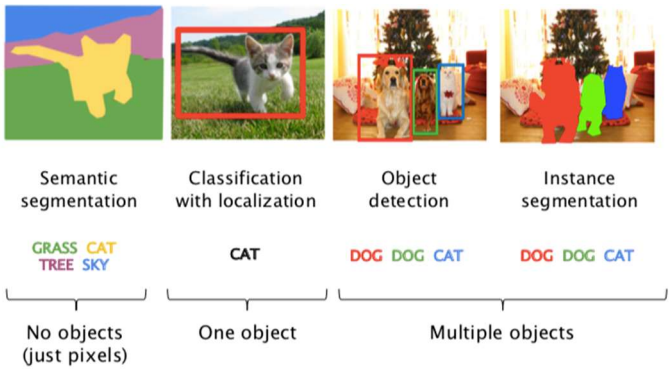
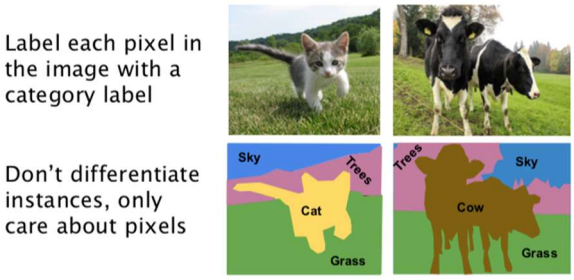
12 - BEYOND CLASSIFICATION AND DETECTION

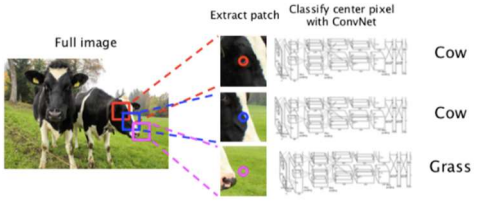
Segmentation



Nella **semantic segmentation** si tratta di sapere quali pixel appartengono ad una classe. Fa coppia con la instance segmentation dove non si estraggono i bounding box come nella object detection ma tutti i singoli pixel appartenenti ad un oggetto.
Nota: ci sono algoritmi nati per fare instance segmentation ma che possono essere usati per fare object detection inserendo bounding box.



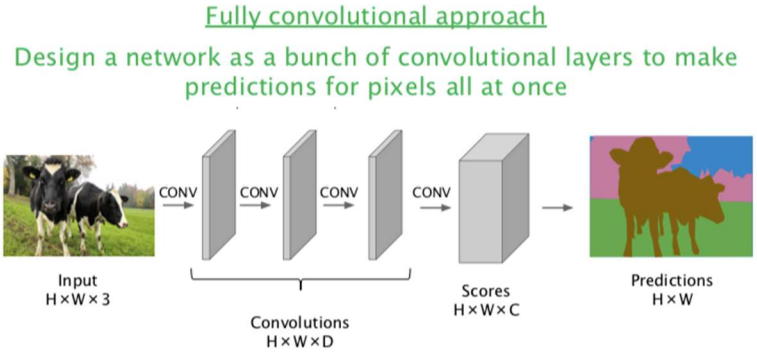
Non si ha il concetto di oggetto, ma viene **aggregato tutto alla classe**.



L'approccio sliding window, chiamato in questo caso Single Pixel Classification, è molto inefficiente.

Si sceglie invece un approccio **fully convolutional**, ovvero progettiamo una rete con una serie di livelli convolutivi che vanno a fare la predizione su tutti i pixel in un colpo solo.

- D: numeri di canali
- C: numero di classi che voglio predire
- Mantengo le dimensioni HxW fino alla fine perché voglio mantere tutta l'immagine
- Nella rete faccio variare solo la profondità che alla fine dovrà essere pari al nr di classi



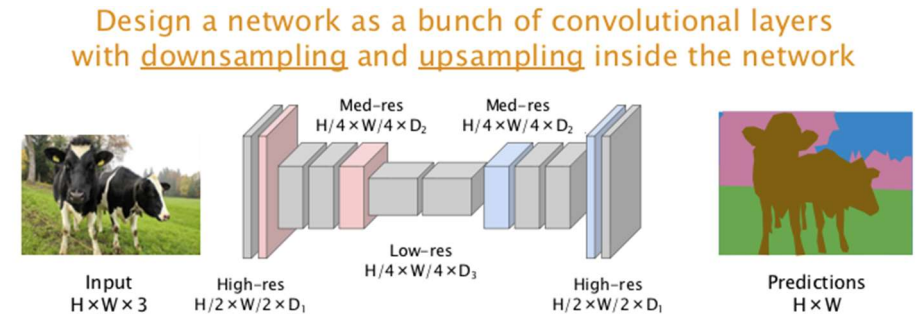
Vantaggi:

