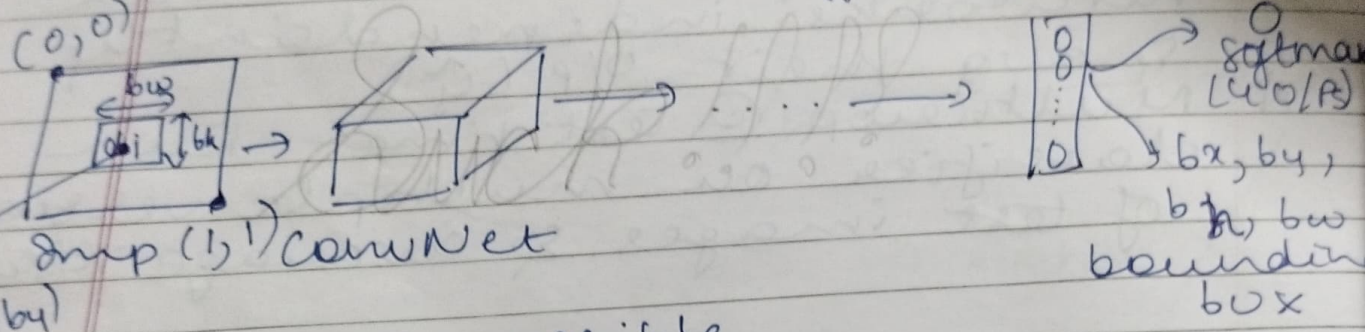
OBJECT LOCALIZATION

Image classification
classification with localization } 1 object

Detection } Multiple objects of diff. types



mid point (bx, by)

4 outputs possible
→ pedestrian → Motorcycle
→ car → Background

$0 \leq b_x, b_y, b_h, b_w \leq 1$
(since opp. corners are (0,0) & (1,1))

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

Is there an object?
1 if obj is there / 0 if obj not there (ie background)

(target label) if obj present

c_1 — class 1
c_2 — class 2
c_3 — class 3

we assume img has only 1 object

Eg if car in image

$$y = \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

— class 2 is car

If no obj

$y = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ \vdots \\ ? \end{bmatrix}$ — we dont care about the values as no obj found

If $y_1 = 1$ $\quad L(\hat{y}, y) = (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 \ldots + (\hat{y}_8 - y_8)^2$

If $y_1 = 1$

(y_1 to y_8 are the elements of y)

If $y_1 = 0$ $\quad L(\hat{y}, y) = (\hat{y}_1 - y_1)^2$ since we dont care of rest of the values

→ Here we squared to simplify but in practice we use a log function like feature loss for C_1, C_2, C_3 to the softmax O/P

→ We can use squared error on something like squared error for bounding boxes

→ & for P_c we can use logistic regression loss.

Landmark Detection

→ If we want to find position for particular features in an object we can have the final layer output all the (x, y) coordinates along the

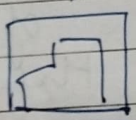
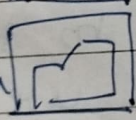
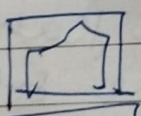
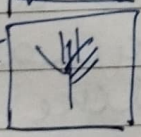
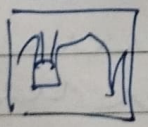
border of the landmarks

OBJECT DETECTION

→ using convnet to perform object detection using sliding windows detection algorithm

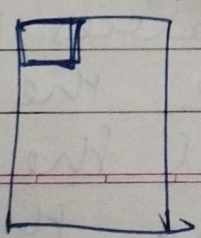
→ make a training set where the images containing the object to be detected are closely cropped ie the object takes up most of the img

training set:

	x	y	
car		1	used to train convnet to detect cars
car		1	
car		1	
not car		0	then we use it in sliding windows dection
not car		0	

