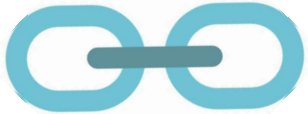
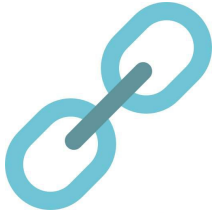
TensorFlow





PYTORCH

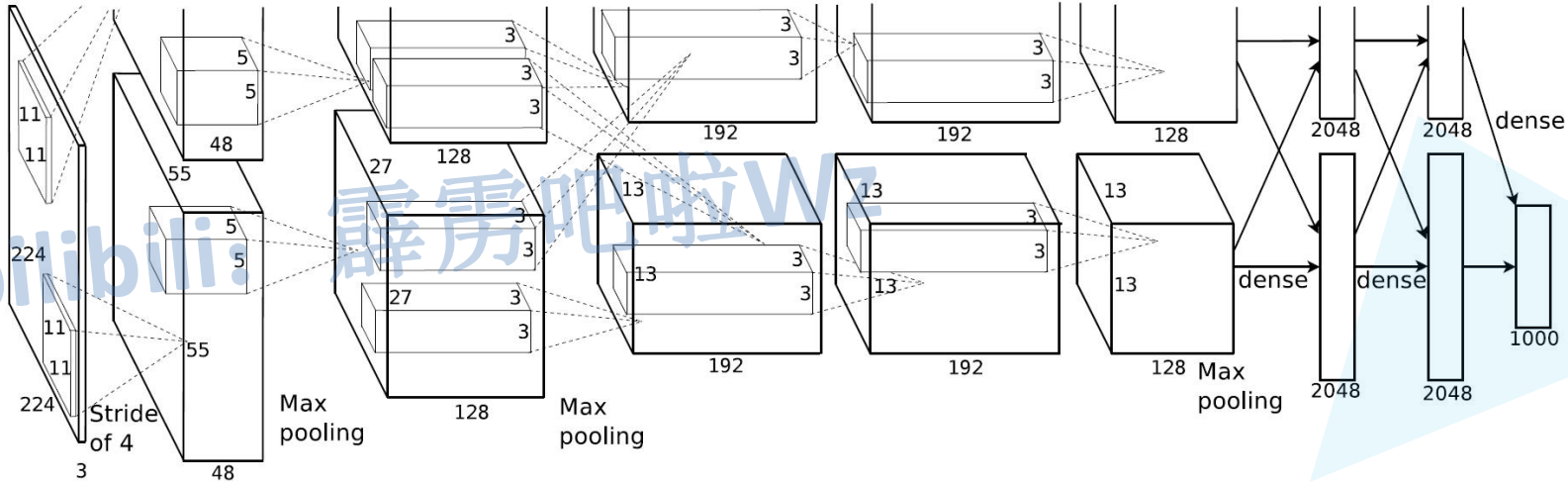
深度学习-图像处理篇

作者：神秘的wz

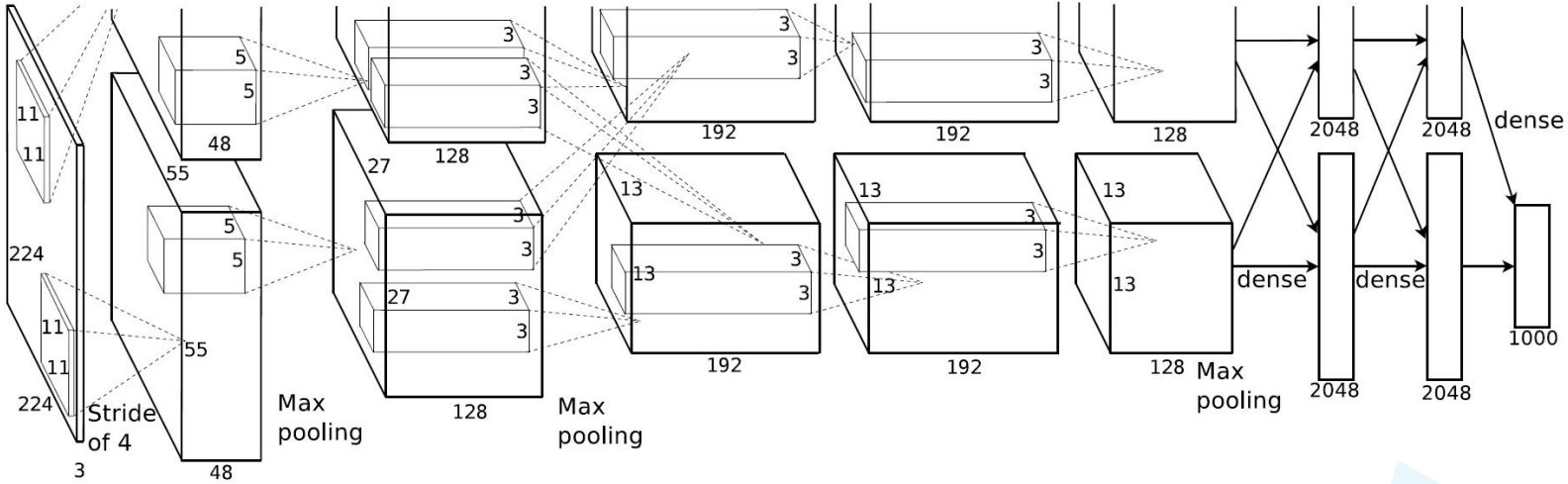
AlexNet详解

AlexNet是2012年ISLVRC 2012（ImageNet Large Scale Visual Recognition Challenge）竞赛的冠军网络，分类准确率由传统的 70%+提升到 80%+。它是由Hinton和他的学生Alex Krizhevsky设计的。也是在那年之后，深度学习开始迅速发展。

ISLVRC 2012
