Week 1 Foundations

Thursday, December 14, 2017 8:45 PM

CNNs

Thursday, December 14, 2017 8:46 PM

Computer Vision

Classification, Object detection, Neural Style Xfer

Large images req very large weight matrices, making it difficult to avoid overfitting

Solution: Convolutions!

Edge Detection

Vertical edge detection

convolve

$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

6x6x1 image

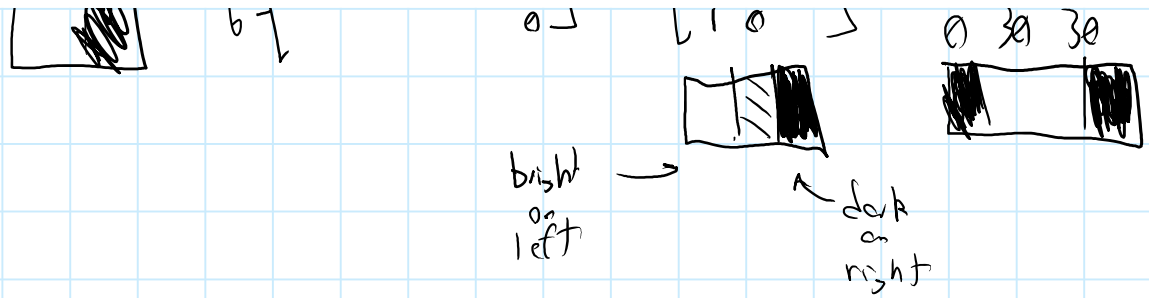
3x3 filter

Python: conv_forward tf: tf.nn.conv2d Keras: Conv2D

ex. image

matrix

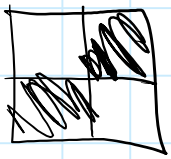
$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & \vdots & \vdots & \\ 0 & & & \\ 0 & 30 & 30 & 0 \end{bmatrix}$$



What if it's flipped?

→ same convolution = same output * −1

Horizontal $\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

 * horizontal convolution =

$$\begin{matrix} 0 & 0 & 0 & 0 \\ 30 & 10 & -10 & -30 \\ 30 & 10 & -10 & -30 \\ 0 & 0 & 0 & 0 \end{matrix}$$

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

Sobel filter
(emphasizes center)

Can also use numbers themselves as parameters

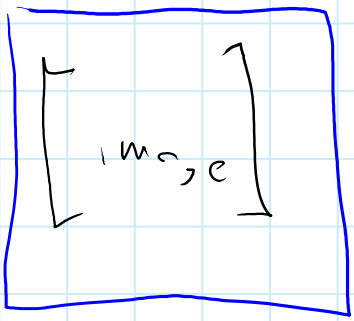
$$\begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix}$$

Padding

$n \times n$ image convolved w/ $f \times f$ filter $= (n-f+1) \times (n-f+1)$ output

Downsides: filter shrinks images
edge pixels are underutilized, center overemphasized

Sol'n: Padding



6×6 w/ 1 px padding $= 8 \times 8$
put thru 3×3 filter $\rightarrow$ 6×6 output!

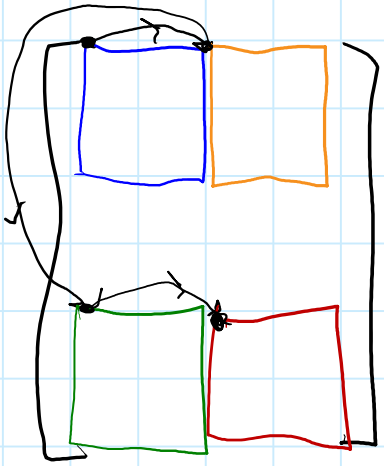
$\text{Output} = (n+2p-f+1) \times (n+2p-f+1)$

How much to pad? Termed valid & same

"Valid": no padding
"Same": Pad so output is $n \times n$ ($p = \frac{f-1}{2}$)

Strided Convolutions

Stride = pixel shift



stride 2

1st

2nd

3rd

4th

$\lfloor z \rfloor = \text{floor}(z)$

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

↑ rounds for stride outside image

Note in math texts convolution matrices are flipped before multiplication

"cross-correlation"

$$\begin{matrix} 3 & 4 & 5 \\ 1 & 0 & 2 \\ -1 & 9 & 7 \end{matrix} \leftrightarrow \begin{matrix} 7 & 2 & 5 \\ 9 & 0 & 4 \\ -1 & 1 & 3 \end{matrix}$$

why? preserves associativity

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

Convolutions Over Volumes

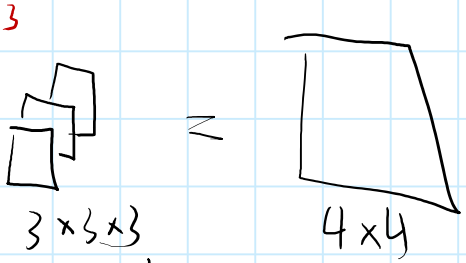
ex. RGB images

height width #channels

$n \times m \times 3$

6px img

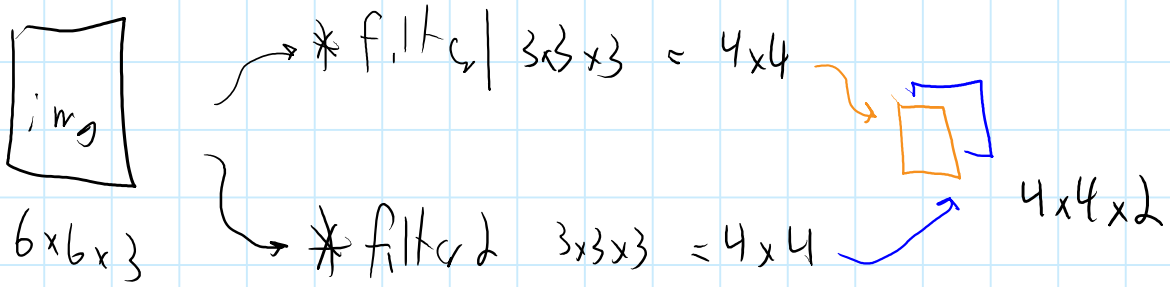
$6 \times 6 \times 3 * \;\; 3 \times 3 \times 3 = 4 \times 4$