→ select a window of certain small size □

→ input it into the img



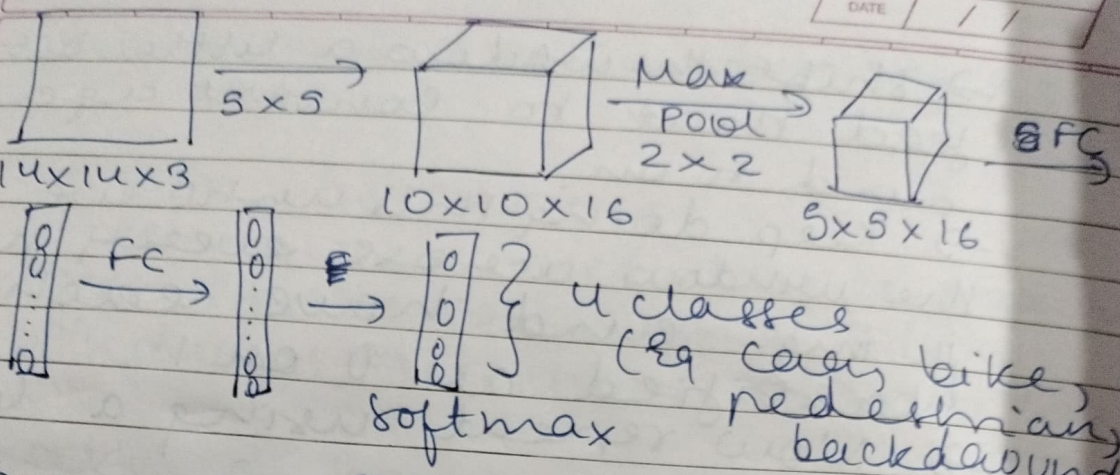
→ have the convnet make a prediction

→ Shift the window a little bit & feed that to convnet again and so on
→ Keep doing this until we slide the window across every position in img. and have each position classified as 0 or 1
→ Now repeat using a larger window
→ Resize to whatever size convnet expects and feed it into convnet & O/P 0 or 1
→ Keep doing this using larger windows and so on...
→ If object is present it will be detected at some position & window size where we will get O/P as 1

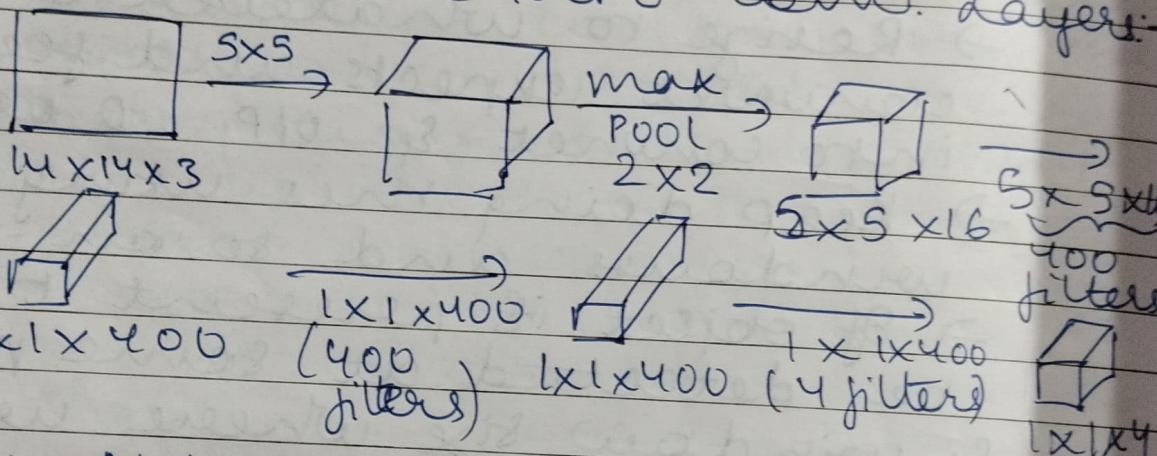
→ Huge disadvantage of sliding windows detection is computational cost
→ Using large slide hurts performance but small stride increases comp. cost

