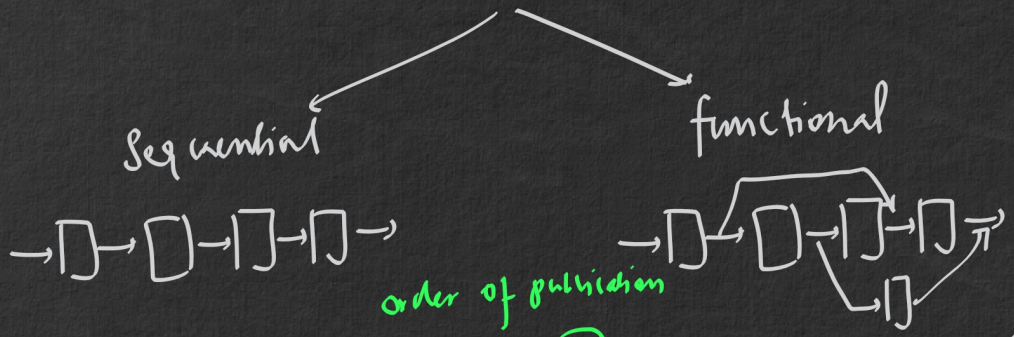
CNN Architectures



order of publication

- LeNet ✓
- AlexNet ✓
- VGG 16/19

depth ↑ parameters ↑
size of Model ↑
accuracy ↑
Latency ↑ slow

- GoogLeNet {Inception}
- ResNet

depth ↑ parameters ↓
size of model {modules}
accuracy ↑
Latency ↓

Latency
↳ late

cat & dog
binary classification {simples / create your own}

Total No. of parameters = Trainable + Non Trainable
= N

32 bit / parameter

32 × N bit

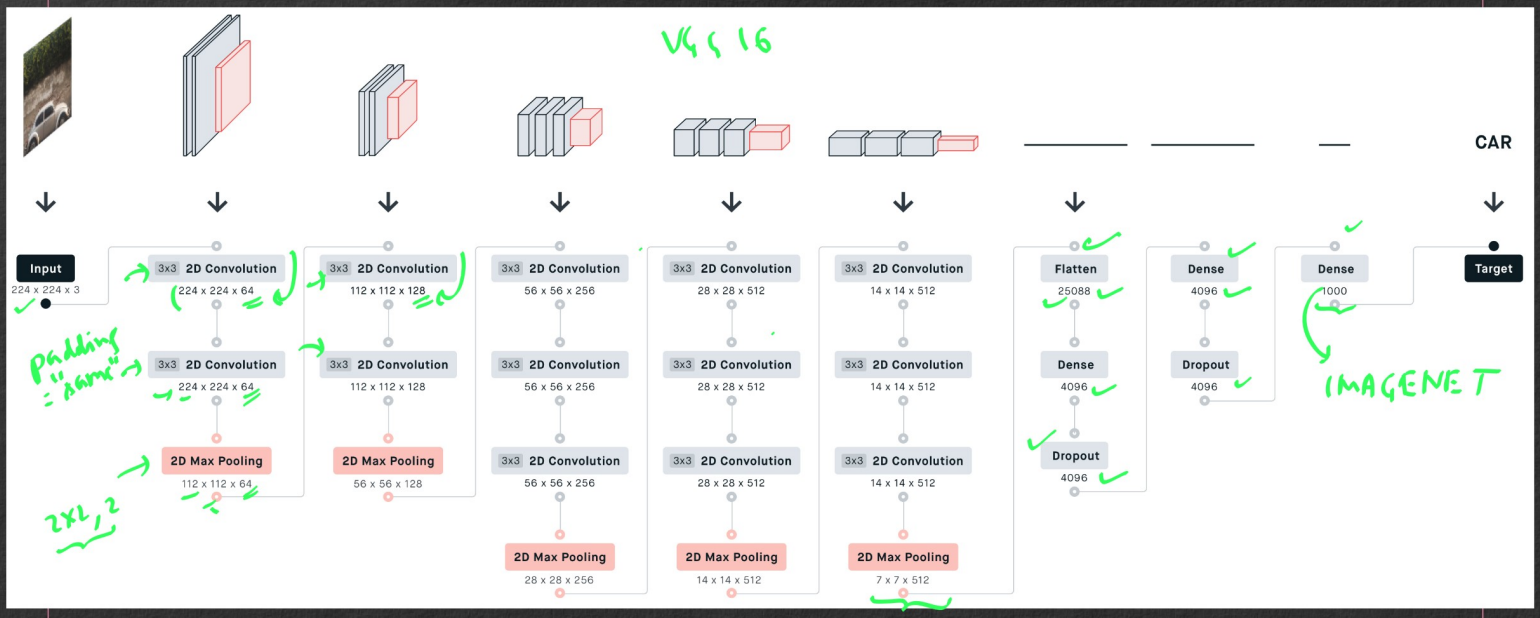
8

x byte

1 byte = 8 bit

1 KB = 1024 byte

Tradeoff accuracy vs latency / speed vs memory

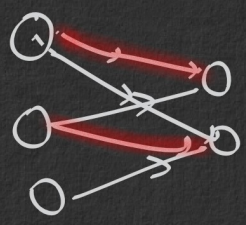