- Rete totalmente convolutiva
  - Tutto in un colpo solo
  - Filtri -> alcune cose imparate per una determinata regione di un'immagine serve anche in un'altra regione

Ha dei problemi:

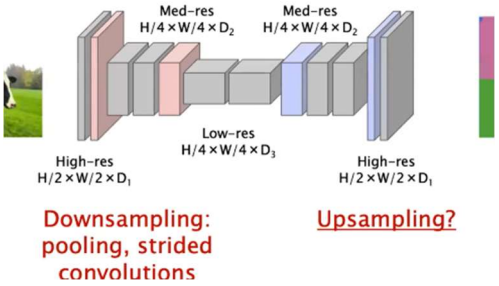
- Uso la stessa dimensione dell'input HxW nelle convoluzioni → oneroso per il calcolo e non scala con l'input size

La soluzione adottata è il modello encoder-decoder



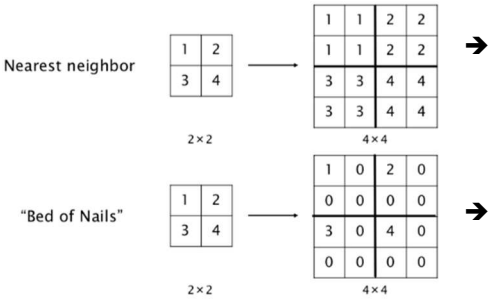
All'interno della rete si compatta la rappresentazione usando pooling/ strided convolution (filtro a passi più grandi) (l*h), nr di filtri diversi (profondità).

In pratica abbiamo una rete con dei livelli convolutivi chiamati di downsampling ottenuti con il pooling o con le strided convolutions, seguiti da dei livelli di upsampling per ri-incrementare la classificazione per fornire l'output.



In-network upsampling: Unpooling

Voglio far diventare la matrice 2*2 -> 4*4.



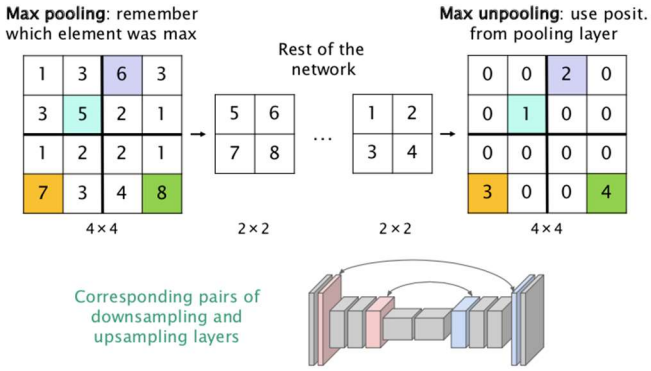
➔ Le celle adiacenti prendono quel valore.

➔ Riporto una sola volta il valore, il resto a 0

Si tiene traccia delle localizzazioni spaziali di dove si trovano i valori per collegare la fase di downsampling con quella di upsampling.
È importante notare che non ci sono parametri.

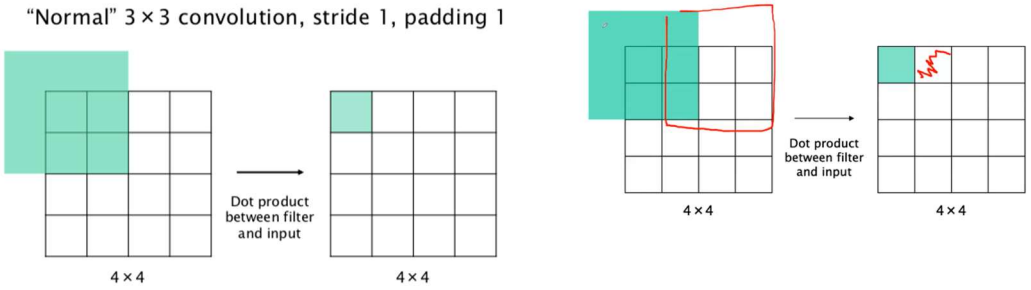
Varie tipologie:

