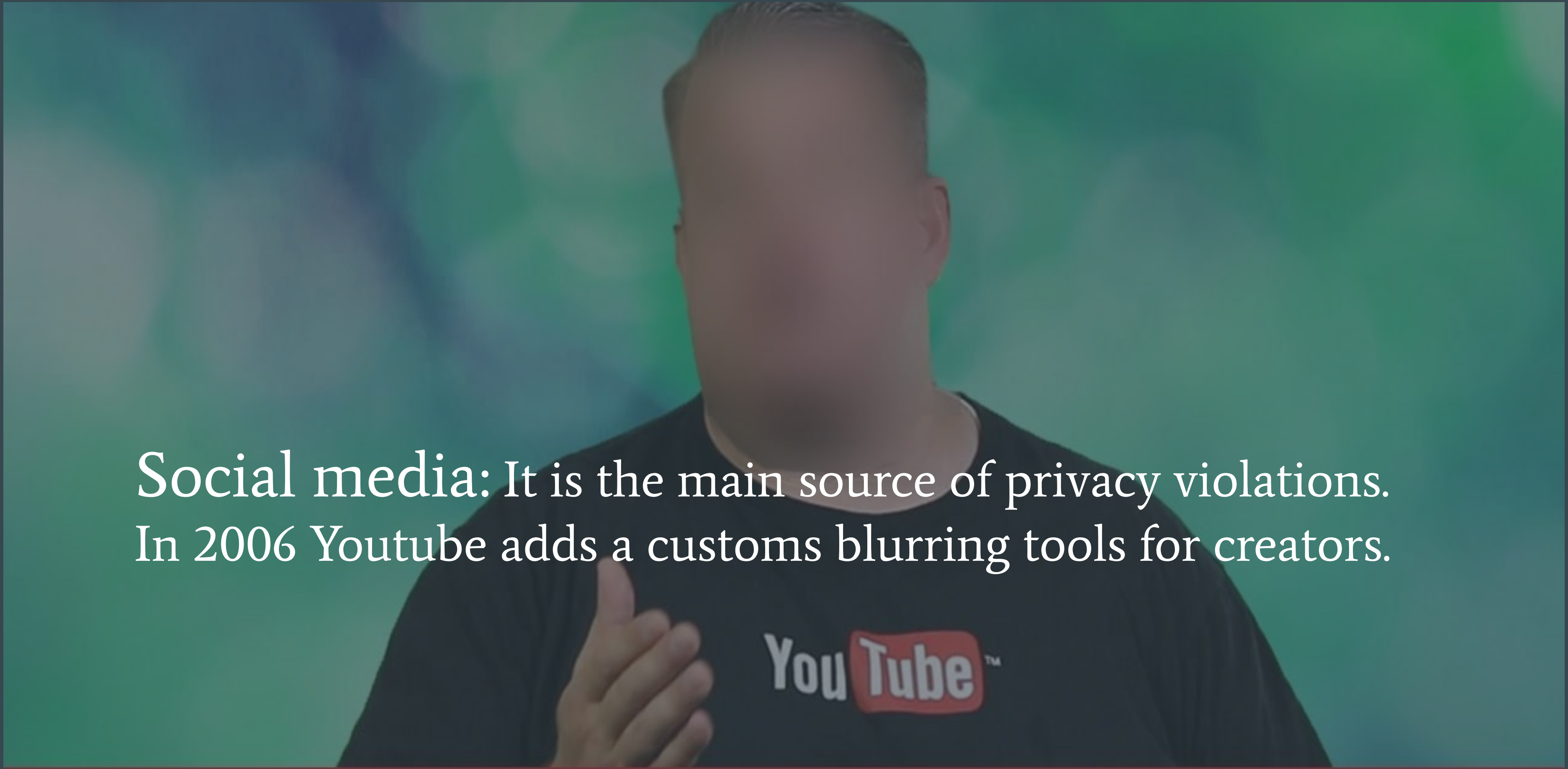
Obfuscated Human Faces Reconstruction

Where?

In photographic media: faces are often obfuscated to protect the identity of those pictured

Where?