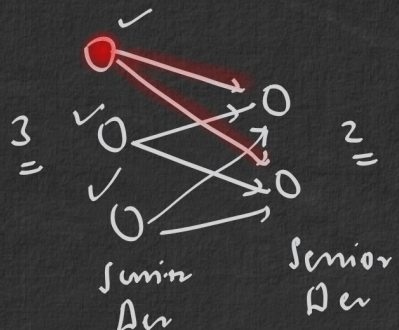
VGG16 VGG19

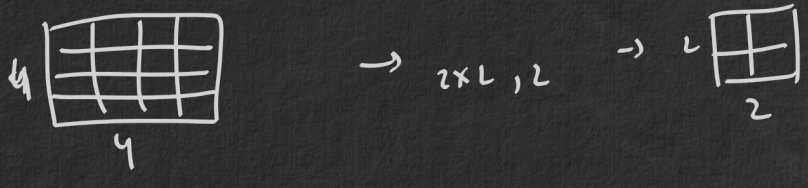
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

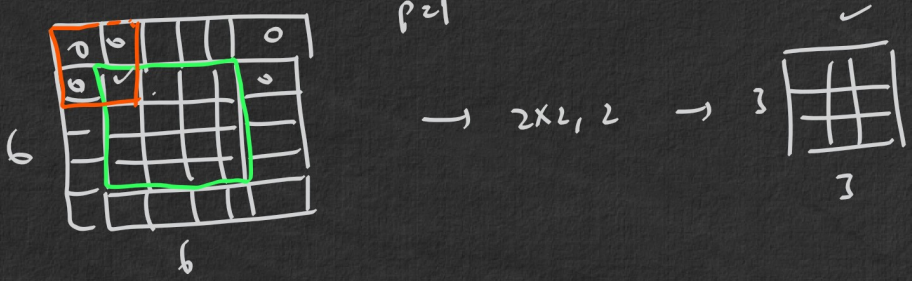
Table 2: **Number of parameters** (in millions).

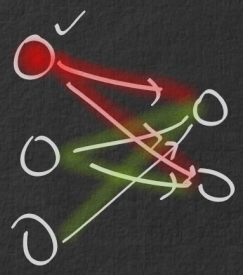
Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144



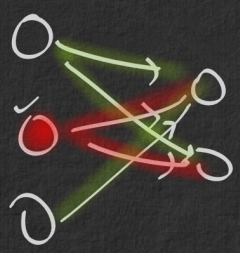








Arch 1st ✓



Arch 2nd ✓

3 years
{ Reading }
→ Hands on ML Aurelien Geron
→ ISLR
→ Python ML Sebastian Raschka

Ensemble Technique

Observation :- { VGG models }

i) Based on simple classical sequential architecture

ii) Successful variants → VGG16 & VGG19.