Max unpooling

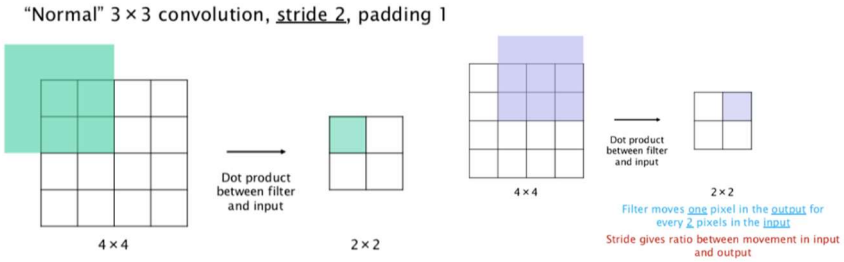


Alto sx: devo usare 1 per popolare la regione, lo inserisco nella posizione in cui era stato trovato il massimo -> il massimo di quel quadrato deve essere quel quadratino.

Learnable upsampling

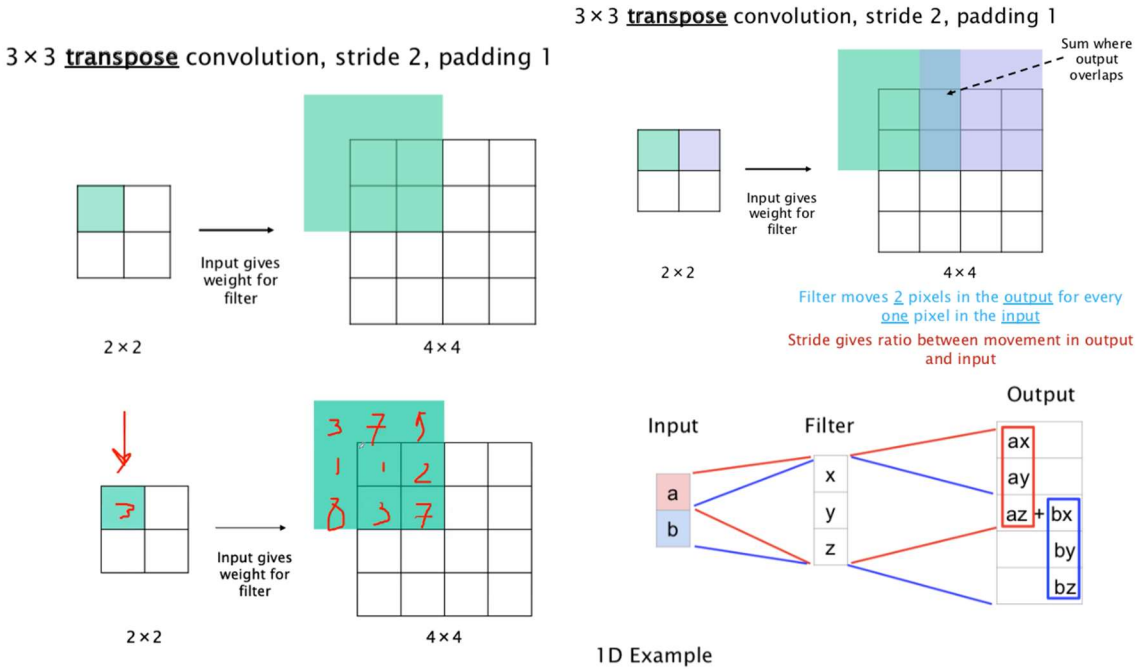


Stride convolution in cui compatto l'informazione ma imparo anche qualcosa.



In questo modo faccio un downsampling

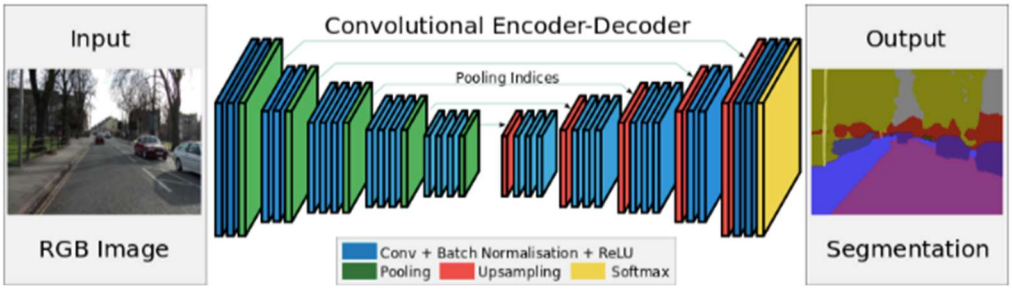
Transpose convolution



Dove c'è sovrapposizione, sommo

SegNet (Reti Encoder-Decoder o AutoEncoder)

