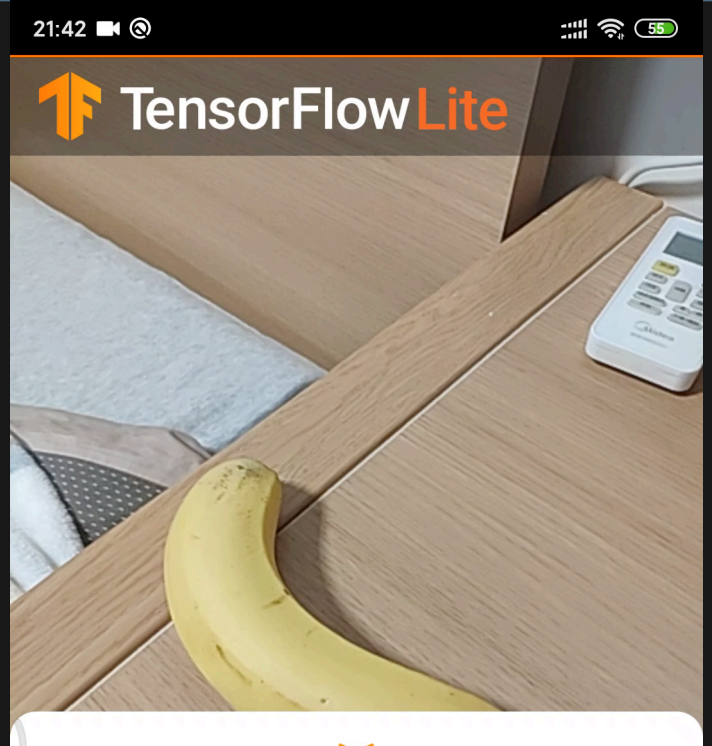


Step 4：在 Android Studio 中安装物品识别 APP



Step 5：在 Android Studio 中运行物品识别 APP





tray	12.14%
file	10.22%
spindle	10.06%
Frame	640x480
Crop	224x224
View	480x480
Rotation	90
Inference Time	32ms
Threads	− N/A +
Model:	Float_MobileNet
Device:	GPU

Try it


EDU
CSDN学院 IT实战派