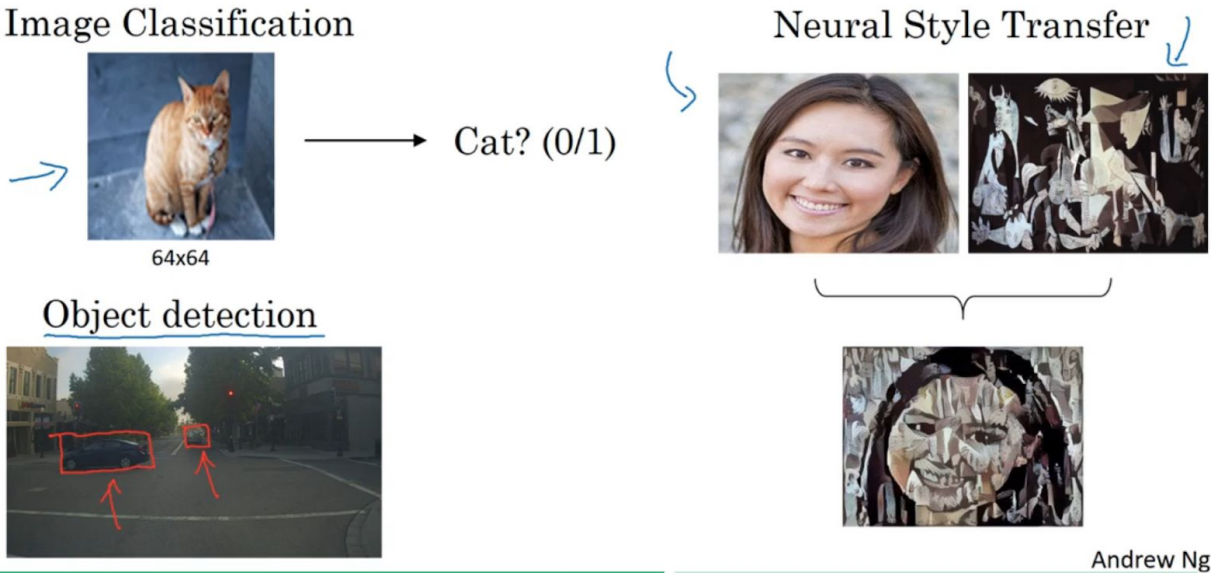
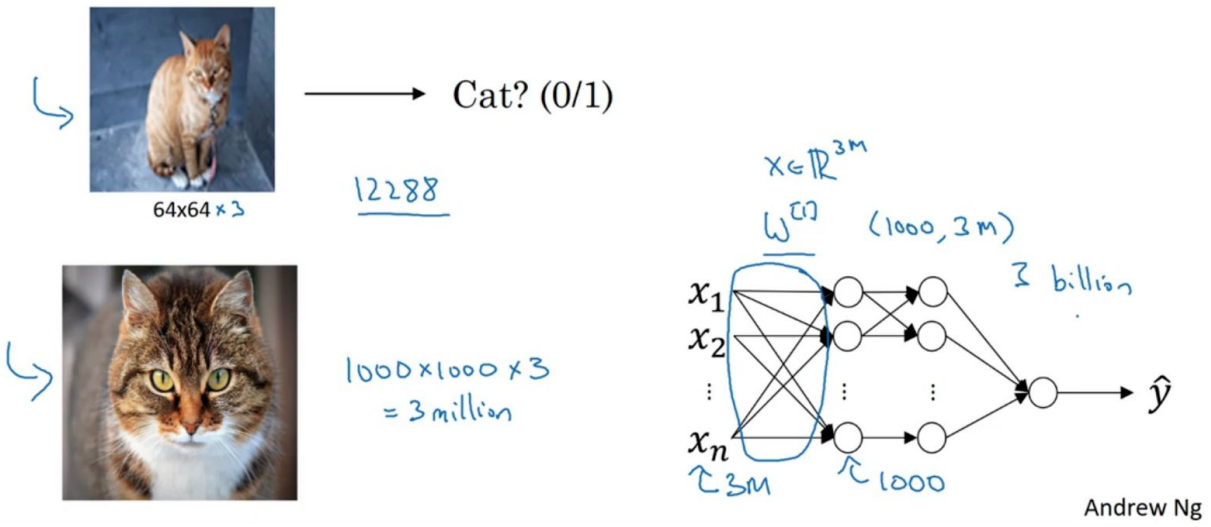
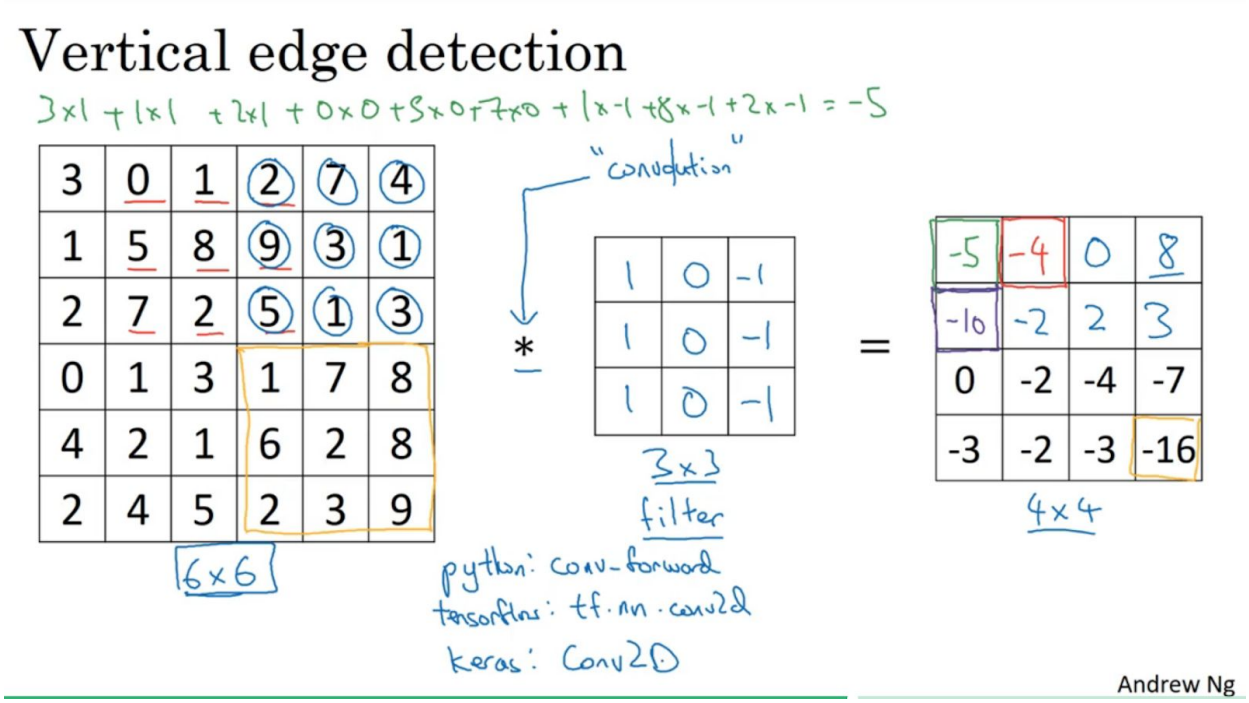
Computer Vision Problems



Deep Learning on large images

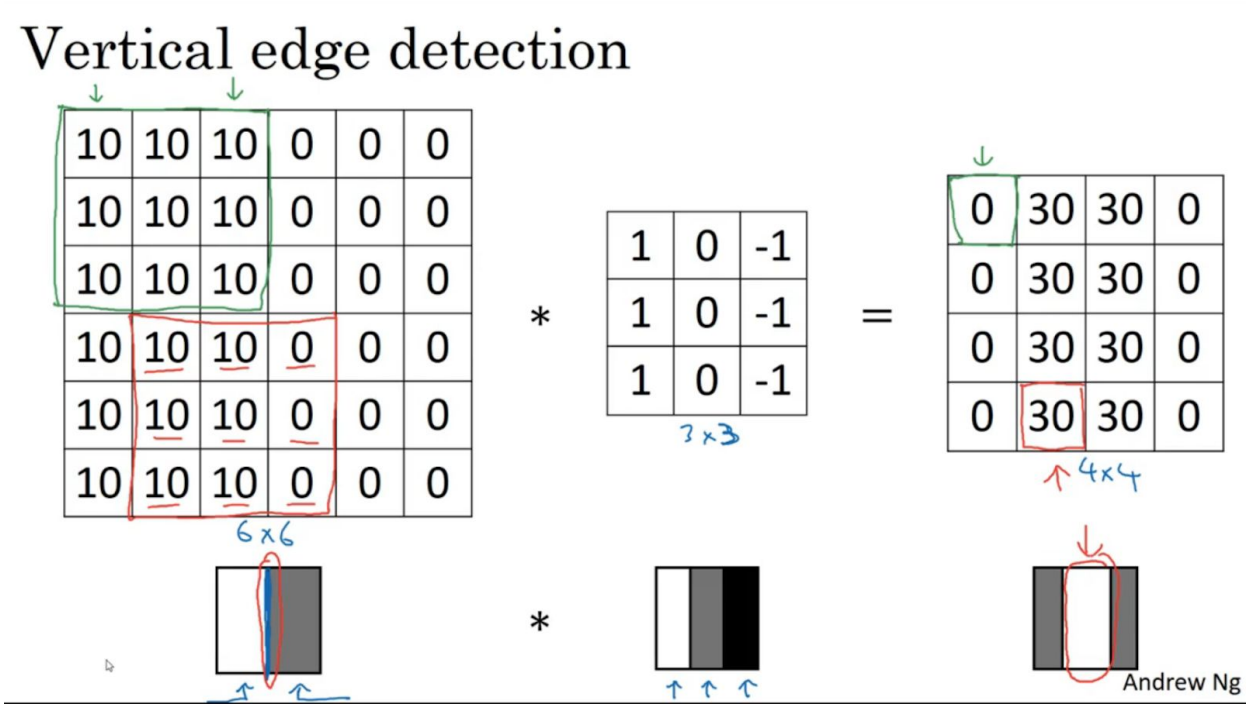


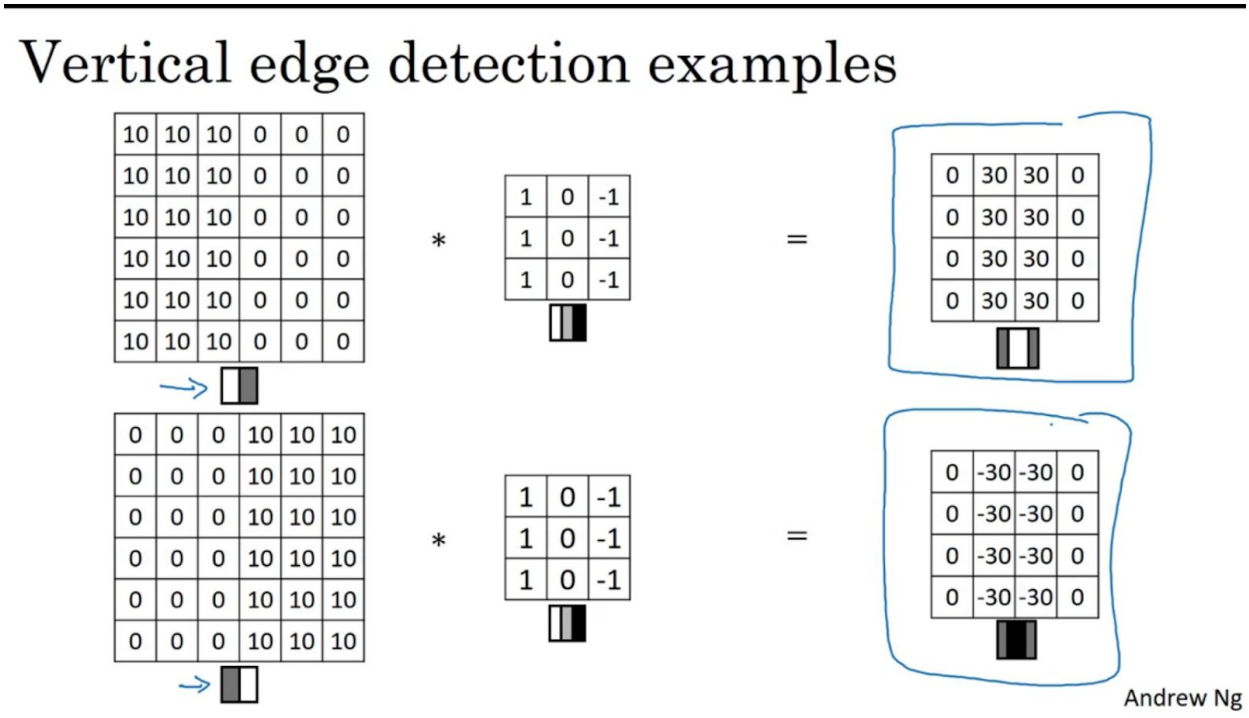
Small images are ok - but what about larger images? Need a way to manage 3b parameters!

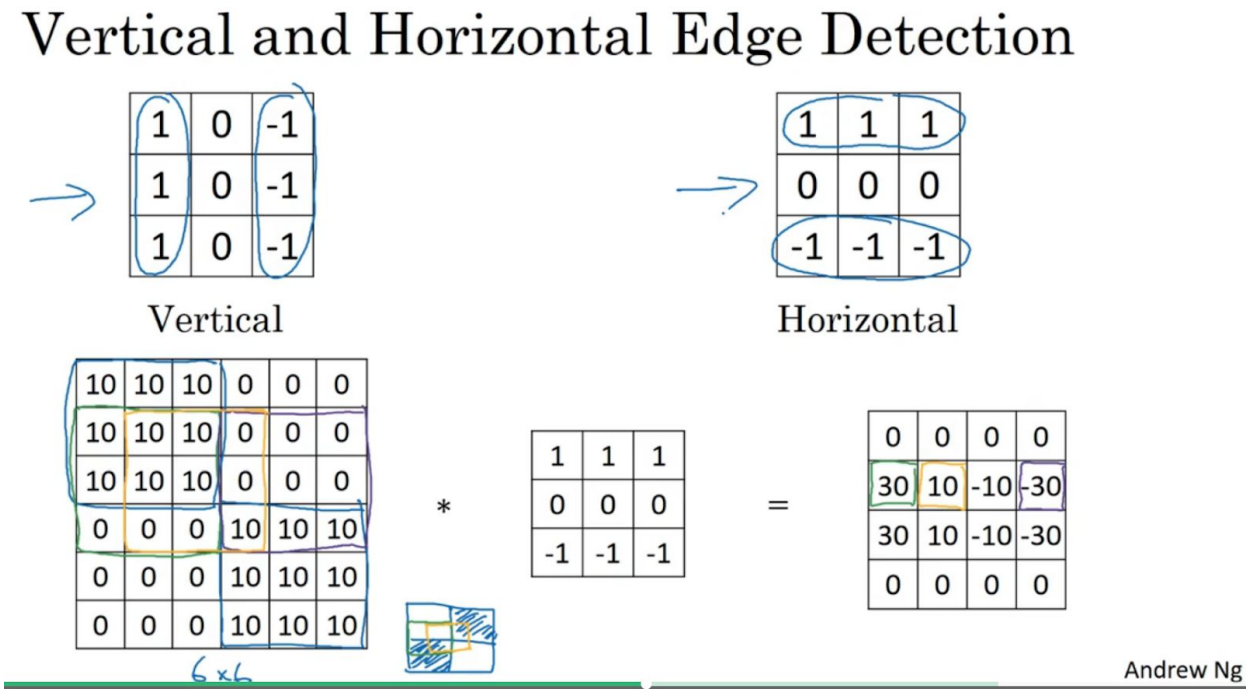


Element wise multiply all parts of the matrix with numbers in the image to get the convolution outputs.

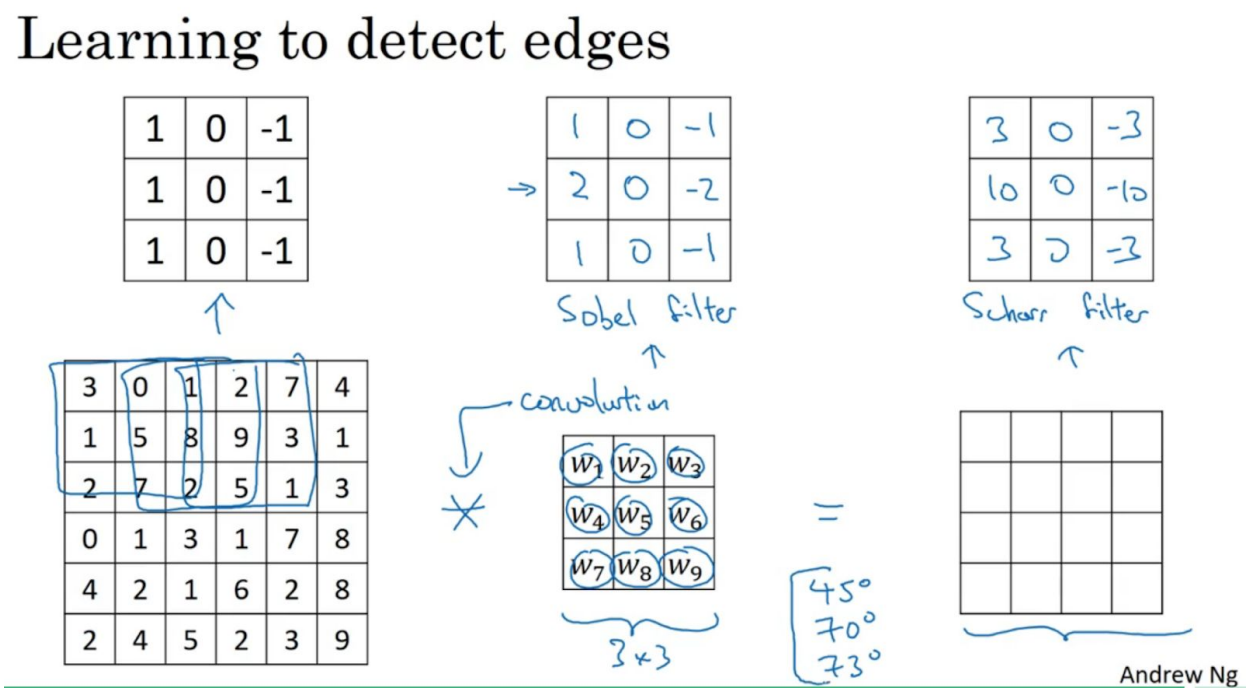
BRILLIANT! Why does kernel actually work? Convolution actually masks on various shades of color and then tries to find the region of space where edge might occur - here in the middle!