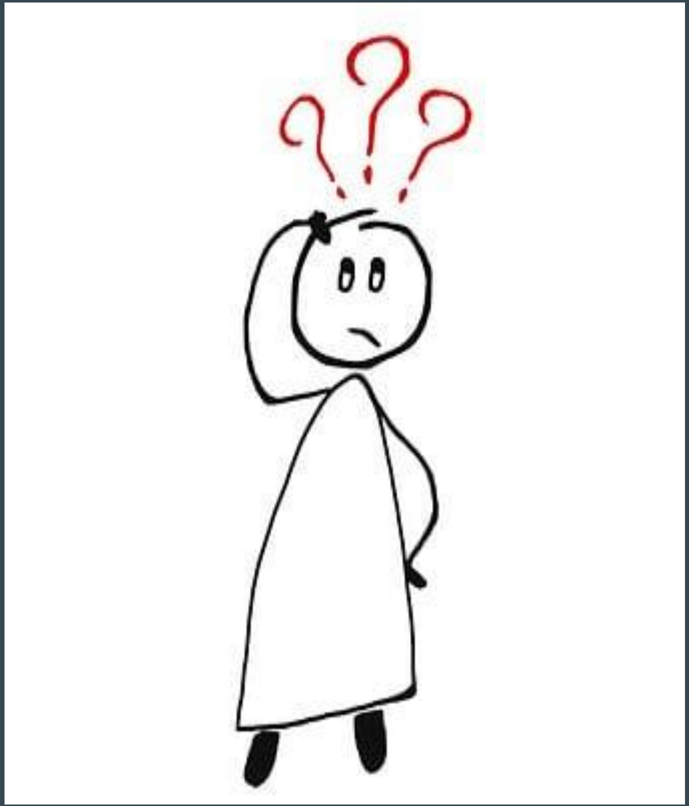


Social media: It is the main source of privacy violations.
In 2006 Youtube adds a customs blurring tools for creators.

Where?

Social networks: Users adopt obfuscations to prevent identity's stealing, facial recognitions.

Is it secure?



Privacy is argued informally, by appealing to the human users' inability to recognize faces and other sensitive objects in the transformed images.

Problem is that humans may no longer be the "gold standard" for extracting information from visual data.

It has been proven that commonly used obfuscation techniques can not stand deep learning architectures.

In our work

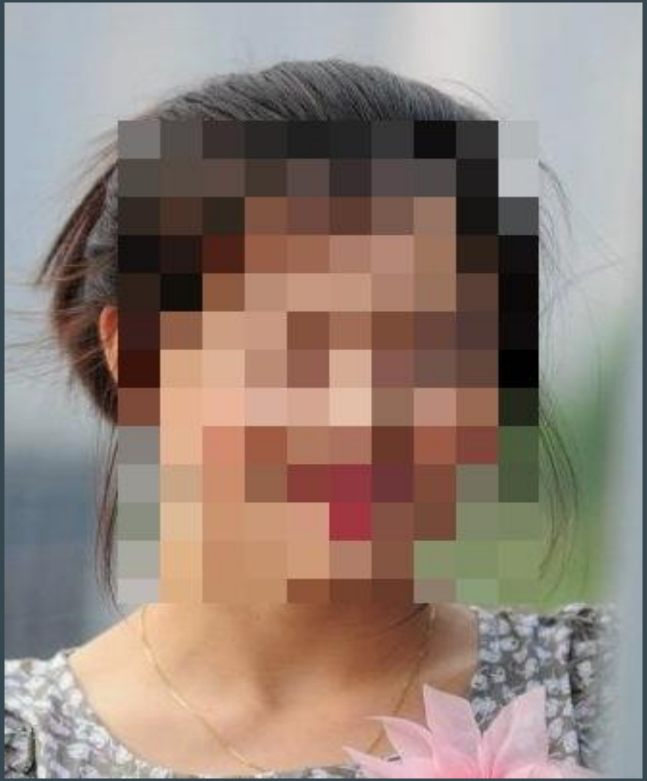
GOAL:

- Build a baseline model for image restoration.
- Investigate the capabilities of two single image super resolution model such as ESPCN and SRGAN for an image restoration task.
- Compare the results obtained from different losses, upscaling factors and filter size.

In our work

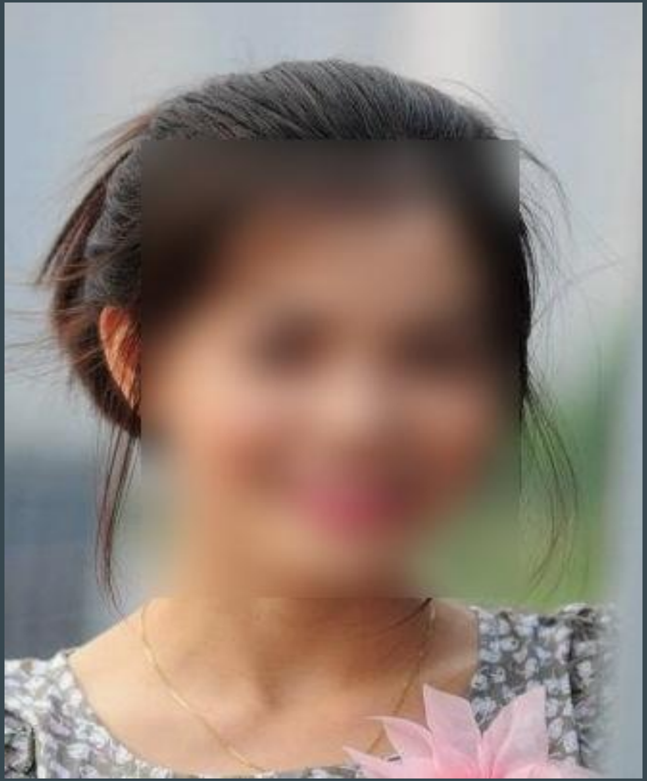
Obfuscation considered

Pixelation



Mean value of boxes of size r

Gaussian Blurr



GaussianBlur(r), window_size= (4*r +0.5)