Convolutional implementation of Sliding Windows

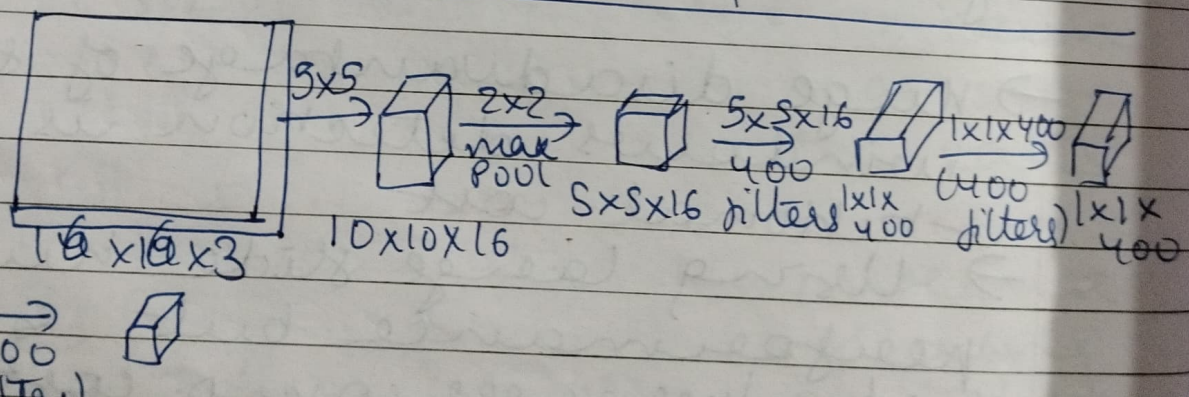
Turning fully connected layers to convolutional layers



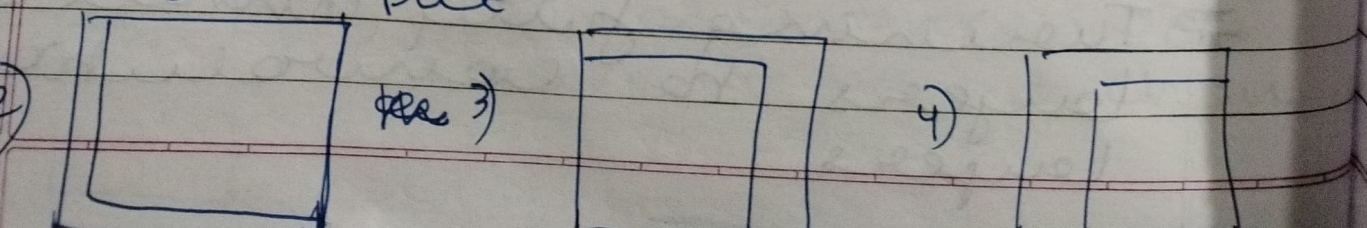
To make FC into conv. layers:-



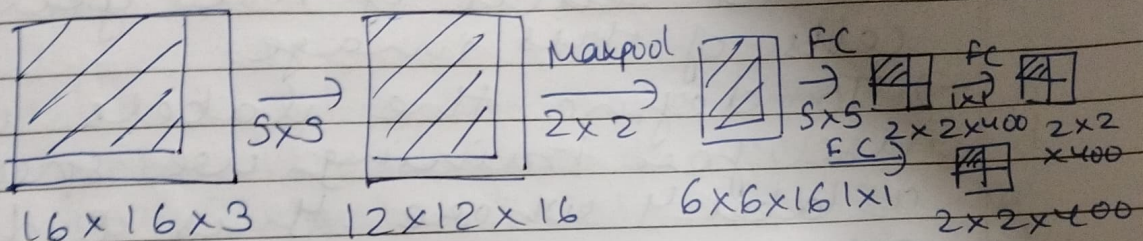
ConvNet in sliding window



→ Suppose our image is 16x16x3 then make a 14x14x3 window & input that in convnet then input



∴ We run 4 diff windows thro the convnet
→ But lot of computation done by 4 convnets is highly duplicative
→ So convolutional implementation of sliding windows allows 4 parts in the convnet to share computation



→ What convolutional implementation does is instead of forcing you to run all 4 propagation on 4 subsets it combines all 4 into 1 form of computation. & shares computation in regions that are common

Bounding Box predictions
→ How to get bounding box predictions to be more accurate
→ None of the windows form a perfect bound box
Sometimes the shape of the perfect bounding is diff from the window we have taken

This can be solved by YOLO - you only look once algo.

→ We place a grid on the input image we are going to take the img classification & localization algo and apply it to each of the grid cells of this image.