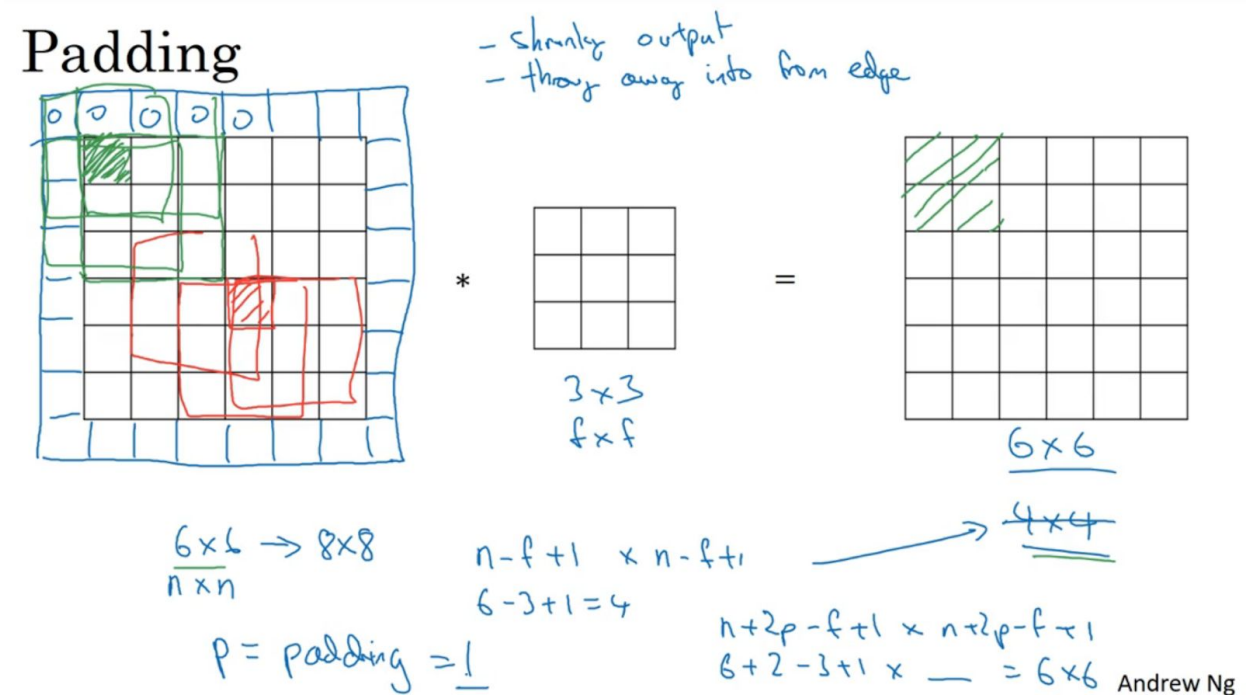




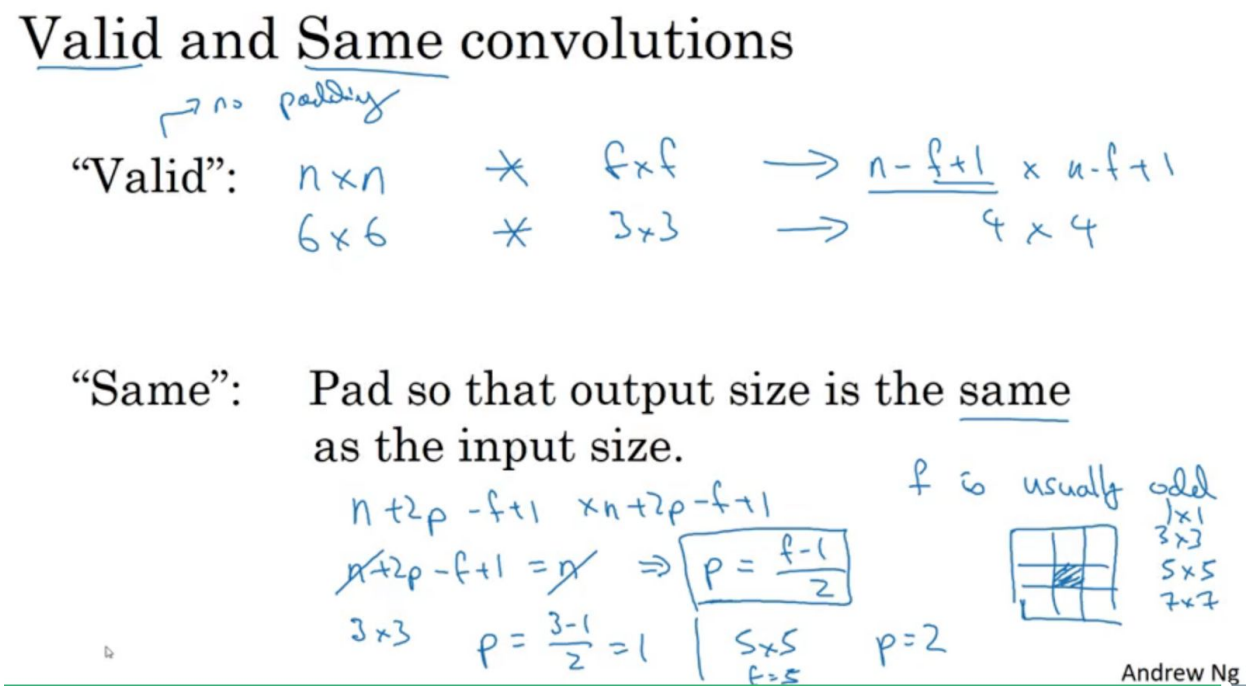
If you go to larger say 1000x1000 images - you won't see the transitions actually (10,-10)



Use deep learning to learn the weights instead of hand coding them! YOu can even detect random edges like at 73 degrees etc. Whatever the model chooses to learn using backprop. Convolution is still the key though.

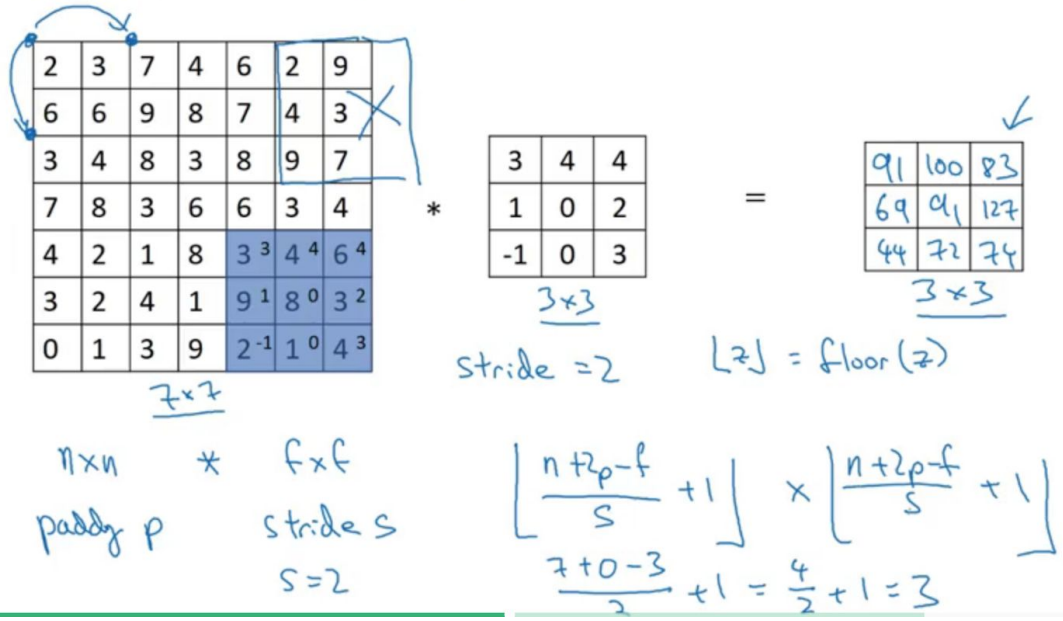


Padding --- > keeps same image dimension which is important for deep networks, otherwise you will lose info! Also preserves info about corners and keeps them.



Odd number f : symmetric padding instead of asymmetric and also has central pixel

Strided convolution



Andrew Ng

Summary of convolutions

$n \times n$ image $\quad f \times f$ filter

padding p $\quad$ stride s

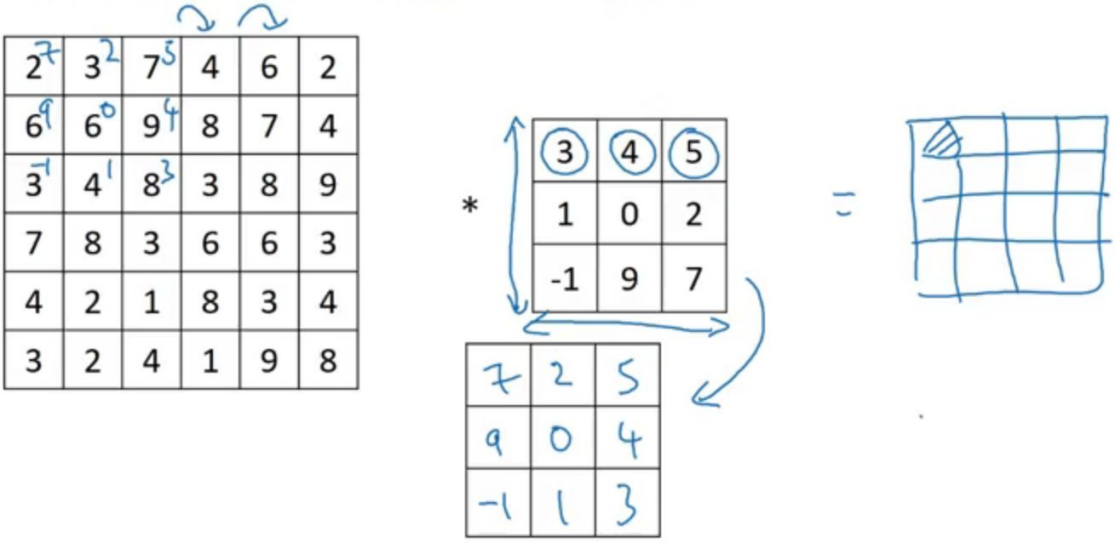
Output

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

OUTPUT SIZE with floor if not an integer

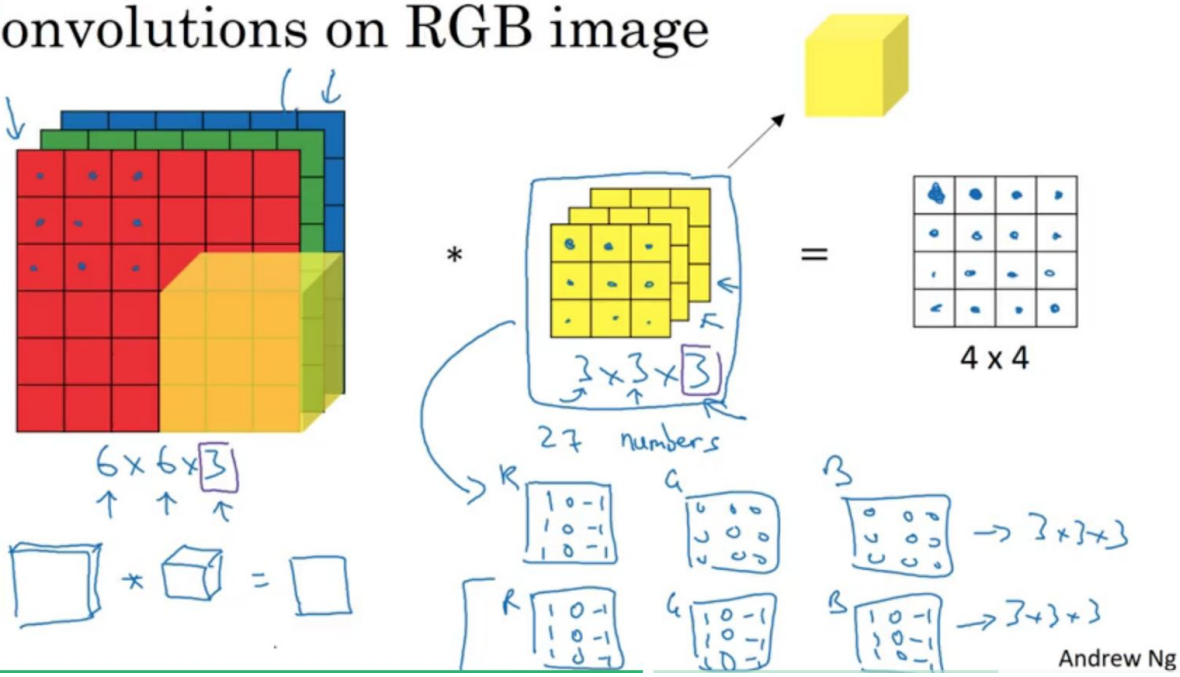
Technical note on cross-correlation vs. convolution

Convolution in math textbook:

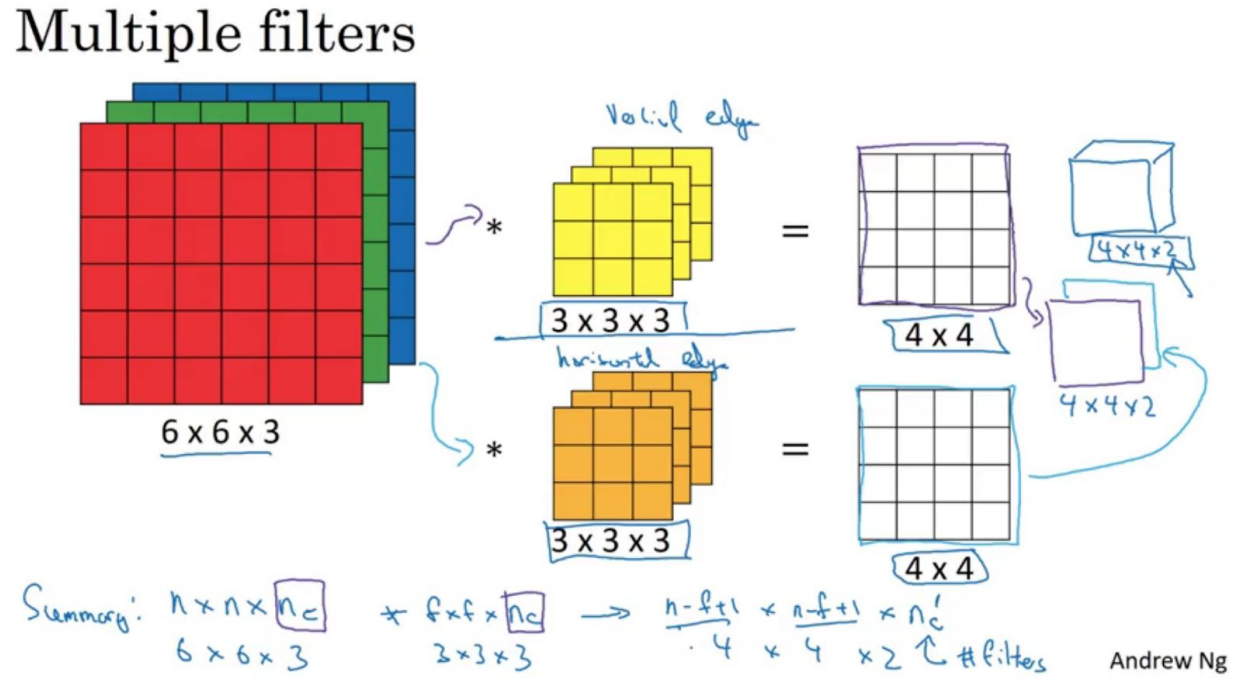


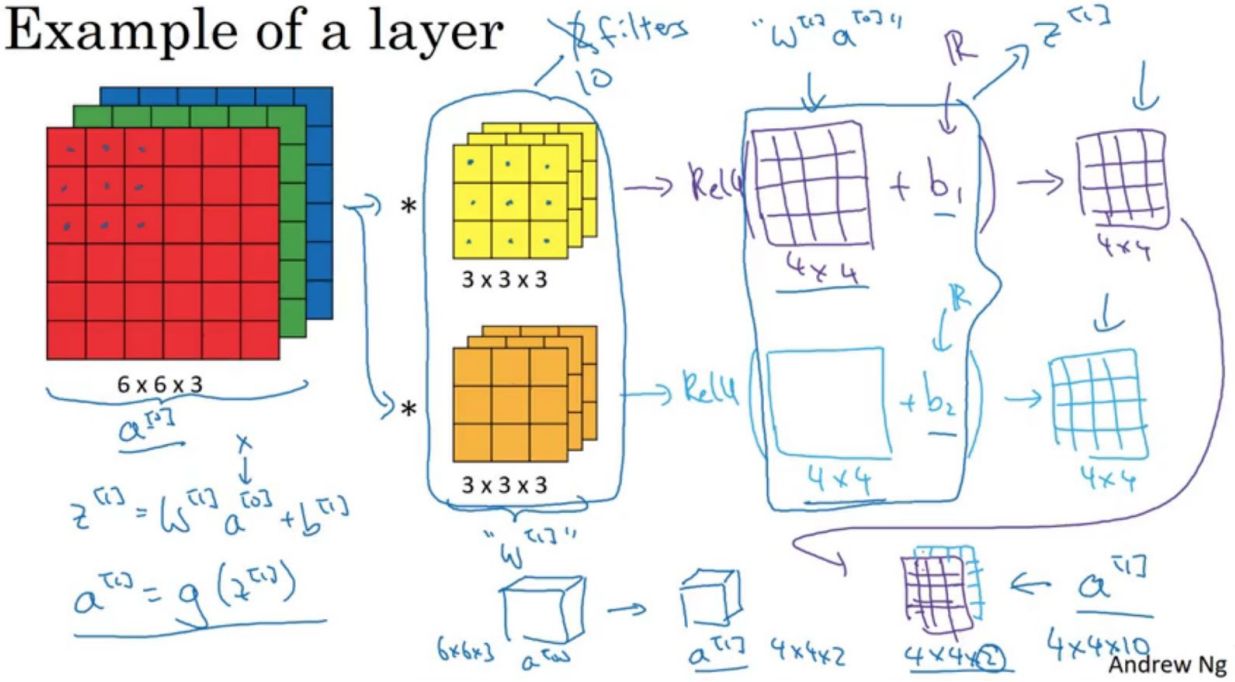
CONVOLUTIONS OVER A VOLUME

Convolutions on RGB image



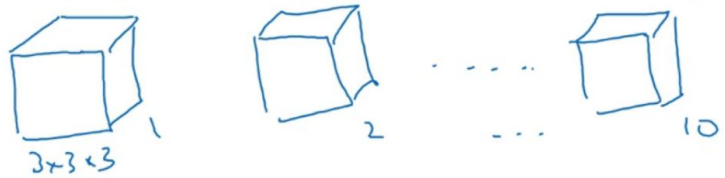
Channels in image and channels in filter is exactly the same! To detect only red edges you can make 1,1,1,0,0,0,-1,-1,-1 and rest all 0s and if you dont care which color edge it is then all 3 filter dimensions will have this same value.





Number of parameters in one layer

If you have 10 filters that are 3 x 3 x 3 in one layer of a neural network, how many parameters does that layer have?



Andrew Ng

Even if image is 10000*10000 parameters are still only 280! NO OVERFITTING DUE TO THIS IN CNNs!

Summary of notation

If layer l is a convolution layer:

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

$p^{[l]}$ = padding

$s^{[l]}$ = stride

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

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

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

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

bias: $n_c^{[l]} - (1,1,1,n_c^{[l]})$ ← #filters in layer l.