The channels of the filter scan over the respective channel of the image

Multiple Filters

Multiple filters

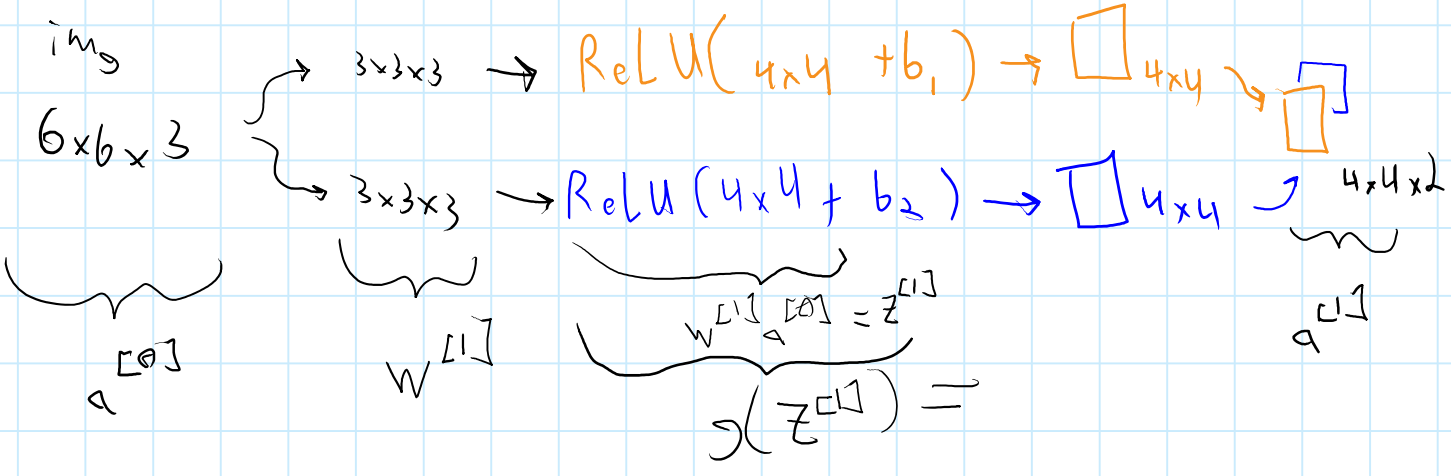


$n \times n \times n_c \quad * \quad f \times f \times n_c \rightarrow (n-f+1) \times (n-f+1) \times (n_c')$

n_c' = # filters

dependent on stride as usual

One layer of a CN



Notation

Notation

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

$p^{[l]}$ = padding

$s^{[l]}$ = stride

$n_c^{[l]}$ = # filters

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

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

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

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

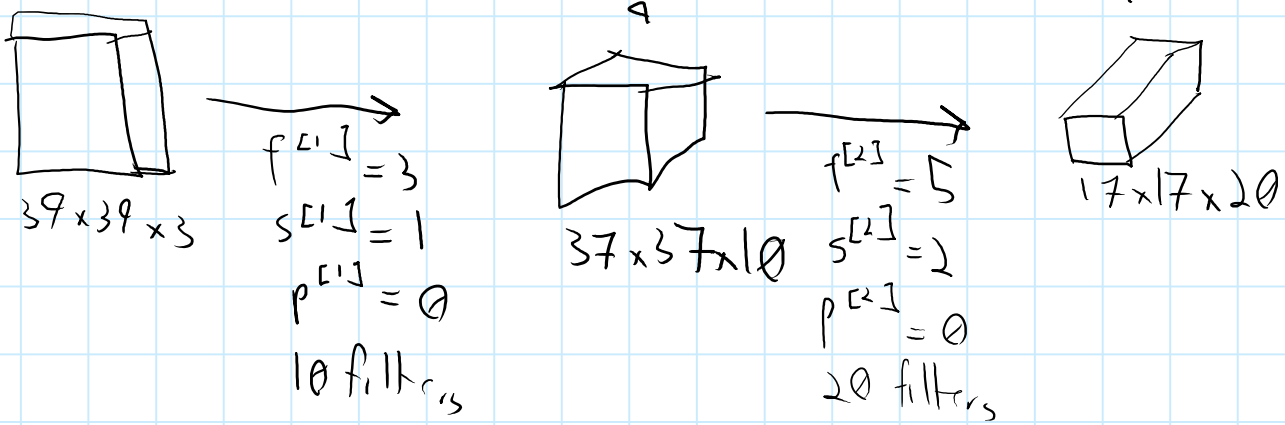
Activations: $a^{[l]} \rightarrow n_H^{[l]} \times n_w^{[l]} \times n_c^{[l]}$

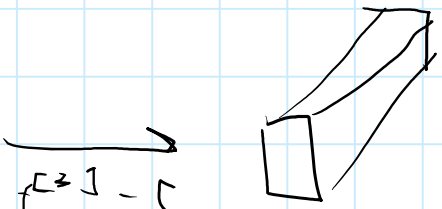
$A^{[l]} \rightarrow m \times n_H^{[l]} \times n_w^{[l]} \times n_c^{[l]}$

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

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

Simple CN Ex.





3 types of layers in a CN

Convolution (CONV)
Pooling (POOL)
Fully Connected (FC)

Pooling Layers

Max pooling

1	3	2	1
2	9	1	1
1	3	2	3
5	6	1	2

→ $f=2$, $s=2$

9	2
6	3

2×2

4×4

if a feature is detected in an area, its presence is preserved

n_c is preserved

Average pooling

1	3	2	1

Not used as often as max pooling

1	3	2	1
2	9	1	1
1	3	2	3
5	6	1	2

→

3.75	1.25
4	2

Summary

f: filter size

s: stride

Max or avg

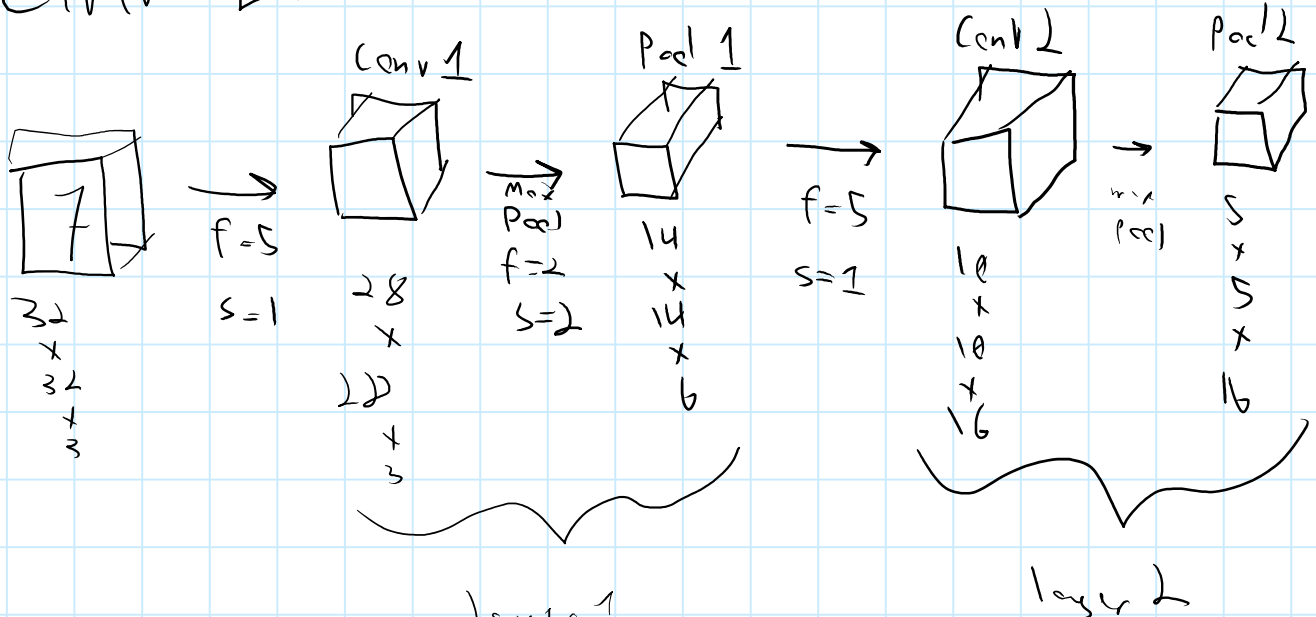
p: padding (always 1)

