Deep Learning

Notes : Perceptron , Activation Function , Gradient Descent , backpropagation , Loss Function , Metrics , Optimizer , Learning Rate , CNN , RNN , Structured Data Unstructured Data , Research Papers , History

Convolutional Neural Network

PCA color augmentation

Edge detection example

"convolution"

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

6x6

*

1	0	-1
1	0	-1
1	0	-1

3x3 Filter

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	16

4x4

→ 3+1+1x1+ ... = -5 - - - - -

Vertical edge detection

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

6x6

*

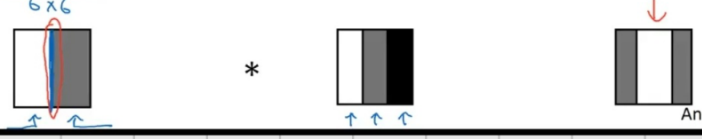
1	0	-1
1	0	-1
1	0	-1

3x3

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

4x4



Andrew Ng

1	1	1
0	0	0
-1	-1	-1

Horizontal

Vertical and Horizontal Edge Detection

1	0	-1
1	0	-1
1	0	-1

Vertical

1	1	1
0	0	0
-1	-1	-1

Horizontal

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

*

1	1	1
0	0	0
-1	-1	-1

=

0	0	0	0
30	10	-10	-30
30	10	-10	-30
0	0	0	0

Andrew Ng

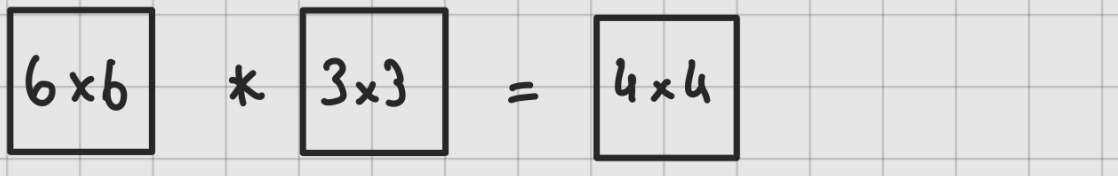
1	0	-1
2	0	-2
1	0	-1

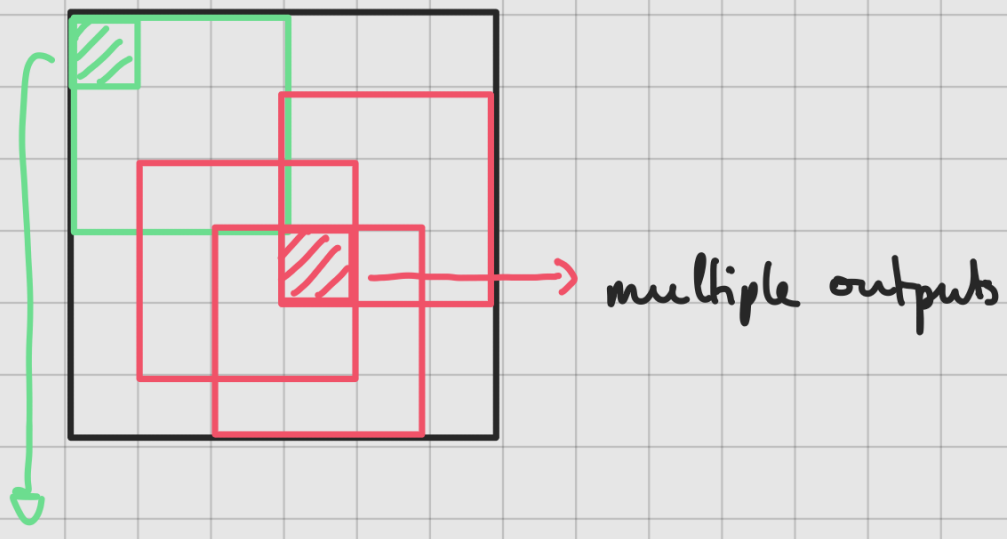
Sobel Filter

3	0	-3
10	0	-10
3	0	-3

Scharr Filter

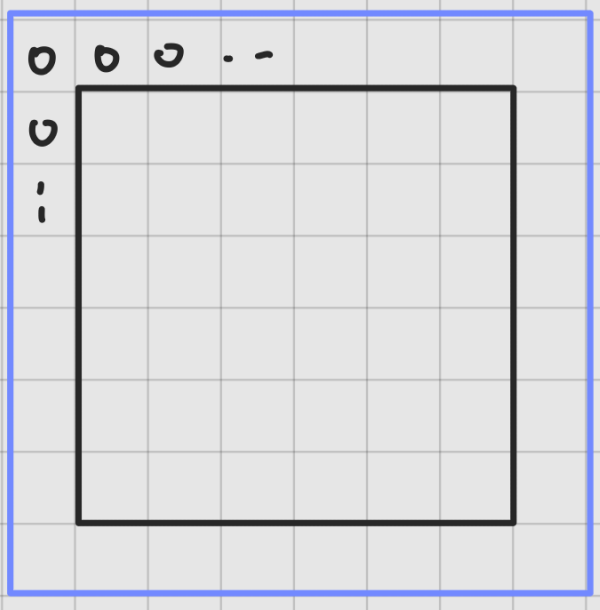
Padding





Problems:
- Shrinky output
- throw away info from edges

Padding $p = 1$



$6 \times 6 \rightarrow 8 \times 8$

$8 \times 8 \ * \ 3 \times 3 \rightarrow 6 \times 6 \quad (n + 2p - f + 1 = 6)$

p = padding amount = 1

Valid and Same convolut

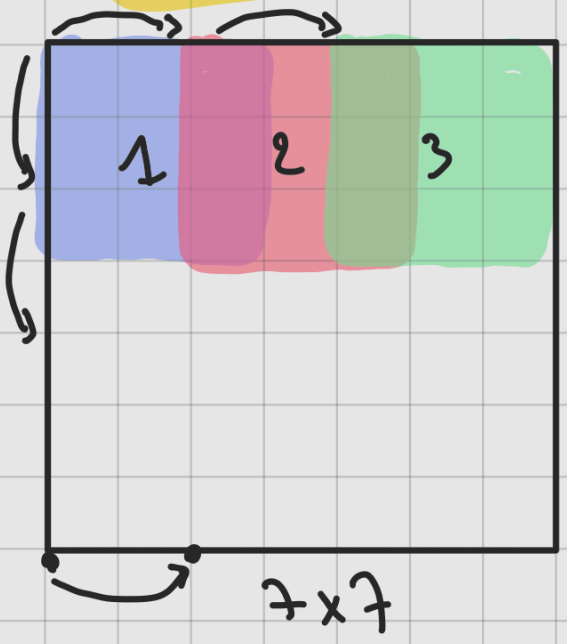
- "Valid" $n \times n \ * \ f \times f \rightarrow n - F + 1 \ \times \ n - F + 1$
  ↳ (no padding)

- "Same": Pad so output size same as input

$$\Rightarrow p = \frac{f - 1}{2}$$

f is usually odd

Strided Convolution



$* \quad 3 \times 3 \quad \longrightarrow \quad 3 \times 3$

stride = 2

$n \times n \quad * \quad f \times f \quad \longrightarrow \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \dots$

padding p stride s

↑ floor

Technical note on cross-correlation vs. convolution

Convolution in math textbook:

2	3	7	4	6	2
6	6	9	8	7	4
3	4	8	3	8	9
7	8	3	6	6	3
4	2	1	8	3	4
3	2	4	1	9	8

$*$

3	4	5
1	0	2
-1	9	7

$=$

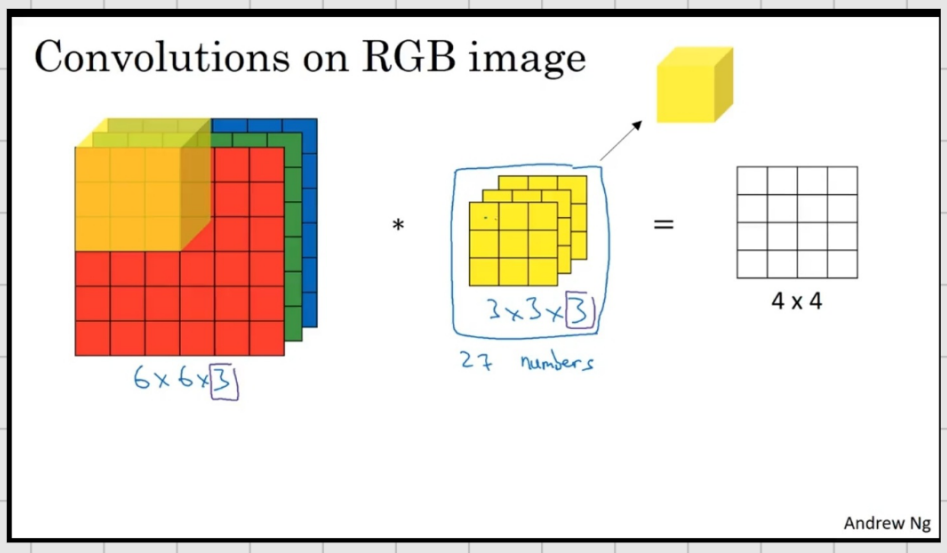
7	2	5
9	0	4
-1	1	3

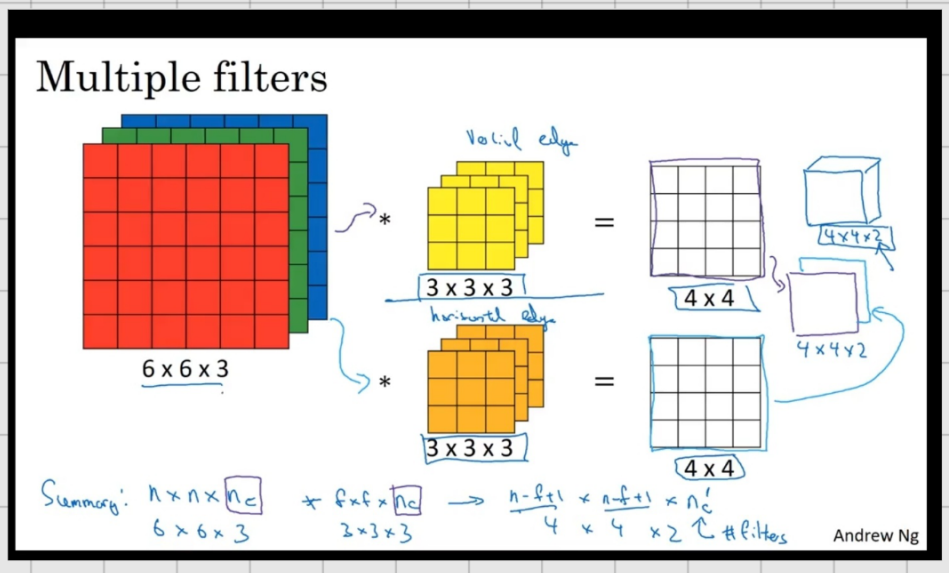
$(A * B) * C = A * (B * C)$

Andrew Ng

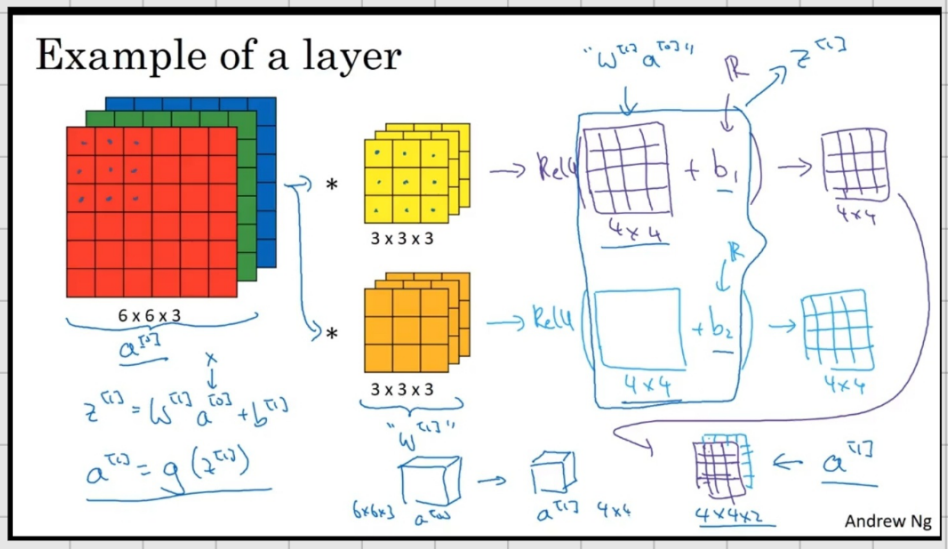
math lit

Convolution over volume





One layer of a Convolutional Network



$f^{[l]}$ = filter size

$p^{[l]}$ = padding

$s^{[l]}$ = stride

$n_c^{[l]}$ = number of filters

Input : $n_H^{[l-1]} \times n_w^{[l-1]} \times n_c^{[l-1]}$

Output: $n_H^{[l]} \times n_w^{[l]} \times n_c^{[l]}$

$$n^{[l]} = \left\lfloor \frac{n^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