训练集：1,281,167张已标注图片
验证集：50,000张已标注图片
测试集：100,000张未标注图片



AlexNet详解



该网络的亮点在于：

（1）首次利用 GPU 进行网络加速训练。

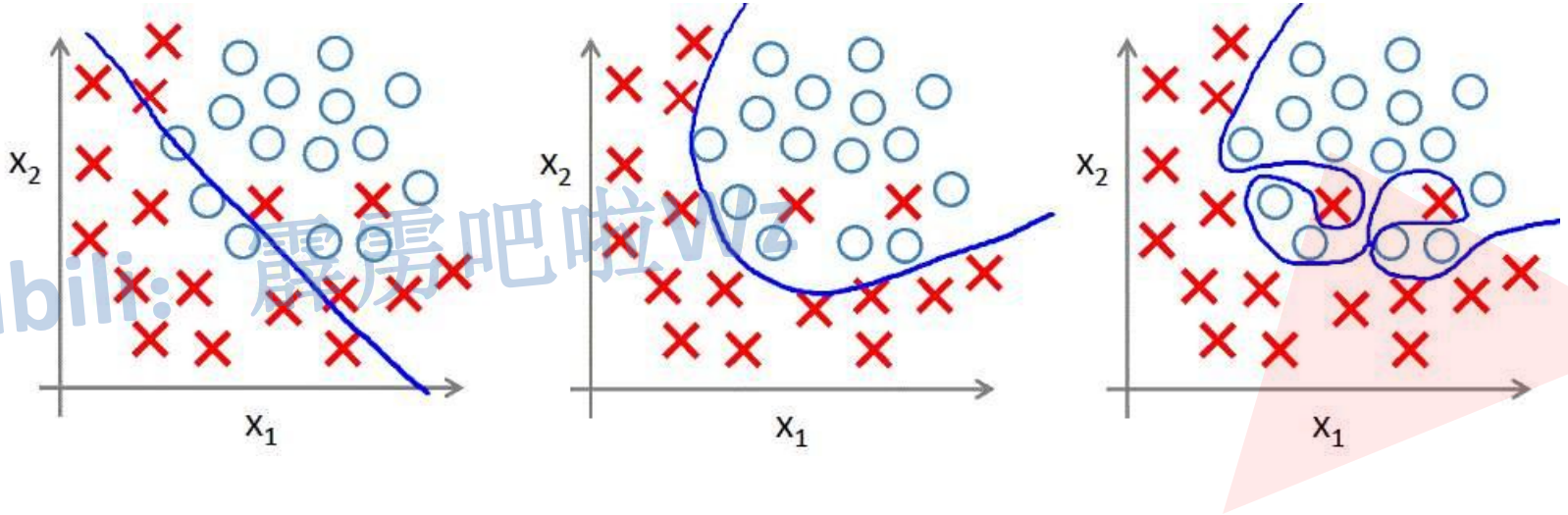
（2）使用了 ReLU 激活函数，而不是传统的 Sigmoid 激活函数以及 Tanh 激活函数。

（3）使用了 LRN 局部响应归一化。

（4）在全连接层的前两层中使用了 Dropout 随机失活神经元操作，以减少过拟合。

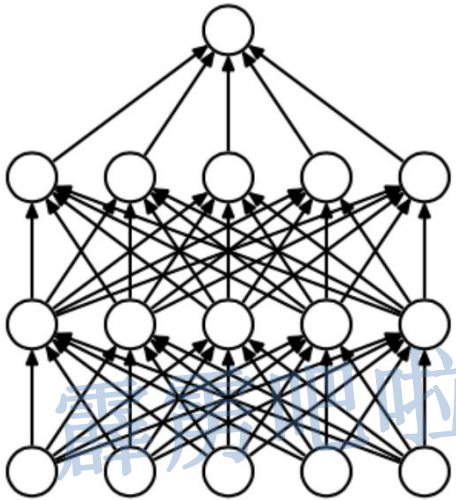
AlexNet详解

过拟合：根本原因是特征维度过多，模型假设过于复杂，参数过多，训练数据过少，噪声过多，导致拟合的函数完美的预测训练集，但对新数据的测试集预测结果差。 过度的拟合了训练数据，而没有考虑到泛化能力。