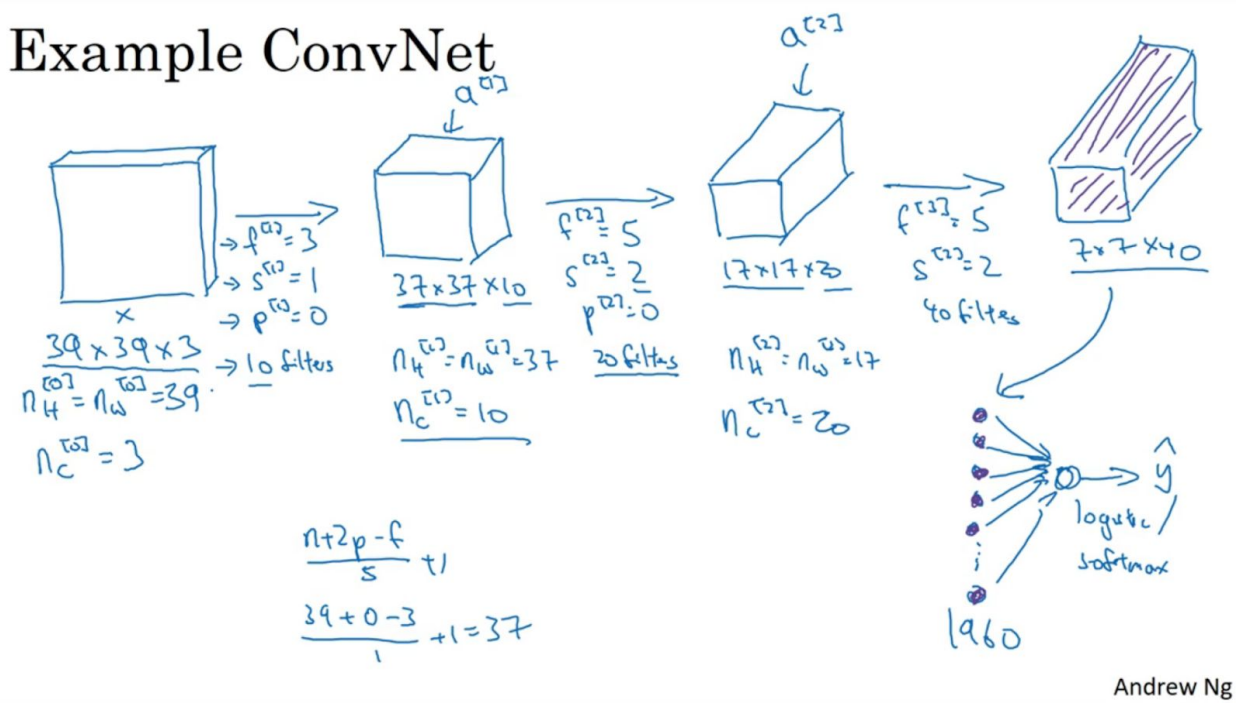
Input: $n_H^{[l-1]} \times n_W^{[l-1]} \times n_c^{[l-1]}$ ←

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

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

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

Andrew Ng

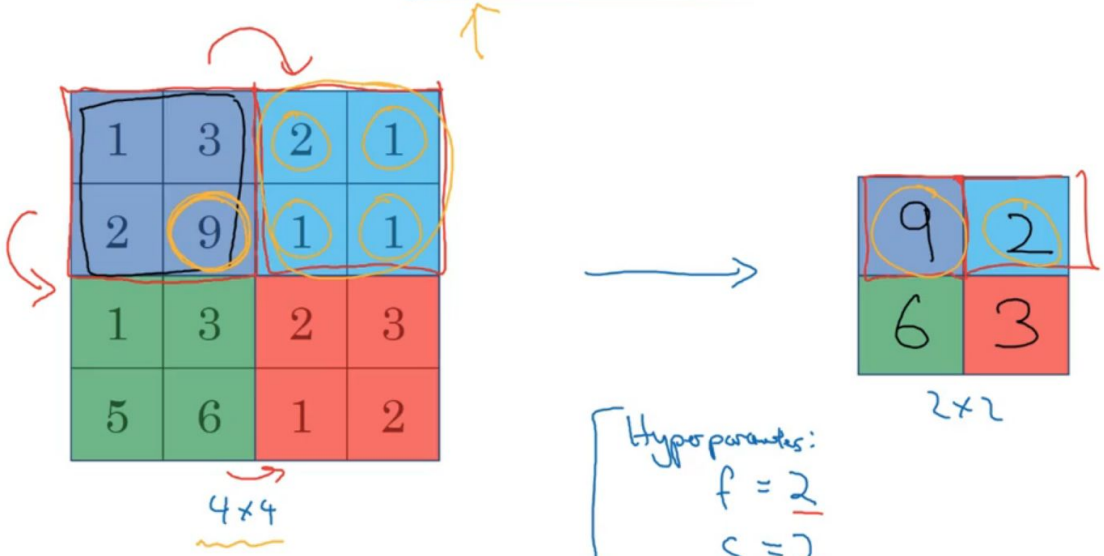


Andrew Ng

Types of layer in a convolutional network:

- Convolution (CONV) ←
- Pooling (POOL) ←
- Fully connected (FC) ←

Pooling layer: Max pooling



Summary of pooling

Hyperparameters:

f : filter size $f=2, s=2$ $f=3, s=2$

s : stride

Max or average pooling

~~p : padding~~

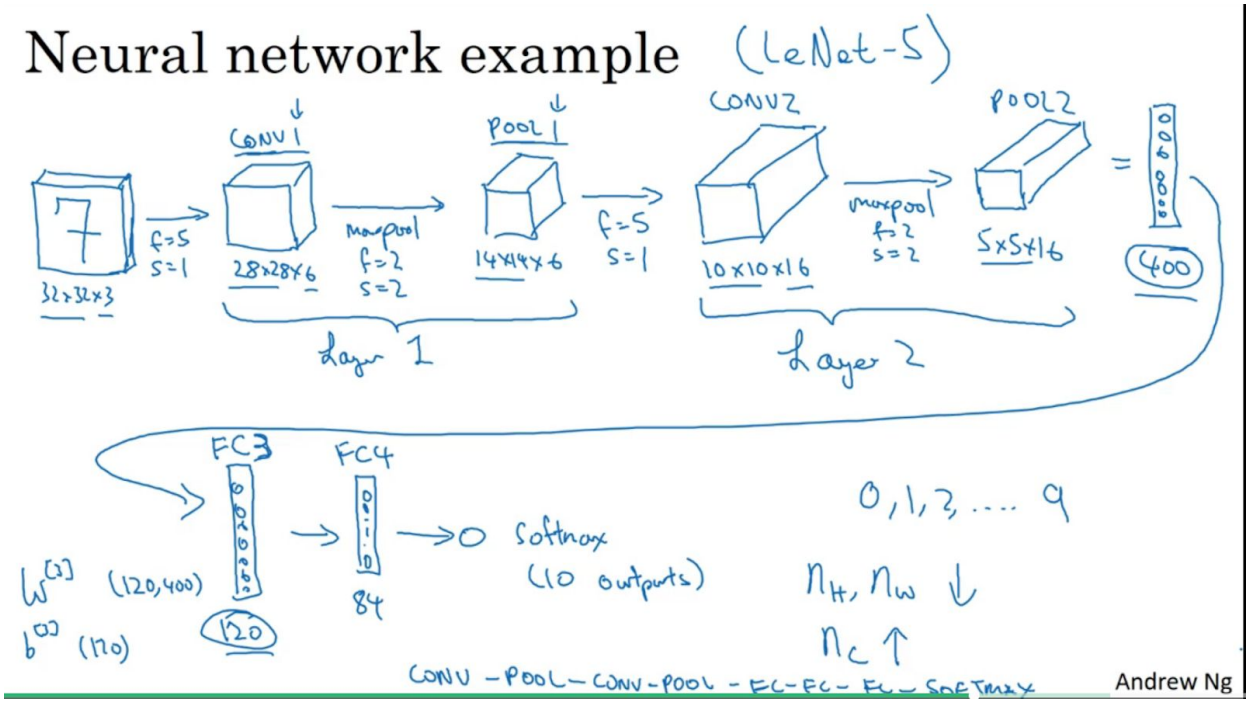
No parameters to learn!

$n_H \times n_w \times n_c$

$\downarrow$

$\left\lfloor \frac{n_H - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n_w - f}{s} + 1 \right\rfloor$

$\times n_c$



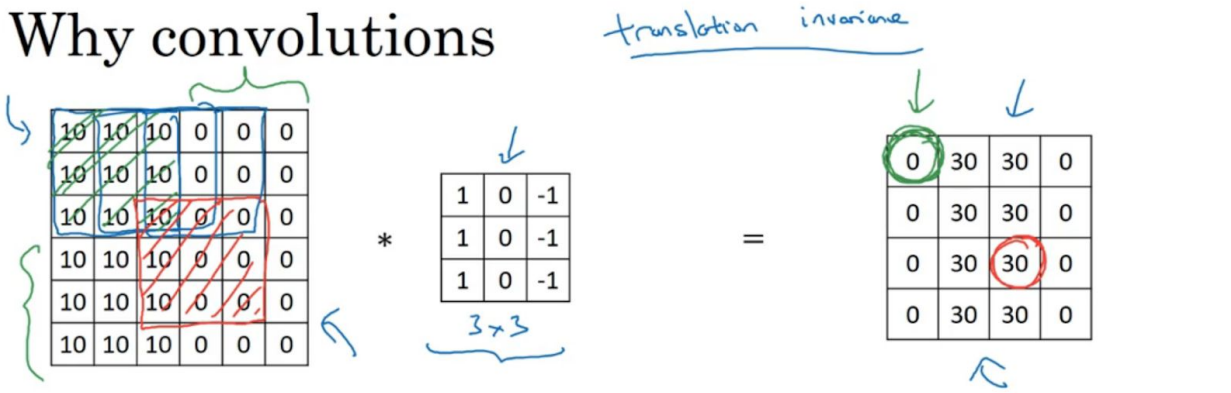
CONV→ POOL → CONV → POOL → FC → FC → FC → SOFTMAX

Neural network example

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072 $a^{[0]}$	0
CONV1 (f=5, s=1)	(28,28,8)	6,272	208
POOL1	(14,14,8)	1,568	0
CONV2 (f=5, s=1)	(10,10,16)	1,600	416
POOL2	(5,5,16)	400	0
FC3	(120,1)	120	48,001
FC4	(84,1)	84	10,081
Softmax	(10,1)	10	841

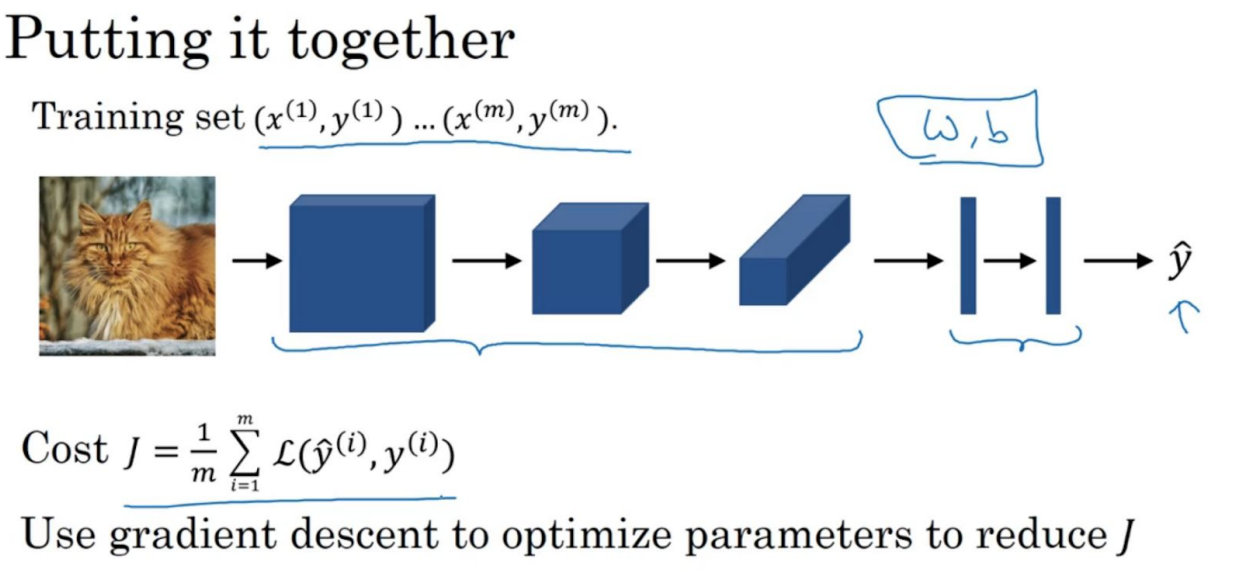
Activation size should gradually decrease -- not too sharply! Slowly reduce it. Most parameters are in FC!

WHY CONVOLUTIONS? Sparsity, parameter sharing and translation invariance. Cat is a cat even if shifted by a few pixels!



Parameter sharing: A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.

→ **Sparsity of connections:** In each layer, each output value depends only on a small number of inputs.



Reduce cost function with the weights and biases and build a CNN!

WEEK 2 (CASE STUDY OF NETWORKS)

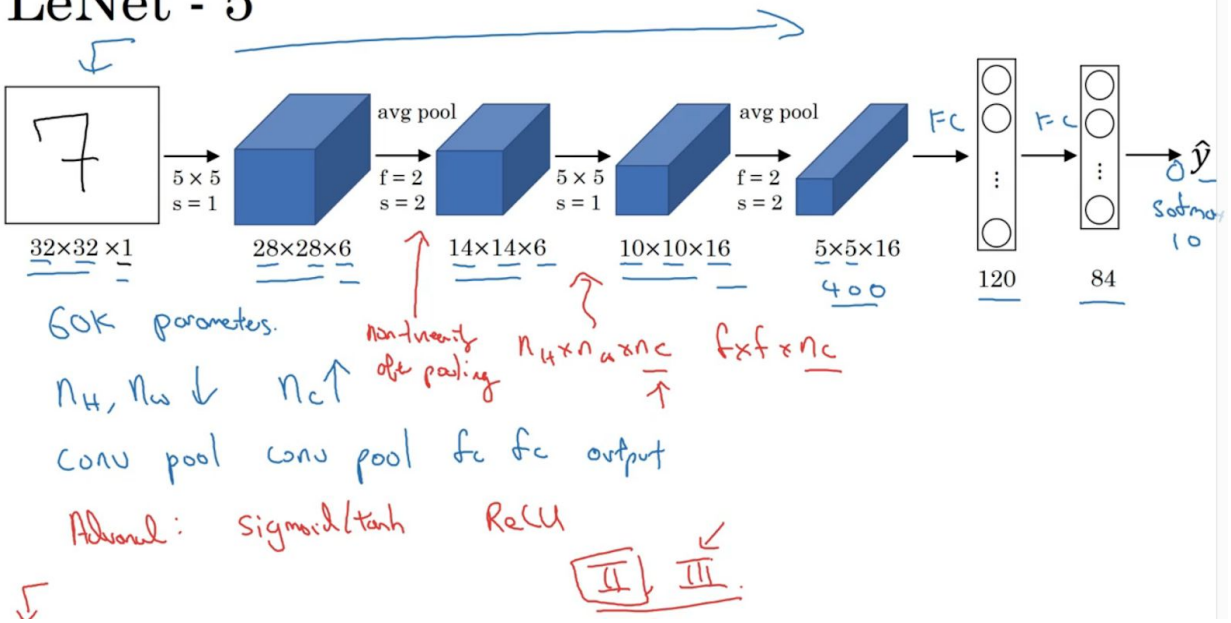
Classic networks:

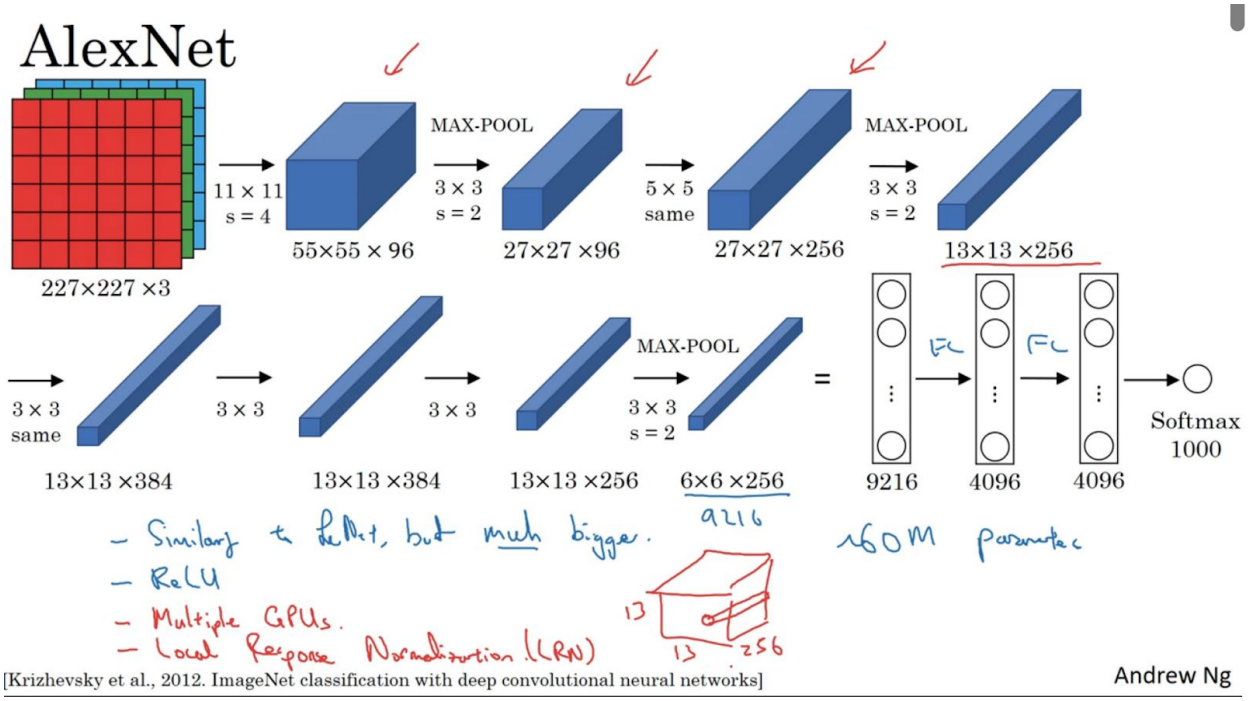
- LeNet-5
- AlexNet
- VGG

ResNet (152)

Inception

LeNet - 5

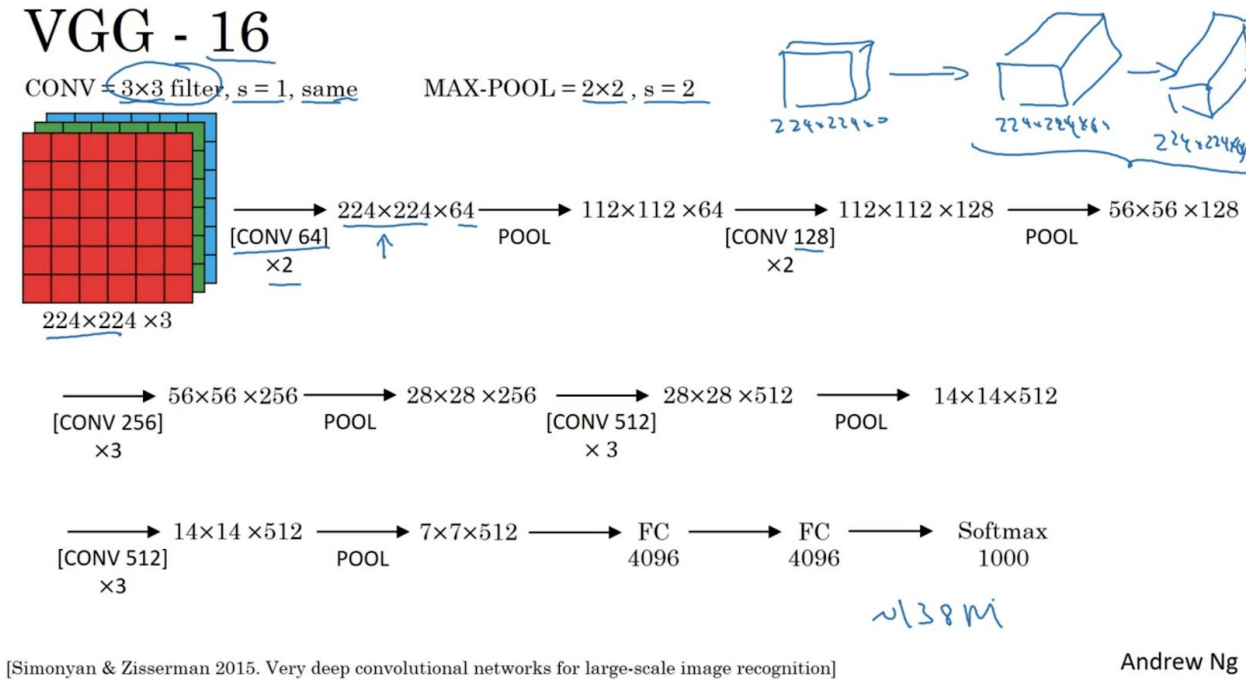




[Krizhevsky et al., 2012. ImageNet classification with deep convolutional neural networks]

Andrew Ng

LRN is not used today as much. Very influential paper in general though!
EASIER PAPER TO READ!