Each filter is : $f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$

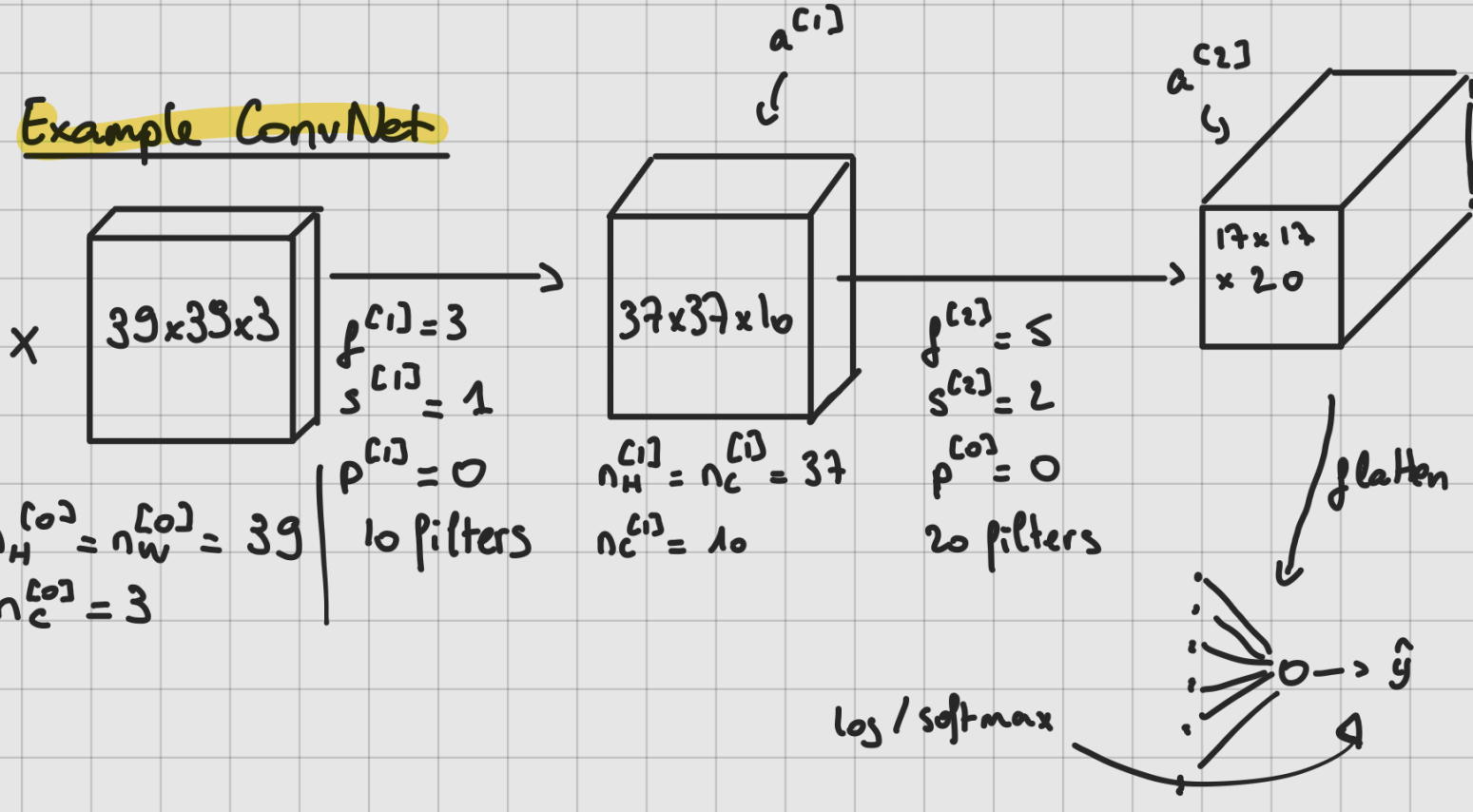
Activations : $a^{[l]} \rightarrow n_H^{[l]} \times n_w^{[l]} \times n_c^{[l]}$

$$A^{[l]} \rightarrow m \times n_H^{[l]} \times n_w^{[l]} \times n_c^{[l]}$$

Weights : $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$

bias : $n_c^{[l]} - (1, 1, 1, n_c^{[l]})$

Example ConvNet



$n_H^{[0]} = n_w^{[0]} = 39$

$n_c^{[0]} = 3$

$n_H^{[1]} = n_c^{[1]} = 37$

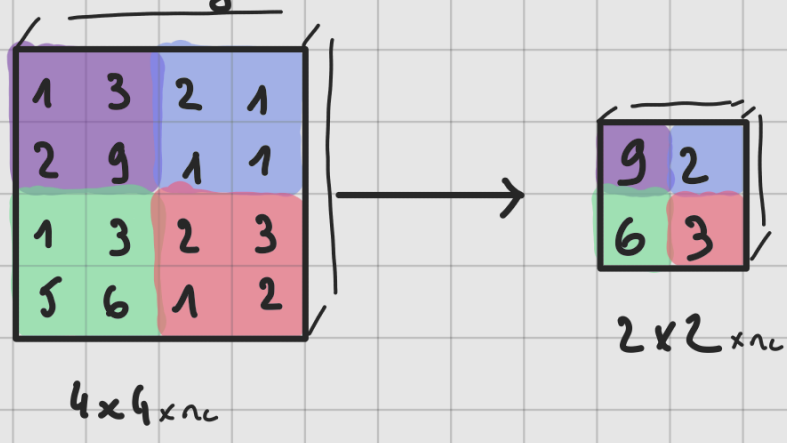
$n_c^{[1]} = 10$

Layers in ConvNet :

- Convolution (Conv)
- Pooling (POOL)
- Fully Connected (FC)

Pooling layers

Max pooling



Hyperparams :

$f = 2$

$s = 2$

No learning params

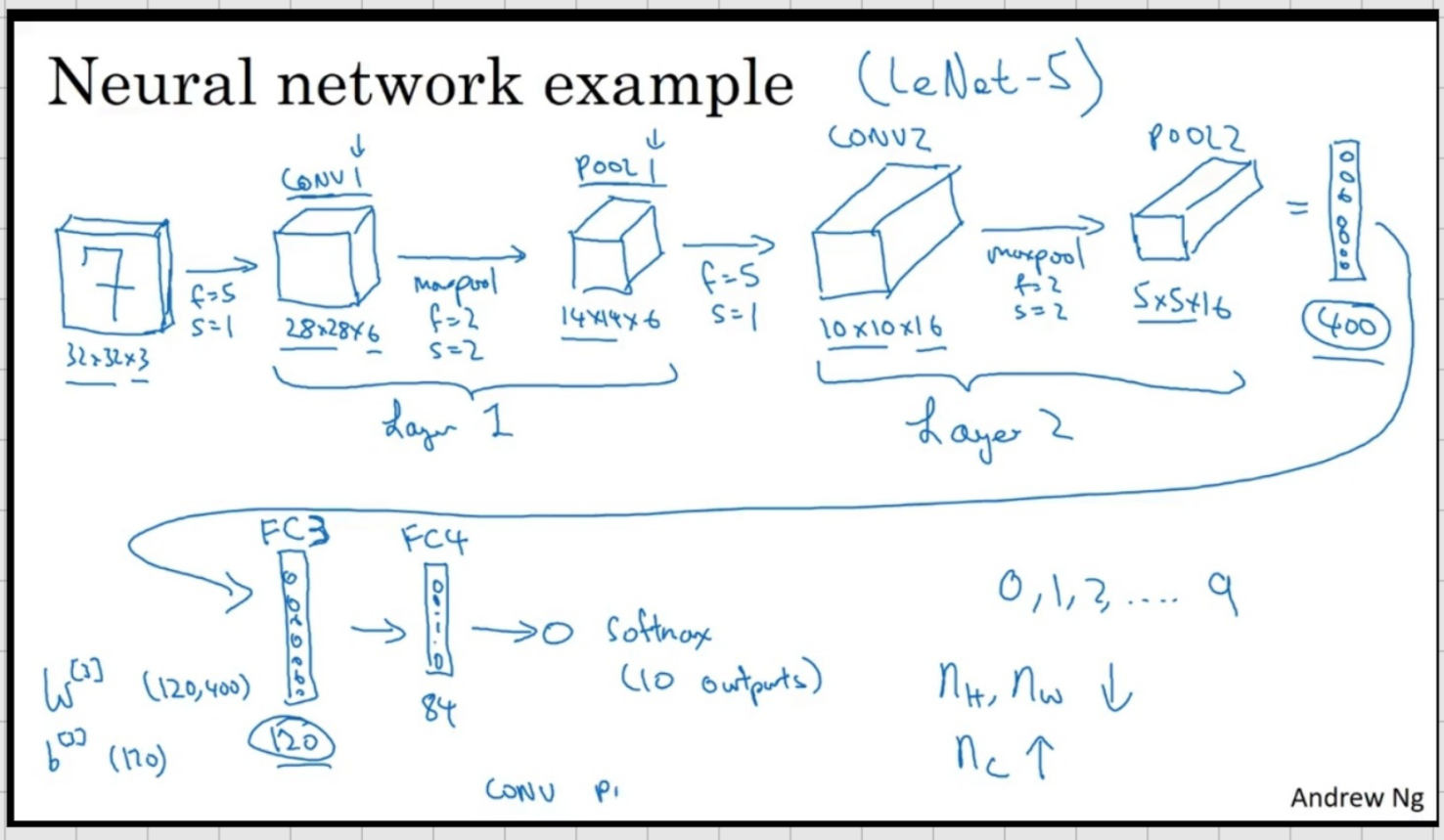
Average pooling : take average

↳

$f = 2$, $s = 2$

usually not use padding

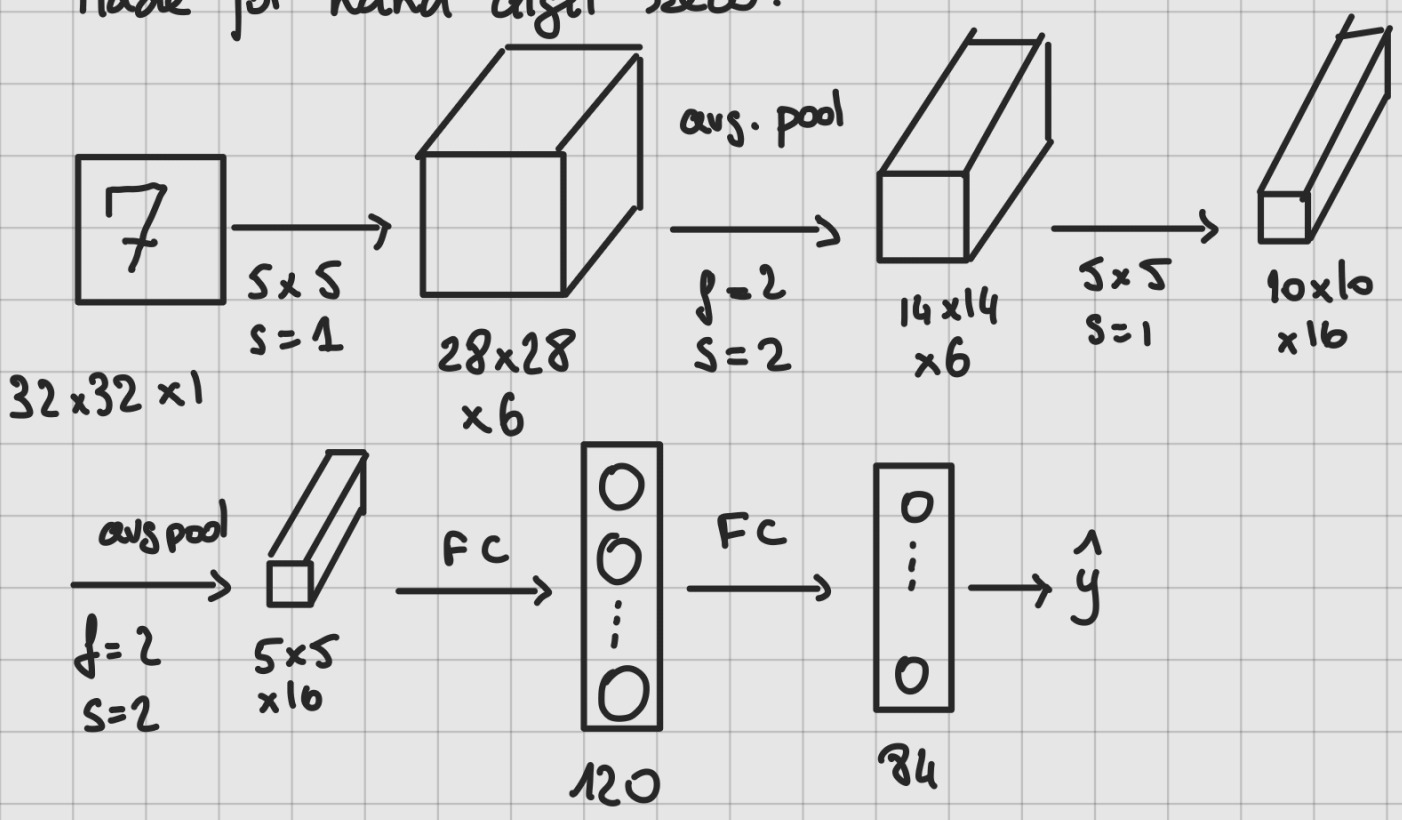
Example


Neural network example (LeNet-5)
CONV1
POOL1
CONV2
POOL2
7
f=5
s=1
32×32×3
28×28×6
maxpool
f=2
s=2
14×14×6
f=5
s=1
10×10×16
maxpool
f=2
s=2
5×5×16
400
Layer 1
Layer 2
FC3
FC4
120
84
Softmax
(10 outputs)
$W^{[3]}$ (120,400)
$b^{[3]}$ (120)
0, 1, 2, ... 9
n_H, n_W ↓
n_C ↑
CONV POOL
Andrew Ng

Classic network

LeNet-5 (1998 LeCun)

Made for hand digit reco.