iii) Dense layers → 2 → 4096 size each

iv) Activation function → hidden layer → ReLU*
→ O/p layer → softmax

v) Conv. filter size 3x3 with stride = 1

vi) Max pooling 2x2 stride = 2

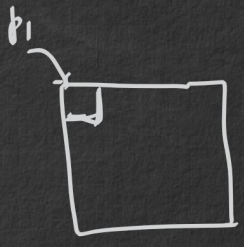
vii) Padding → "same"

viii) Demonstrated → Depth is beneficial for classification accuracy.

ix) Preprocessing → subtracting the mean RGB values from each pixel

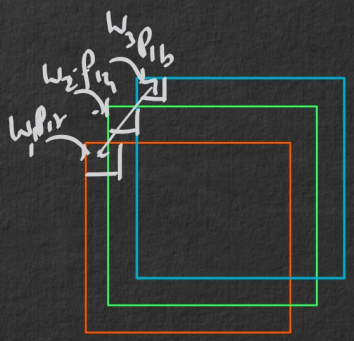
x) Augmentation → Random flipping, Random RGB colour shifting

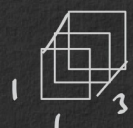
xi) Tried 1x1 convolution => Linear transformation of i/p channel → followed by non-linearity

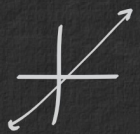


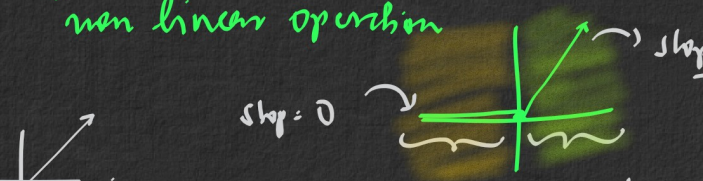
relu $\left(\underbrace{(W \cdot p_1 + W p_2 \dots .)}_{\text{linear transformation}} + b \right)$