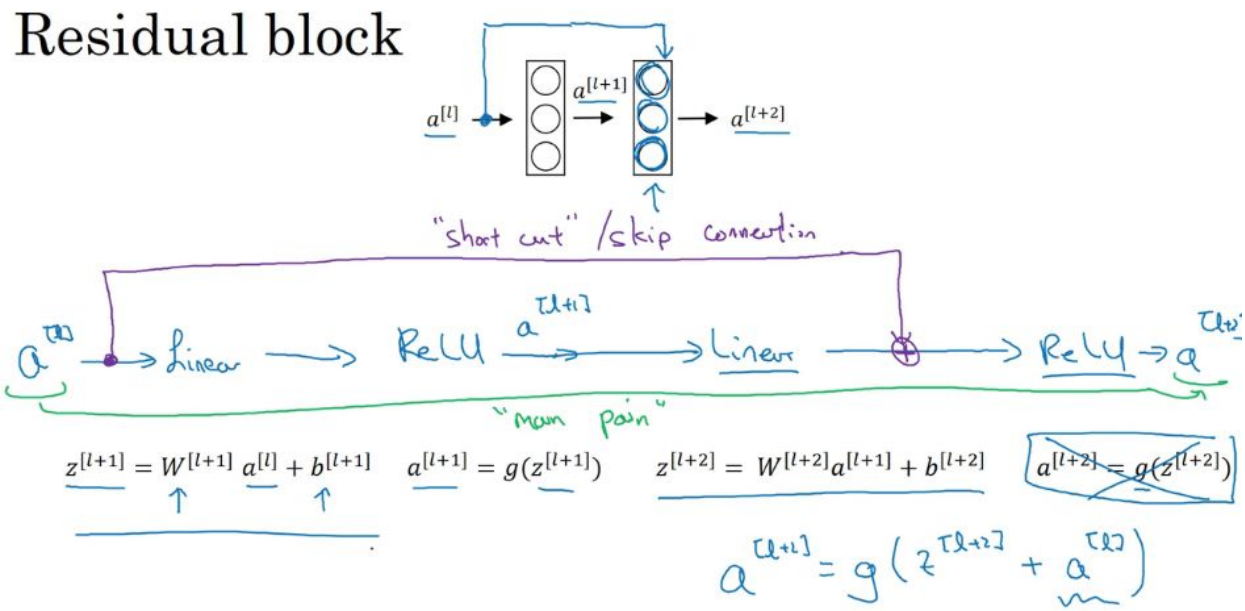


[Simonyan & Zisserman 2015. Very deep convolutional networks for large-scale image recognition]

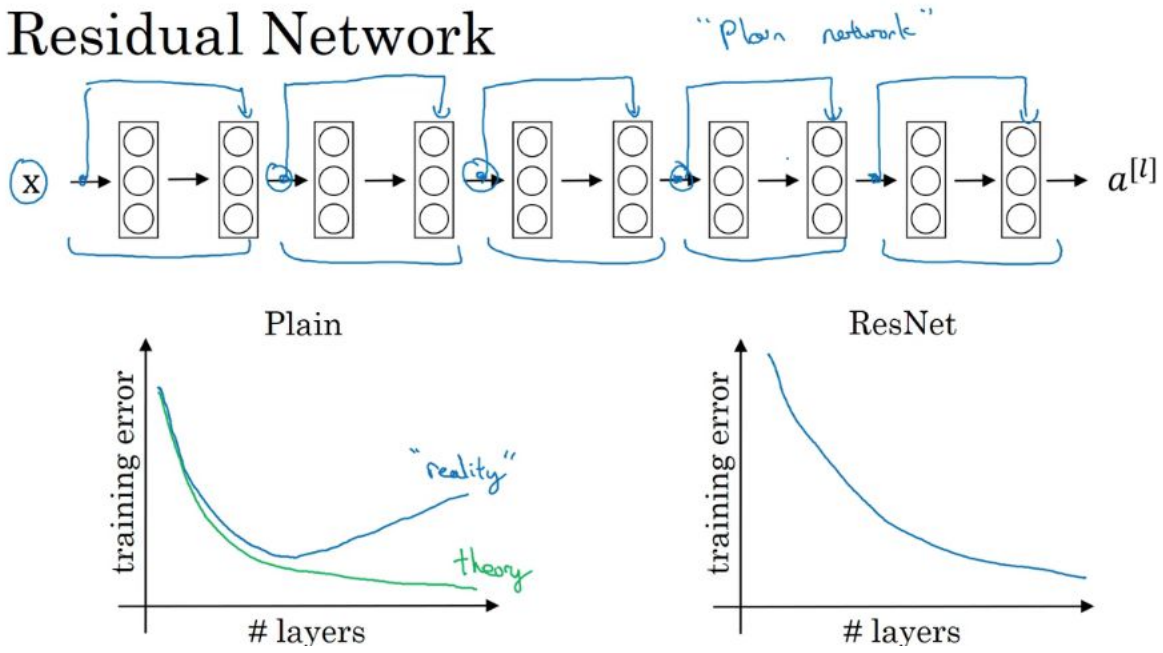
Andrew Ng

Pattern of height width goes down but channels go up → systematic and attractive!
Start from reading ALexnet paper to VGG net paper

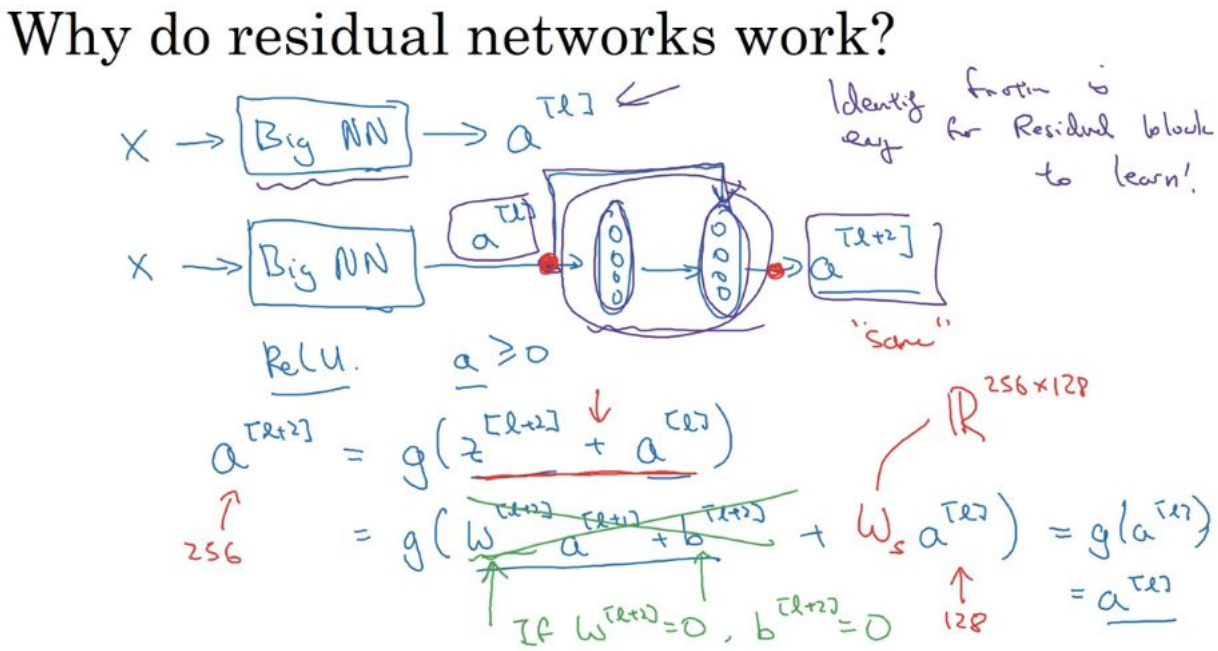
RESNETS

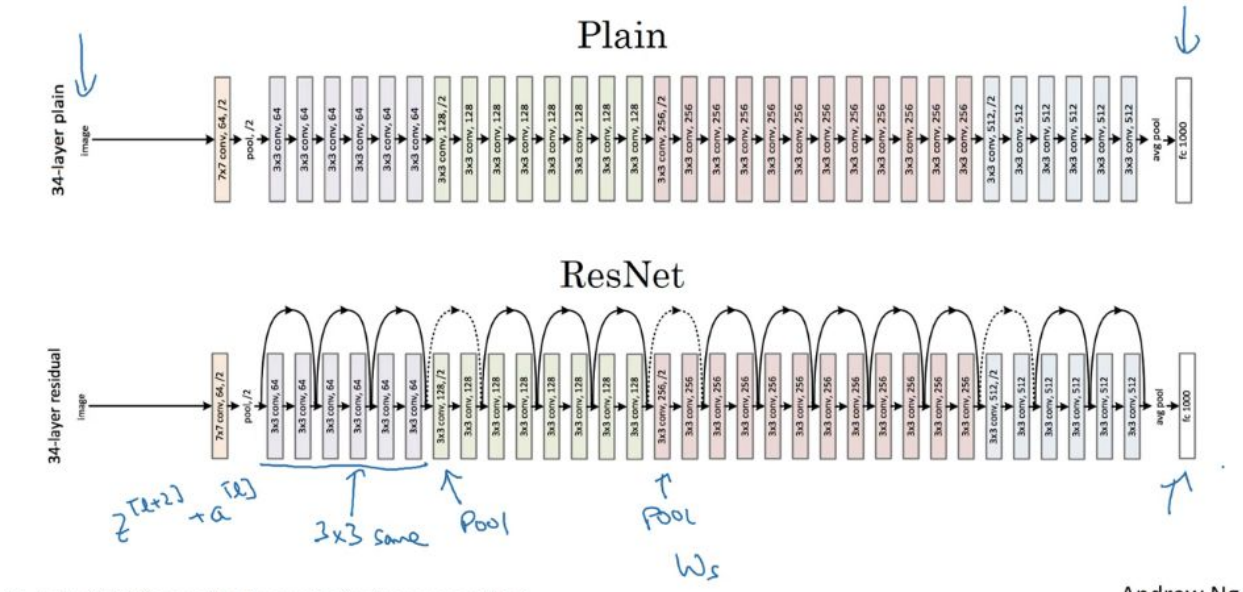


Skipping over and feed into much deeper layers - can train even deeper ResNet!



Residual networks work because -- its easy to learn the identity function. Basically if there is no additional complexity to be learnt, it can just feed a[l] to a[l+2] but if there is an additional complexity - BONUS!
Thats why generally deeper and deeper layers are only good to certain extent, after which your error increases. But with resnet - you have a backup option of identity being moved forward so you can choose more complexity or choose no complexity.
Assume same dimensions of Z and A -- therefore SAME convolutions, but if not you can just do padding using Ws.



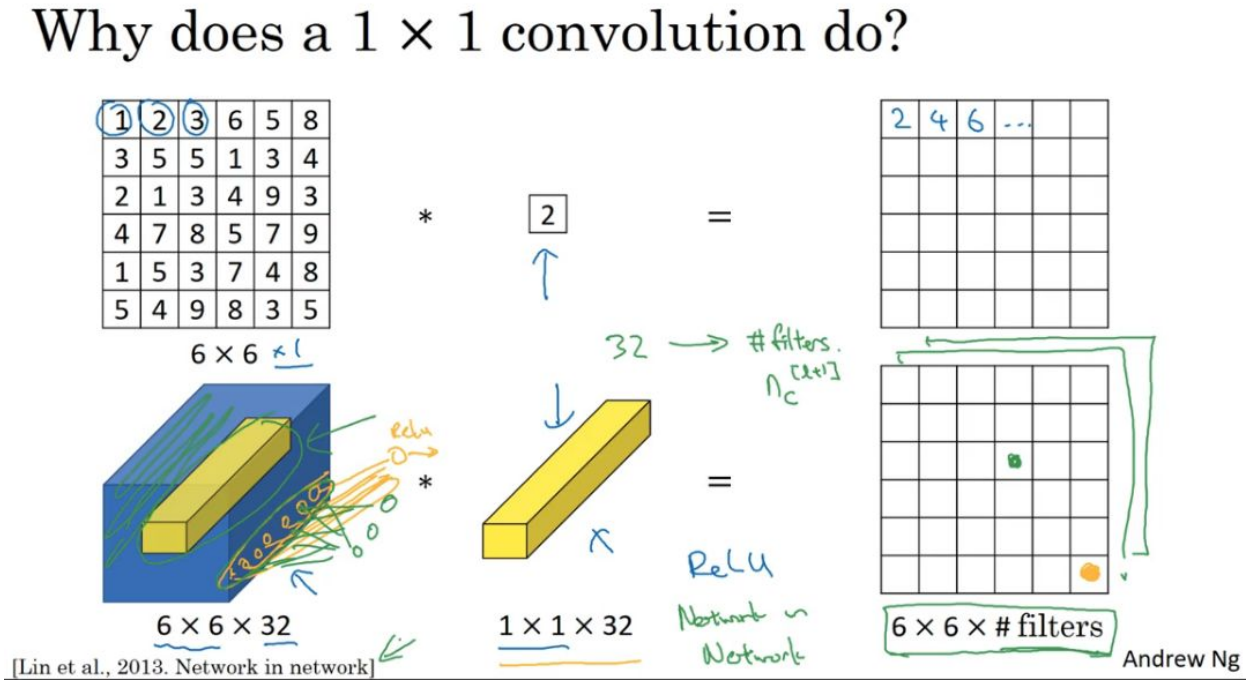


He et al., 2015. Deep residual networks for image recognition]

Andrew Ng

NETWORK IN NETWORK

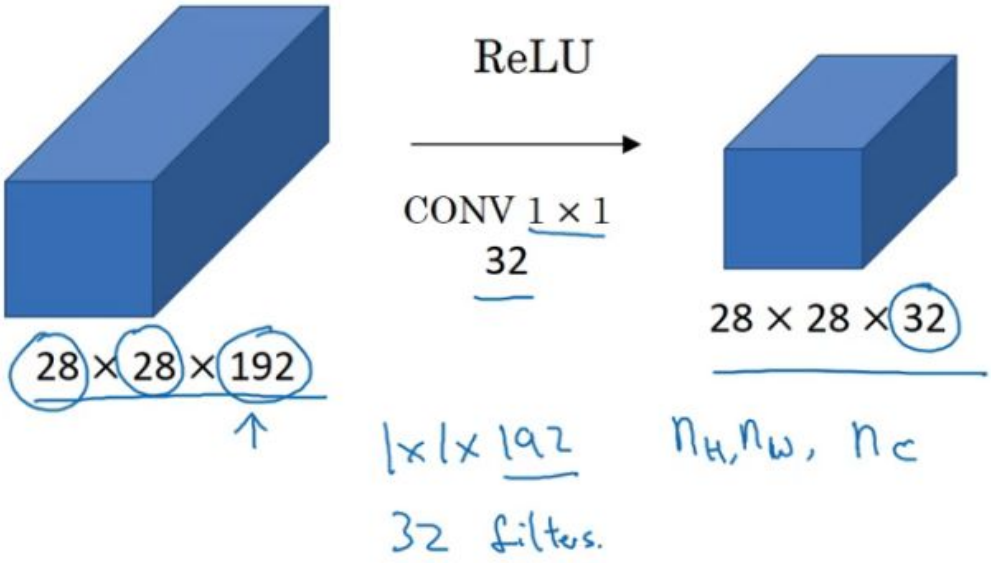
1x1 convolution : If number of channels is too much , you can use 1x1 conv. For shrinking height and width - use max pooling!



[Lin et al., 2013. Network in network]

Andrew Ng

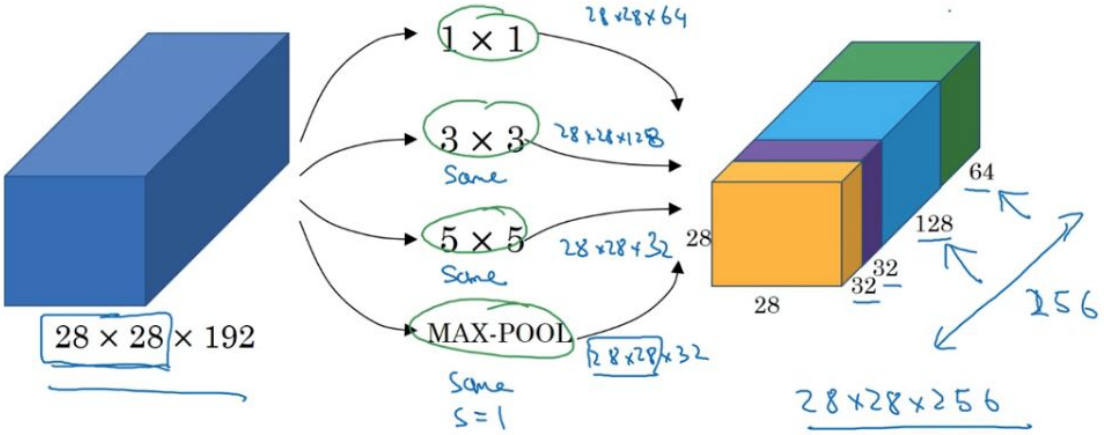
Using 1×1 convolutions



INCEPTION NETWORK

DO ALL YOU MIGHT WANT! If you are not sure - just stack them all!
For last max pool ,there is 1 x 1 conv at the end to reduce the channels.