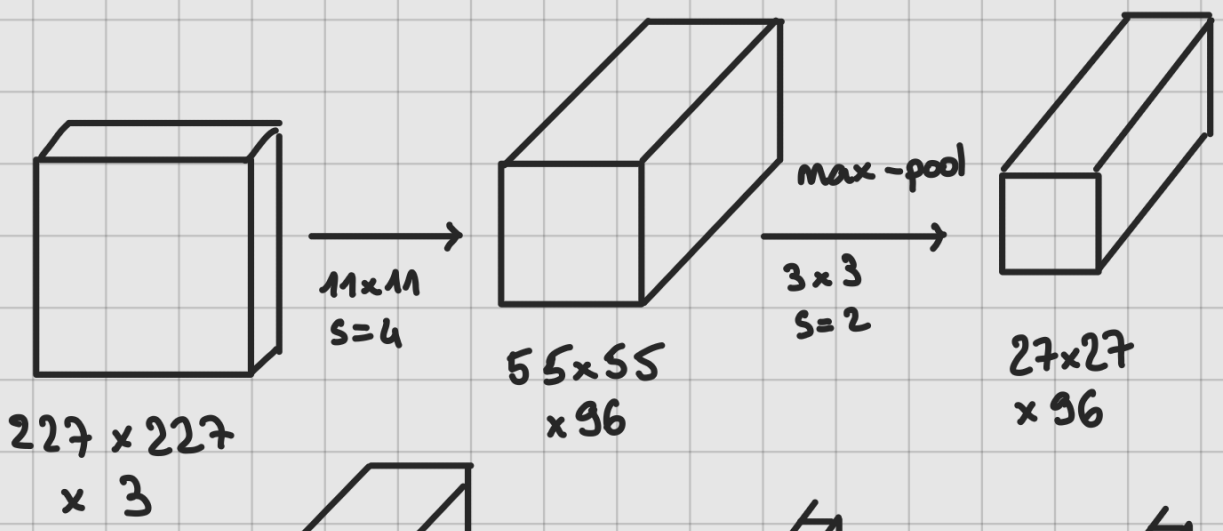
~ 60K parameters

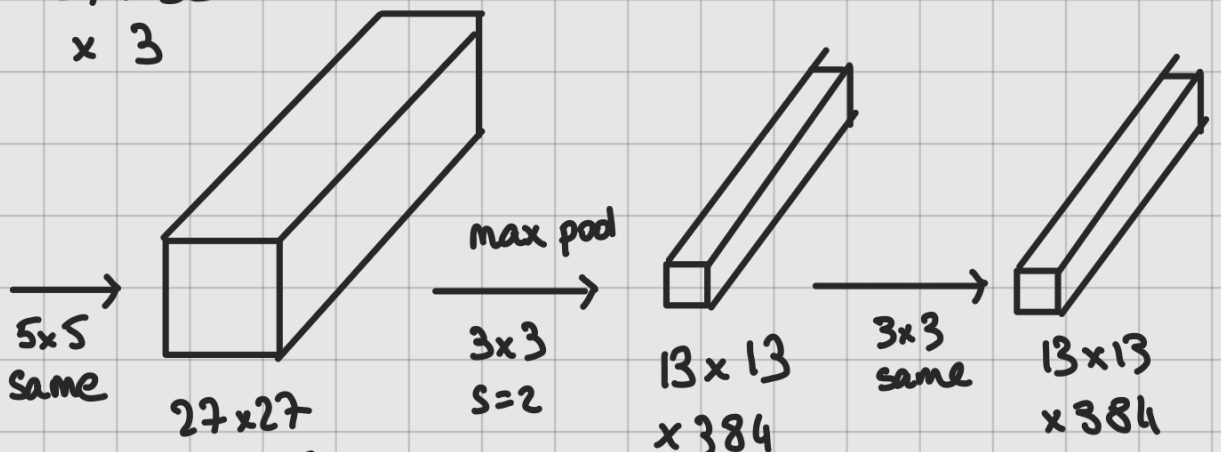
$n_H, n_W \downarrow \quad n_C \uparrow$

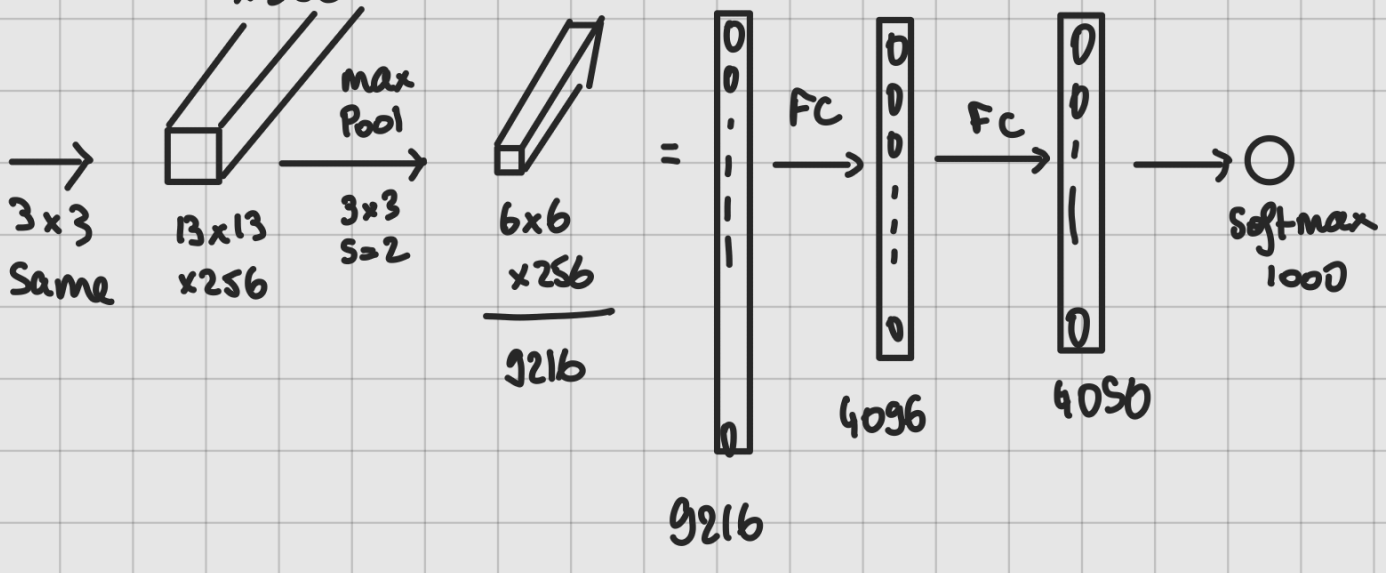
[Conv Pool] - [Conv Pool] - FC - FC - Output

Use sigmoid / tanh

AlexNet (2012 Krizhevsky)







~ 60M parameters

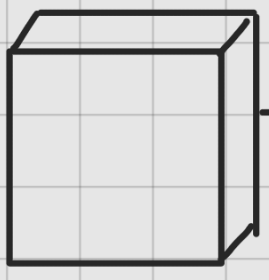
Use ReLu

(multiple GPUs
Local Response Normalized (LRN)
↓
Not use)

VGG-16 (2015 Simonyan, Zisserman)

Conv = 3x3 filter, s = 1, same

Max Pool = 2x2, s = 2



224 x 224 x 3

→ [Conv 64] x 2 → 224 x 224 x 64 → Pool → 112 x 112 x 64 -->

→ [Conv 128] x 2 → 112 x 112 x 128 → Pool → 56 x 56 x 128 → [Conv 256] x 3 → 56 x 56 x 256 -->

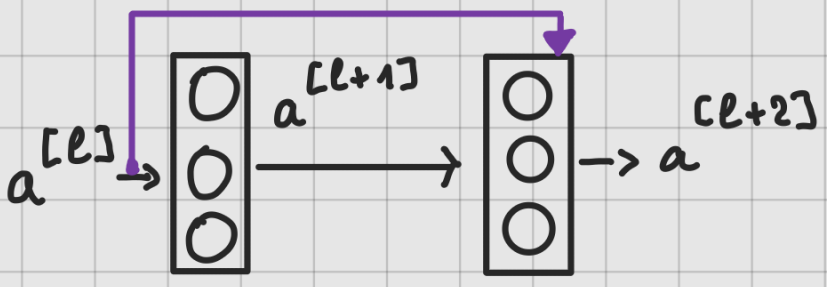
→ Pool → 28 x 28 x 256 → [Conv 512] x 3 → 28 x 28 x 512 → Pool → 14 x 14 x 512 -->

→ [Conv 512] x 3 → 14 x 14 x 512 → Pool → 7 x 7 x 512 → FC 4096 → FC 4096 → Softmax 1000

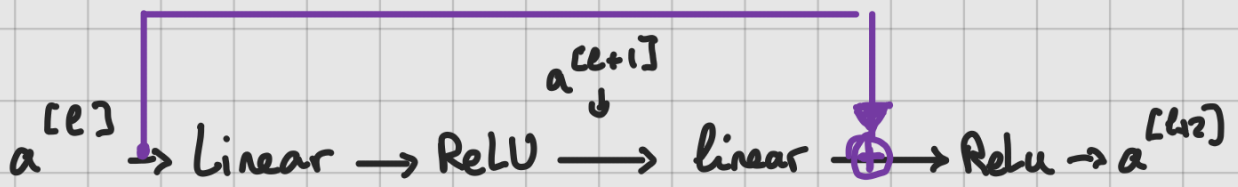
~ 138 M params

ResNets

Residual Block



"short cut" / Skip connection

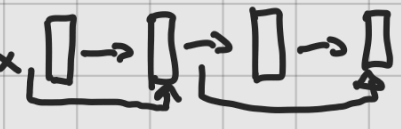


$$z^{[l+1]} = W^{[l+1]} a^{[l]} + b^{[l+1]} \qquad a^{[l+1]} = g(z^{[l+1]})$$

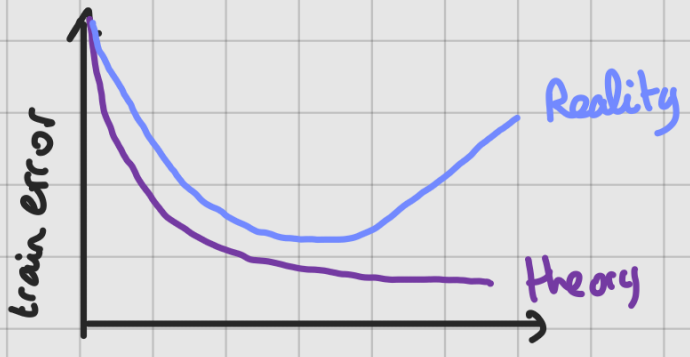
$$z^{[l+2]} = W^{[l+2]} a^{[l+1]} + b^{[l+2]} \qquad a^{[l+2]} = g(z^{[l+2]})$$

$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$

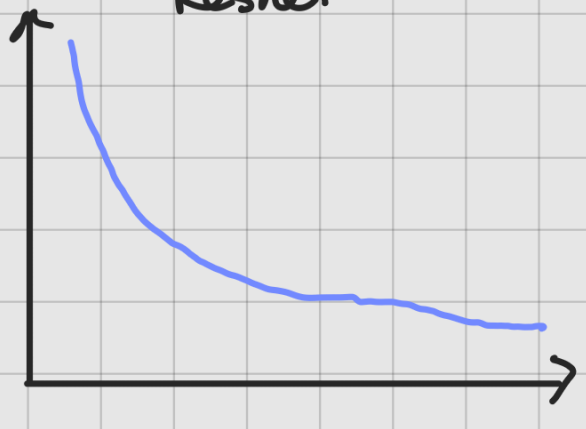
"Plain Network" 

"Residua Net" 

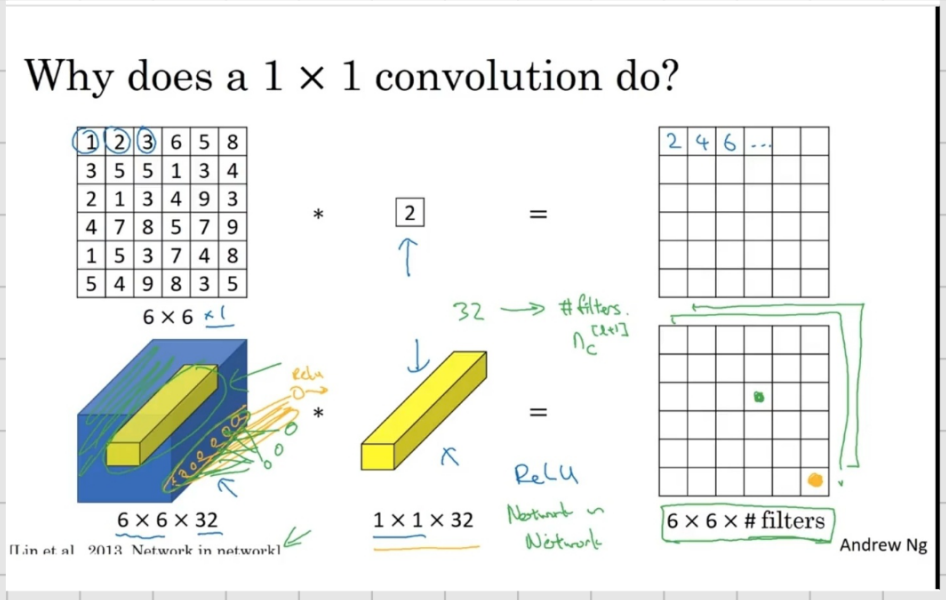
Plain



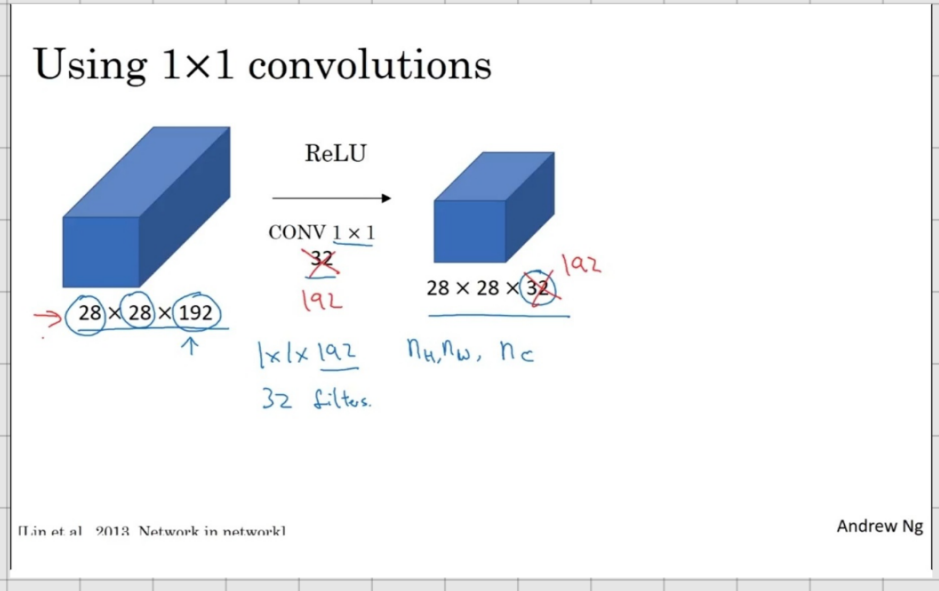
Resnet



Networks in Networks and 1x1 convolution (2013, Lin)