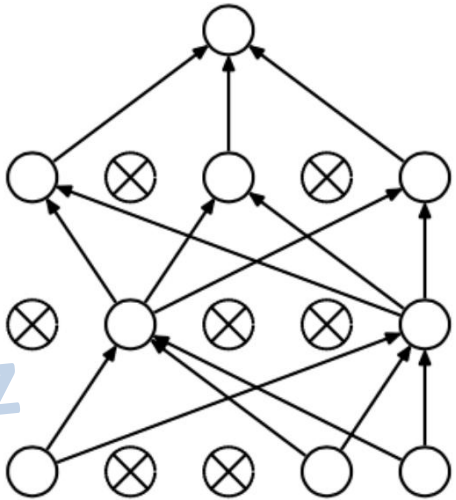


AlexNet详解

使用 Dropout 的方式在网络正向传播过程中随机失活一部分神经元



未使用Dropout的
正向传播



使用Dropout后的
正向传播

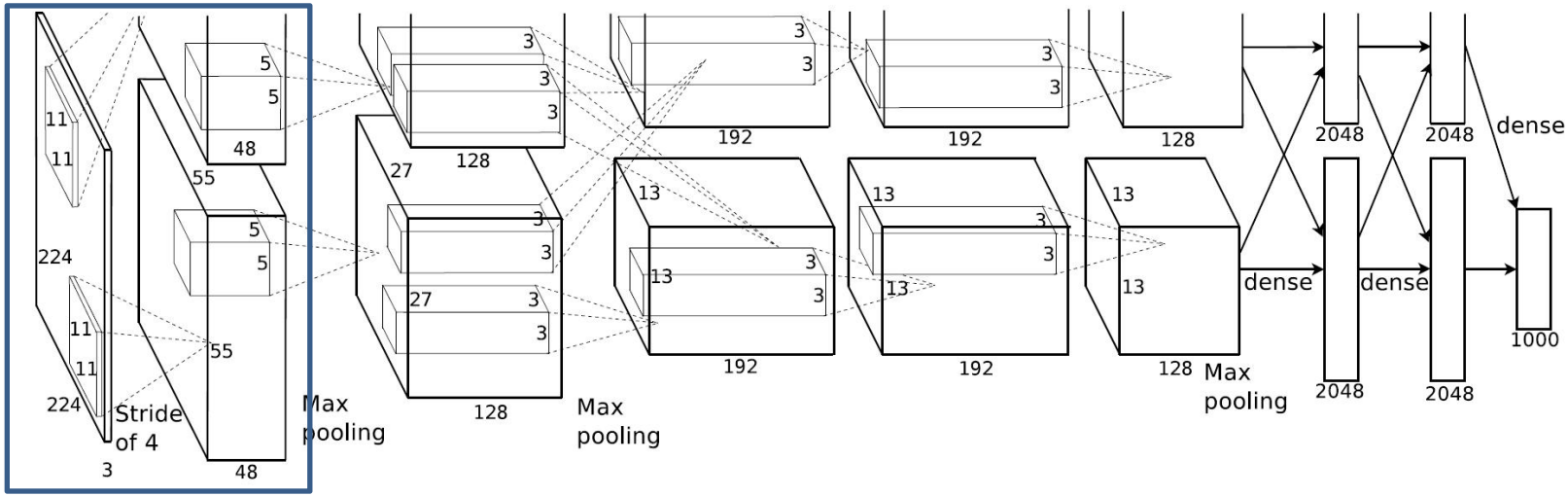
AlexNet详解

经卷积后的矩阵尺寸大小计算公式为：

$$N = (W - F + 2P) / S + 1$$

① 输入图片大小 W×W
② Filter大小 F×F
③ 步长 S
④ padding的像素数 P

AlexNet详解



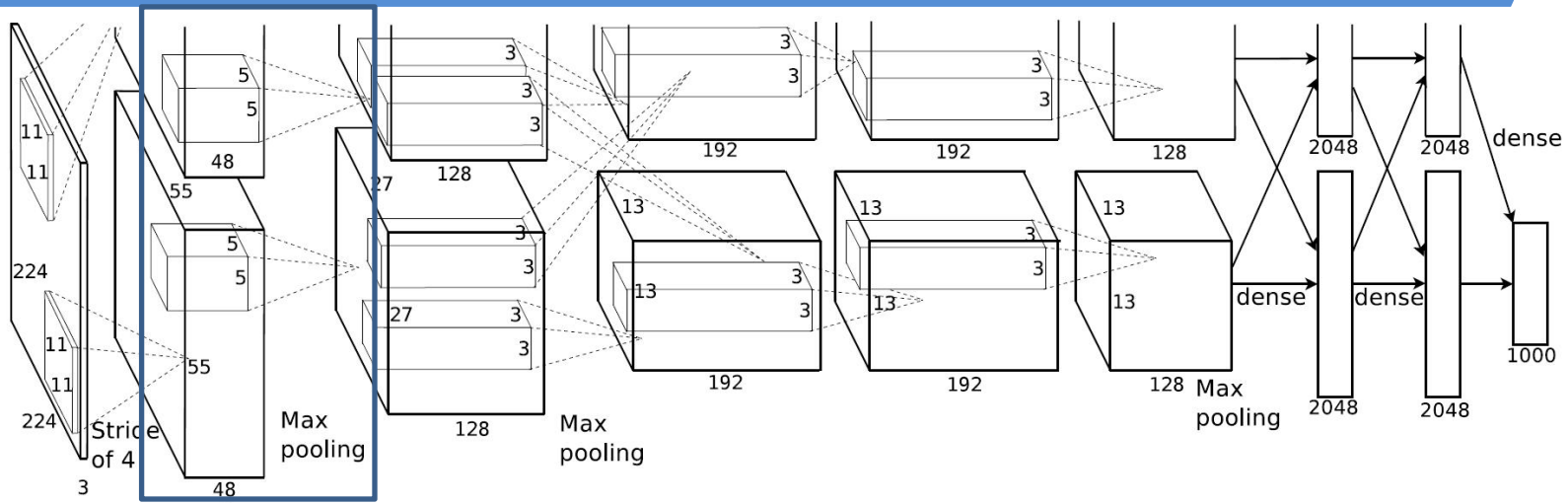