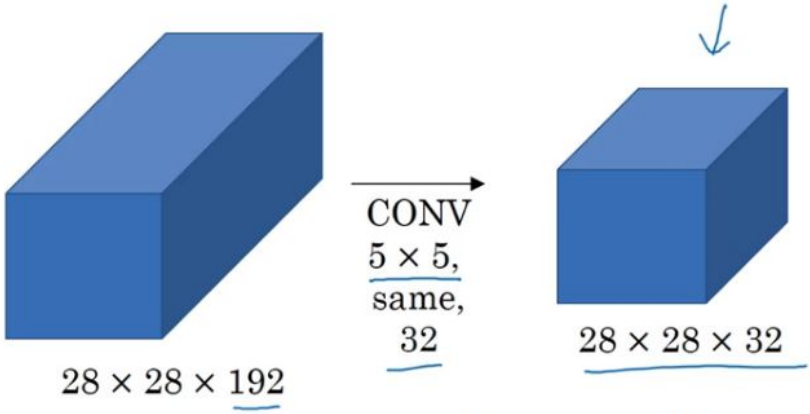
Motivation for inception network



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

Andrew Ng

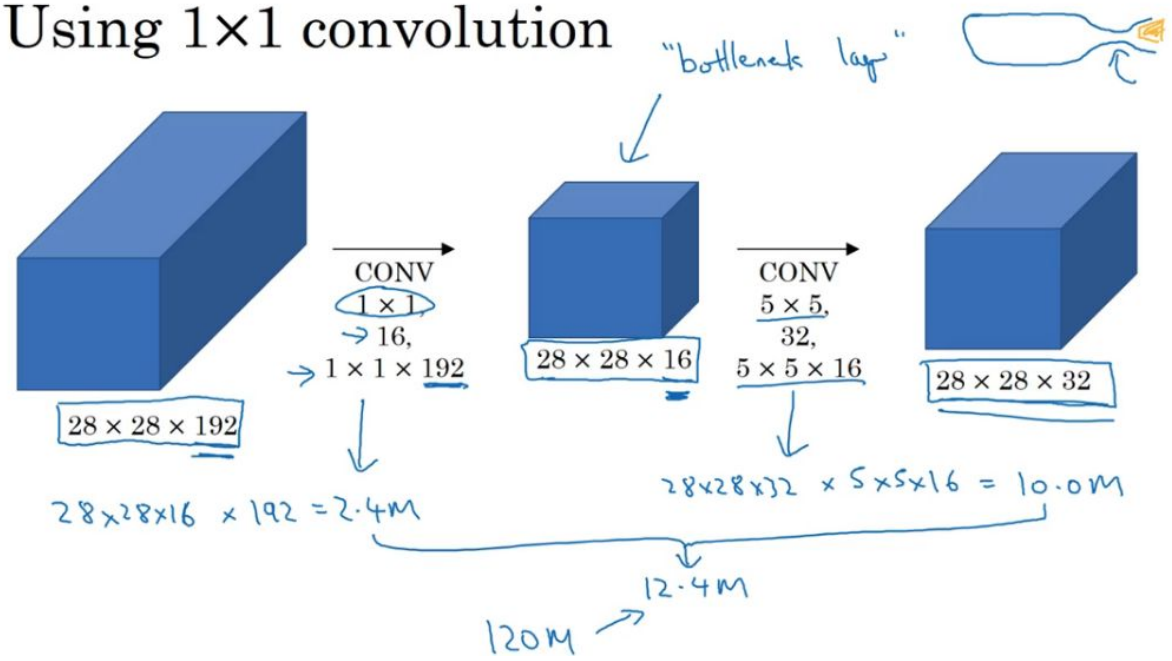
The problem of computational cost



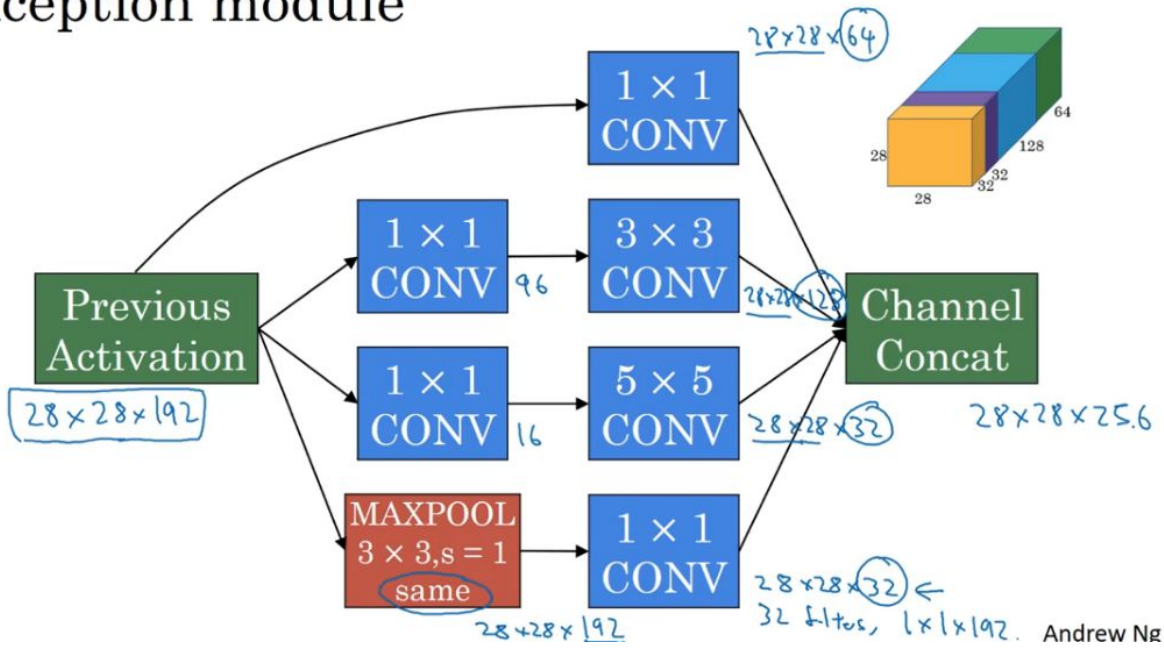
32 filters. filter one 5×5×192.

28×28×32 × 5×5×192 = 120M..

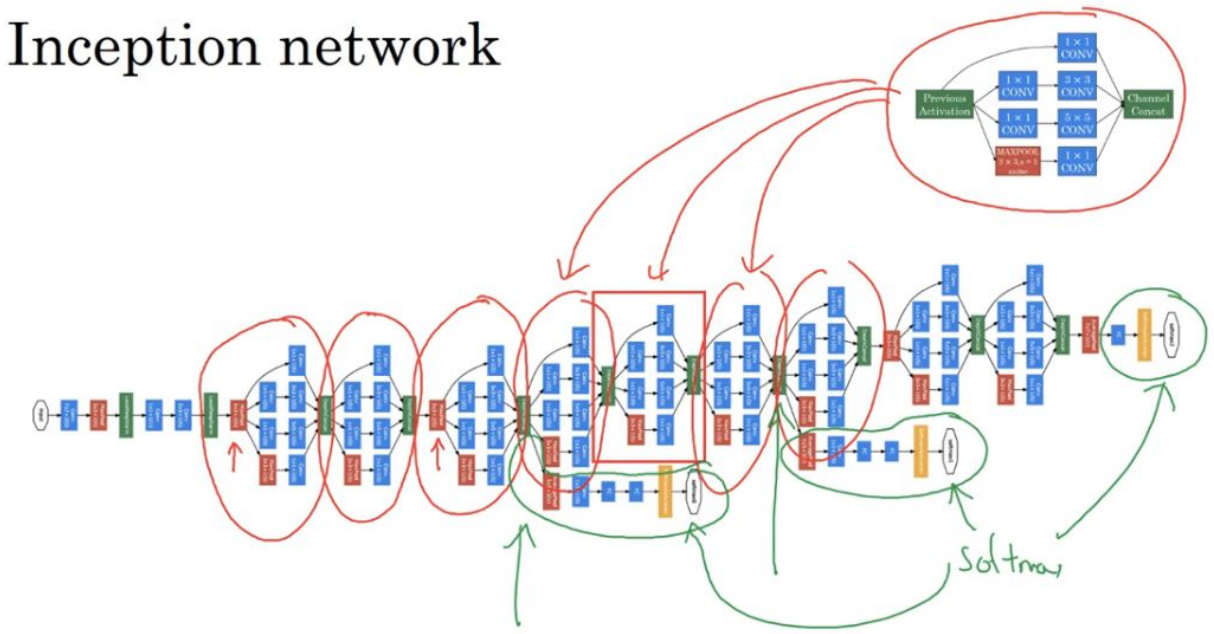
Using 1×1 convolution



Inception module



Andrew Ng



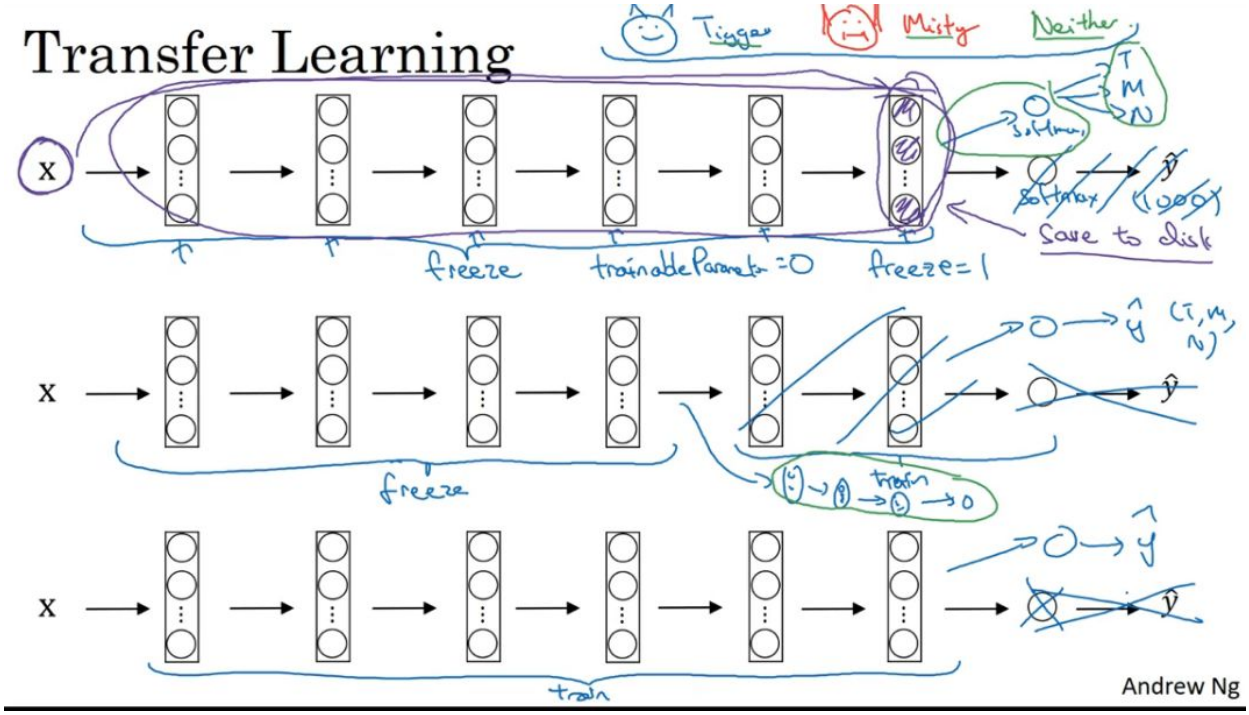
there are a very few softmax layers even in between - to have regularization. To make sure even the hidden data is not bad for predicting the output.

There are ideas of inception network combined with Res nets!

PRACTICAL ADVICE

TRANSFER LEARNING:

Trainable parameter = 0 -- tensorflow and keras might have it to freeze earlier layers and only add your own softmax function in the end.



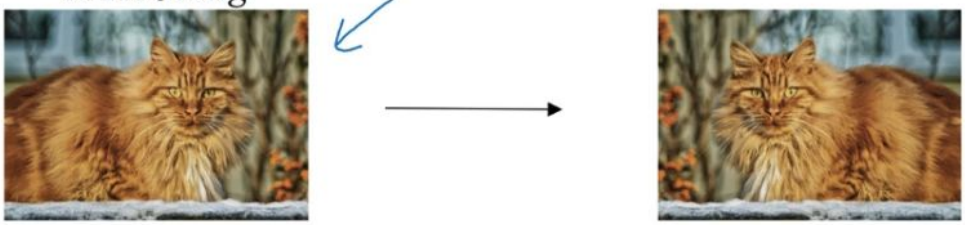
More data you have → more layers you can train! If you have insanely high data, you can use weights and architecture as initialization to your problem and train the whole network!
People have spent weeks and months training some models, good to re-use them!

DATA AUGMENTATION:

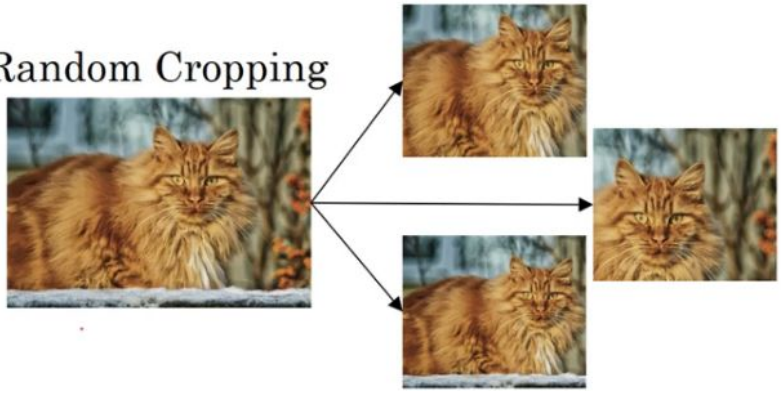
you can train! If you have insanely high data, you can use weights and architecture

Common augmentation method

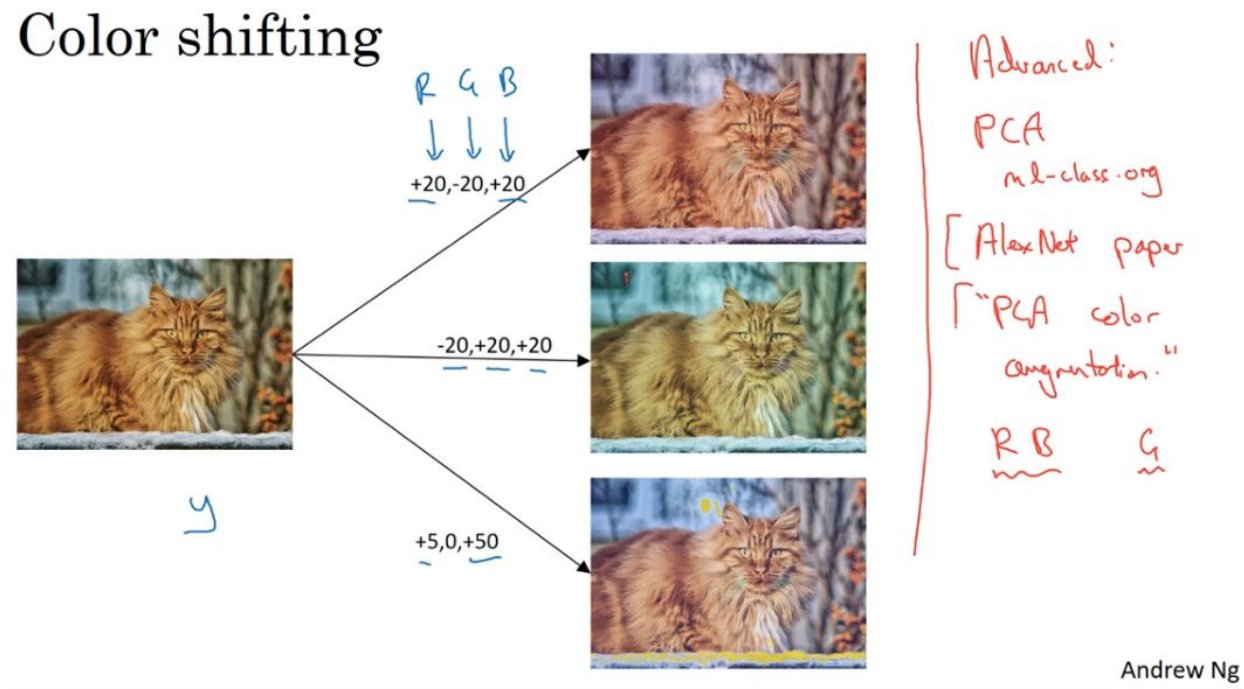
Mirroring



Random Cropping

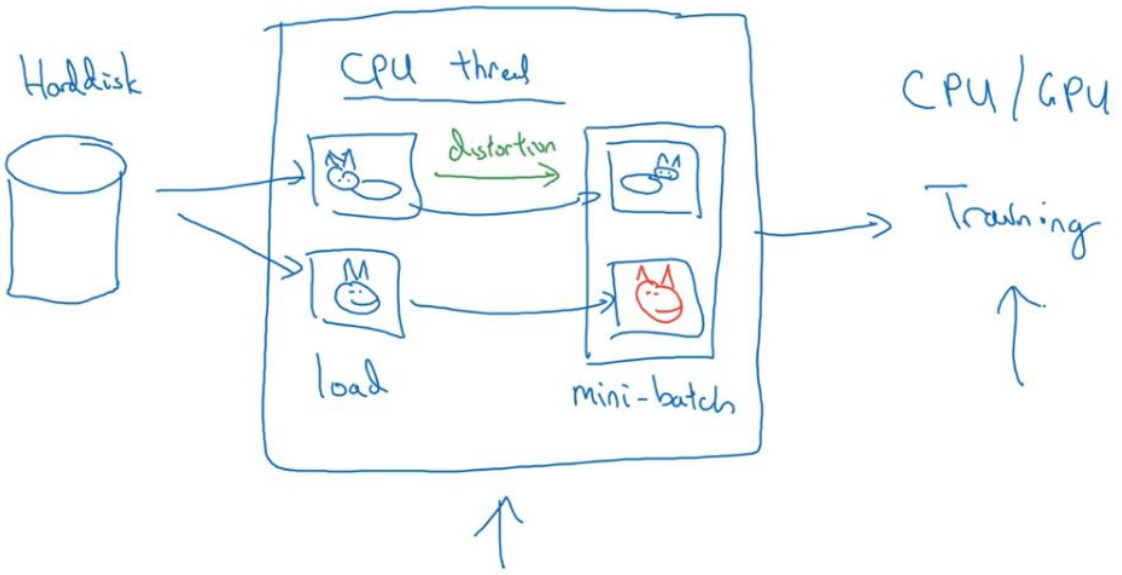


Rotation, Shearing, Local Warping etc!



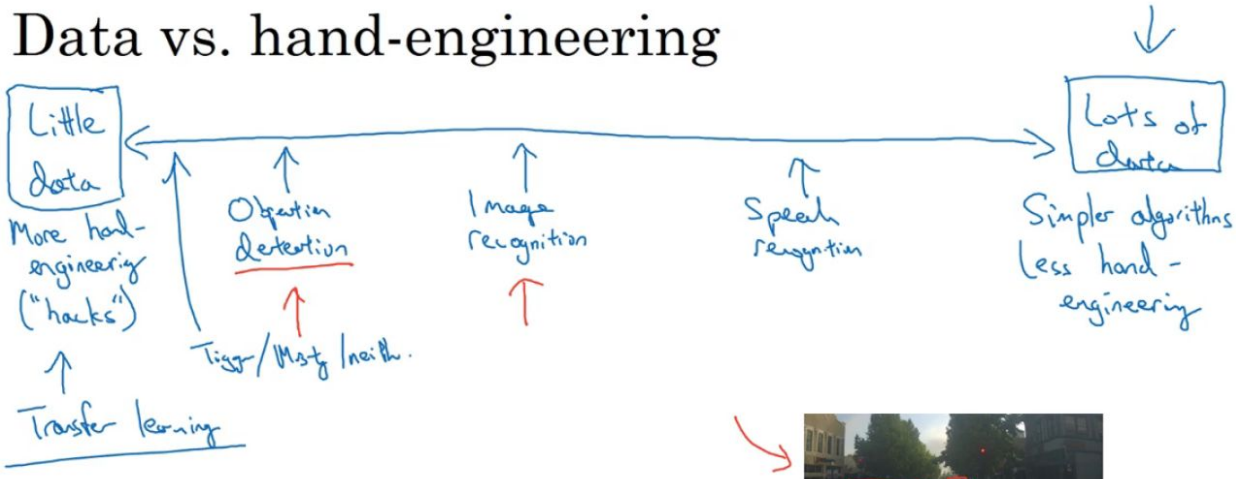
You can sample the increase or decrease of R G B pixels based on PCA _ so you if you have lower G tint, your new images will continue to have lower G tint. You will not have very very high G or something.