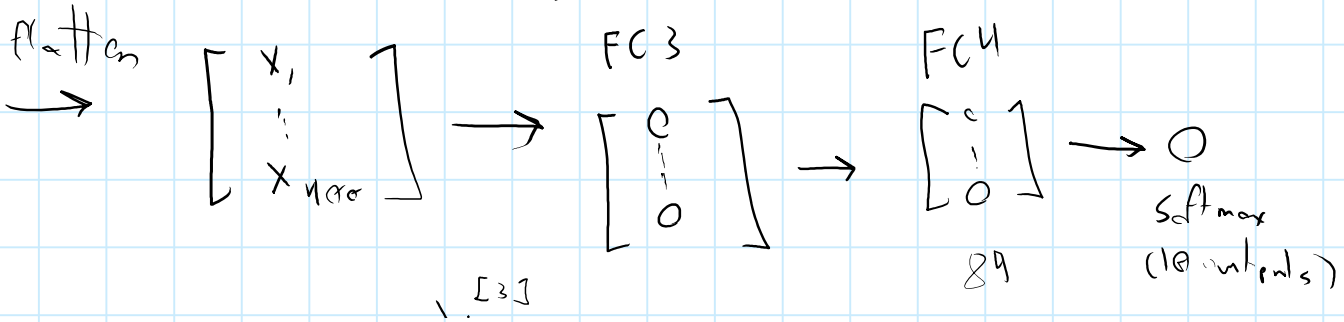
$n_{H} \times n_{w} \times n_{c}$

↓

$\left[\frac{n_H - f}{s} + 1\right] \times \left[\frac{n_w - f}{s} + 1\right] \times n_c$

CNN Ex

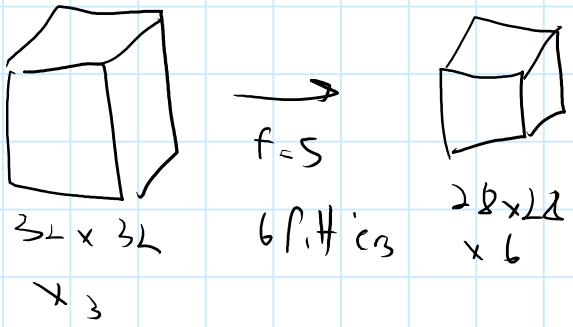




(120,400)

Why Convolutions?

Parameter sharing



each filter has 5x5 + 1 params

26 x 6 filters = 156 parameters

3,072 | | 4,704

Slid NN → 14M parameters!

Feature detectors are useful in diff parts of an image (e.g., edge detection)

Connection sparsity:

Output values only depend on small # of inputs

Week 2 Deep Conv Models - Case Studies

Thursday, February 1, 2018 11:12 PM

Case Studies

Thursday, February 1, 2018 11:13 PM

Why look at case studies?

Intuition building

Outline:

Classic networks

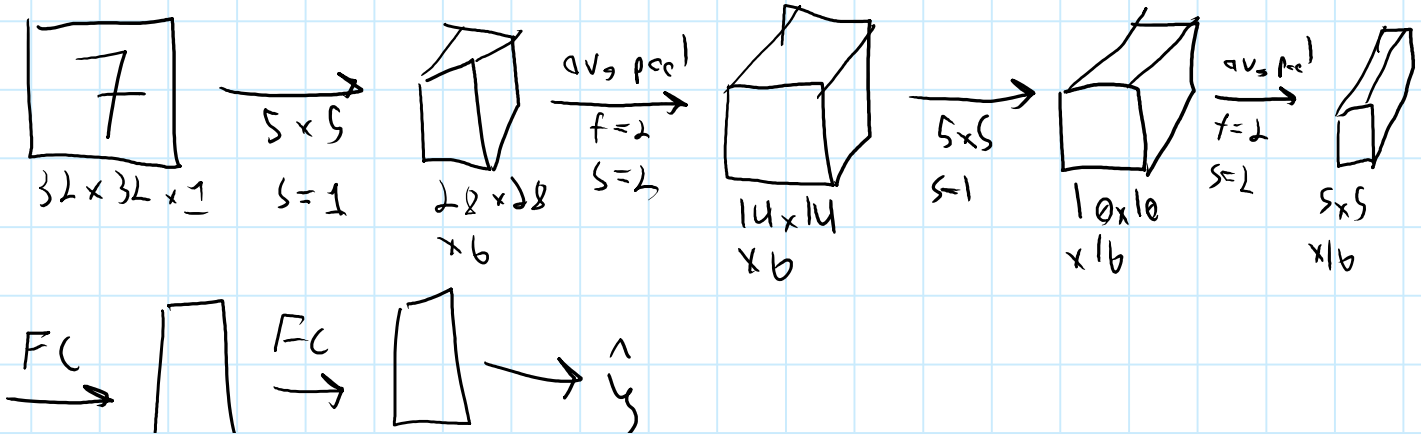
- LeNet-5
- AlexNet
- VGG

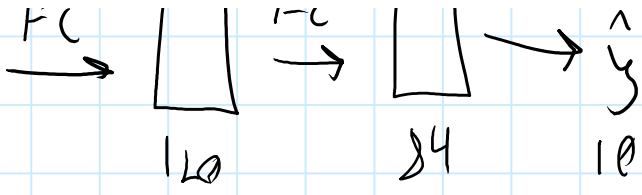
Res (idual) Net (work)