Conv1

Conv1:
kernels:48*2=96
kernel_size:11
padding:[1, 2]
stride:4

input_size: [224, 224, 3]
output_size: [55, 55, 96]

$$N = (W - F + 2P) / S + 1$$
$$=[224-11+(1+2)]/4+1$$

AlexNet详解



Maxpool1

Conv1:
kernels:48*2=96
kernel_size:11
padding: [1, 2]
stride:4

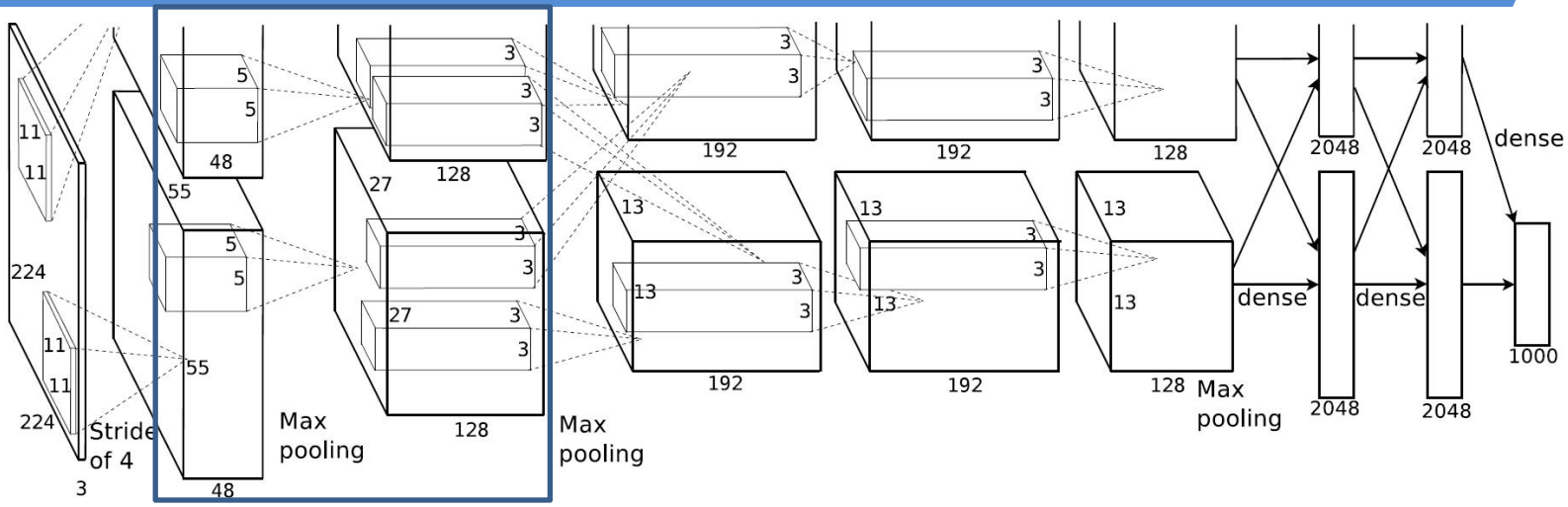
output_size:
[55, 55, 96]