↓ non linear operation



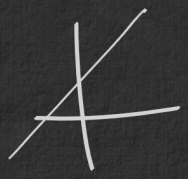


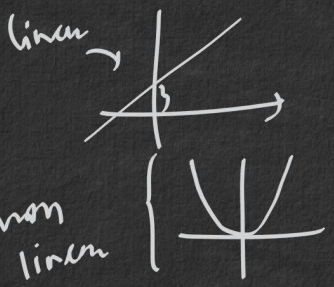




linear transformation

$(\underbrace{W \cdot p}_{\uparrow} + b)$





$f(x) = x + 3$ → linear operation

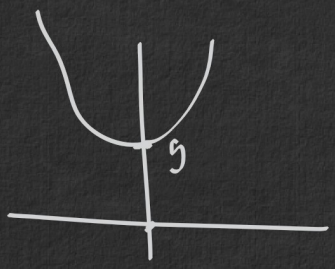
$g(x) = x^2$

$g(f(x)) = (f(x))^2$

$= (x+3)^2$

$= x^2 + 9 + 6x$

$0 \quad 0$

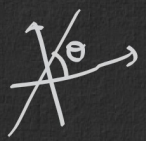


$f(x) = mx + c$

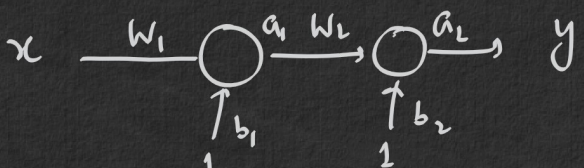
$= m$

$g(x) = mx^2 + c$

$g'(x) = 2mx$



Not applying any non linearity



$a_1 = W_1 x + b_1$

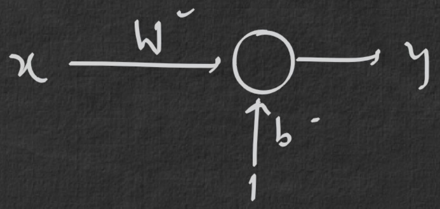
$a_2 = W_2 a_1 + b_2$

$y = a_2$

$y = W_2(W_1 x + b_1) + b_2$

$= W_2 W_1 x + W_2 b_1 + b_2$

$y = Wx + b$



$a_1 = 0.5x + 0.3$

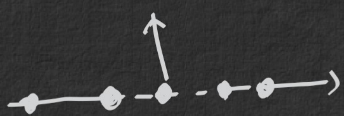
$a_2 = 0.6\,a_1 + 0.7$

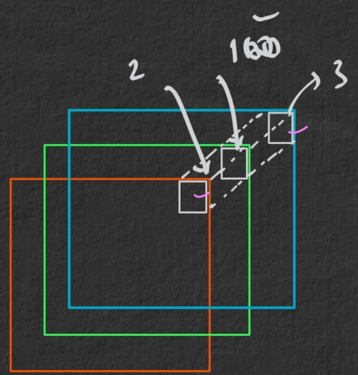
$y = a_2$

$y = \underbrace{0.6 \times 0.5}\,x + \underbrace{0.6 \cdot 0.3 + 0.7}$

$y = Wx + b$

LRN → Local Response Normalization | Batch Norm ✓





RGB

Ch → 64

$z = W_r r + W G + W b + \text{bias}$

2   100   3

1 [4] 1

64

$\overbrace{0 \rightarrow 63}^{i}$

$i = 64 \quad r = 2$

$$b_i = \frac{a_i}{\left[k + \alpha \sum_{j_{low}}^{j_{high}} (a_j)^2\right]^{\beta}}$$

(j_{high} = 65, j_{low} = 63)