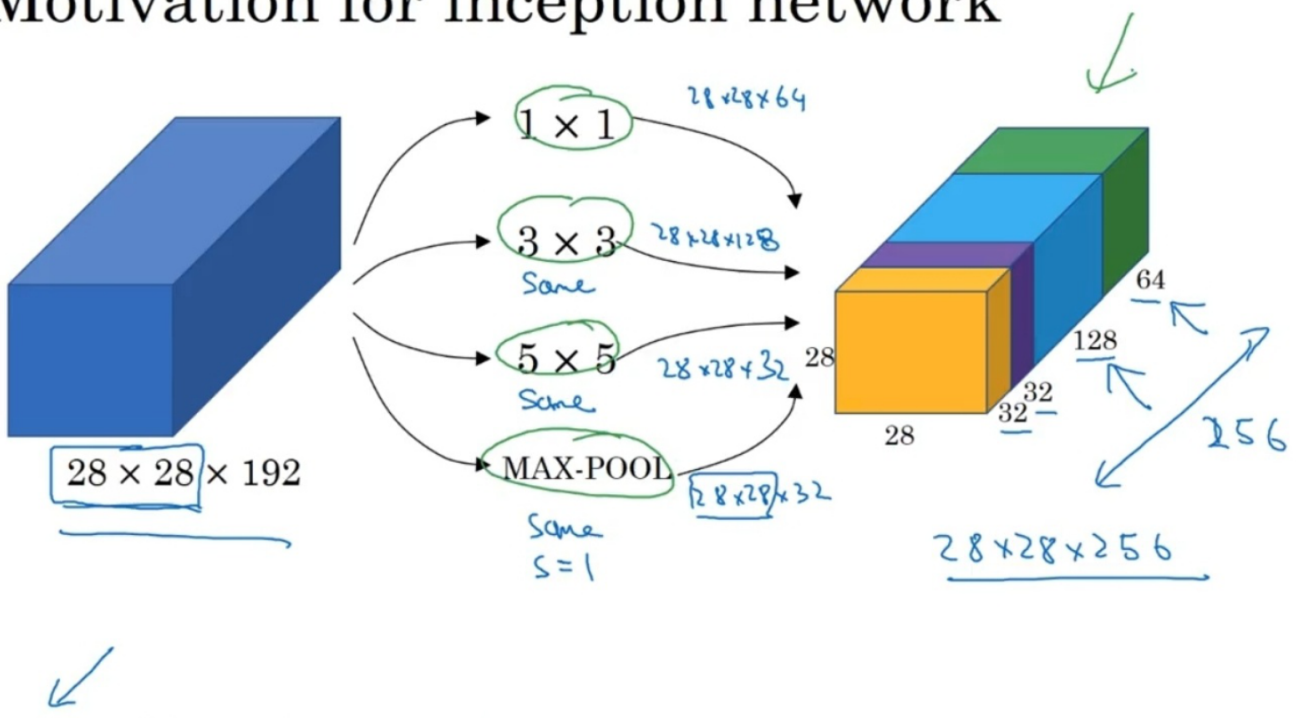


Shrink nb of filters



Inception Network (2014 Szegedy)

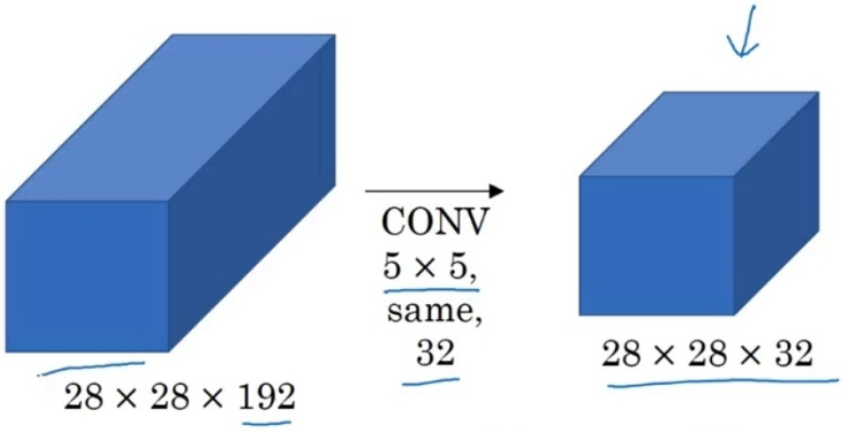
Motivation for inception network



[Szegedy et al. 2014. Going deeper with convolutions]

Andrew Ng

The problem of computational cost



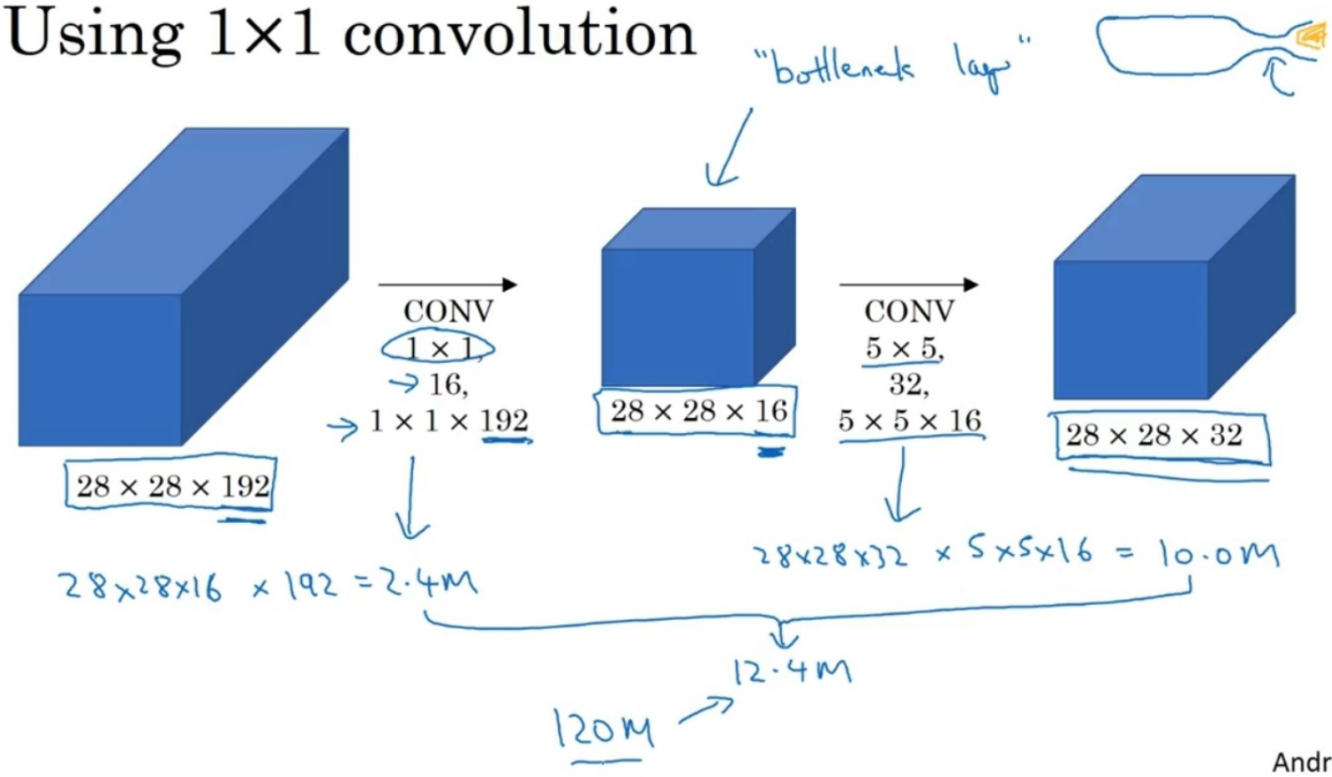
32 filters. filter one 5×5×192.

28×28×32 × 5×5×192 = 120M.

Andrew Ng

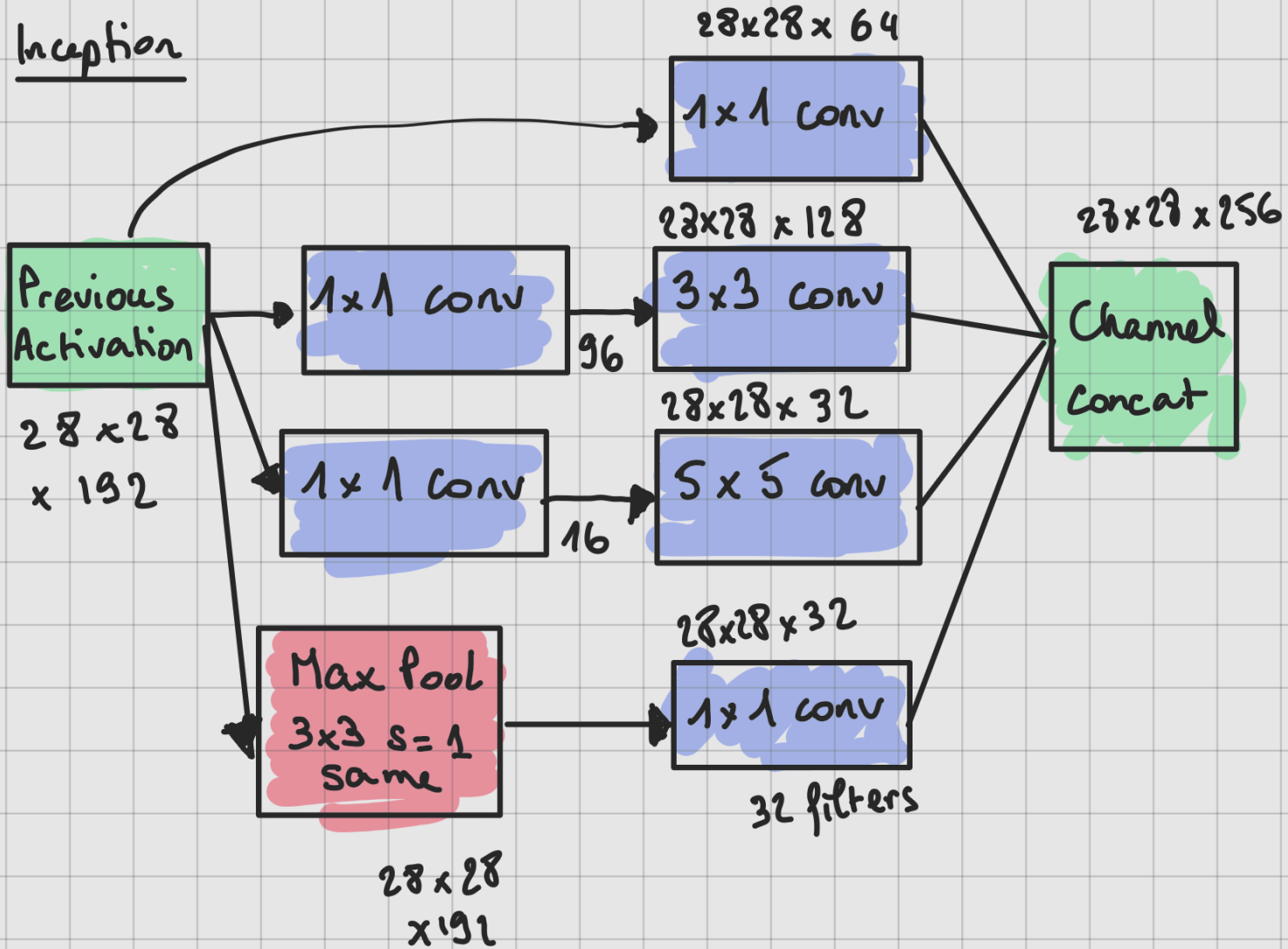
Solution ↓

Using 1×1 convolution



Andrew Ng

Inception

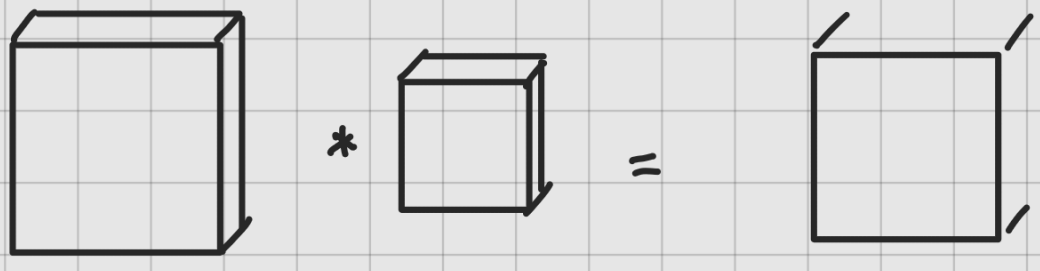


googLeNet

MobileNet (2017 Howard)