Maxpool1:
kernel_size:3
pading: 0
stride:2

input_size: [55, 55, 96]
output_size: [27, 27, 96]

$$N = (W - F + 2P) / S + 1$$
$$=(55-3)/2+1$$

AlexNet详解



Conv2

Conv1:
kernels:48*2=96
kernel_size:11
padding: [1, 2]
stride:4

output_size:
[55, 55, 96]

Maxpool1:
kernel_size:3
pading: 0
stride:2

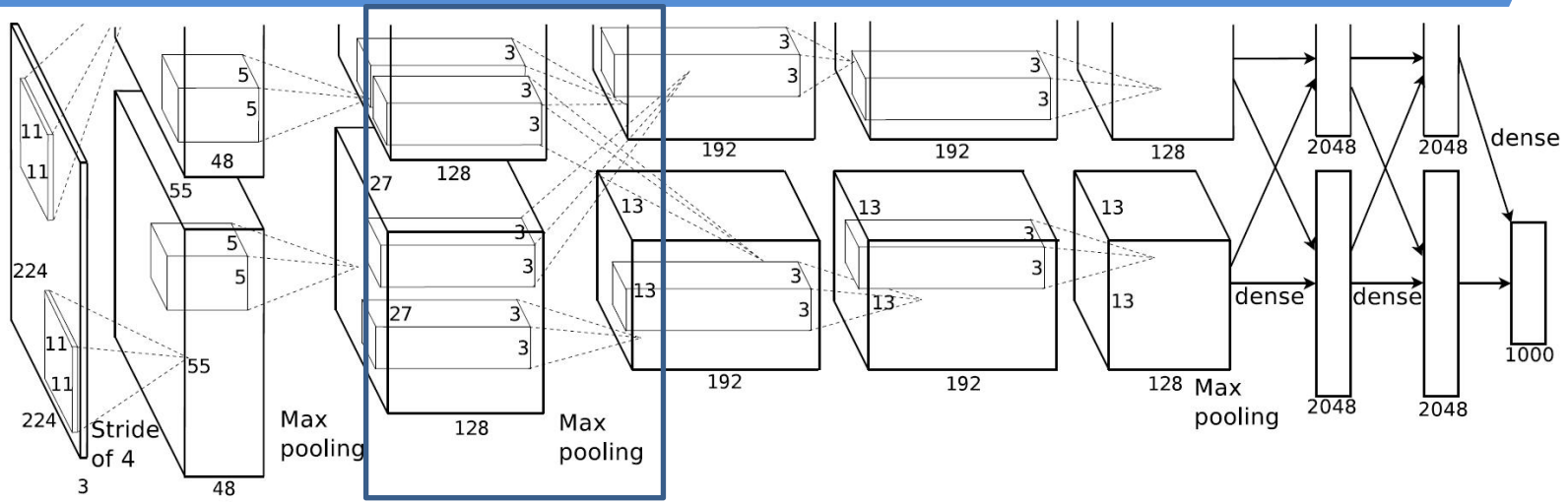
output_size:
[27, 27, 96]

Conv2:
kernels:128*2=256
kernel_size:5
padding: [2, 2]
stride:1

input_size: [27, 27, 96]
output_size: [27, 27, 256]

$$N = (W - F + 2P) / S + 1$$
$$=(27-5+4)/1+1$$

AlexNet详解



Maxpool2

Conv1:
kernels:48*2=96
kernel_size:11
padding: [1, 2]
stride:4

output_size:
[55, 55, 96]

Maxpool1:
kernel_size:3
pading: 0
stride:2

output_size:
[27, 27, 96]