Posso usare vgg-16, resnet, etc.
Solitamente vengono usati i primi 13 livelli di VGG-16 buttando i FC (così restano solo i convolutivi), riducendo il numero di parametri da 100 milioni a 10 milioni. Alla fine c'è un softmax multiclasse che produce l'immagine segmentata.
Non usa informazioni temporali -> lavora immagine per immagine, dunque ha una latenza bassa.



(linee orrizzontali, collegano le operazioni di pooling della fase di downsampling alle operazioni di unpooling della fase di upsampling)

Nota: non necessariamente simmetrica.

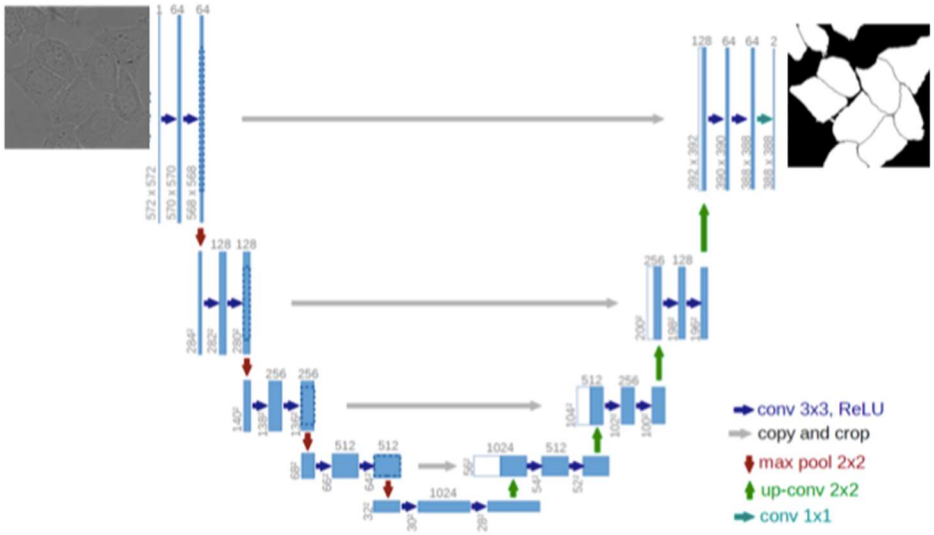
ENet (EfficientNet)

Versione migliore di SegNet

Model	NVIDIA TX1						NVIDIA Titan X					
	480×320		640×360		1280×720		640×360		1280×720		1920×1080	
	ms	fps	ms	fps	ms	fps	ms	fps	ms	fps	ms	fps
SegNet	757	1.3	1251	0.8	-	-	69	14.6	289	3.5	637	1.6
ENet	47	21.1	69	14.6	262	3.8	7	135.4	21	46.8	46	21.6

Model	Building	Tree	Sky	Car	Sign	Road	Pedestrian	Fence	Pole	Sidewalk	Bicyclist	Class avg.	Class IoU
1	75.0	84.6	91.2	**82.7**	36.9	93.3	55.0	47.5	**44.8**	74.1	16.0	62.9	n/a
2	**88.8**	**87.3**	92.4	82.1	20.5	**97.2**	57.1	49.3	27.5	84.4	30.7	65.2	**55.6**
3	74.7	77.8	**95.1**	82.4	**51.0**	95.1	**67.2**	**51.7**	35.4	**86.7**	**34.1**	**68.3**	51.3

U-Net



Si comporta molto bene nell'identificare e separare oggetti che vengono a contatto.