- Low computational cost at deployment
- Useful for mobile and embedded visions apps
- Key idea: Normal vs depthwise-separable conv

Normal conv:



6x6x3 * 3x3x3 = 4x4x5

nxnxnc fxfxnc

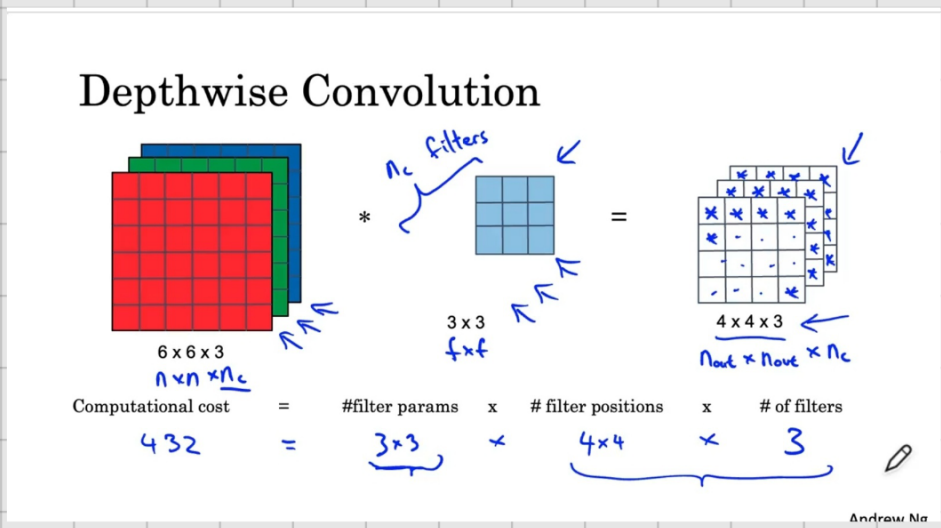
Computational cost = # filter params x # filter position x # filters

3x3x3 x 4x4 x 5

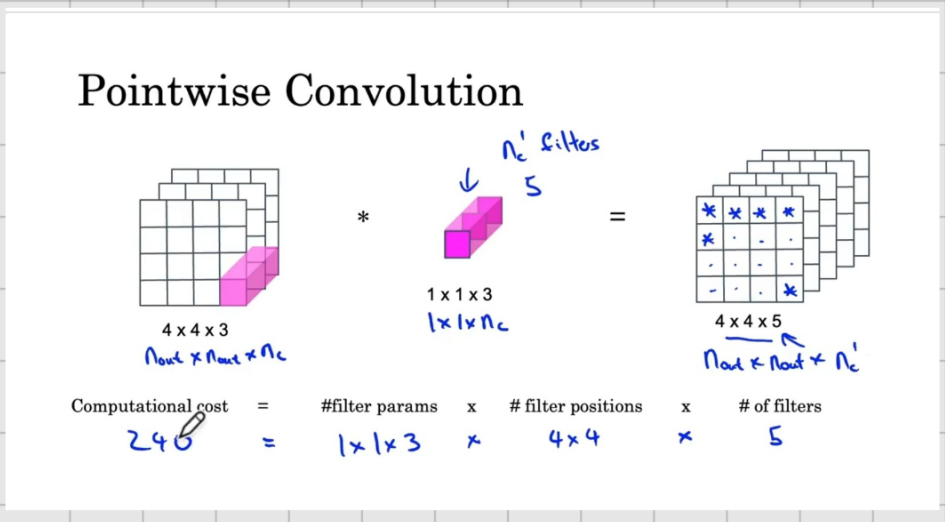
= 2160

Depthwise Separable Convolution

2 steps:



- 432



- 240

Total : 672 vs Normal : 2160

$\frac{672}{2160} = 0{,}31$

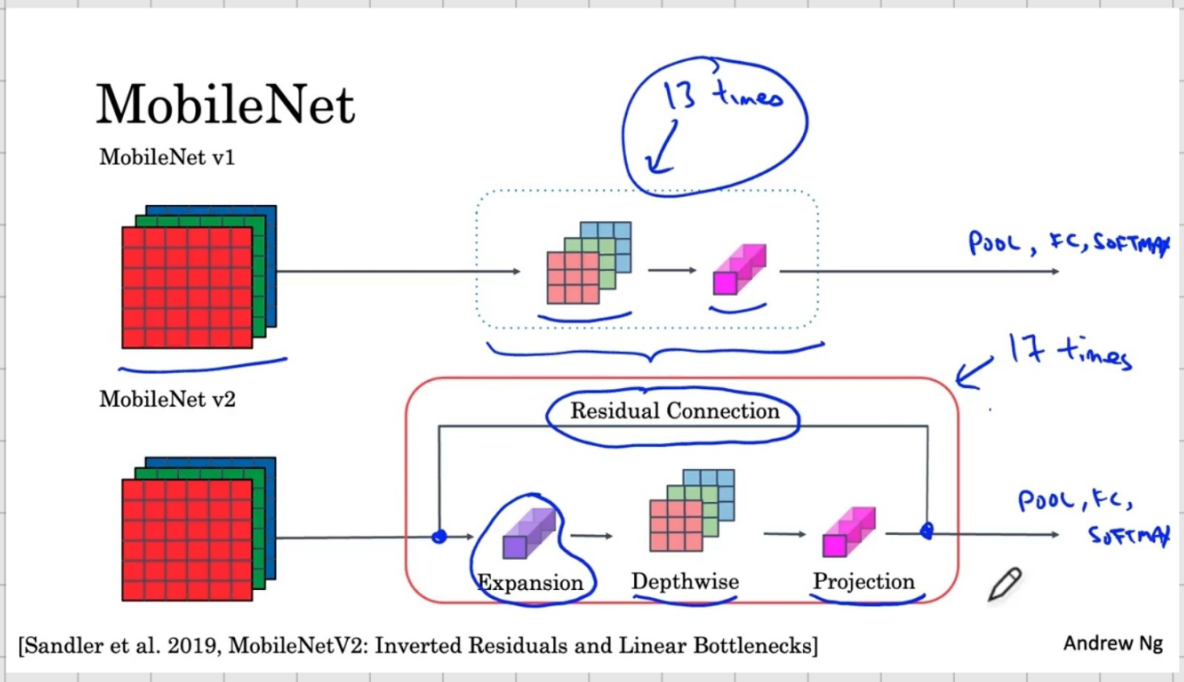
General rule:

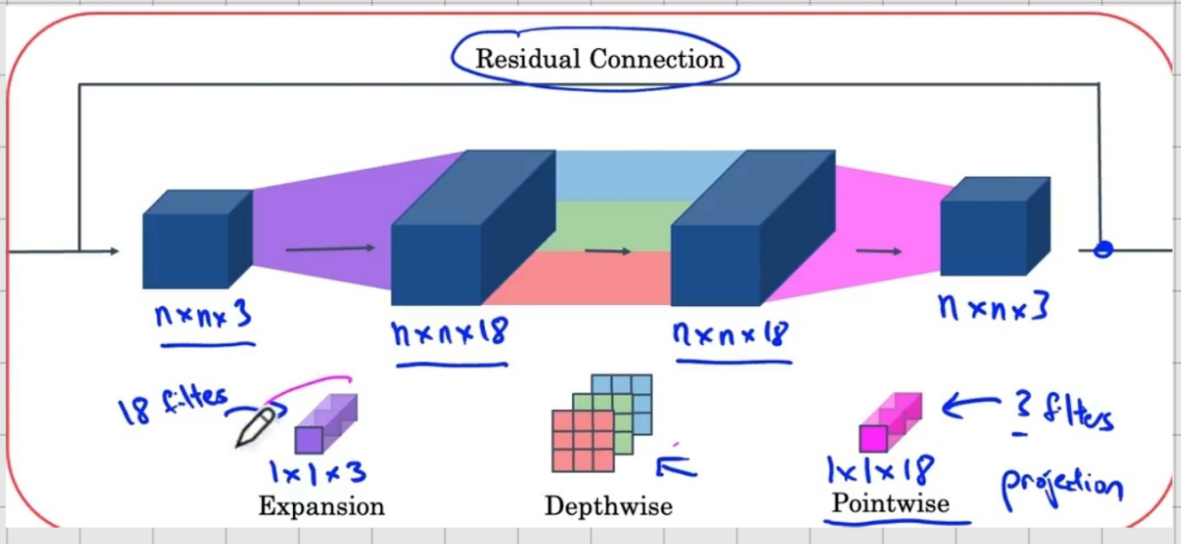
$$= \frac{1}{n_c'} + \frac{1}{f^2}$$

$$\text{exp} = \frac{1}{512} + \frac{1}{3^2}$$

~ 10 times cheaper

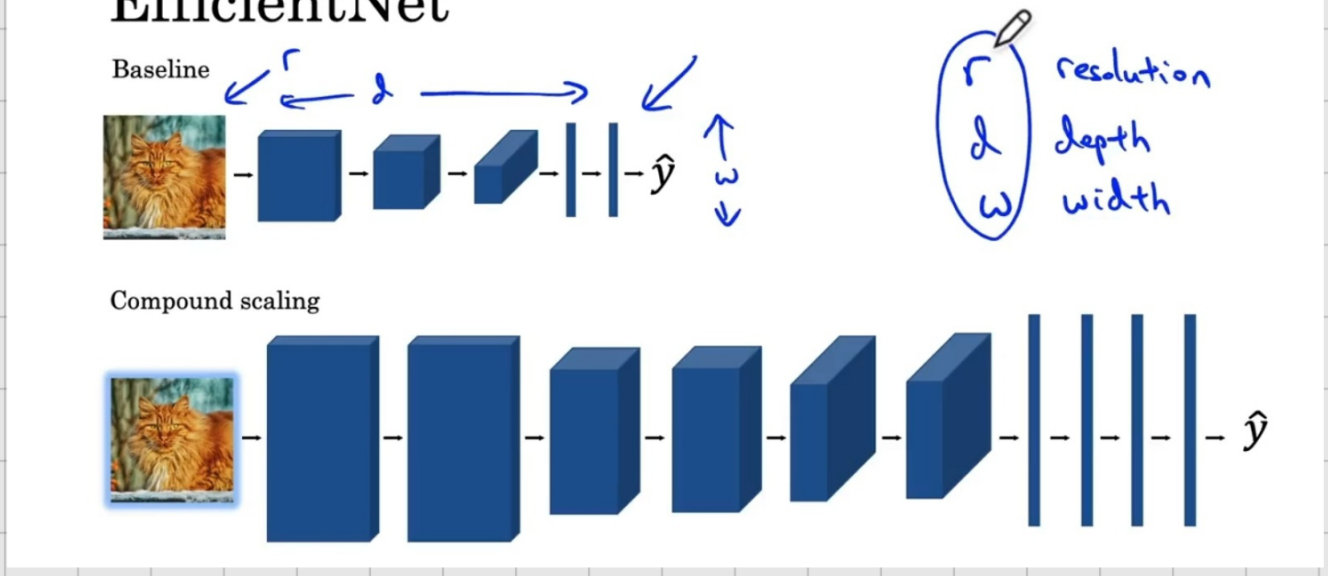
Archi





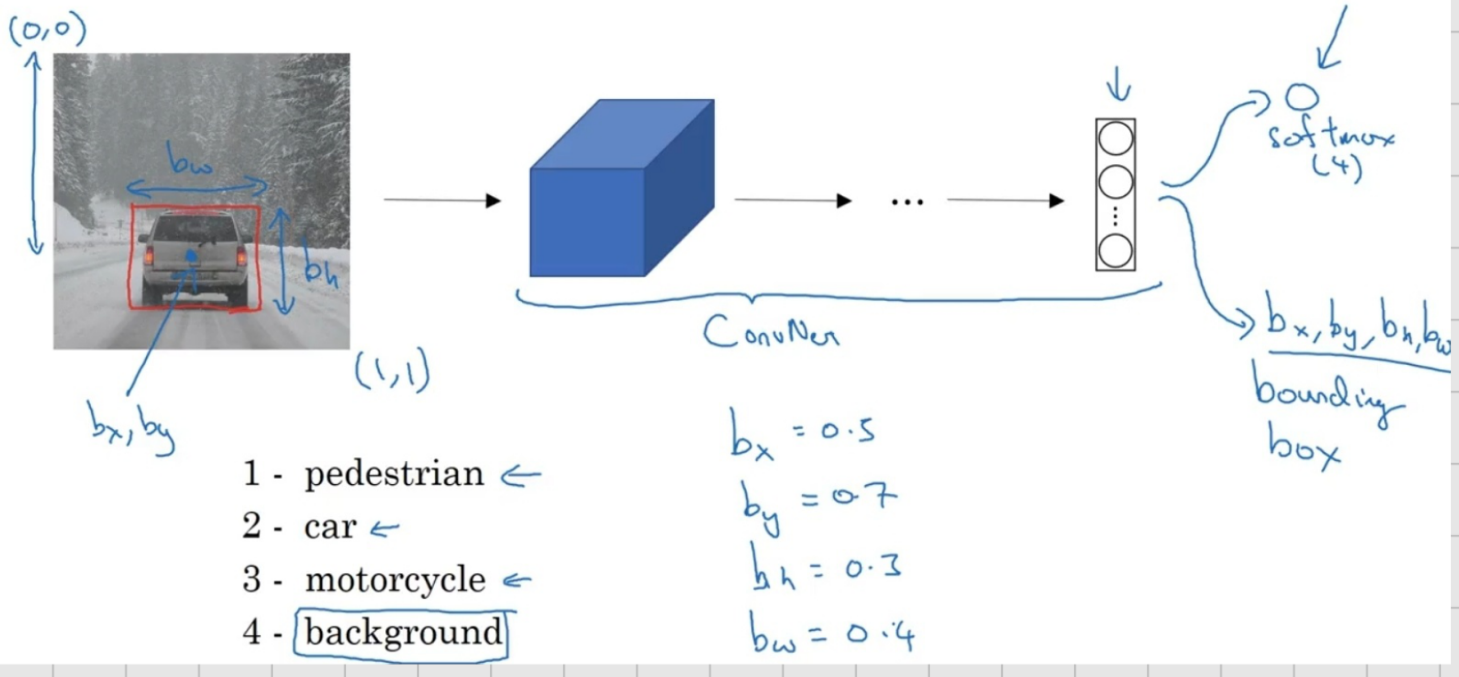
$30 \times 5 + 20 \times 30$

EfficientNet (2019 Tan and Le)



Object Localization

Classification with localization



1 object

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$ ← is their any object?