Conv2:
kernels:128*2=256
kernel_size:5
padding: [2, 2]
stride:1

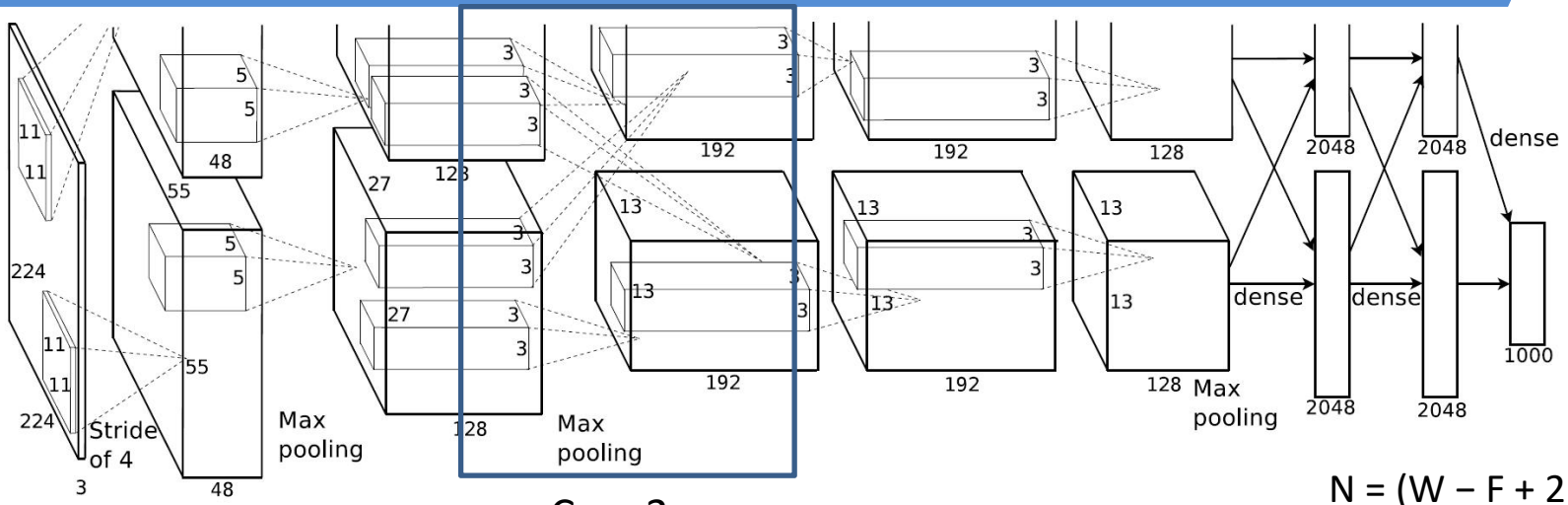
output_size:
[27, 27, 256]

Maxpool2:
kernel_size:3
pading: 0
stride:2

input_size: [27, 27, 256]
output_size: [13, 13, 256]

$$N = (W - F + 2P) / S + 1$$
$$=(27-3)/2+1$$

AlexNet详解



Conv3

$$N = (W - F + 2P) / S + 1$$
$$=(13-3+2)/1+1$$

Conv1:
kernels:48*2=96
kernel_size:11
padding: [1, 2]
stride:4

output_size:
[55, 55, 96]

Maxpool1:
kernel_size:3
pading: 0
stride:2

output_size:
[27, 27, 96]

Conv2:
kernels:128*2=256
kernel_size:5
padding: [2, 2]
stride:1

output_size:
[27, 27, 256]

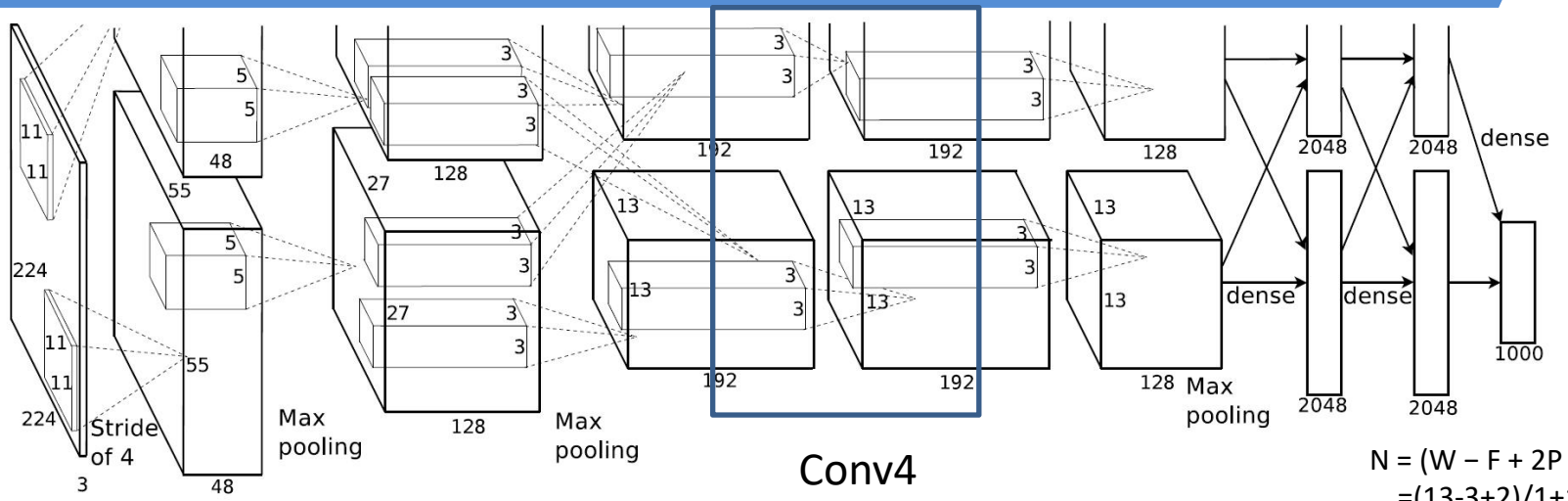
Maxpool2:
kernel_size:3
pading: 0
stride:2

output_size:
[13, 13, 256]