Inception

Classic Networks

LeNet-5 Handwritten characters

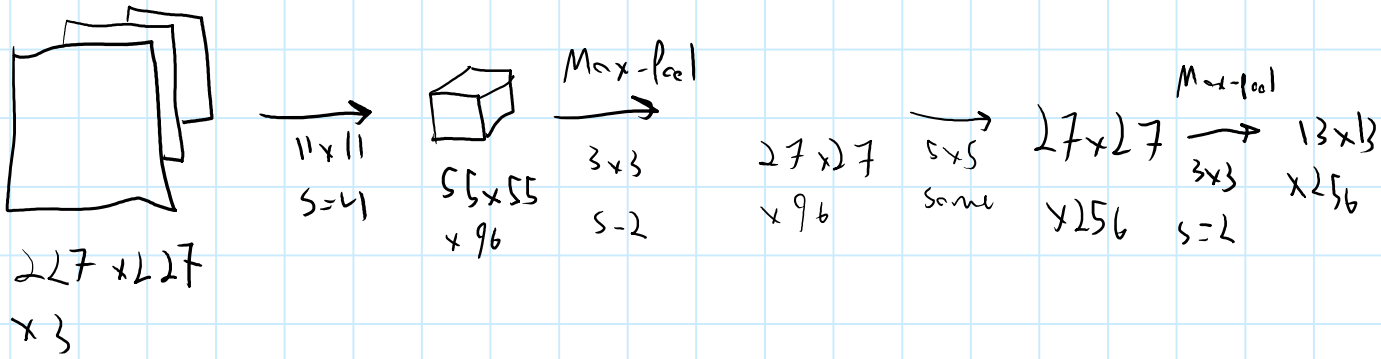




$n_{H,w}$ ↓ n_c ↑

conv → pool → conv → pool → fc → fc → output

AlexNet



→ 3×3 same → 13×13 ×384 → 3×3 → 13×13 ×384 → 3×3 → ... → 1000 possible outputs

Similar to LeNet but much bigger, used ReLU

Local Response Normalization (no longer used)

VGG-16

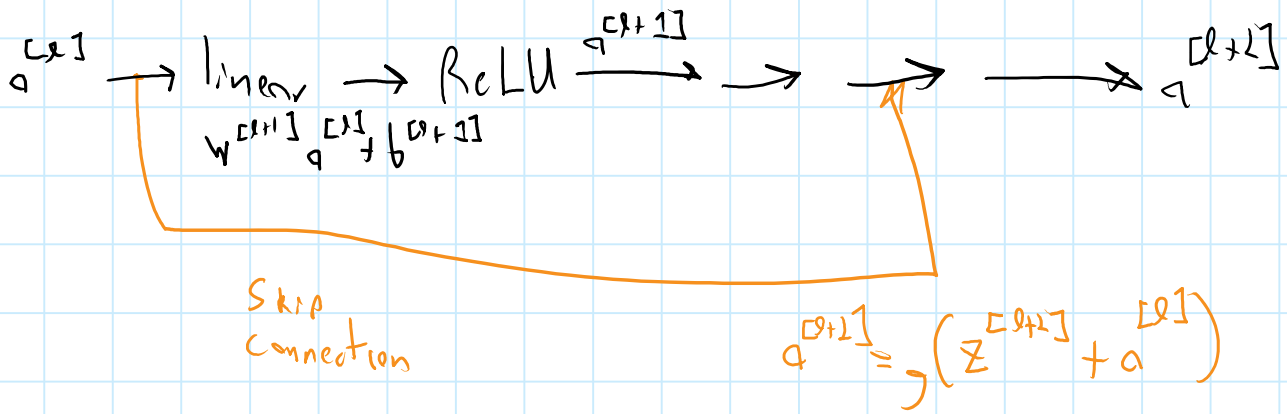
Conv = 3×3 filter, s=1, same Max-Pool = 2×2, s=2

→ Conv 64 → 224 ×224 → →

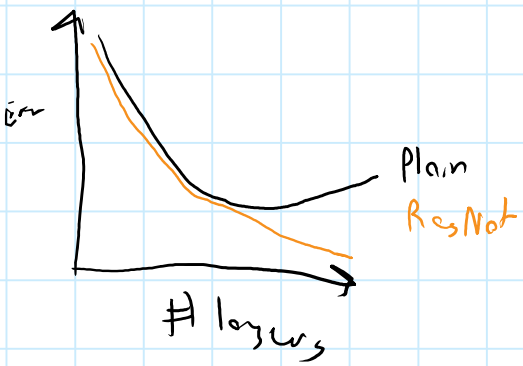
224 x 224 x3 x2 → x64 Pool conv x2 ...

Res Nets

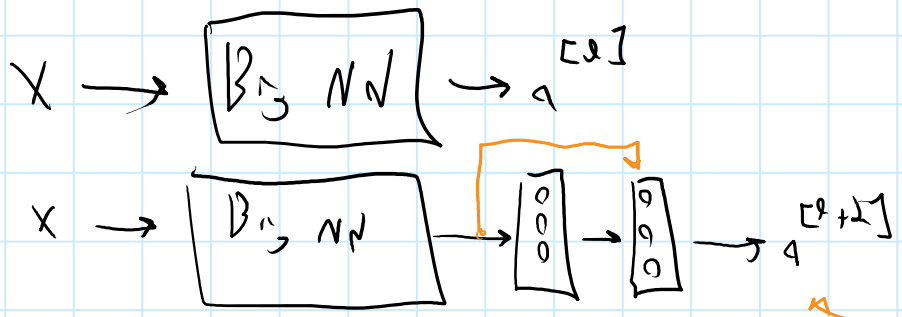
Residual Block



Repeated every 2 layers, helps w/ vanishing / exploding gradients



Why Res Nets Work



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

if $W^{[l+2]}, b^{[l+2]} = 0$, $a^{[l+2]} = a^{[l]}$

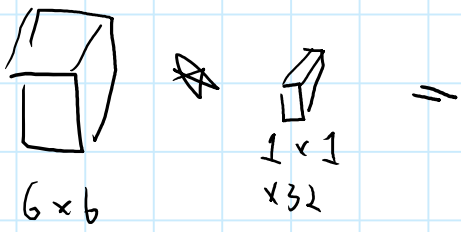
Identity function is easy to learn,
std network has trouble w/ this

Mismatch size correction

$a^{[l+2]} \in \mathbb{R}^m$, but $a^{[l]} \in \mathbb{R}^n$

$a^{[l+2]} = g(z^{[l+2]} + W_s a^{[l]})$, $W_s \in \mathbb{R}^{m \times n}$

Network in Network, 1×1 convolution



takes a slice through layers, like 32 neurons into 1 ReLU neuron

Can be used to reduce n_c just like pooling can shrink n_H, n_W
Can also introduce nonlinearity