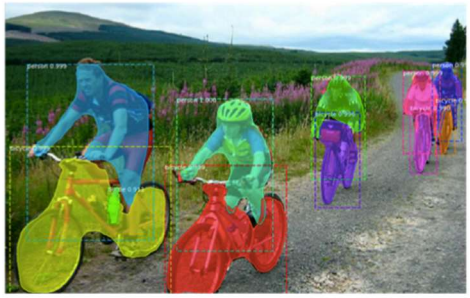
Portano avanti nella rete l'informazione attraverso dei modelli di copy and crop -> copie dell'immagine di partenza tagliata (con la speranza che questo aiuti la rete a convergere)

La complessità in questi modelli è produrre un insieme di immagini e più o meno manualmente tracciare i contorni delle varie istanze -> istance segmentation.

Instance segmentation

Label each pixel in the image with a category label

Differentiate instances



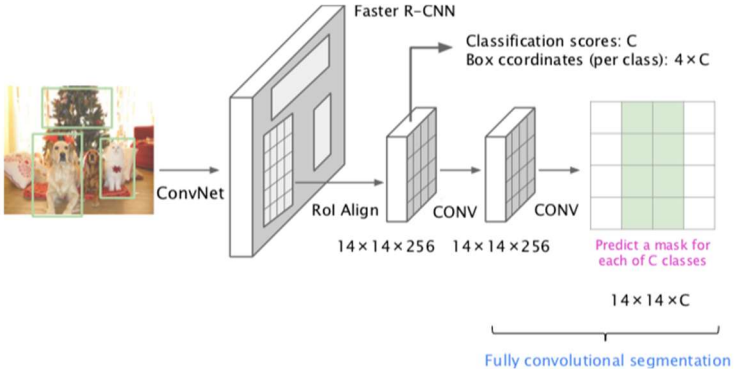
produrre queste annotazioni è molto complicato (separare ciclista dalla bici)

Mask R-CNN

Usa il meglio della detection usando Faster R-CNN e usa il Fully Convolutional per la segmentazione.

- tira fuori delle ROI su cui fa classificazione degli oggetti
- tira fuori i bounding box degli oggetti
- usa un'altra parte per tirare fuori la mask *(fully convolutional segmentation)* → **identificare gli oggetti.**

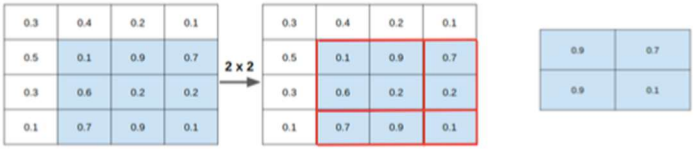
Questo viene fatto per tutti gli oggetti.
Nota: maschera= insieme dei pixel che rappresentano l'oggetto.



Rispetto alle Faster R-CNN c'è una parte di ROI pooling che prende le ROI estratte e va a proiettarle in regioni di dimensione fissa e per farlo va a fare una media dei valori all'interno delle varie feature maps.

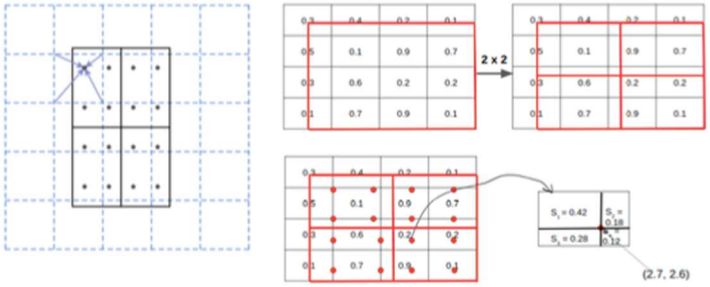
ROI Align vs Pooling

RoI Pooling



pooling-> per passare da una dimensione all'altra

RoI Align



Align: filtraggio bi-dimensionale che dovrebbe permettere di mantenere l'informazione più inalterata

Viene applicata una bilinear interpolation
Risultati:

Circa 5 frame al secondo ovvero 200 ms per immagine
Questa rete può essere usata anche per estrapolare i landmark

FACE VERIFICATION AND RECOGNITION

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

➔ data un'immagine di input, dire se quell'immagine corrisponde ad una determinata persona

- Recognition
  - Given a database of K persons
  - Get an input image
  - Output ID if the image is any of the K persons (or "not recognized")

➔ Dato un data-set che contiene n persone, identificare di che persona si tratti

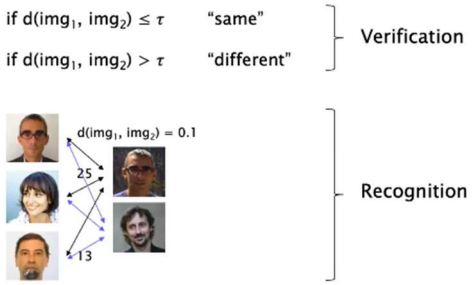
One shot learning

Si chiamano one-shot learning perché si ha un'unica immagine da cui imparare il riconoscimento per il seguito. Questo perché, se dovessi voler aggiungere un nuovo soggetto con gli approcci standard dovrei riallenare di nuovo tutta la rete, ri-addestrare l'algoritmo.