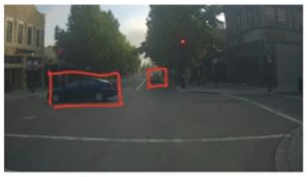
Implementing distortions during training



STATE OF COMPUTER VISION!

Data vs. hand-engineering





Two sources of knowledge

- Labeled data (x, y)
- Hand engineered features/network architecture/other components

Andrew Ng

Tips for doing well on benchmarks/winning competitions

Ensembling

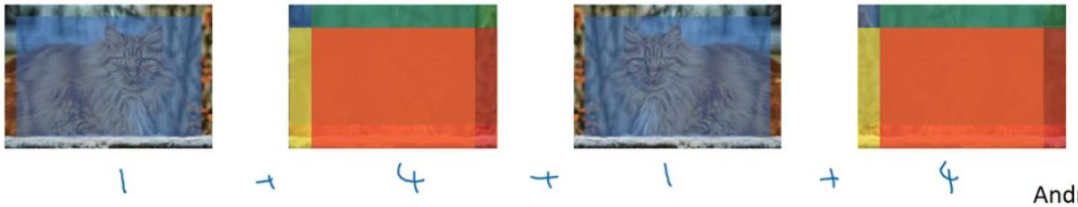
3-15 networks → ŷ

- Train several networks independently and average their outputs

Multi-crop at test time

- Run classifier on multiple versions of test images and average results

10-crop



Andrew Ng

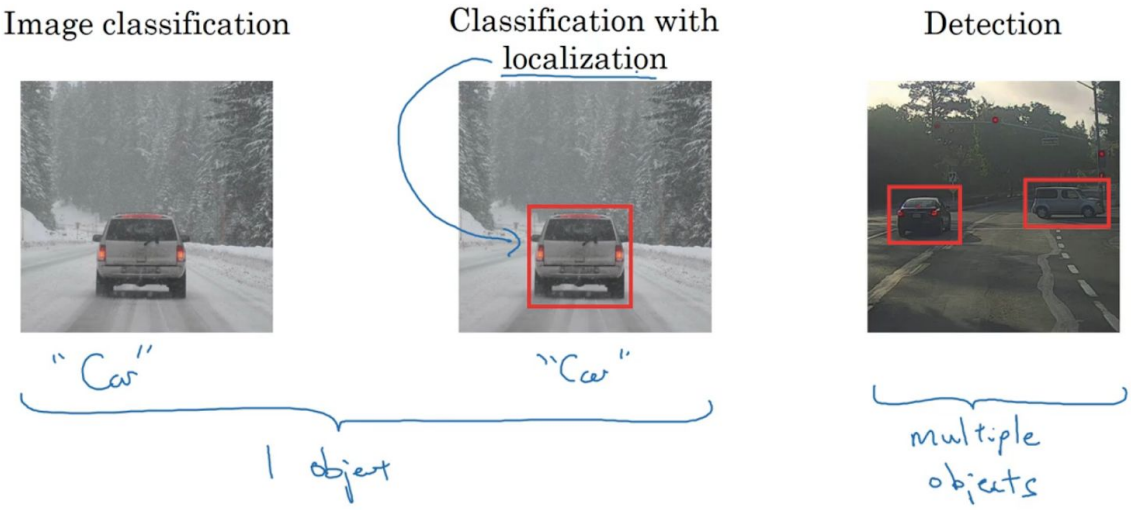
Cant use them at production for customers as its more expensive! But great for winning competitions or benchmarks!

Use open source code

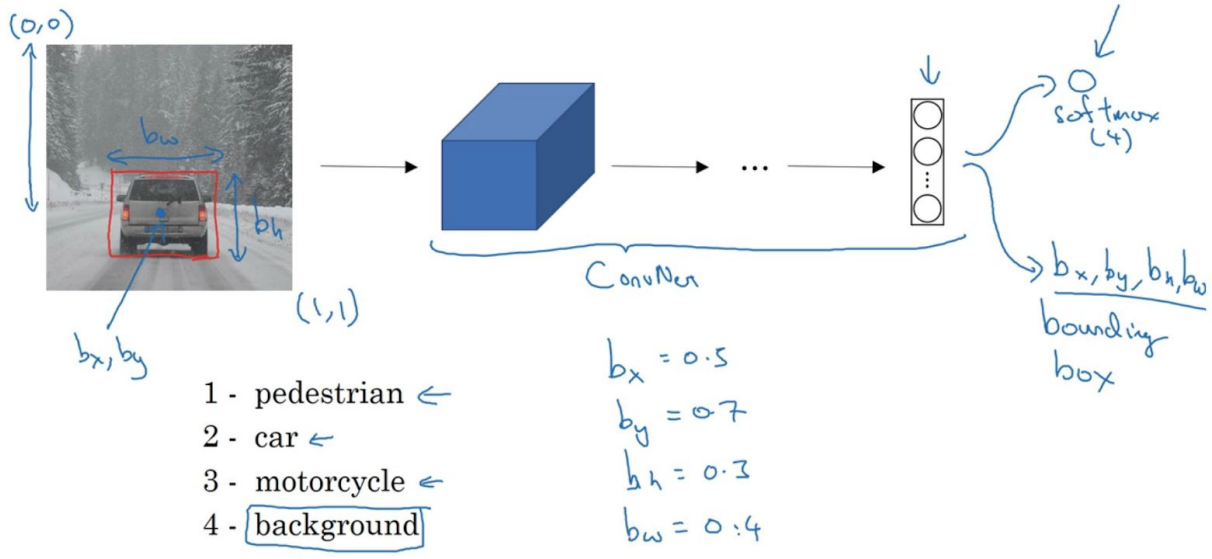
- Use architectures of networks published in the literature
- Use open source implementations if possible
- Use pretrained models and fine-tune on your dataset

OBJECT LOCALIZATION

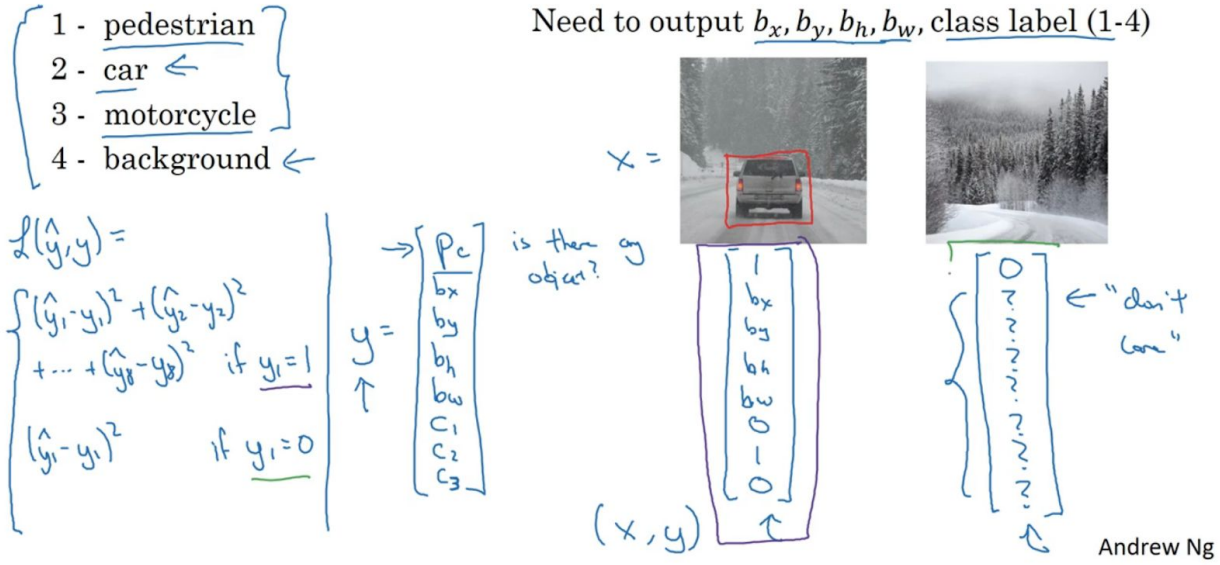
What are localization and detection?



Classification with localization

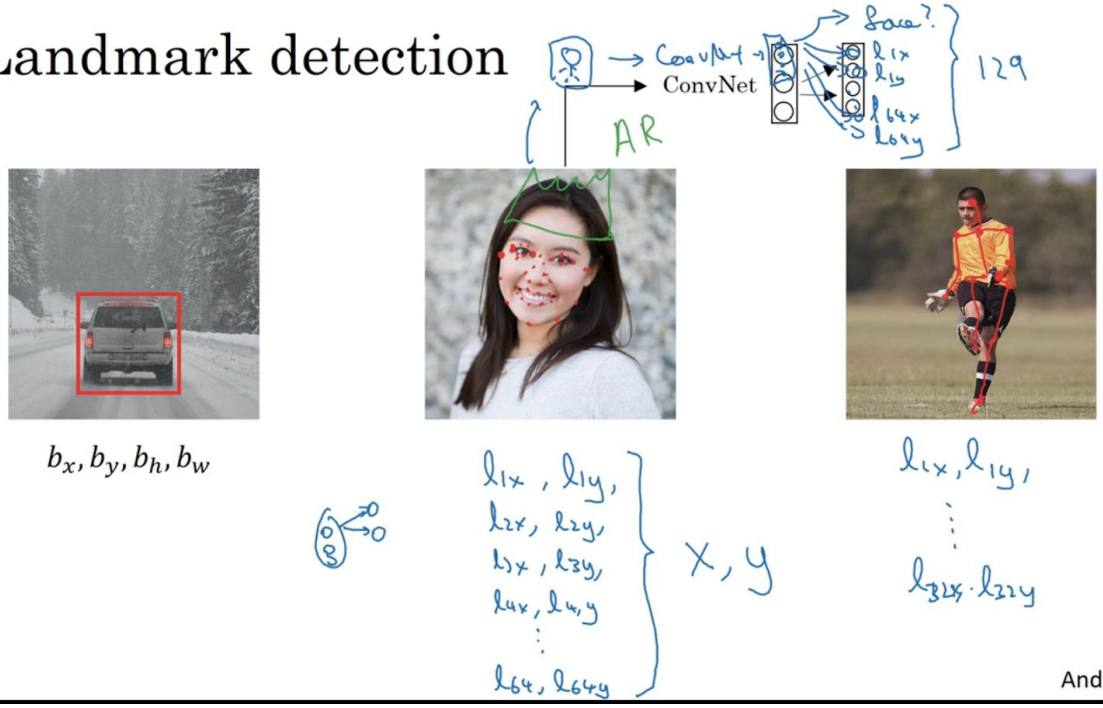


Defining the target label y

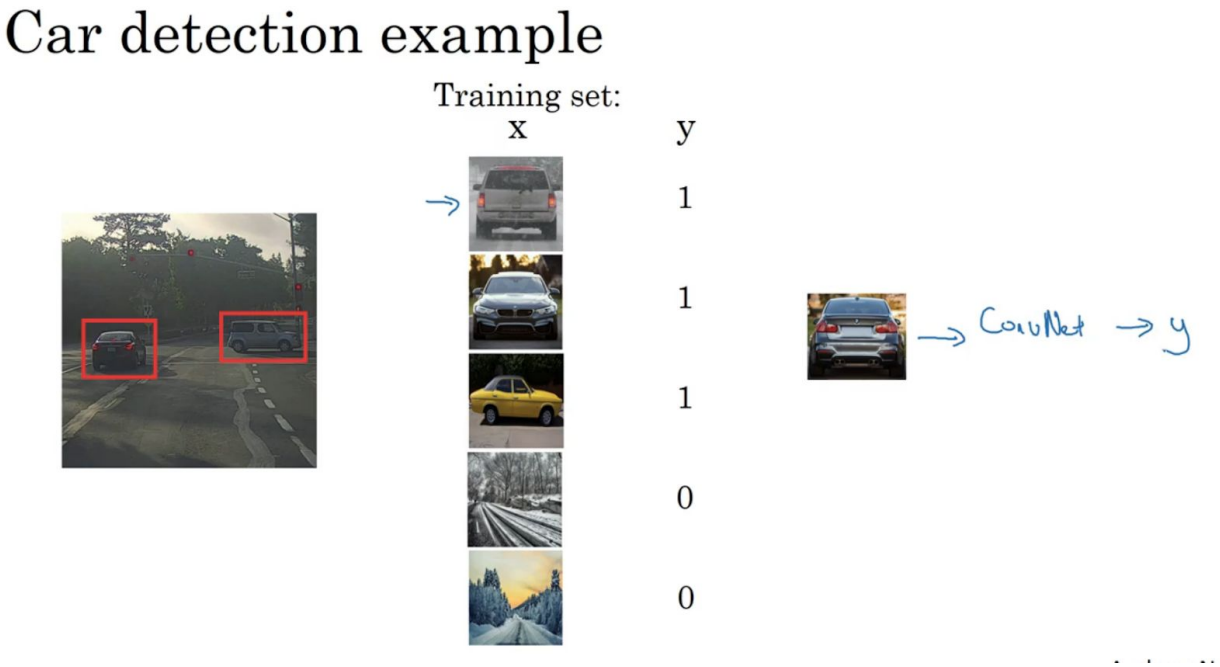


If object exists, then all square or log-loss errors for all 8 elements for bounding boxes as well as if object exists or not etc. If object is not there, then rest of the elements are dont care ,therefore loss is only with Pc (Pc → Does object exists in the image?)

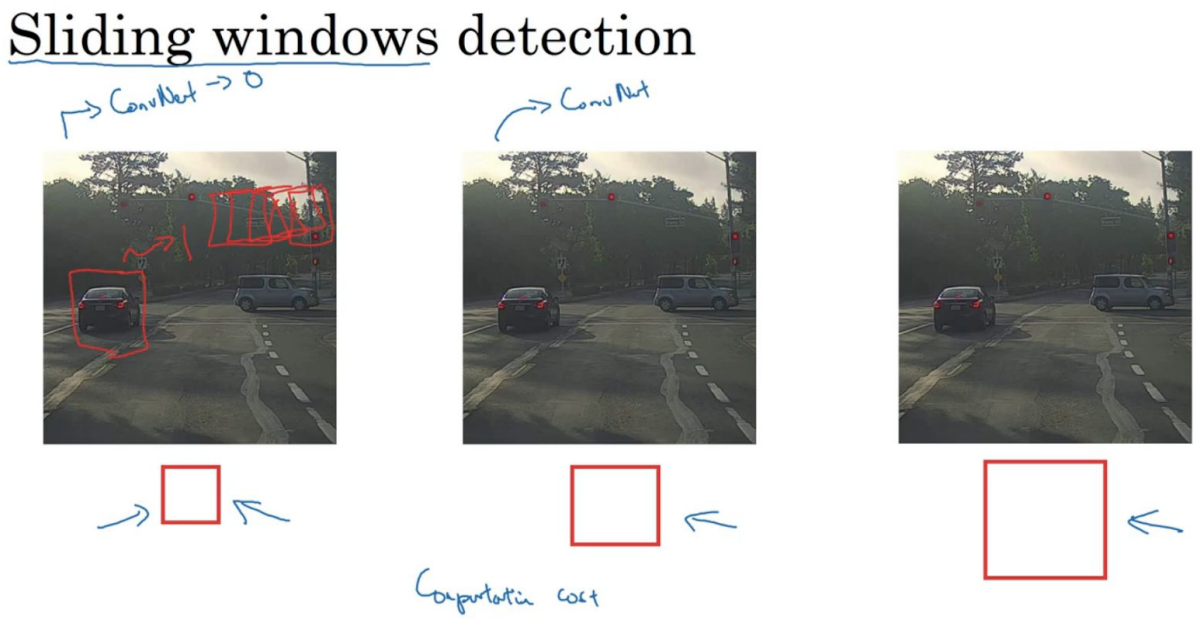
Landmark detection



SnapChat AR filters for creating a crown etc needs landmarks on face to identify where various face things appear.

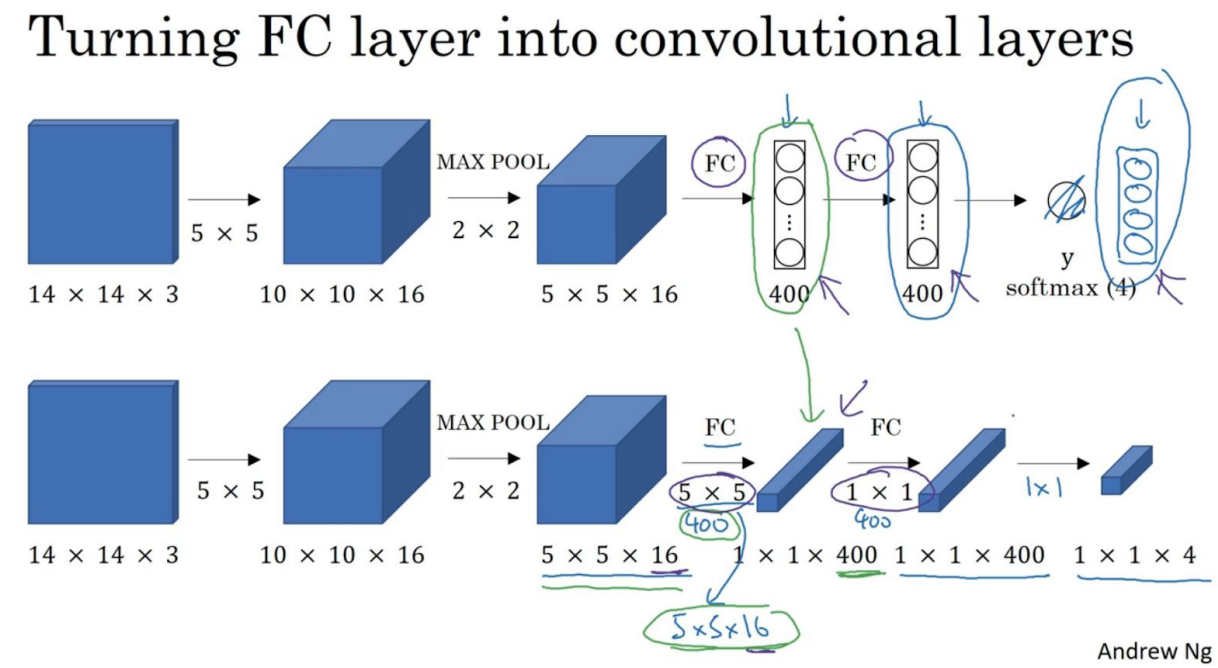


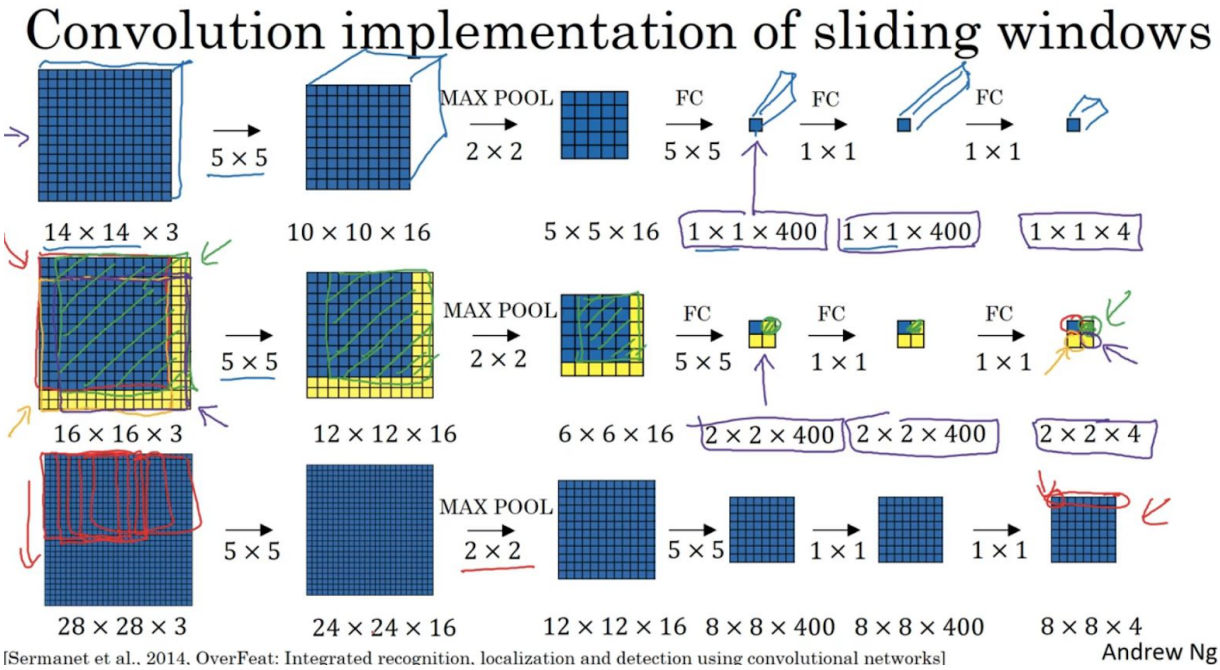
Just train a convnet first on a closely cropped image of car. Then use sliding window to identify all possible cars in an image.