Inception Network Key Ideas

Stack multiple filters into one output volume

Reducing computation cost:

Run 1x1 before larger mxm, get some volume but drastically reduce cost

↳ "bottleneck layer"

Inception Network

Week 3 Detection Algorithms

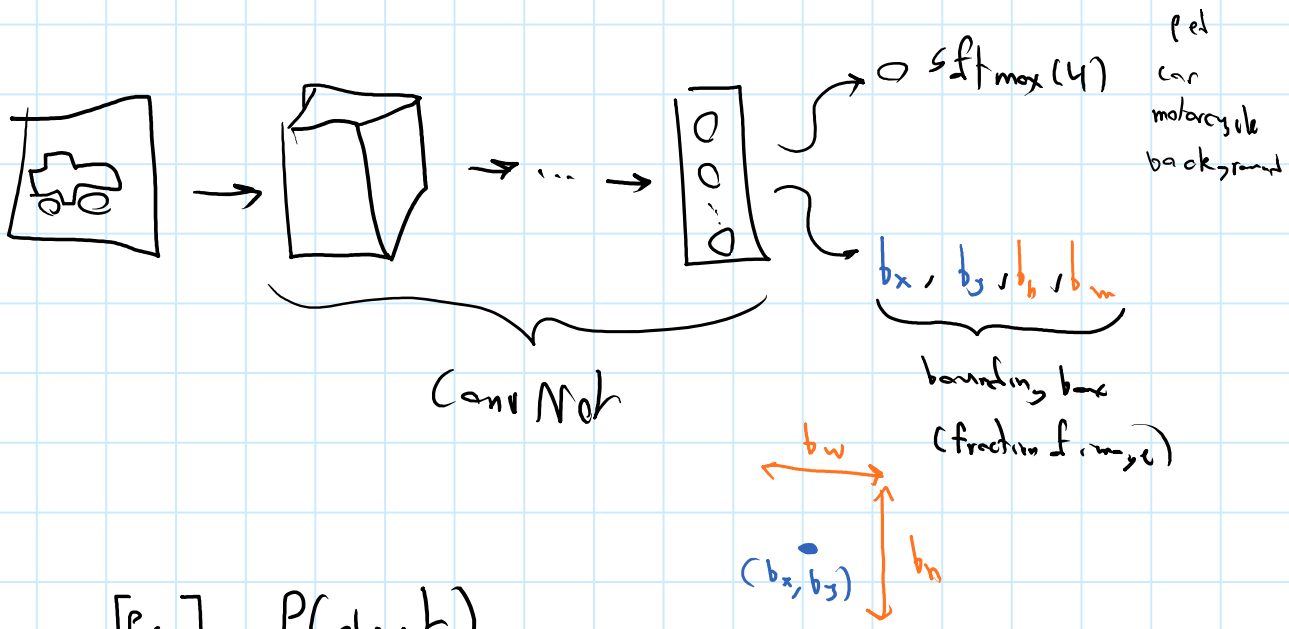
Monday, February 19, 2018 12:14 PM

Detection Algorithms

Monday, February 19, 2018 12:15 PM

Object Detection

Object Localization (bounding box around object in image)

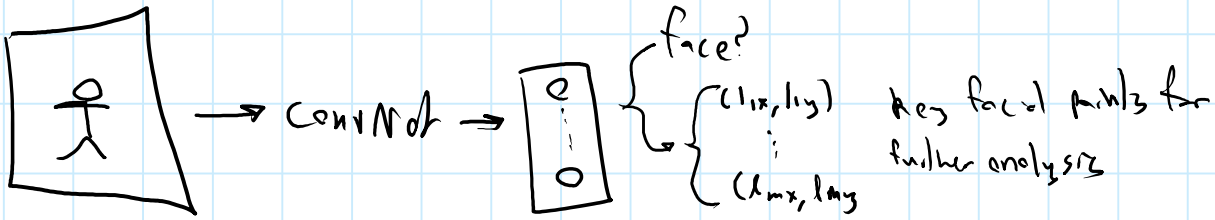


$$y = \begin{bmatrix} p_c \\ b_x \\ \vdots \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \begin{matrix} \leftarrow P(\text{object}) \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \text{bounding box} \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \text{which class} \end{matrix}$$

$$\mathcal{L}(\hat{y}, y) \begin{cases} y_1 = 1 \rightarrow \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 \\ y_1 = 0 \rightarrow (\hat{y}_1 - y_1)^2 \end{cases}$$

Landmark Detection

x & y coord of important features (e.g. corner of eye, body joints)



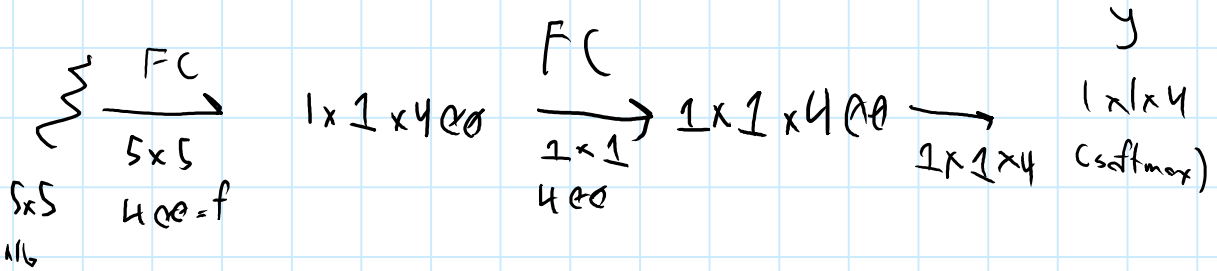
Object Detection

Closely cropped labeled images, then sliding window detection
Repeat w/ larger windows