$j_{high} = \min\left(i + \frac{r}{2},\ fn - 1\right)$ — 65, 65, 127

$j_{low} = \max\left(0,\ i - \frac{r}{2}\right)$ — 63, 63

b_i → Normalized output of the neuron located in feature map i

a_i → activation of that neuron after ReLU step

Hyper parameters w.r.t AlexNet

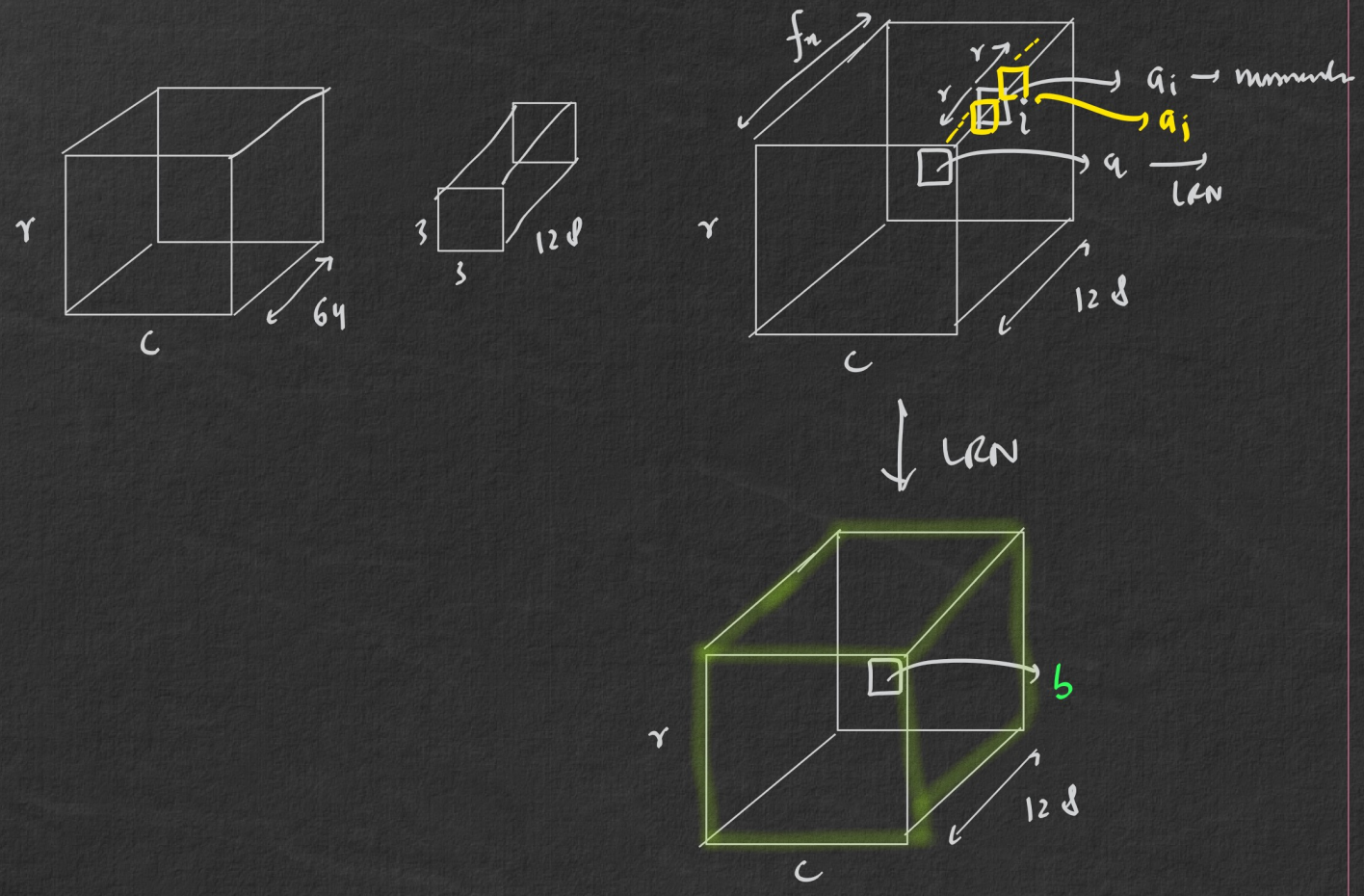
k → bias = 1

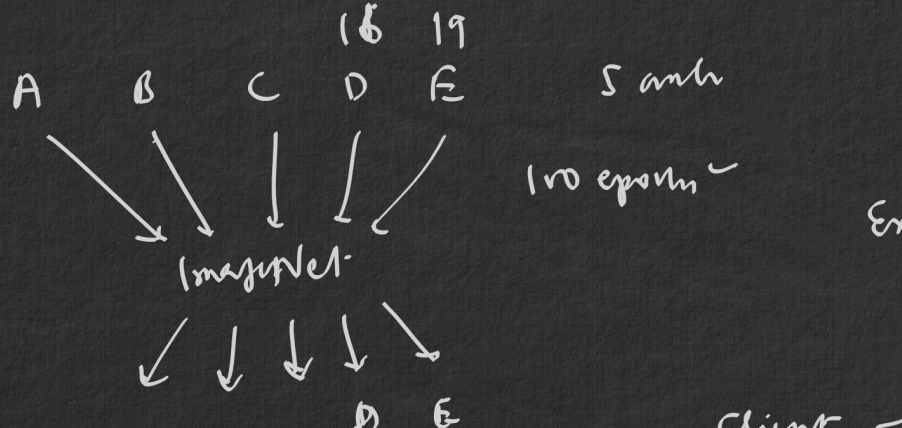
α → coeff. = 0.00002

r → depth radius = 2

β → coeff. = 0.75

$f_n \rightarrow$ no. of feature maps



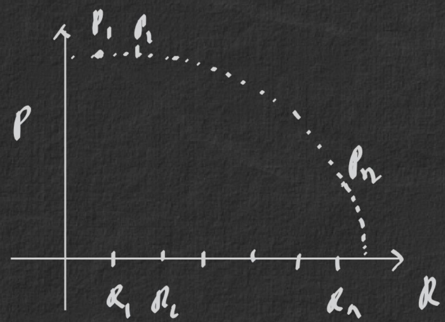


Healthcare → Life → latency
accuracy ↑ → 19

Traffic management → latency ↓
accuracy →

→ Mean Average Precision :-

Precision vs Recall PR



Average Precision

Per class 1000

mean Average Precision
for all the class

car, bike ....

~~for every~~ class 1

for recall values

↳ AP_1

class 2

for recall values

↳ AP_2

⋮

1000 classes → $AP_1 + AP_2$ .... AP_{1000}

mean of APs $\frac{AP_1 \ldots\ldots AP_{1000}}{1000}$ = mAP