→ To define the labels we use for training we specify a label y where y is 8 dimensional vector as seen previously

→ First O/P is Pc which is either 0/1 then b_x, b_y, b_h, b_w to specify bounding box if there is an image obj in the grid cell. and then c_1, c_2, c_3 ($c_i = 0$ as obj present) → so we have a vector for each grid cell.

→ If no of grid cells is 3×3

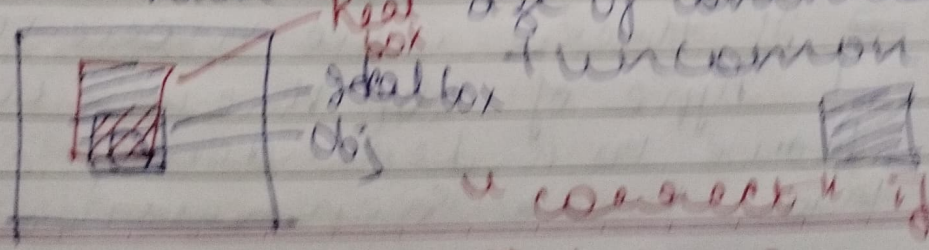
→ Then the target O/P volume is $3 \times 3 \times 8$

→ To train the NN input is Eg:- $100 \times 100 \times 3$ then we have usual convnet with conv, maxpool layers & so on on that in the end we shd choose conv & max pool layers & so on that maps to a $3 \times 3 \times 8$ O/P vol (if grid is 3×3)

Then we backpropagate the network to train the network
→ The advantage of this algo is that N N O/P rectangle bounding boxes
→ The obj is only assigned to 1 grid cell ie the one in which its mid pnt lie
→ bx, by, bh, bw are specified wrt the grid cell and not the whole I/P image
bx, by has to be btw 0 & 1 as mid pnt is within bounds of grid cell
→ bh, bw can be >1

Intersection over union (IoU)
→ used to evaluate obj det algo and also to make it more better
→ IoU O/P the intersection of the actual bounding box and the ideal bounding box

Intersection over Union = Size of common area / Size of common function



"correct" if IoU > 0.5

→ We can use a more stringent threshold than 0.5 also

Non Max Suppression

→ Algorithm may find multiple detections of same objects → Non max suppression is a way to make sure algo detects object only once.

→ It looks at the probabilities associated with each of the detections. (actually P_c times C_1 / C_2 / C_3 → It takes the largest ones as the most confident detection → The non max supression next looks at the rem. windows & all the ones with high overlap with high IOU get supressed

→ STEP 1: ~~Pick box with large~~ Discard all boxes with $P_c \leq$ some threshold value

→ STEP 2: Pick O/P box with largest P_c & O/P it as a prediction

→ STEP 3: Discard any remaining boxes

→ If there are more than 1 type of obj then we have to independently carry non max supression

on each o/p class i

Anchor boxes

→ If we have 2 objects with midpnt coinciding and lying on same grid cell then the output y vector won't be able to detect both and will pick one of the 2.

→ With the idea of anchor boxes is that we predefine 2 shapes called anchor boxes and associate 2 diff predictions with 2 diff bokes

→ We can use more anchor boxes also (> 5)

→ $y = \begin{bmatrix} p_c \\ b_2 \\ b_4 \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_2 \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$ anchor box 1 / anchor box 2

Each object in training img is assigned to grid cell that contains objects midpoint and anchor box for grid cell with highest IOU

5+ no of classes

Output = grid x grid x (8 x no of anchor boxes)

→ If we have 2 anchor boxes & 3 obj then the algo doesn't handle it well.

→ If 2 obj in a grid cell have same anchor box shape then also not handled well

Region proposals - RCNN

→ It picks just a few regions that make sense to run convolutional network rather than running sliding windows on every single window & select few windows & run it on them

→ The way to region region proposals is to run an algo called segmentation algo that results in O/P to figure out what could be objects

It is still quite slow to be faster algorithms:

→ R-CNN: Propose regions. Classify proposed regions one at a time. O/P label + bounding box

→ Fast R-CNN Propose regions Use convol. implementation

5+ no of classes

PAGE No.
DATE / /

Output = grid x grid x (8x no of anchor boxes)

→ If we have 2 anchor boxes & 3 obj then the algo doesn't handle it well.