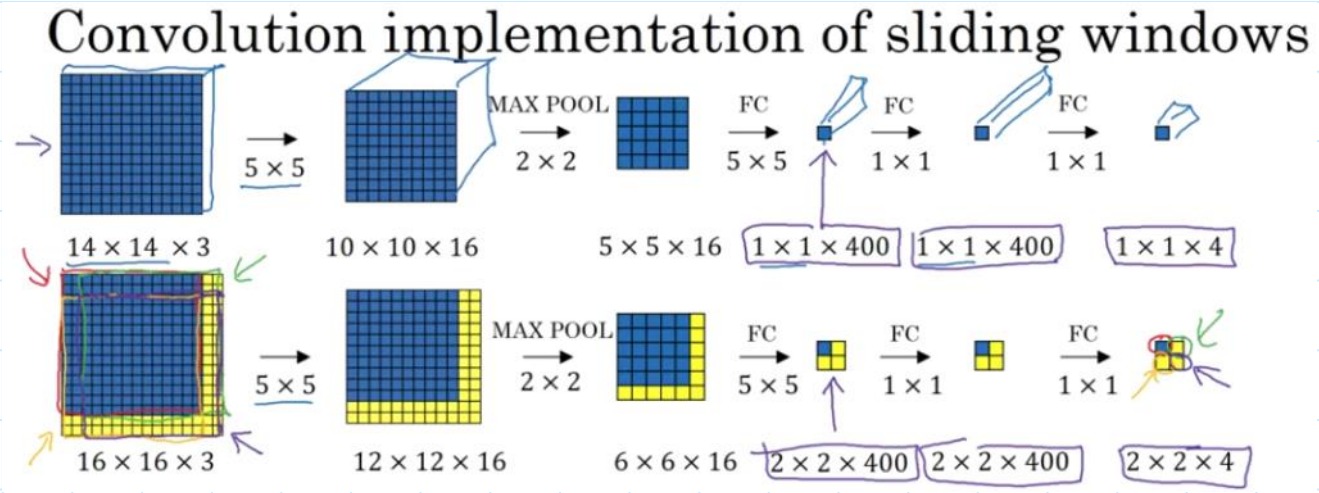
High computational cost to be accurate, solved w/ convolutions

CNN Implementation

Turning FC layers into CN



CNN implementation of sliding windows



Screen clipping taken: 2/20/2018 12:42 PM

Bounding Box Predictions

YOLO Algorithm

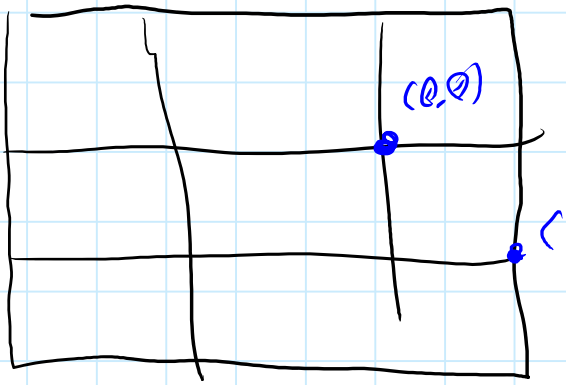
(You Only Look Once)

Divide image into grid,

Iterate one, each grid cell

(a 3x3 grid will have a 3x3x8 output $y = \begin{bmatrix} P_c \\ b_1 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$)

Specifying bounding boxes



relative to grid cell

b_h & b_w may be > 1

Intersection Over Union (IOU)

$$\frac{\text{Size of intersection}}{\text{Size of union}}$$

If $IOU \geq 0.5$

(of computed & ground truth bounding boxes)

Non-max Suppression

Solution to issue of algorithm detecting single object multiple times

Take bounding box with max p_c, negate ones which overlap

repeat for all remaining

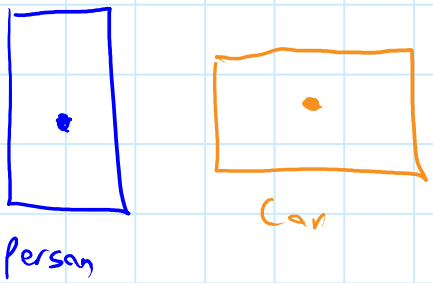
$IoU \geq 0.5$

Best to carry out independently for each class of object (c_n)

Anchor Boxes

Predefined shapes for object classes

e.g.



$$y = \begin{bmatrix} p_c \\ \vdots \\ c_3 \\ p_c \\ \vdots \\ c_3 \end{bmatrix}$$

Each object in training image is assigned to a grid cell that contains object's midpoint & anchor box w/ highest IoU