Learning a "similarity function"

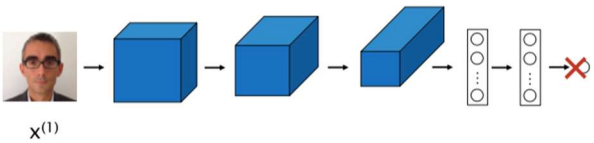
Mi dice se due immagini sono simili o no, in base ad una determinata soglia.

Date due immagini si definisce d(img1, img2) come il grado di differenza tra queste due e impostando una soglia posso dire se sono la stessa o meno.

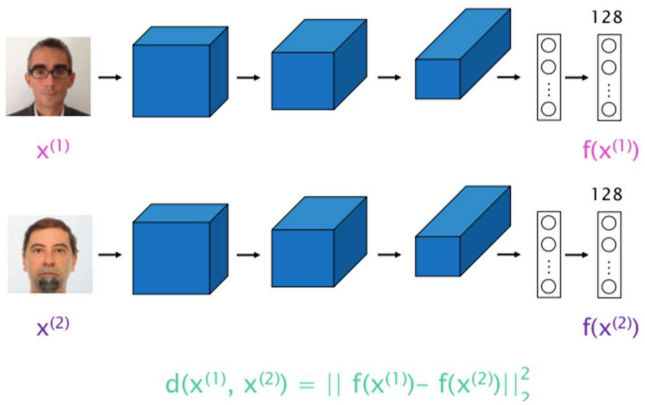


Siamese network

Prendiamo una rete e buttiamo l'ultima parte di classificazione

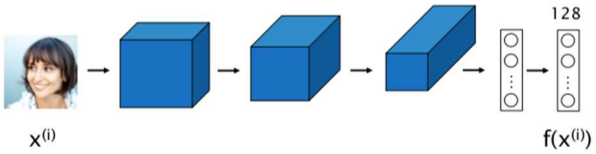


Il nuovo ultimo livello sarebbe l'encoding $x^{(1)}$ e contiene le feature calcolate dai livelli precedenti e lo chiamiamo $f(x^{(1)})$.
Prendiamo un'altra immagine e prendiamo la stessa rete e abbiamo $f(x^{(2)})$.



Se possiamo assumere che gli encoding sono una buona rappresentazione delle immagini allora si può definire d come sopra.

Goal of learning



Parameters of the network define an encoding of $f(x^{(i)})$

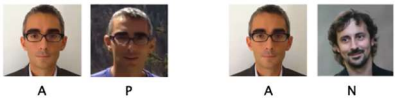
Learn parameters so that:

if $x^{(i)}$, $x^{(j)}$ are the same person, $d(x^{(i)}, x^{(j)}) = || f(x^{(i)}) - f(x^{(j)})||^2$ is small

if $x^{(i)}$, $x^{(j)}$ are different persons, $d(x^{(i)}, x^{(j)}) = || f(x^{(i)}) - f(x^{(j)})||^2$ is large

Triplet loss

Nella realtà abbiamo a che fare con una rete siamese a cui passiamo delle immagini e dobbiamo confrontare essenzialmente l'anchor con un esempio positivo e con uno negativo.

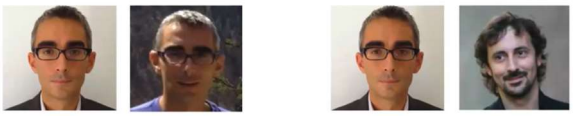


Want

$$\underbrace{||f(A) - f(P)||^2}_{d(A,P)} \leq \underbrace{||f(A) - f(N)||^2}_{d(A,N)}$$

La nostra rete potrebbe imparare un encoding per cui a tutte le immagini associo un vettore di zero per avere la regola sempre verificata.