Conv3:
kernels:192*2=384
kernel_size:3
padding: [1, 1]
stride:1

input_size: [13, 13, 256]
output_size: [13, 13, 384]

AlexNet详解



N = (W − F + 2P) / S + 1
=(13-3+2)/1+1

Conv1:
kernels:48*2=96
kernel_size:11
padding: [1, 2]
stride:4

output_size:
[55, 55, 96]

Maxpool1:
kernel_size:3
pading: 0
stride:2

output_size:
[27, 27, 96]

Conv2:
kernels:128*2=256
kernel_size:5
padding: [2, 2]
stride:1

output_size:
[27, 27, 256]

Maxpool2:
kernel_size:3
pading: 0
stride:2

output_size:
[13, 13, 256]

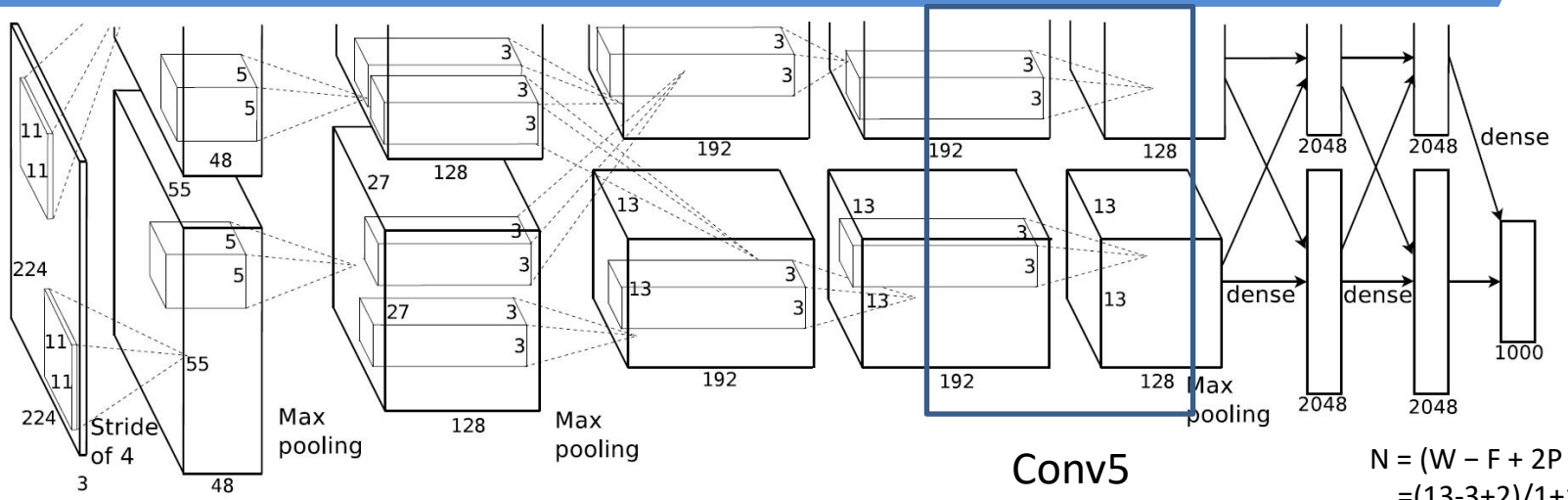
Conv3:
kernels:192*2=384
kernel_size:3
padding: [1, 1]
stride:1

output_size:
[13, 13, 384]

Conv4:
kernels:192*2=384
kernel_size:3
padding: [1, 1]
stride:1

input_size: [13, 13, 384]
output_size: [13, 13, 384]

AlexNet详解



Conv5

N = (W − F + 2P) / S + 1
=(13-3+2)/1+1

Conv1:
kernels:48*2=96
kernel_size:11
padding: [1, 2]
stride:4

output_size:
[55, 55, 96]

Maxpool1:
kernel_size:3
pading: 0
stride:2

output_size:
[27, 27, 96]

Conv2:
kernels:128*2=256
kernel_size:5
padding: [2, 2]
stride:1

output_size:
[27, 27, 256]

Maxpool2:
kernel_size:3
pading: 0
stride:2

output_size:
[13, 13, 256]

Conv3:
kernels:192*2=384
kernel_size:3
padding: [1, 1]
stride:1

output_size:
[13, 13, 384]