No obj:

$$y = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ : \\ ? \end{bmatrix}$$ ← "don't care"

$$\mathcal{L} = \begin{cases} (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 + \\ \quad \dots + (\hat{y}_8 - y_8)^2 & \text{if } y_1 = 1 \\ (\hat{y}_1 - y_1)^2 & \text{if } y_1 = 0 \end{cases}$$

↑

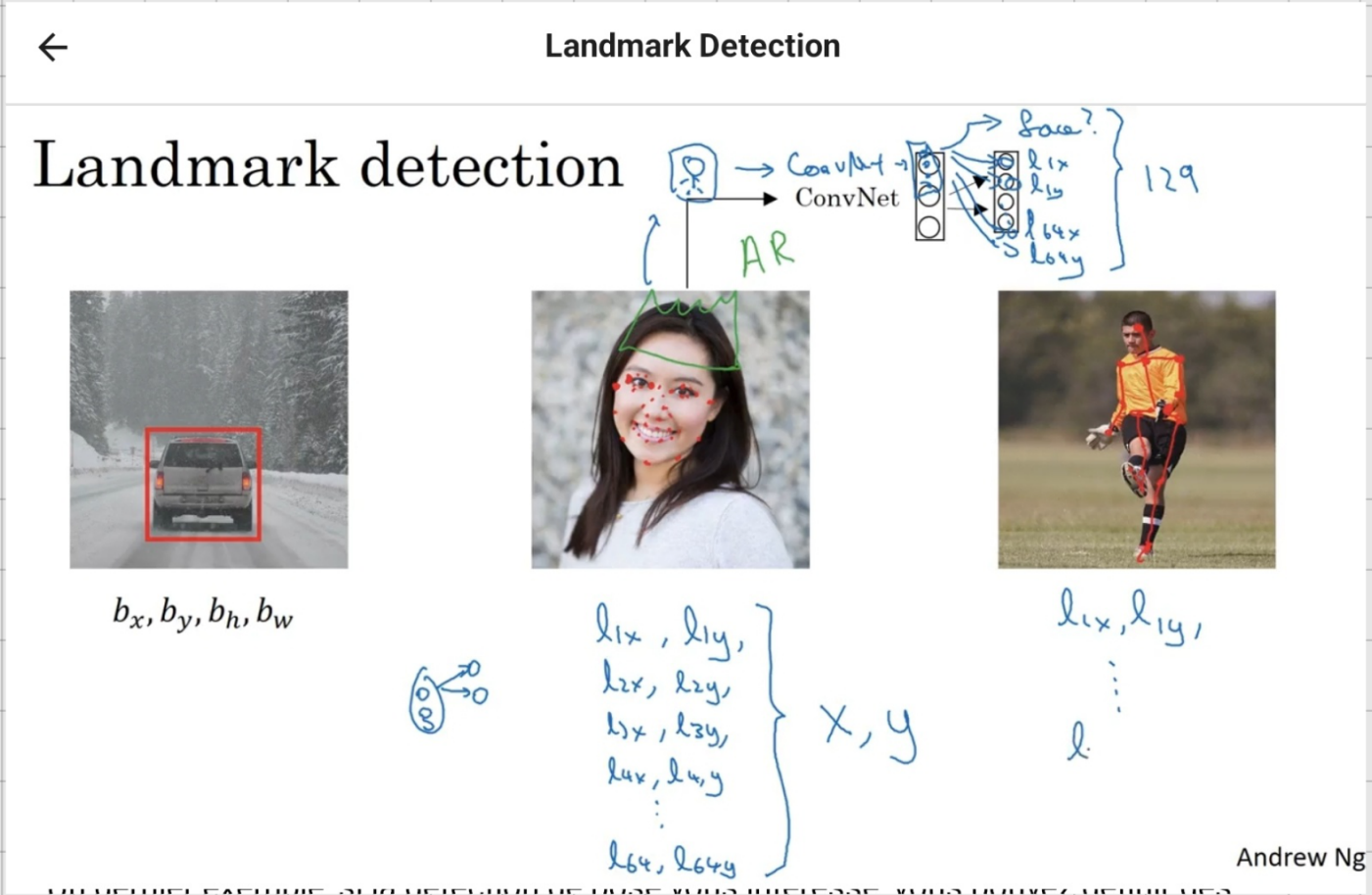
for squared error

Can use | log like feature loss for c_1, c_2, c_3

| squared error for bounding box

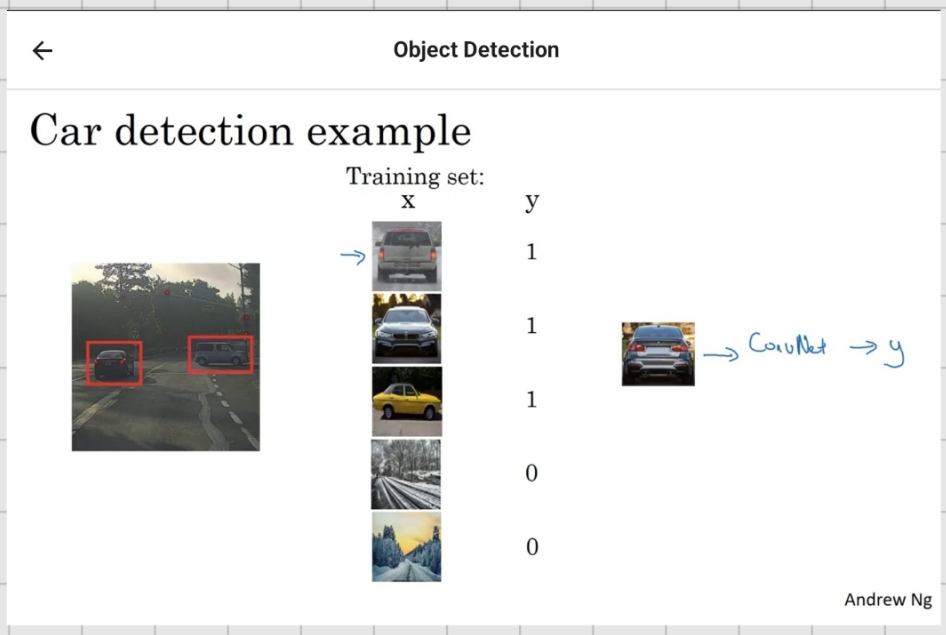
| logistic reg loss for p_c

Landmark Detection



Object Detection

Sliding window detection :



1) Train with car that take entire image

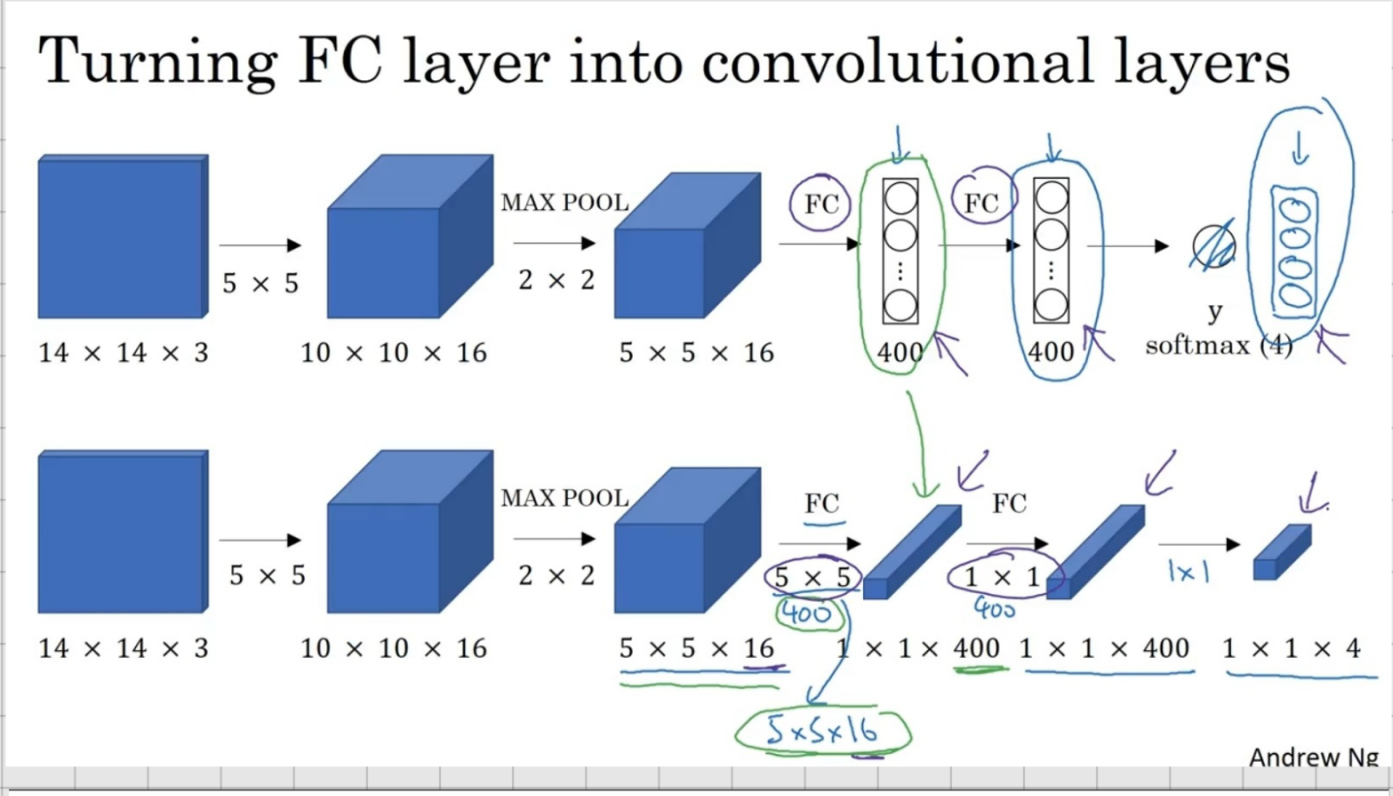
2) Crop part of the image and predict car or not

↑ larger window

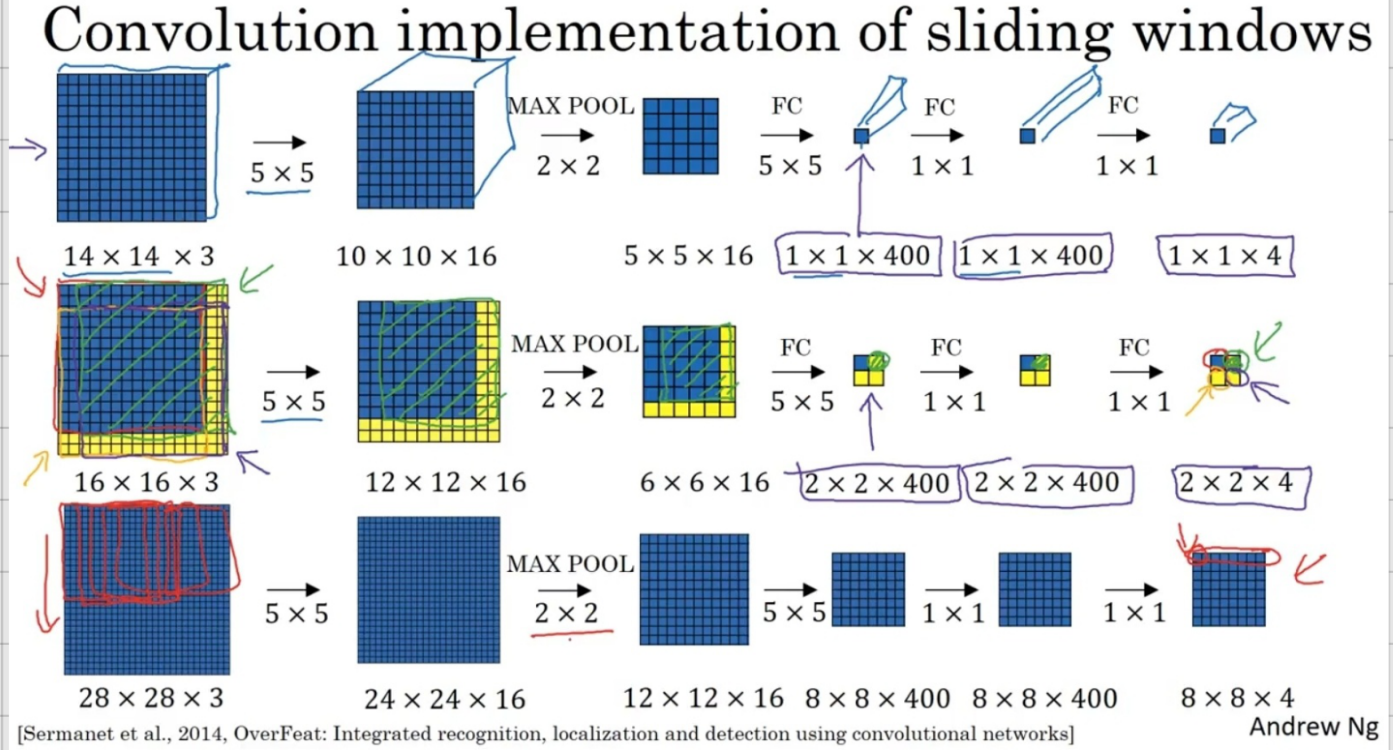
disavantage : Computational cost

Convolutional Implementation of sliding windows

Turning FC layer into conv layer

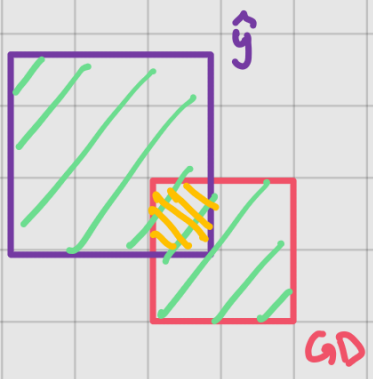


← Convolutional Implementation of Sliding Windows



Intersection vs Union (IoU)

$$= \frac{\text{size of } I}{\text{size of } U}$$

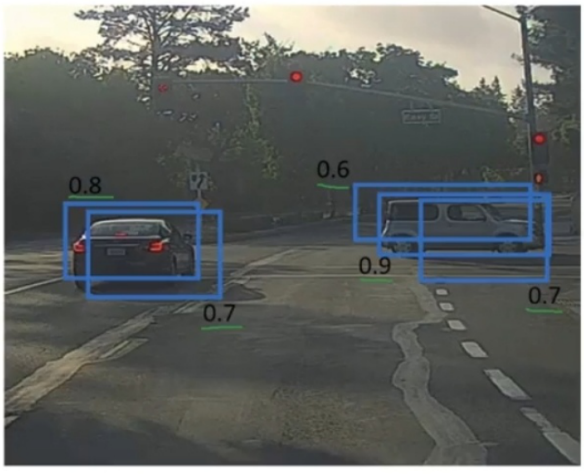


"correct" if IoU ≥ 0.5
0.6
..