Quindi **si introduce un margine** che è un hyperparametro della rete



Want

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

Definisco questo nuovo loss per la tripletta

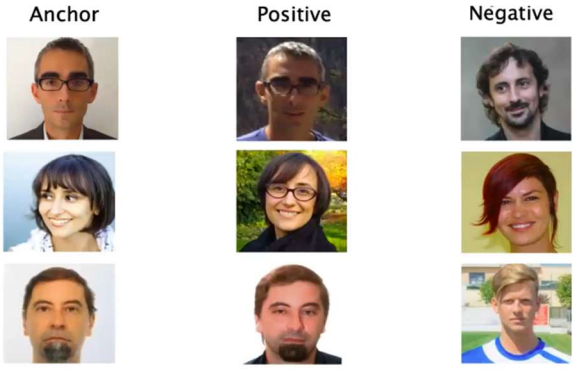
Given three images A, P, N

$$Loss(A,P,N) = \max\left(||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + \mu, 0\right)$$

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

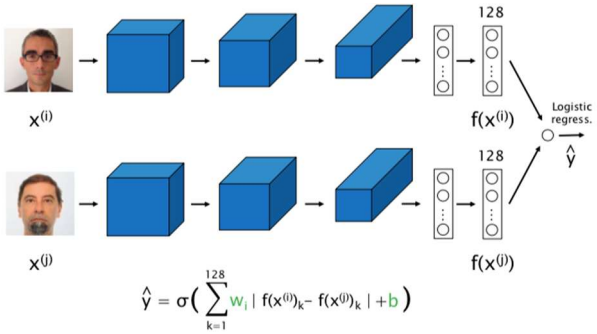
During training, if A, P, N triplets are chosen randomly, $d(A,P)+\mu \leq d(A,N)$ is easily satisfied, hence triplets shall be chosen so that they are "hard" to train on

Per ogni soggetto serve un esempio positivo e uno negativo e bisogna fornire esempi difficili per addestrare bene la rete.

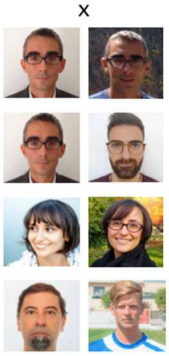


Servono circa 10000 immagini di 1000 persone però una volta addestrata si può usare per problemi di one shot learning.

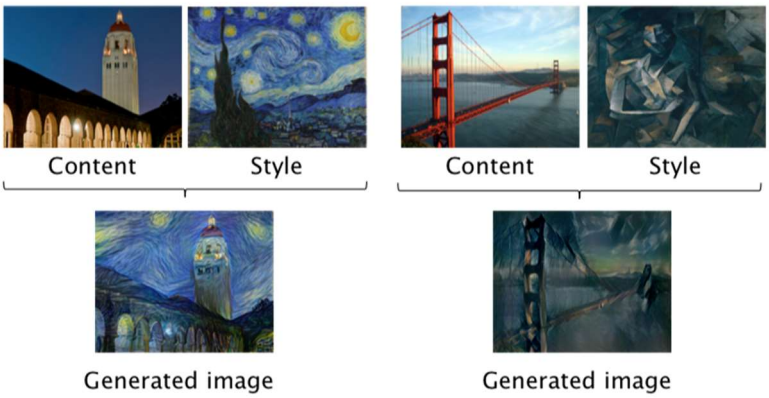
Learning similiarity with binary classificationceh



1 se rappresenta la stessa persona, 0 altrimenti

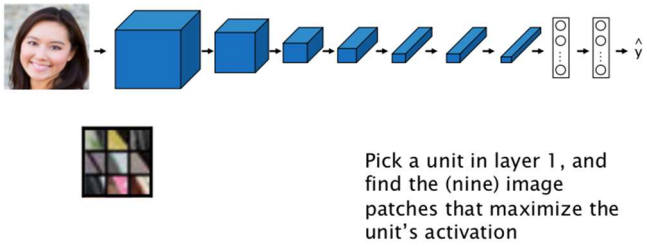


Neural style transfer



➔ Immagine che contiene le informazioni della prima immagine con lo stile della seconda

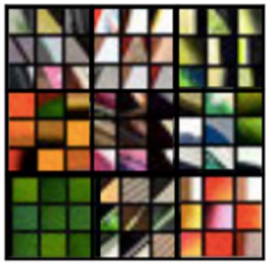
Visualize what network is learning



Presa una rete convolutiva, cosa impara la rete?