→ If 2 obj in a grid cell have same anchor box shape then also not handled well

Region proposals – RCNN

→ It picks just a few regions that make sense to run convolutional network rather than running sliding windows on every single window & select few windows & run it on them

→ The way to propose region proposals is to run an algo called segmentation algo that results in O/P to figure out what could be objects

It is still quite slow to be faster algorithms:

→ R-CNN: Propose regions. Classify proposed regions one at a time. O/P label + bounding box

→ Fast R-CNN Propose regions Use convol. implementation

of sliding windows to classify proposed regions

Faster R-CNN — Convol. NN to propose regions

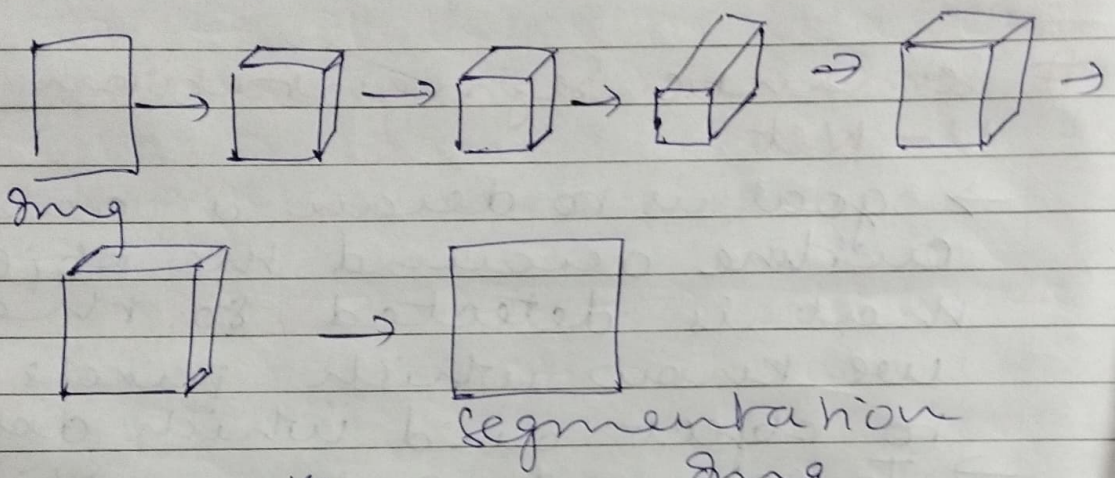
Semantic Segmentation with U-Net

→ goal is to draw a careful outline around the object that is detected so that we know which pixels belong to object and which dont

→ To segment out an obj from background we have 2 class labels 1 for obj & 0 for not obj. In this case job of segmentation algo is to o/p either 1/0 for every single pixel in this image → if we have one more obj we wld have a 2nd class, class 2 and so on in which case the learning algo would label every pixel.

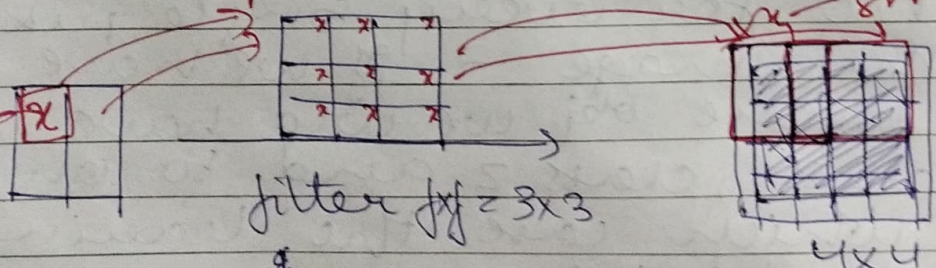
→ One key step of semantic segmentation is as dimensions in CNN get smaller from left to right, it now needs to get bigger so they can gradually blow it up back to a full size image which

→ as we go deeper height & width would go back up while no of channels will decrease



→ To make image bigger again we need to implement transpose convolution.