COMPUTATION COST is very high. Previously linear classifiers was ok to implement so many times, but now neural networks are more expensive.

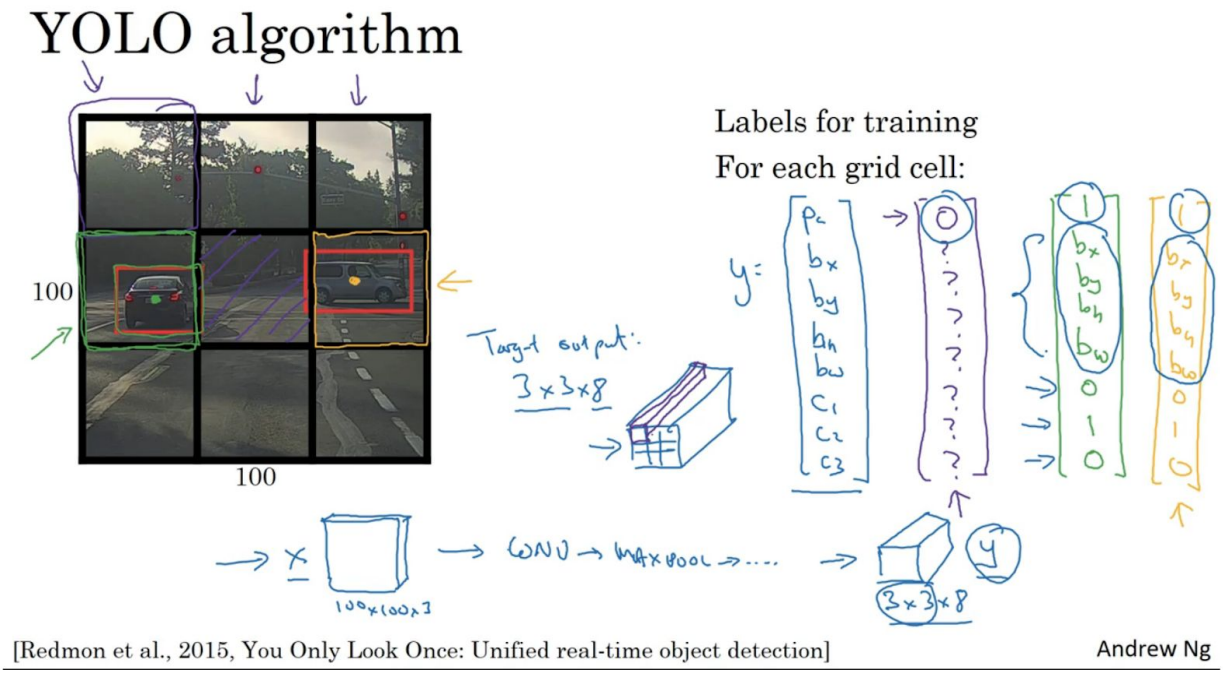
COnverting FC to Convolutional layers:





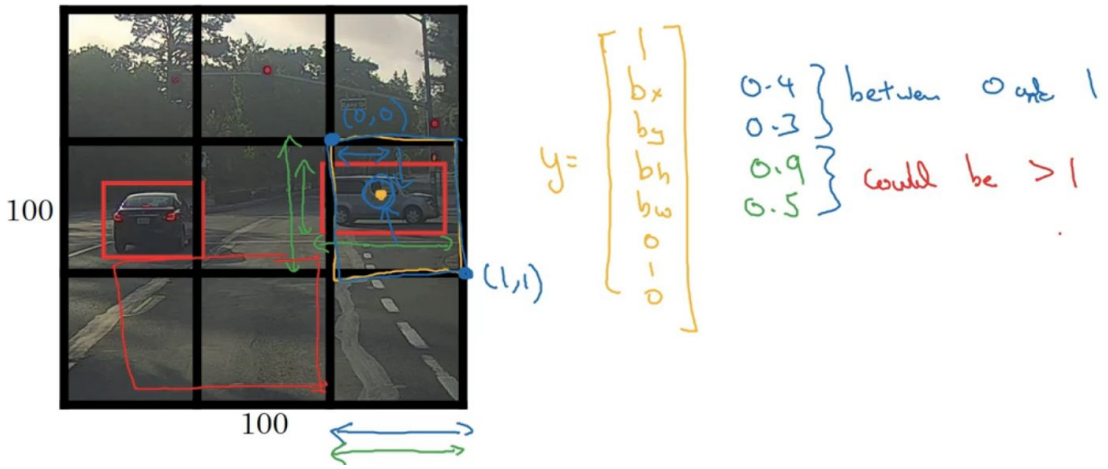
Instead of repeating computation several times, you can just run on the whole image, and now instead of seeing FC as flat layer but see it as CNN. Weakness is that this method cant catch bounding box very well.

Just use the last 4 boxes and softmax output and see which output for every box.



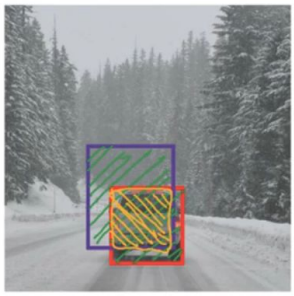
See where the mid point of the object lies in a grid. In real life the grid is 19x19. Its not running it 9 times, but its conv implementation where you re-use parameters.

Specify the bounding boxes



YOU ONLY LOOK ONCE or Yolo algo is really popular for object detection.

Evaluating object localization



Intersection over Union (IoU)

$$= \frac{\text{Size of } \square}{\text{Size of } \square}$$

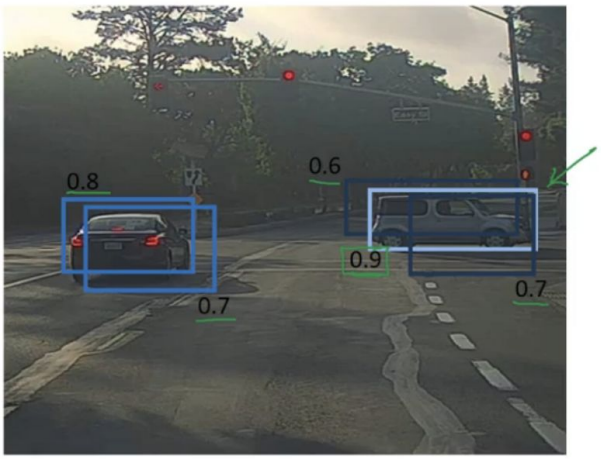
"Correct" if IoU $\geq$ 0.5 ←

0.6 ←

More generally, IoU is a measure of the overlap between two bounding boxes.

Andrew Ng

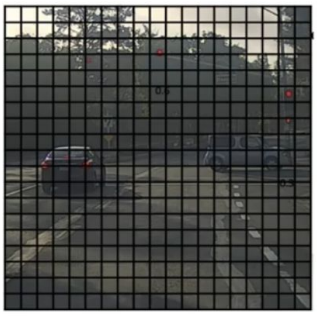
Non-max suppression example



p_c

Remove boxes with non max probabilities, its non max suppression. Use IOU to find closely overlapped boxes and remove the non max ones and keep only the max one.

Non-max suppression algorithm



19× 19

Each output prediction is: $\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \end{bmatrix}$

Discard all boxes with $p_c \leq 0.6$

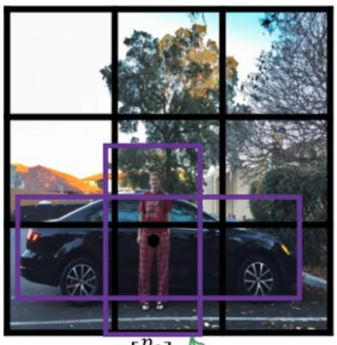
While there are any remaining boxes:

- Pick the box with the largest p_c Output that as a prediction.
- Discard any remaining box with IoU ≥ 0.5 with the box output in the previous step

Andrew Ng

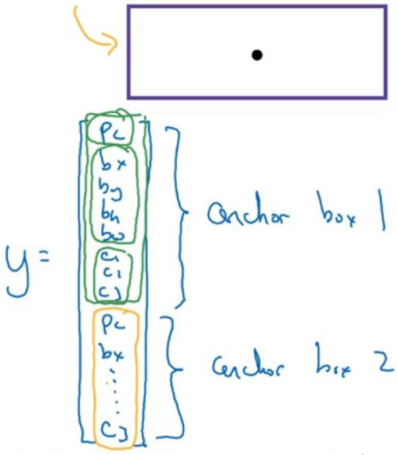
You need to do non max suppression for each classes, each for pedestrian, car etc.

Overlapping objects:



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

Anchor box 1: Anchor box 2:



[Redmon et al., 2015, You Only Look Once: Unified real-time object detection]

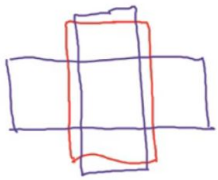
Andrew Ng

Anchor box algorithm

Previously:

Each object in training image is assigned to grid cell that contains that object's midpoint.

Output y: $3 \times 3 \times 8$



With two anchor boxes:

Each object in training image is assigned to grid cell that contains object's midpoint and anchor box for the grid cell with highest IoU.

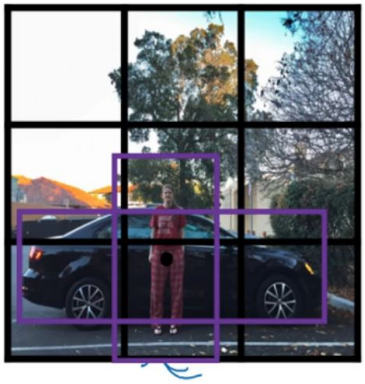
(grid cell, anchor box)

Output y: $3 \times 3 \times 16$ or $3 \times 3 \times 2 \times 8$

Andrew Ng

2 anchor boxes for each grid, find which anchor box is most IOU with the ground truth.

Anchor box example



Anchor box 1: Anchor box 2:

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

car only?

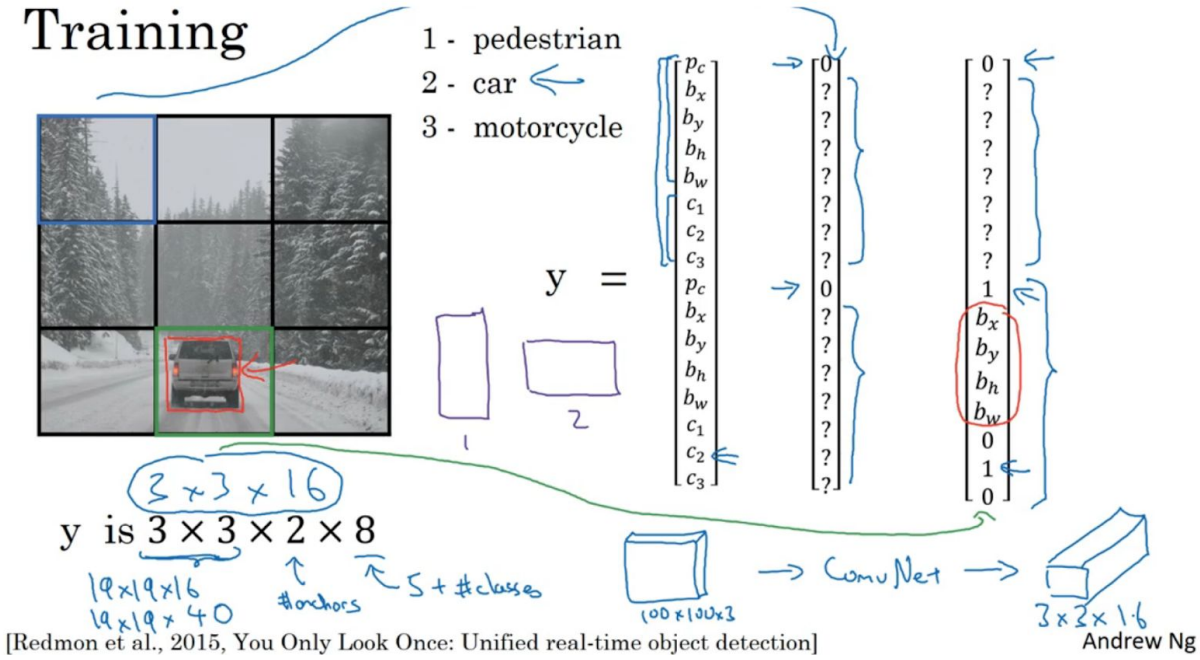
anchor box 1

anchor box 2

Andrew Ng

Cant handle more than 2 objects in one grid --- but usually 19x19 doesnt have overlapping objects in same grid.
People choose 5 or 10 anchor boxes.

YOLO ALGORITHM BELOW

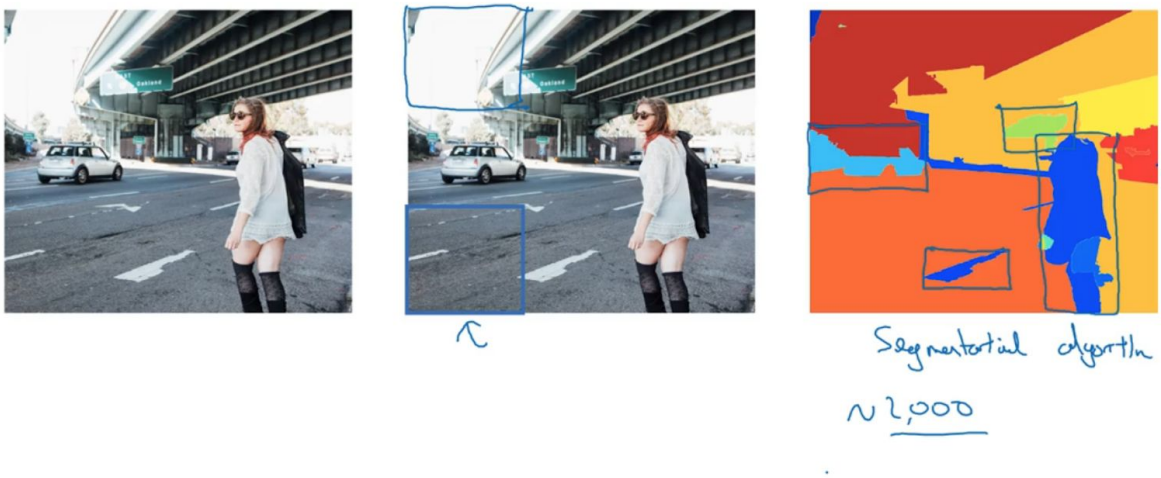


Outputting the non-max supressed outputs

- 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.

Another approach instead of YOLO: Find regions where to run sliding window CNN

Region proposal: R-CNN



[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

WEEK 4: FACE RECOGNITION

Face verification vs. face recognition

→ Verification

1:1 99%

- Input image, name/ID
- Output whether the input image is that of the claimed person

99.9

→ Recognition

1:K

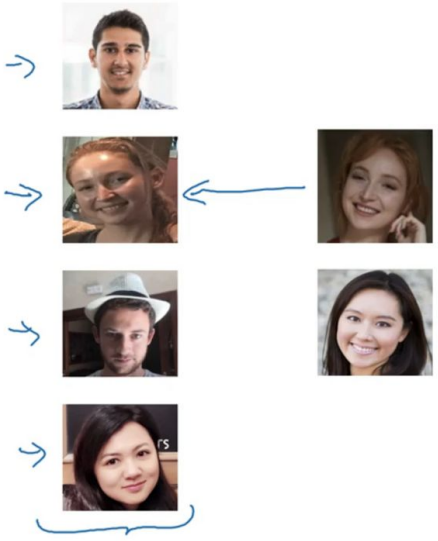
- Has a database of K persons
- Get an input image

K=100 ←

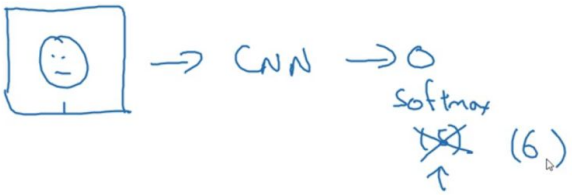
- Output ID if the image is any of the K persons (or "not recognized")

Easier to verify if a person is what he is claiming to be, harder to do recognition. 99% accuracy is ok in verification, but recognition is not good at all with 99% if say there are 100 people, it might get 1 person wrong.

One-shot learning



Learning from one example to recognize the person again



You only have one image for every employee.

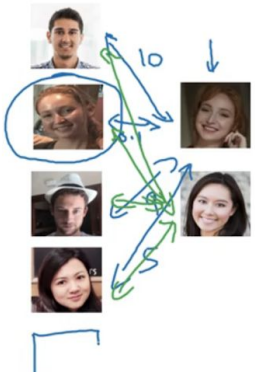
Learning a "similarity" function

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

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

$> \tau$ "different"

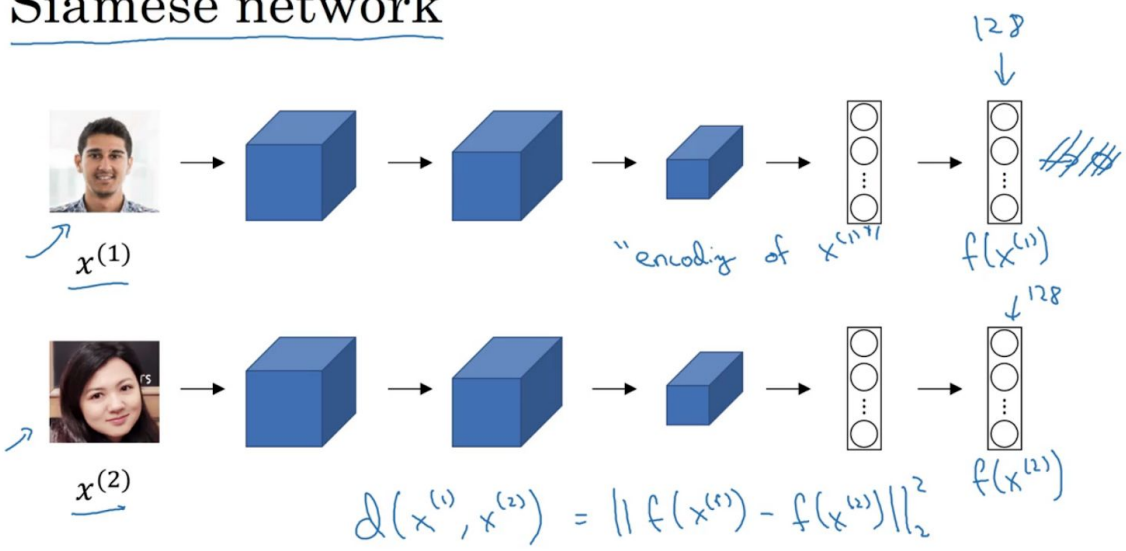
} Verification.



d(img1, img2)

You can add a new employee easily if you can find a similarity function of two images.