- Prendiamo un neurone al primo livello, quali sono le immagini che massimizzano l'attivazione di que neurone?

Se andiamo a prendere per nove unità del layer 1

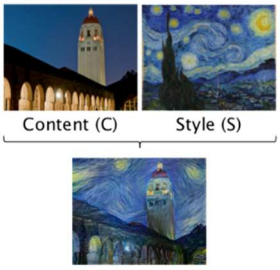


Nine units in layer 1

Ognuna di esse viene attivata da un certo tipo di elemento di basso livello presente nell'immagine.

Neural style transfer cost function

La rete dovrebbe minimizzare la funzione di costo tra l'immagine di contenuto e quella da realizzare e al tempo stesso, cercare di minimizzare la funzione di costo tra immagine di stile e l'immagine da generare.



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

Alfa e beta sono hyperparametri.

Generate the output image

Given C and S

Initiate G randomly

Then use, e.g., gradient descent to minimize J(G)

➔ Man mano che la rete impara, prova a realizzare un'immagine che assomigli ad entrambe le immagini.

Si inizia settando G randomica e si definisce una funzione di costo settando i parametri per allenare la rete.

Content cost function component

Considerando il costo della funzione di content utilizzo un layer intermedio per valutarlo in quanto alla fine della rete otterei semplicemente un output quasi identico al content.
Si prende una rete convolutiva pre-addestrata (es. VGG), e si immagini di avere le attivazioni calcolate per quel livello calcolate sul content e sull'immagine generata.
Se le attivazioni sono simili allora le immagini sono simili e misuro il costo come differenza tra queste.

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

Say you use a hidden layer l to compute content cost

Use pre-trained ConvNet (e.g., VGG)

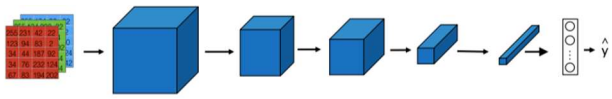
Let $a^{[l](C)}$ and $a^{[l](G)}$ be the activation of layer l on the images

If $a^{[l](C)}$ and $a^{[l](G)}$ are similar, both images have similar content

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

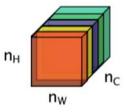
Misurare lo stile di un'immagine

Come prima si prende un livello intermedio per misurare lo stile di un'immagine che viene definito come la correlazione tra le attivazioni sui diversi canali.



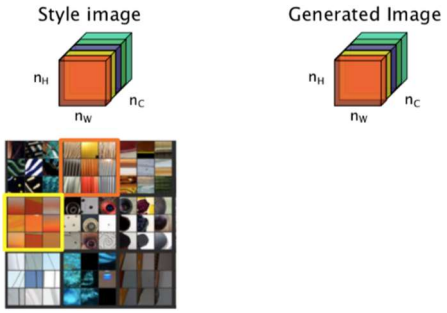
Say you are using layer l's activation to measure "style"

Define style as the correlation between activations across channels



How correlated are the activations across different channels?

Se c'è correlazione positiva tra i canali allora le immagini hanno lo stesso stile.



Style matrix

Si crea la matrice di stile che mi dice quanto sono correlate in profondità le attivazioni di ogni canale.
k è l'indice di profondità che si muove lungo i canali.
L'elemento della matrice è dato dalla somma dei prodotti delle attivazioni. Questo mi dice quanto sono correlate in profondità le attivazioni.

Let $a_{ijk}^{[l]}$ = activation at (i,j,k)
$G^{[l]}$ is $n_C^{[l]} \times n_C^{[l]}$

$$G_{kk'}^{[l]} = \sum_{i=1}^{n_H^{[l]}} \sum_{j=1}^{n_W^{[l]}} a_{ijk}^{[l]} \, a_{ijk'}^{[l]}$$

Concettualmente è una covarianza non normalizzata e la matrice che contiene tali elementi è una matrice di GRAM.
Lo si fa sia per lo stile che per l'immagine generata

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

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

Style cost function component

Si definisce quindi la funzione di costo dello stile come la differenza tra le due matrici di stile. In realtà si prende la norma di Flobenius dove si estraggono le informazioni da più livelli.

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

Defined as (for a layer l)

$$J_{Style}^{[l]}(S,G) = \frac{1}{\dots} \, || \, G^{[l](S)} - G^{[l](G)} \, ||_F^2$$

More layers considered

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

Dando un peso ad ogni livello come parametro.