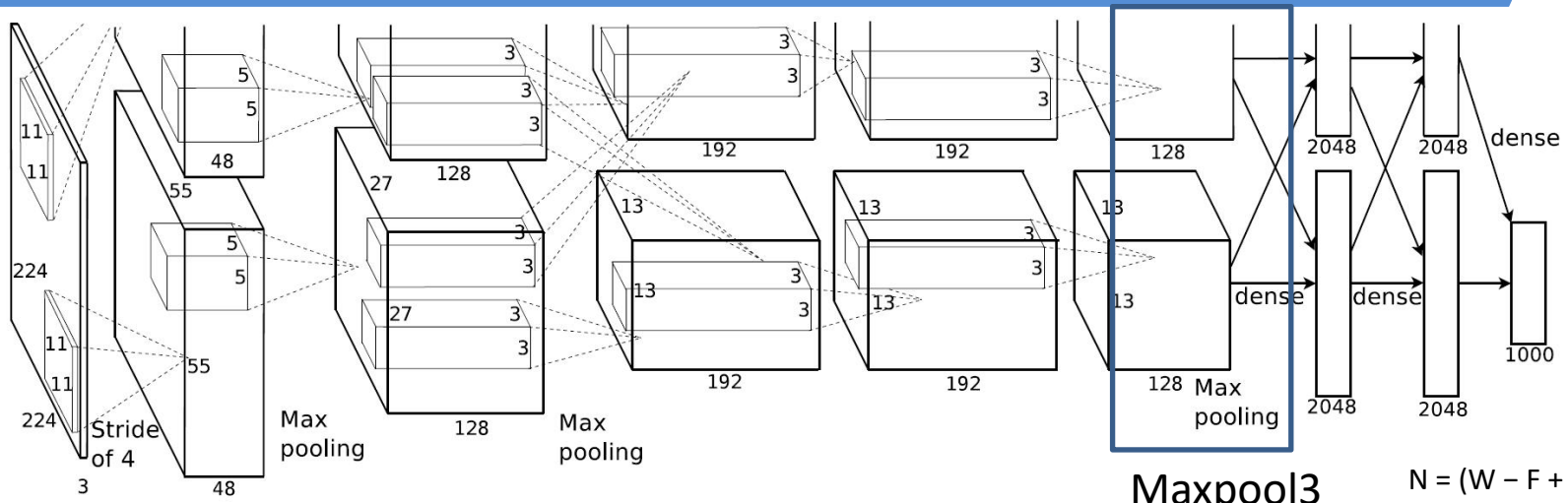
Conv4:
kernels:192*2=384
kernel_size:3
padding: [1, 1]
stride:1

output_size:
[13, 13, 256]

Conv5:
kernels:128*2=256
kernel_size:3
padding: [1, 1]
stride:1

input_size: [13, 13, 384]
output_size: [13, 13, 256]

AlexNet详解



Maxpool3

$N = (W - F + 2P) / S + 1$
$=(13-3)/2+1$

Conv1:
kernels:96
kernel_size:11
padding: [1, 2]
stride:4

output_size:
[55, 55, 96]

Maxpool1:
kernel_size:3
pading: 0
stride:2

output_size:
[27, 27, 96]

Conv2:
kernels:256
kernel_size:5
padding: [2, 2]
stride:1

output_size:
[27, 27, 256]

Maxpool2:
kernel_size:3
pading: 0
stride:2

output_size:
[13, 13, 256]

Conv3:
kernels:=384
kernel_size:3
padding: [1, 1]
stride:1

output_size:
[13, 13, 384]

Conv4:
kernels:384
kernel_size:3
padding: [1, 1]
stride:1

output_size:
[13, 13, 384]

Conv5:
kernels:128*2=256
kernel_size:3
padding: [1, 1]
stride:1

output_size:
[13, 13, 256]

Maxpool3:
kernel_size:3
padding: 0
stride:2

input_size: [13, 13, 256]
output_size: [6, 6, 256]

AlexNet详解

layer_name	kernel_size	kernel_num	padding	stride
Conv1	11	96	[1, 2]	4
Maxpool1	3	None	0	2
Conv2	5	256	[2, 2]	1
Maxpool2	3	None	0	2
Conv3	3	384	[1, 1]	1
Conv4	3	384	[1, 1]	1
Conv5	3	256	[1, 1]	1
Maxpool3	3	None	0	2
FC1	2048	None	None	None
FC2	2048	None	None	None
FC3	1000	None	None	None

AlexNet详解

下载花分类数据集

daisy
雏菊

dandelion
蒲公英

roses
玫瑰

sunflower
向日葵

tulips
郁金香

沟通方式

1.github

https://github.com/WZMIAOMIAO/deep-learning-for-image-processing

2.CSDN

https://blog.csdn.net/qq_37541097/article/details/103482003

3.bilibili

https://www.bilibili.com/video/av79436317

尽可能每周更新

感谢各位的观看！