Siamese network

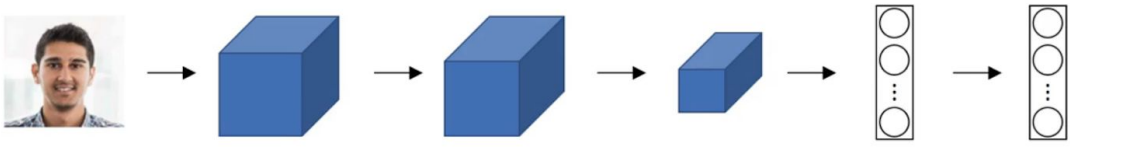


[Taigman et. al., 2014. DeepFace closing the gap to human level performance]

Andrew Ng

Take a norm of the two 128d vectors and see how similar are two images.

Goal of learning



Parameters of NN define an encoding $f(x^{(i)})$ — 128

Learn parameters so that:

If $x^{(i)}, x^{(j)}$ are the same person, $\|f(x^{(i)}) - f(x^{(j)})\|^2$ is small.

If $x^{(i)}, x^{(j)}$ are different persons, $\|f(x^{(i)}) - f(x^{(j)})\|^2$ is large.

TRIPLET LOSS

Learning Objective

$d(A,P) = 0$

Want: $\|f(A) - f(P)\|^2 + \alpha \leq \|f(A) - f(N)\|^2$ — α: 0.2

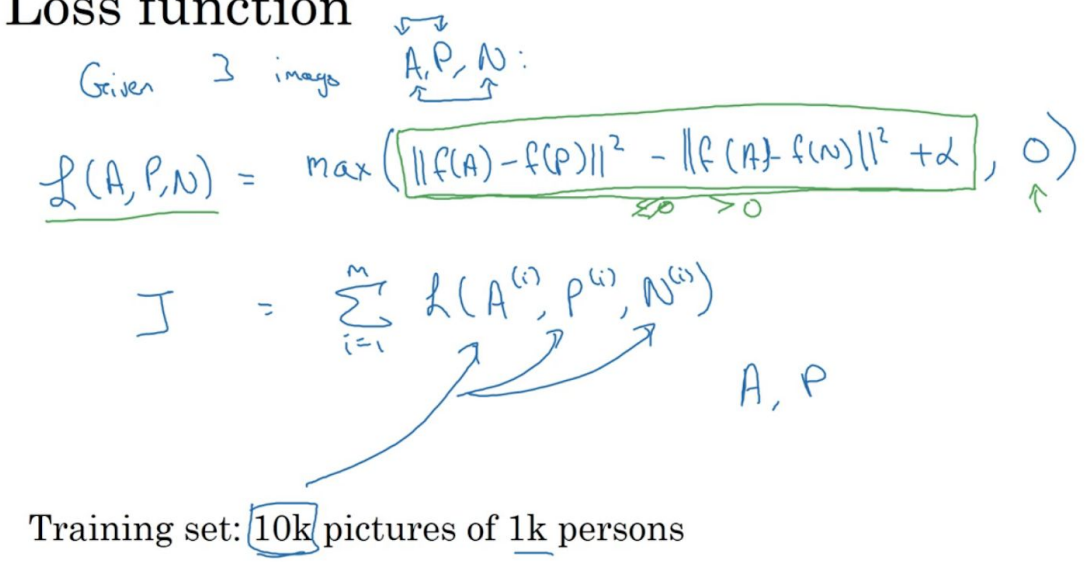
$d(A,P)$ $\qquad$ $d(A,N)$

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

0 $\qquad$ 0 $\qquad$ margin

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

Loss function



Atleast while training need pairs of anchors and positive examples. Once trained can use it for one shot learning case with only 1 picture for an employee, but while training definitely needs more than 1 image per person.

Choosing the triplets A,P,N

During training, if A,P,N are chosen randomly,
$d(A,P) + \alpha \leq d(A,N)$ is easily satisfied.

$\|f(A)-f(P)\|^2 + \alpha \leq \|f(A)-f(N)\|^2$

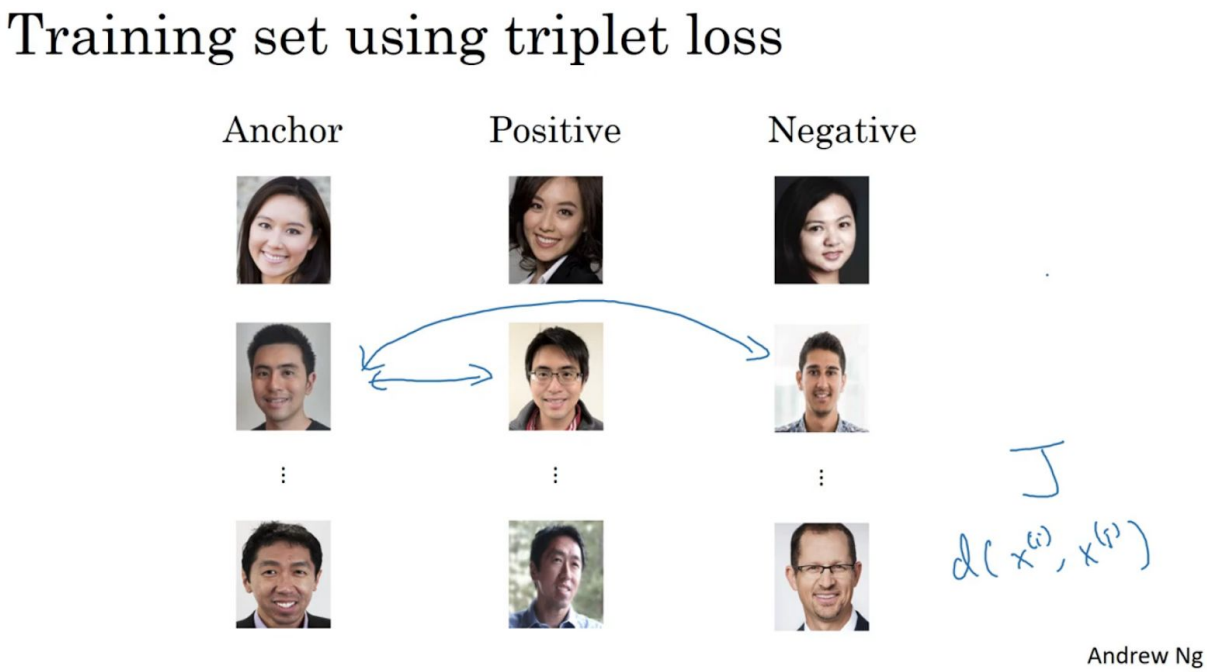
Choose triplets that're "hard" to train on.

$d(A,P) + \alpha \leq d(A,N)$

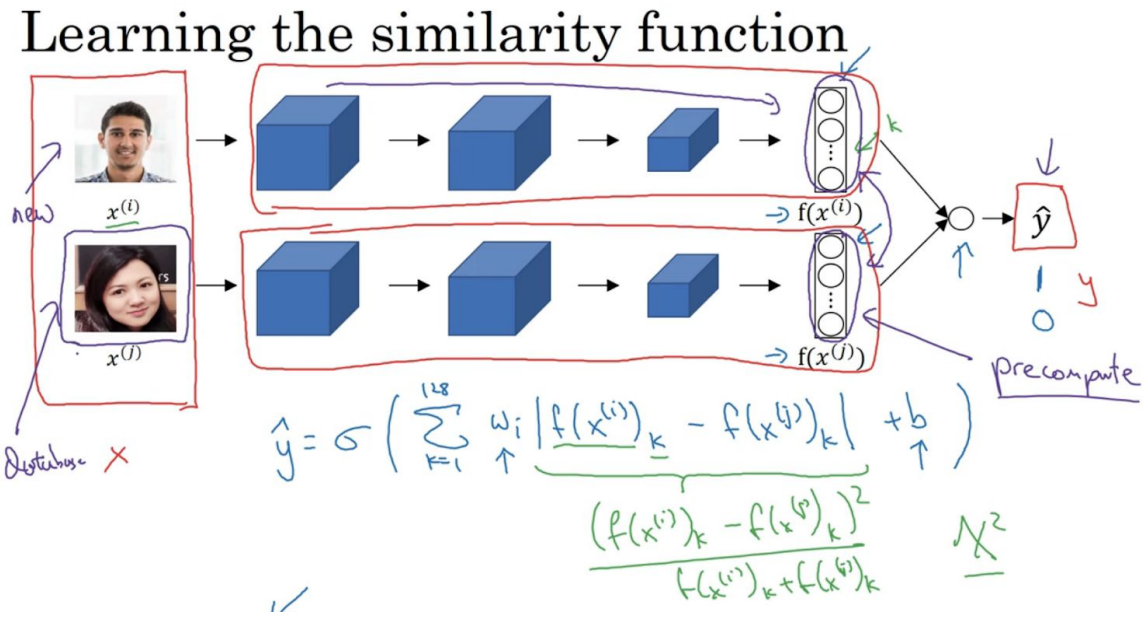
$d(A,P) \approx d(A,N)$

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

Have to choose triplets that are hard to train.



You can also use logistic unit for saying 1 if pictures are similar else 0. You can also keep pre-computed values of employees in database - so you don't lose time.

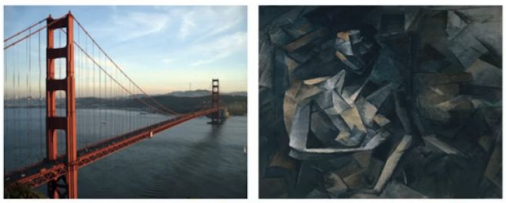


NEURAL STYLE TRANSFER

Neural style transfer

Content (C) Style (S)

Generated image (G)



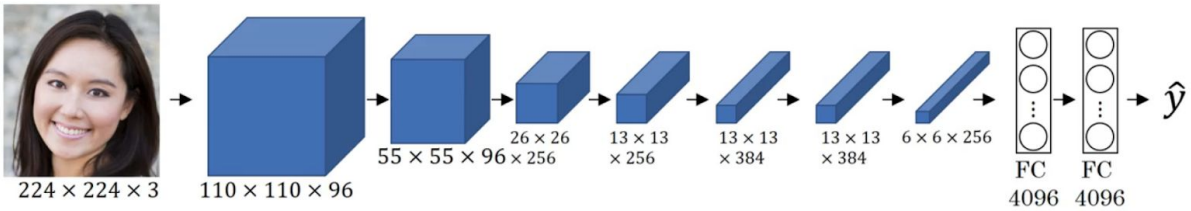
Content (C) Style (S)

Generated image (G)

[Images generated by Justin Johnson]

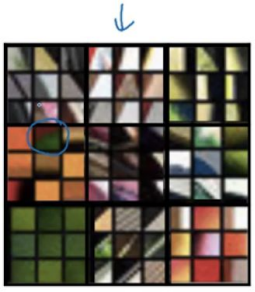
Andrew Ng

Visualizing what a deep network is learning



Pick a unit in layer 1. Find the nine image patches that maximize the unit's activation.

Repeat for other units.

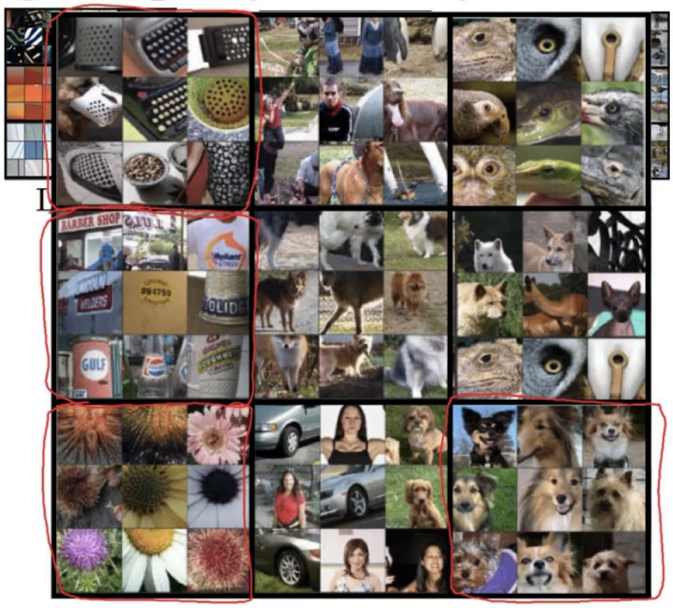


[Zeiler and Fergus., 2013, Visualizing and understanding convolutional networks]

Andrew Ng

Visualizing deep layers: Layer 5

Layer 1



Layer 5

Andrew Ng

Neural style transfer cost function

Content C Style S

Generated image G

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

[Gatys et al., 2015. A neural algorithm of artistic style. Images on slide generated by Justin Johnson]

Andrew Ng

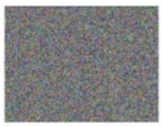
Find the generated image G

1. Initiate G randomly

   G: $100 \times 100 \times 3$ (RGB)

2. Use gradient descent to minimize $J(G)$

$$G := G - \frac{\partial}{\partial G} J(G)$$


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

Content cost function

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

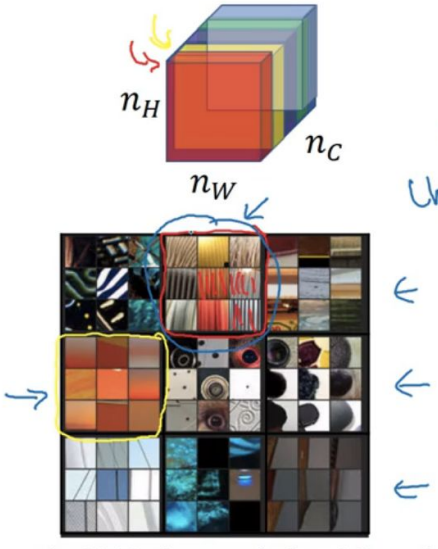
- Say you use hidden layer l to compute content cost.
- Use pre-trained ConvNet. (E.g., VGG network)
- Let $a^{[l](C)}$ and $a^{[l](G)}$ be the activation of layer l on the images
- If $a^{[l](C)}$ and $a^{[l](G)}$ are similar, both images have similar content

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

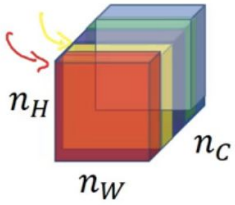
[Gatys et al., 2015. A neural algorithm of artistic style]

Intuition about style of an image

Style image



Generated Image



How co-related are activations across channels? This will help get style cost function.

Style matrix

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

$$G^{[l](S)}_{kk'} = \sum_{i=1}^{n_H^{[l]}} \sum_{j=1}^{n_W^{[l]}} a^{[l](S)}_{ijk} \, a^{[l](S)}_{ijk'}$$

$$G^{[l](G)}_{kk'} = \sum_{i=1}^{n_H^{[l]}} \sum_{j=1}^{n_W^{[l]}} a^{[l](G)}_{ijk} \, a^{[l](G)}_{ijk}$$

$k = 1, \ldots, n_c^{[l]}$

"Gram matrix"

$$J^{[l]}_{style}(S,G) = \frac{1}{(\cdots)} \left\| G^{[l](S)} - G^{[l](G)} \right\|_F^2$$

$$= \frac{1}{(2 n_H^{[l]} n_W^{[l]} n_C^{[l]})^2} \sum_k \sum_{k'} \left(G^{[l](S)}_{kk'} - G^{[l](G)}_{kk'} \right)^2$$

Style cost function

$$J_{style}^{[l]}(S,G) = \frac{1}{\left(2n_H^{[l]}n_W^{[l]}n_C^{[l]}\right)^2}\sum_k\sum_{k'}(G_{kk'}^{[l](S)} - G_{kk'}^{[l](G)})$$

$$J_{style}(S,G) = \sum_l \lambda^{[l]} J_{style}^{[l]}(S,G)$$

Style cost function

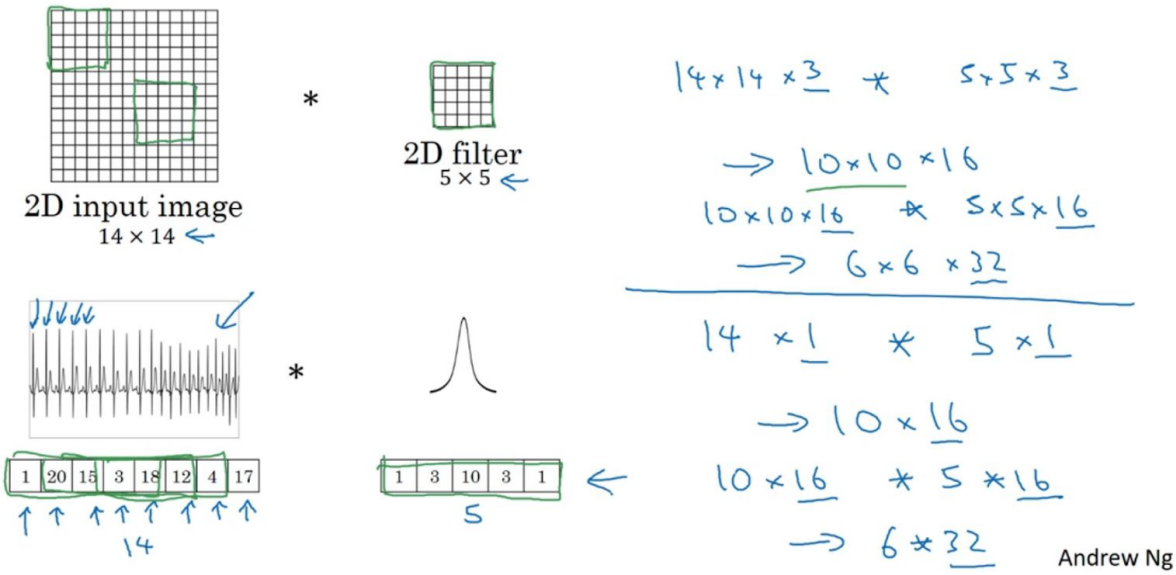
$$J_{style}^{[l]}(S,G) = \frac{1}{\left(2n_H^{[l]}n_W^{[l]}n_C^{[l]}\right)^2}\sum_k\sum_{k'}(G_{kk'}^{[l](S)} - G_{kk'}^{[l](G)})$$

$$J_{style}(S,G) = \sum_l \lambda^{[l]} J_{style}^{[l]}(S,G)$$

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

$$\min_G J(G)$$

Convolutions in 2D and 1D



Sequence models for 1D data → More like RNN but can also use CNN for 1D data

3D convolution