Non-max Suppression

Detect each object only once

Non-max suppression example



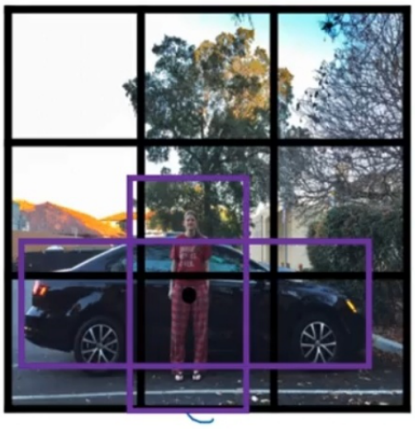
P_c

Andrew Ng

Anchor Boxes

for overlapping obj

Anchor box example



Anchor box 1: Anchor box 2:

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Andrew Ng

YOLO Algorithm

Outputting the non-max supressed outputs

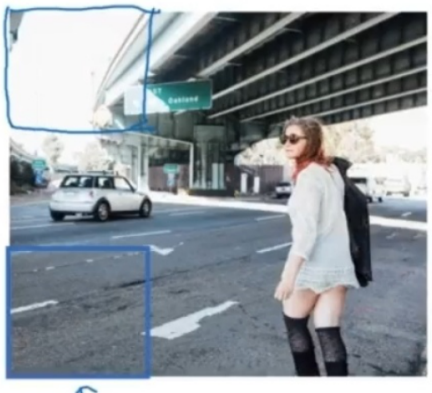
- For each grid call, get 2 predicted bounding boxes.
- Get rid of low probability predictions.
- For each class (pedestrian, car, motorcycle) use non-max suppression to generate final predictions.

Andrew Ng

R-CNN

Region CNN

Region proposal: R-CNN



Segmentation algorithm

~2000

[Girshik et. al, 2013, Rich feature hierarchies for accurate object detection and semantic segmentation] Andrew Ng

Faster algorithms

R-CNN: Propose regions. Classify proposed regions one at a time. Output label + bounding box.

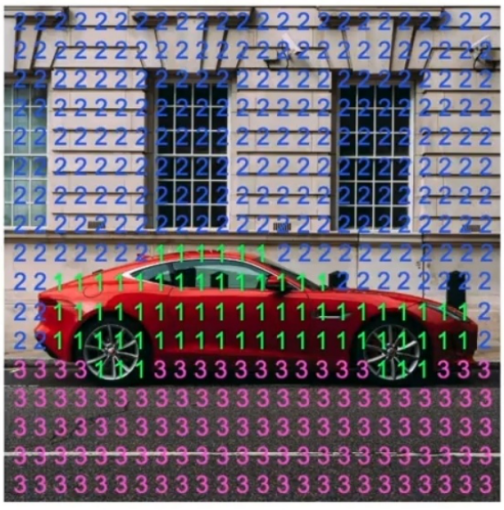
Fast R-CNN: Propose regions. Use convolution implementation of sliding windows to classify all the proposed regions.

Faster R-CNN: Use convolutional network to propose regions.

[Girshik et. al, 2013. Rich feature hierarchies for accurate object detection and semantic segmentation]
[Girshik, 2015. Fast R-CNN]
[Ren et. al, 2016. Faster R-CNN: Towards real-time object detection with region proposal networks] Andrew Ng

Semantic Segmentation with U-Net

Per-pixel class labels



1. Car
2. Building
3. Road

Segmentation Map

Andrew Ng

Transpose Convolution

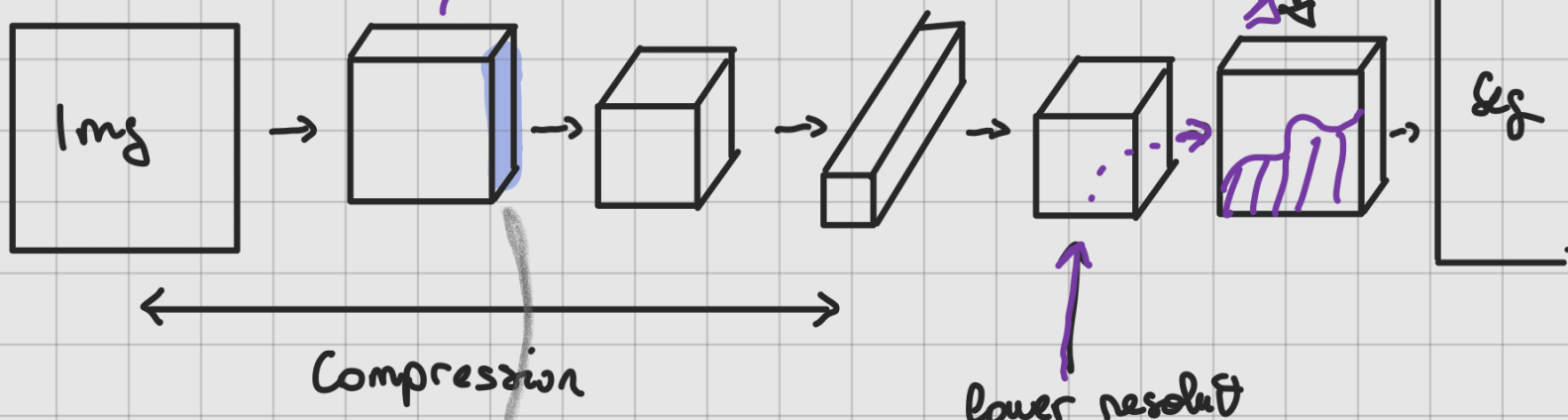
Normal conv : 6x6 x 3 * 3x3 x 3 x5 = 4x4 x5

Transpose conv:

low level info
but high resolut

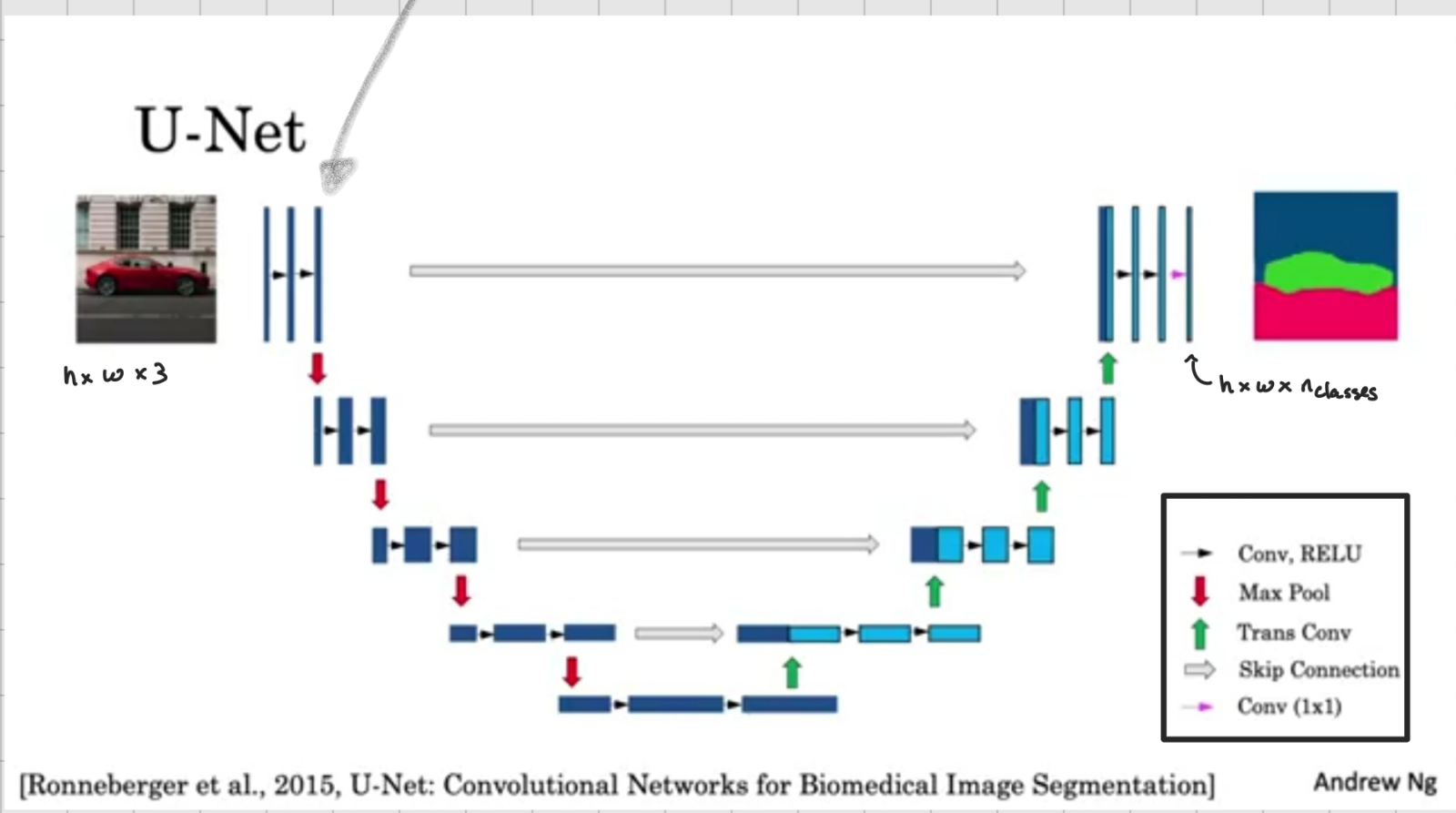
skip connect

U-Net



Compression

lower resolut
but high level info



[Ronneberger et al., 2015, U-Net: Convolutional Networks for Biomedical Image Segmentation]

Andrew Ng

Face Recognition

Face verification:

- Input image, name / ID 1:1
- Output whether the input = claimed person

Face reco:

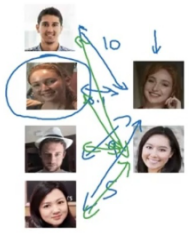
- Has a database of K persons 1:K
- Output ID if the image is any of the K persons

Learning a "similarity" function

→ d(img1,img2) = degree of difference between images

If d(img1,img2) $\leq \tau$ "same" } Verification.