Transpose convolutions

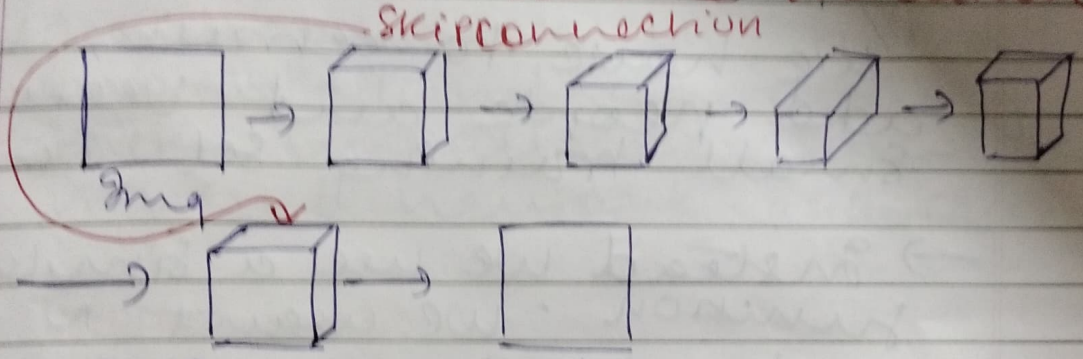


→ In transpose conv we place filter on the output.

→ we take this value and multiply it to every element in filter and take it on upper left

half of the filter
→ stride and padding are used output & not input
→ The padd of filter that overlapping padding remains 0
→ If more than 1 filters are on a pixel of O/P we add them up.

U-Net architecture intuition



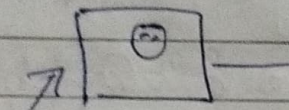
→ For final layer to decide which region is a cat 2 types of info are useful. One is high level, spatial, high contextual which it gets from prev layer. But what is missing is a very fine grained spatial info. The skip connection allows NN to take this very high res, low level feature info where it could capture for every pixel position

Face recognition

PAGE No.
DATE / /

One Shot Learning

→ most face recognition algo don't work well if we have only 1 training eg → it has to learn from just 1 training eg to recognize the person again

→  → CNN → softmax

we have to retrain to add people and very small training set to train

→ Instead we use a similarity function. We want NN to learn a function which is going to denote d, which inputs 2 images & O/P the degree of difference btw 2 images

→ If same person O/P is small no

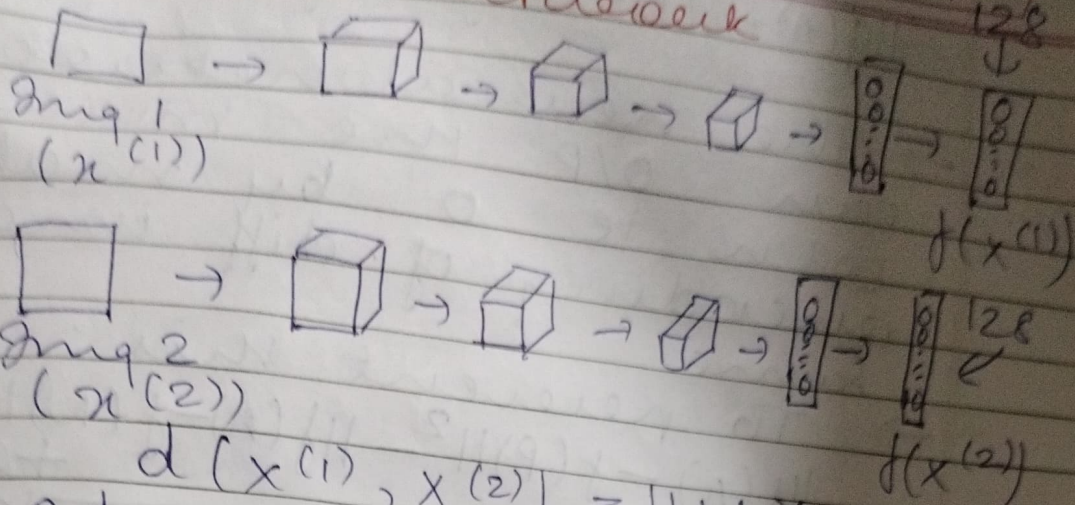
face verification

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

else $> \tau$ "diff"

for face recognition compare with all images in dataset and check if any has low d then they are same