Has trouble making 2 objects w/ similar anchor box overlapping

Can pick object shapes via k-means

YOLO Algorithm

Ex. Detecting people & cars

Split into cells

For each cell get 2 bounding boxes

Eliminate low pc predictions

For each class, use non-max suppression to generate final predictions

---

Region Proposals R-CNN

Run a segmentation algorithm to section out poss. ble objects

Place bounding boxes around "blobs", run classifier only on said blobs

Speed up by using sliding windows over blobs

Mostly still slower than YOLO

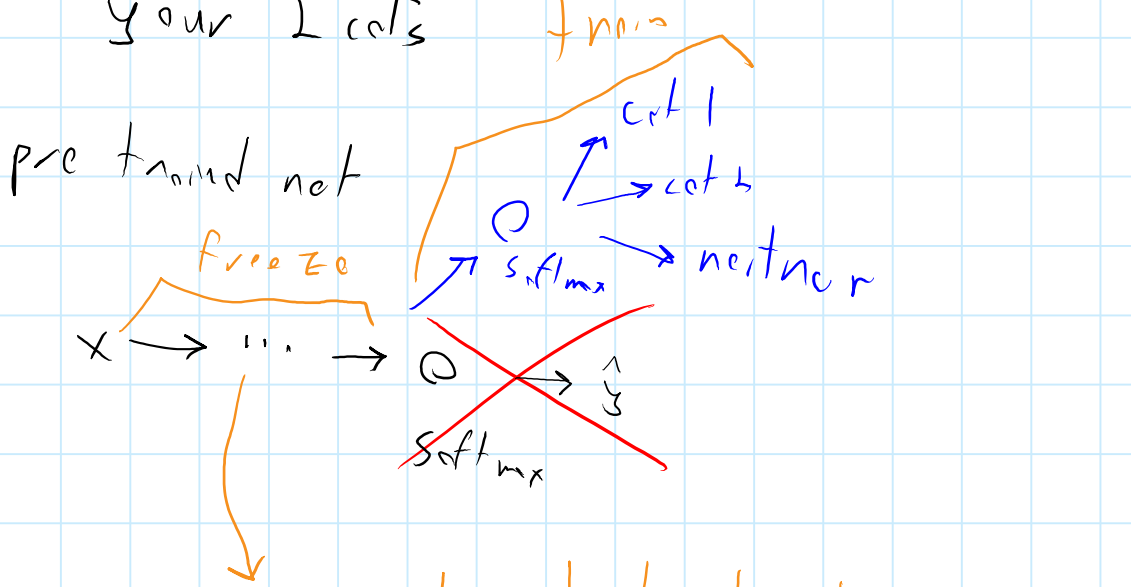
Practical Advice

Monday, March 19, 2018 9:41 PM

Screen clipping taken: 3/19/2018 9:58 PM

~~X~~ for learning

your 2 cats



Partial data? freeze fewer layers, train rest

Full set? Use pretrained weights as "random" initialization

Data Augmentation

Mirroring images

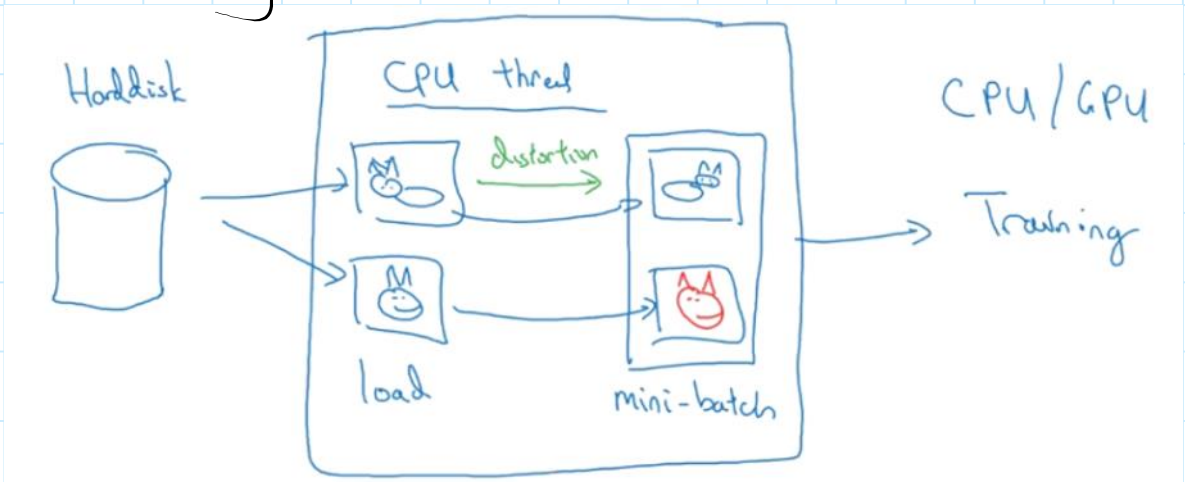
Random cropping

Rotation, shearing, local warping used less

Color shift (random permutation of RGB channels)

- Can be done with PCA
  ex. image is heavily RB, little G
  will shift RB more than G

Implementing



Week 4 Special Applications

Monday, March 26, 2018 8:13 PM

Face Recognition

Monday, March 26, 2018 8:14 PM

What is Facial Recognition?

Verification v. Recognition [Liveness check]

Verification
- Input: image, Name/ID
  Output: image matches credentials

Recognition
- Database of K persons
- Get input image
- Output if image is any of K persons *

* Recall from Bayesian stats that even 99% is not accurate for large K! 99% is not good enough for K=100

One Shot Learning

Learn from one example to recognize person again

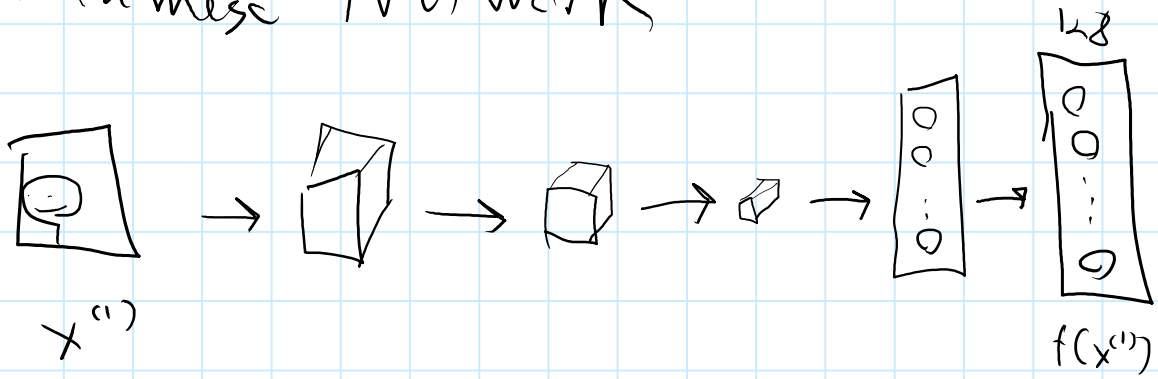
Using a ConvNet is not efficient for this,
use a similarity function

$d(img1, img2)$ = degree of diff between images

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

$d(img1, img2) > \tau$ "different"

Siamese Network



"encoding of $x^{(1)}$"

$x^{(2)}$ → all same

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

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

conversely, large if different

Triplet Loss Function [See Schroff et al, 2015, FaceNet]

A - anchor (reference) P - + (same) N - − (different)

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

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

$\Leftrightarrow \|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 \leq 0 - \alpha$

But a trivial soln is $f(img) = \vec{0}$

Prevent by α

Loss Function

Given 3 images (A, P, N)

$$\mathcal{L}(A, P, N) = \max(d(A, P) - d(A, N) + \alpha, 0)$$

$$J = \sum_{i=1}^{m} \mathcal{L}(A^{(i)}, P^{(i)}, N^{(i)})$$

Training set: ~ 10k pictures of 1k persons

Issue: if A, P, N are chosen randomly,

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

is easily satisfied

Choose triplets that are hard to train on

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

Face Verification & Binary Classification

[Taigman, 2014, DeepFace]

Alternative to triplet loss function

$x^{(i)} \rightarrow \text{ConvNet} \rightarrow f(x^{(i)})$

$x^{(j)} \rightarrow \text{ConvNet} \rightarrow f(x^{(j)})$

$\rightarrow \bigcirc \rightarrow \hat{y}$ (1 or 0)

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

$$\frac{\left(f(x^{(i)})_k - f(x^{(j)})_k\right)^2}{f(x^{(i)})_k + f(x^{(j)})_k} \quad \chi^2$$