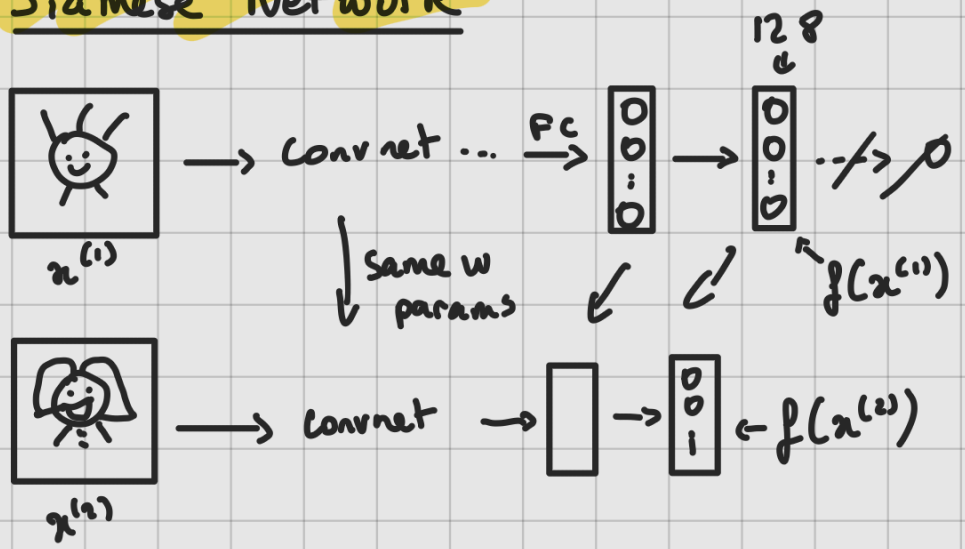
$> \tau$ "different" }



d(img1, img2)

Andrew Ng

Siamese Network



$$d(x^{(1)}, x^{(2)}) = \| f(x^{(1)}) - f(x^{(2)}) \|_2^2$$

Learn parameters so that:

If $x^{(i)}, x^{(j)}$ same person $\rightarrow$ $d(x^{(i)}, x^{(j)})$ is small

If $x^{(i)}, x^{(j)}$ diff person $\rightarrow$ $d(x^{(i)}, x^{(j)})$ is large

Triplet Loss

Learning Objective

Anchor (A)

Positive (P)

Anchor (A)

Negative (N)

$d(A,P) = 0.5$ $\quad d(A,N) = 0.7$

Want: $\underbrace{\| f(A) - f(P) \|^2}_{d(A,P)} + \alpha \leq \underbrace{\| f(A) - f(N) \|^2}_{d(A,N)}$ ($\alpha = 0.2$)

$\| f(A) - f(P) \|^2 - \| f(A) - f(N) \|^2 + \alpha \leq 0$ $\quad f(img) = \vec{0}$

α: margin

[Schroff et al.,2015, FaceNet: A unified embedding for face recognition and clustering]

Andrew Ng

Given 3 images A, P, N:

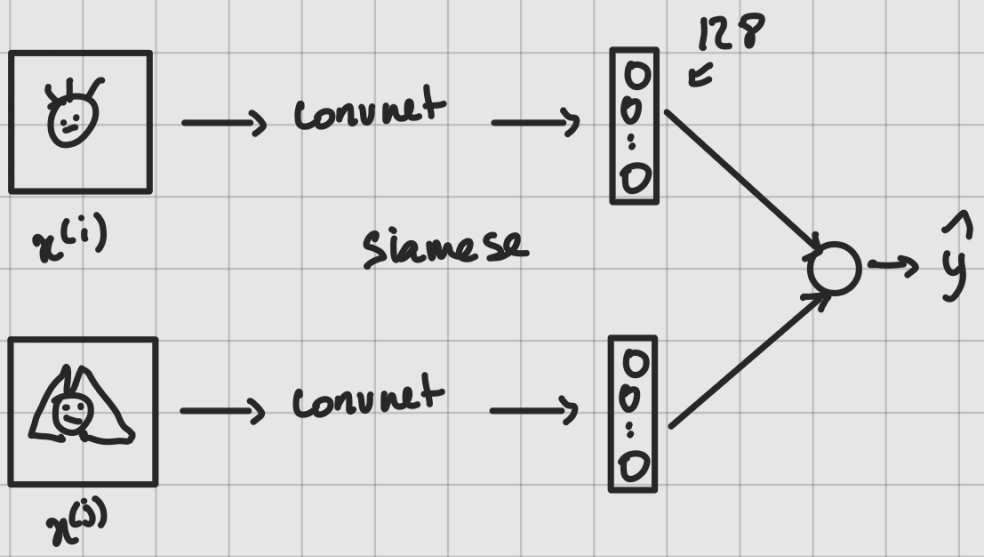
$$\mathcal{L}(A,P,N) = \max\left(\| f(A) - f(P) \|^2 - \| f(A) - f(N) \|^2 + \alpha \, , \, 0 \right)$$

During training, if A, P, N are chosen randomly, its easy to satisfied

→ Chose triplets that're "hard" to train on

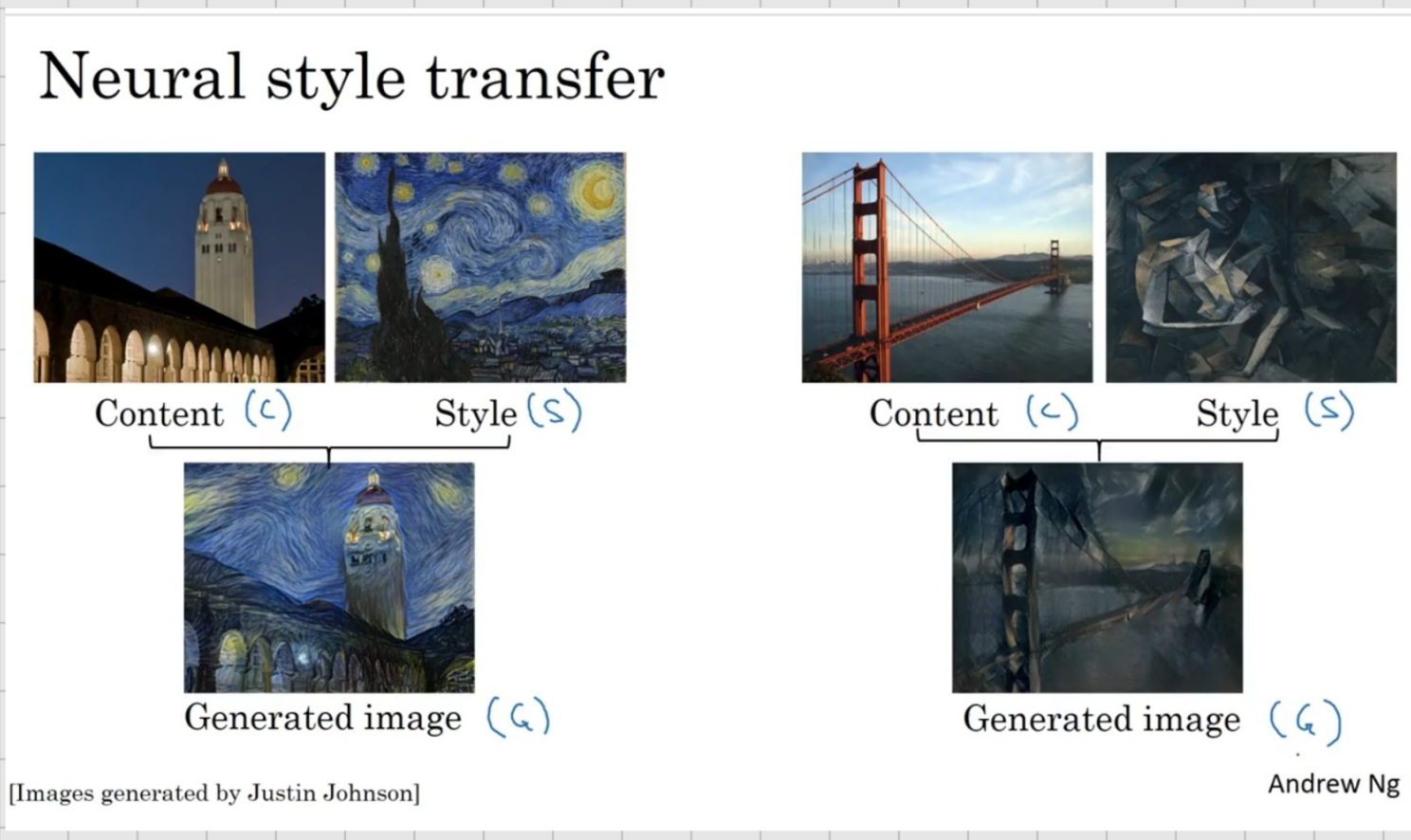
↳ $d(A,P) \approx d(A,N)$

Other way : Binary Classificat For face verif



$$\hat{y} = \sigma\left(\sum_{k=1}^{128} w_k \left| f(x^{(i)})_k - f(x^{(j)})_k \right| + b\right)$$

Neural Style Transfer



Cost Function:

$$J(G) = \alpha J_{content}(C, G) + \beta J_{style}(S, G)$$

Find G:

1. Initiate G randomly G: 100 x 100 x 3
2. Use Gradient Descent to min J(G)

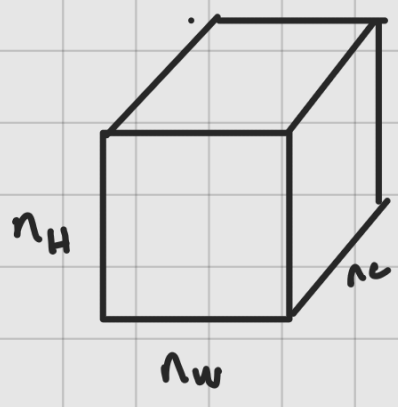
Content Cost Function

- Say you use hidden layer l to compute content cost (l neither to shallow nor to deep)
- Use pre-trained ConvNet (E.g. VGG)
- Let $a^{[l](C)}$ and $a^{[l](G)}$ activation of layer l
- If $a^{[l](C)}$ and $a^{[l](G)}$ are similar -> img similar

$$J_{content}(C,G) = \underbrace{\frac{1}{2}}_{\text{normalizaθ}} \| a^{[l](C)} - a^{[l](G)} \|^2$$

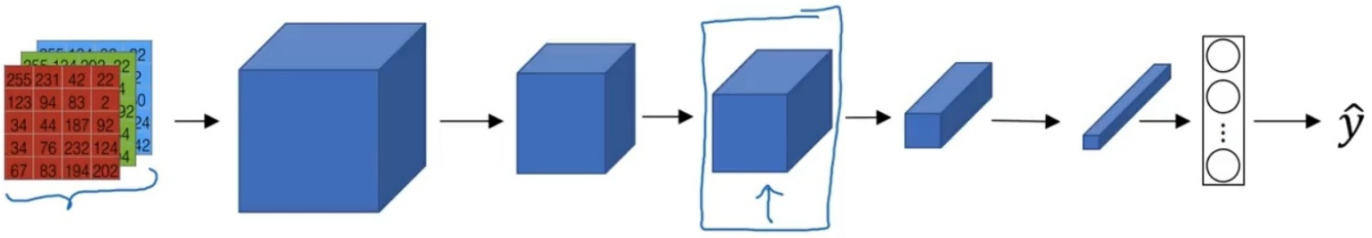
Style Cost Function

Say you are using layer l's activation to measure "style." Define style as correlation between activaθ across chanels.

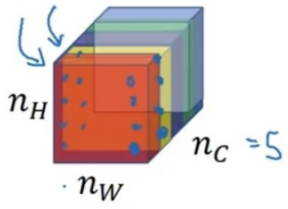


How correlated are the activations across channels?

Meaning of the "style" of an image



Say you are using layer l's activation to measure "style."
Define style as correlation between activations across channels.

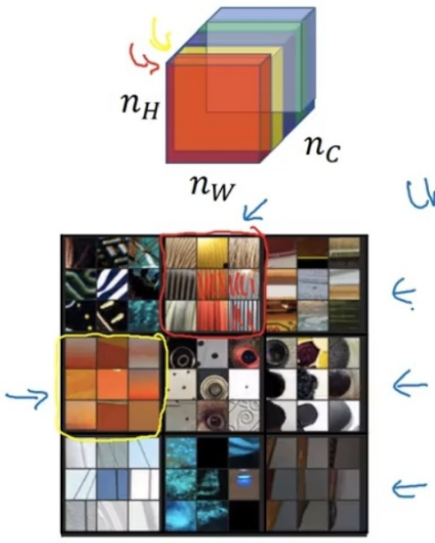


How correlated are the activations across different channels?

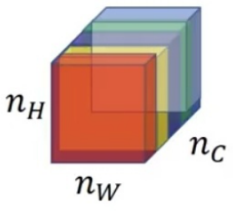
[Gatys et al., 2015. A neural algorithm of artistic style]

Intuition about style of an image

Style image



Generated Image



[Gatys et al., 2015. A neural algorithm of artistic style]

Style matrix

H W C

Let $a^{[l]}_{i,j,k}$ = activation at (i,j,k). $G^{[l]}$ is $n_c^{[l]} \times n_c^{[l]}$

$G^{[l]}_{k,k'}$

$k = 1, \ldots, n_c$

$$G^{[\ell](S)}_{kk'} = \sum_i^{n_H} \sum_j^{n_W} a^{[\ell](S)}_{ijk} \, a^{[\ell](S)}_{ijk'}$$

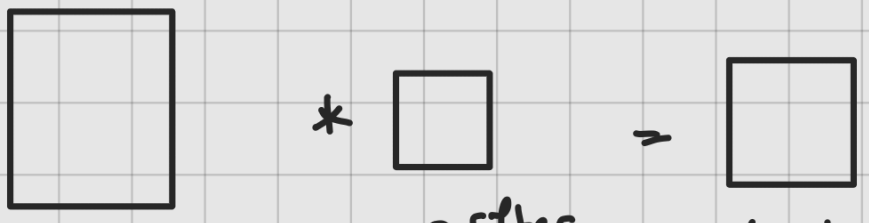
"Gram matrix"

$$G^{[\ell]G}_{kk'} = \sum_i^{n_H} \sum_j^{n_W} a^{[\ell](G)}_{ijk} \, a^{[\ell](G)}_{ijk'}$$

$$J^{[\ell]}_{style}(S,G) = \underbrace{\frac{1}{2 n_H^{[\ell]} n_W^{[\ell]} n_C^{[\ell]}}}_{\text{normalization}} \left\| G^{[\ell](S)} - G^{[\ell](G)} \right\|_F^2$$

$$J_{style}(S,G) = \sum_\ell \lambda^{[\ell]} J^{[\ell]}_{style}(S,G)$$

1D and 3D Generalization



2D input img

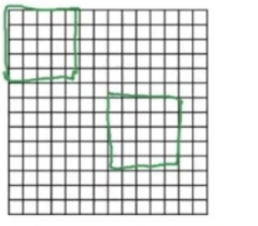
14 x 14

2D filter

5x5

10x10

Convolutions in 2D and 1D



2D input image

14 × 14

2D filter

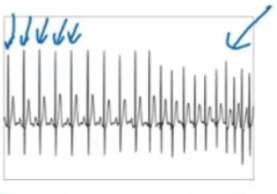
5 × 5

14 × 1 * 5 × 1

→ 10 × 16

10 × 16 * 5 × 16

→ 6 * 32



1	20	15	3	18	12	4	17

14

1	3	10	3	1

5

14 × 14 × 3 * 5 × 5 × 3

→ 10 × 10 × 16

10 × 10 × 16 * 5 × 5 × 16

→ 6 × 6 × 32

Andrew Ng