Siamese Network



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

→ Idea of running 2 identical CNN on two diff ~~networks~~ and comparing them is called a siamese neural network architecture

Triplet Loss

→ One way to learn parameters of N.N so it gives u a good encoding for pics of faces is to define & apply gradient descent on triplet loss function

→ We will always be looking at 3 images an anchor img (A), a +ve img (P) & -ve img (N)

We want: $\|f(A) - f(P)\|^2 \leq \|f(A) - f(N)\|^2$

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

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

→ The neural network can O/P the trivial soln ie 0 by making encoding of diff images same

→ To prevent this:

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

Pushes (anchor, +ve) & (anchor, -ve) pair away from each other.

$+\alpha$ ← margin (hyperparameter)

☆ LOSS FUNCTION

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

→ If loss is -ve then it O/P 0 if it +ve then it O/P 1

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

→ If u have 1 pic of each person you can't actually train this system. But of course, after having trained a system u can apply 1 shot learning problem.

choosing triplets APN

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 are hard to train on

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

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

- face recognition verification & Binary classification

→ we can take the 128 dimensional encoding maybe even higher dimensional & input to a logistic regression to make a prediction. Target OP will be 1 if same or else 0. This is binary classification

→ This is an alternative to triplet loss

→ O/P $\hat{y}$ will be a sigmoid function. Rather than feeding in these encodings we can take difference b/w encodings

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

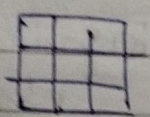
$$\frac{\left(f(x^{(i)})_k - f(x^{(j)})_k\right)^2}{f(x^{(i)})_k + f(x^{(j)})_k}$$

Neural style transfer

What are deep convnets learning?

→ If we want to visualize what hidden units in diff layers are computing:

1) Start with a hidden unit in layer 1 and scan through training sets to find out what are the images / image patches that maximize that units activation ∴ We pause the training set then the NN & figure out what in the image maximizes that particular unit's activation



Unit in Layer 1 — Then we find 9 image patches that influence this layer (since 3 x 3)

→ If we do this for deeper layers a hidden unit will see a larger region of the image. Where at extreme end each pixel could hypothetically affect the O/P of these later layers of NN.

✱ Cost function

→ We define a cost function $J(G)$ that measures how good is a particular generated img and we use gradient descent to minimize $J(G)$ in order to generate this img

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

This is a func of content img & measures how similar contents of generated img are to contents of content img

measures singularity of style of G wrt S

α, β → Specify relative weighting btw content cost & style cost

Find generated img G

1) Initiate G randomly G : 100 × 100 × 3 (Eg)
2) Use gradient descent to minimize $J(G)$

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

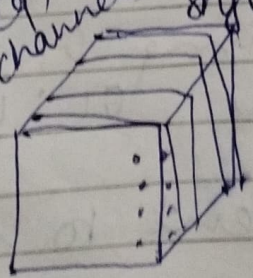
Content cost function

→ For computing content cost layer l need shld be in between (if too deep or not very similar C if too shallow or too similar to S
→ Use a pretrained ConvNet and measure given a content img & gen img how similar are they
→ Let $a^{[l](C)}$ & $a^{[l](G)}$ be activation of layer l on the images
→ If $a^{[l](C)}$ & $a^{[l](G)}$ are similar both have similar content
→ $J_{content}(C, G) = \frac{1}{2} \| a^{[l](C)} - a^{[l](G)} \|^2$

Style cost function

→ Say we use layer l's activation to measure style
→ Define style as correlation bw activations across channels

Layer l activation
(Eg) 5 channels



style img
→ look at first 2 channels and see how correlated activations are
→ & get pairs of nos

at all positions (one in each channel)

→ If 2 channels are correlated the features they detect usually occur together

→ If they are uncorrelated the features won't occur together

→ Then correlate in generated image also and this measures how similar its style is to style img

Style matrix

Let $a^{[l]}_{i,j,k}$ = activation at (i,j,k) in hidden layer l. (height, width, channel)

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

$G^{[l]}_{kk'}$ — will measure correlation bw channels k & k'

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

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

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

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

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

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

allows u to use diff layers in NN

10 & 30 generalizations of models