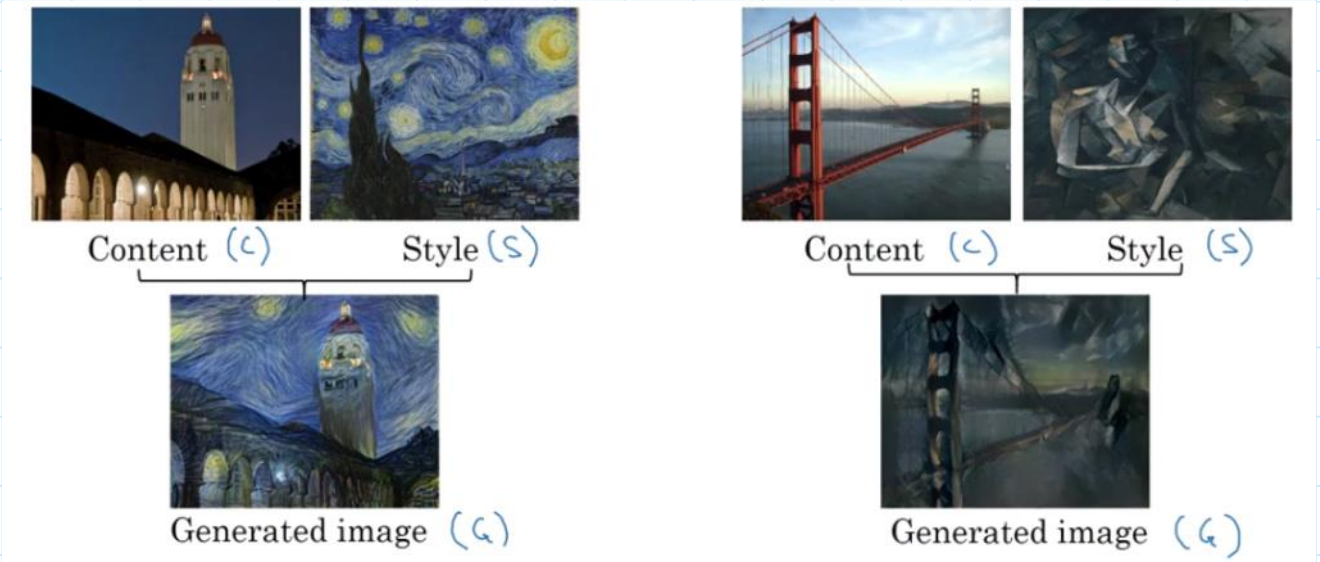
x	y
	1
	0
	0
	1

Screen clipping taken: 3/26/2018 11:09 PM

Neural Style Transfer

Monday, March 26, 2018 8:14 PM

What is Neural Style Transfer



Screen clipping taken: 3/26/2018 11:13 PM

What are Deep ConvNets learning?

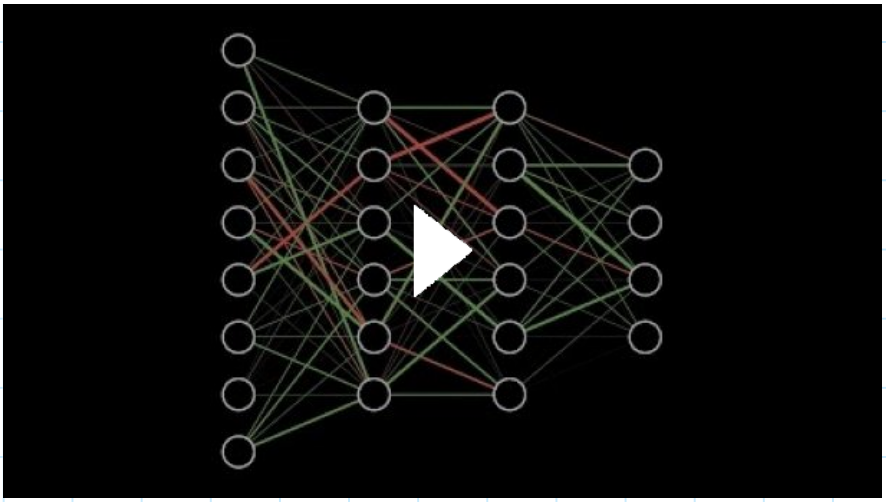
Early layers detect features such as edges & colors

As layers build features become more complex,

Circles → eyes → faces, etc.

See 3Blue 1Brown video for fantastic description

[But what *is* a Neural Network? | Chapter 1, deep learning]



Cost Function

[Gatys, 2015, A neural algorithm of artistic style]

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

$J_{content}$: Similarity to original content

J_{style}: Similarity to reference style

Rough Algorithm

1) Initiate G randomly G: 100 x 100 x 3 (RGB) initially noise

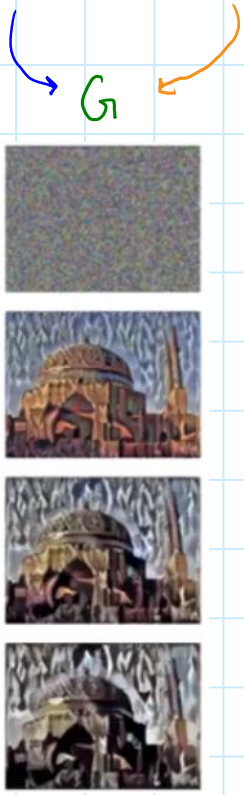
2) Use gradient descent to minimize $J(G)$

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

C S

Screen clipping taken: 3/28/2018 9:01 PM

Screen clipping taken: 3/28/2018 9:01 PM



Screen clipping taken: 3/28/2018 9:01 PM

Content Cost Function

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

- Must carefully select layer l for content cost
  - Too shallow → too similar
  - Too deep → feature placement will be arbitrary
- Use pretrained ConvNet (e.g. VGG)
- Let $a^{[l](C)}$ & $a^{[l](G)}$ be activation of layer l for each image
- If $a \approx a$, content is similar

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

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

L_2 norm of unrolled vectors

Style Cost Function

What does "style" mean?

Correlation between activations across channels

e.g. similarity between R & Y channels at each pixel

Intuition

Red channel detects red lines (texture)

Yellow channel detects orange haze (tint)

Style Matrix

$a^{[l]}_{i,j,k} = \text{activation at } (i,j,k)$, $H \times W \times C$

$G^{[l]}$ is $n_c^{[l]} \times n_c^{[l]}$

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

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

"Unnormalized Cross Covariance"

Define for both S & G

$$J_{style}^{[l]}(S, G) = \left\| G^{[l](S)} - G^{[l](G)} \right\|_F^2$$

$$= \sum_k \sum_{k'} \left(G^{[l](S)}_{kk'} - G^{[l](G)}_{kk'} \right)^2 \quad \text{Frobenius Norm}$$

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

$\lambda^{[l]}$: hyperparameter

1D and 3D Generalizations

Wednesday, March 28, 2018 10:24 PM

1D

2D input $\ast$ 2D filter

14×14 $\quad$ 5×5

$14 \times 14 \times 3$ $\ast$ $5 \times 5 \times 3$

$\rightarrow 10 \times 10 \times 16$

1D EKG Data $\ast$ Gaussian filter

vector of y-values (pk heights)

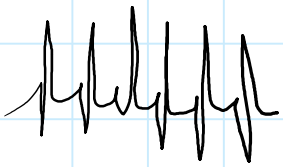
14 $\quad$ 5

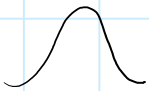
$14 \ast 5 \rightarrow 10 \times 16$

$14 \times 16 \ast 5 \times 16$

$\rightarrow$ n_filters

6×32





3D

e.g. CT Scan, channel is slice

$14 \times 14 \times 14 \times 1$ $\quad$ 16 filters

$\ast$ $\quad$ $\rightarrow 10 \times 10 \times 10 \times 32$

$5 \times 5 \times 5 \times 1$