In our work

Dataset

“Facescrub”:

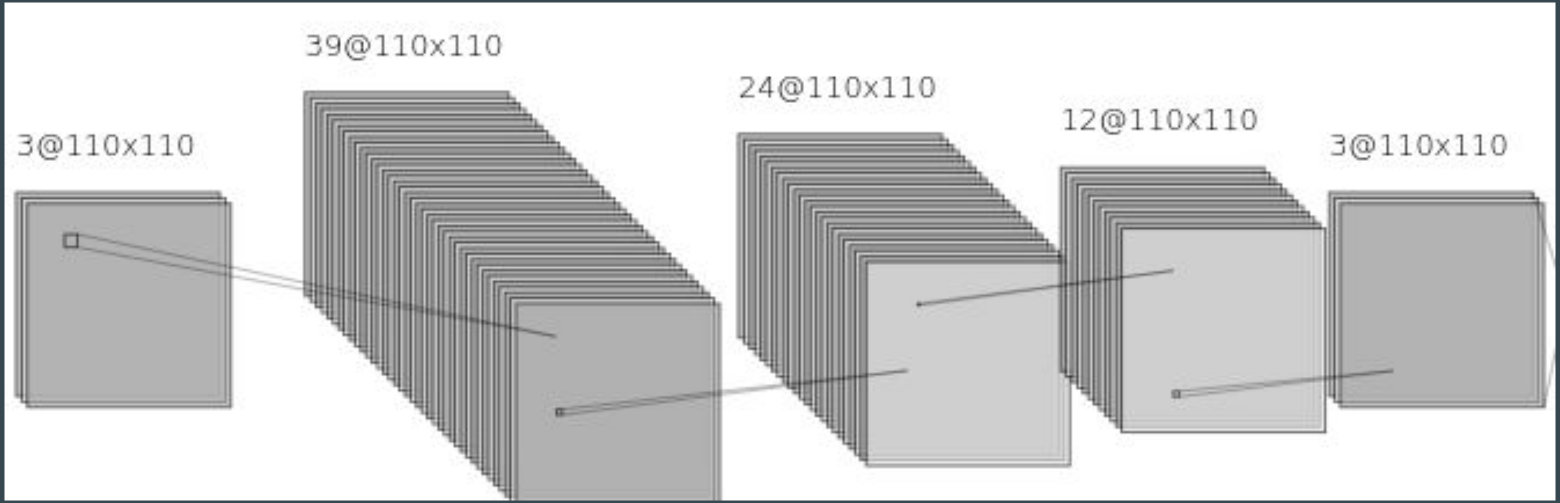
- 106,863 face images
- subsample of 6000 - 4000
- given annotations

“Labeled faces in the wild”:

- 8000-5000
- pretrained Cascade Classifier (OpenCV)
- 110 x 110

Basic model

- **Input:** obfuscated RGB image
- **Kernel sizes:** 7,3,1,3
- **Feature sizes:** 39,24,12,3
- **Padding:** ‘same’
- **Activation:** ReLu
- **Stride:** None



Basic model

Pixel loss: MSE

Pixel wise Mean Square Error (MSE) minimize the distance between the target image and the modelled one.

$$MSE = \frac{1}{n}\frac{1}{m}\sum_{i=1}^{n}\sum_{j=1}^{m}(Y(i,j) - \hat{Y}(i,j))^2$$

Metrics: PSNR and SSIM

- **Peak Signal to Noise Ratio**, measures the quality of a reconstruction in terms of absolute errors.

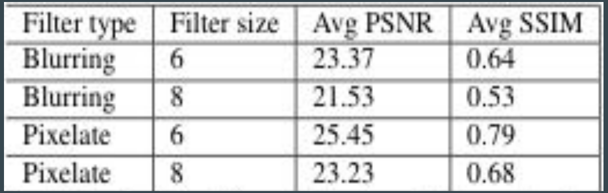
$$PSNR = 20\log_{10}\left(\frac{MAX_f}{\sqrt{MSE}}\right)$$

- Higher is better

- **SSIM** is a perception-based model that considers image degradation as *perceived change in structural information,* (a.k.a the idea that the pixels have strong interdependencies especially when they are spatially close)
- range stays in (-1,1)

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{+}c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

Results

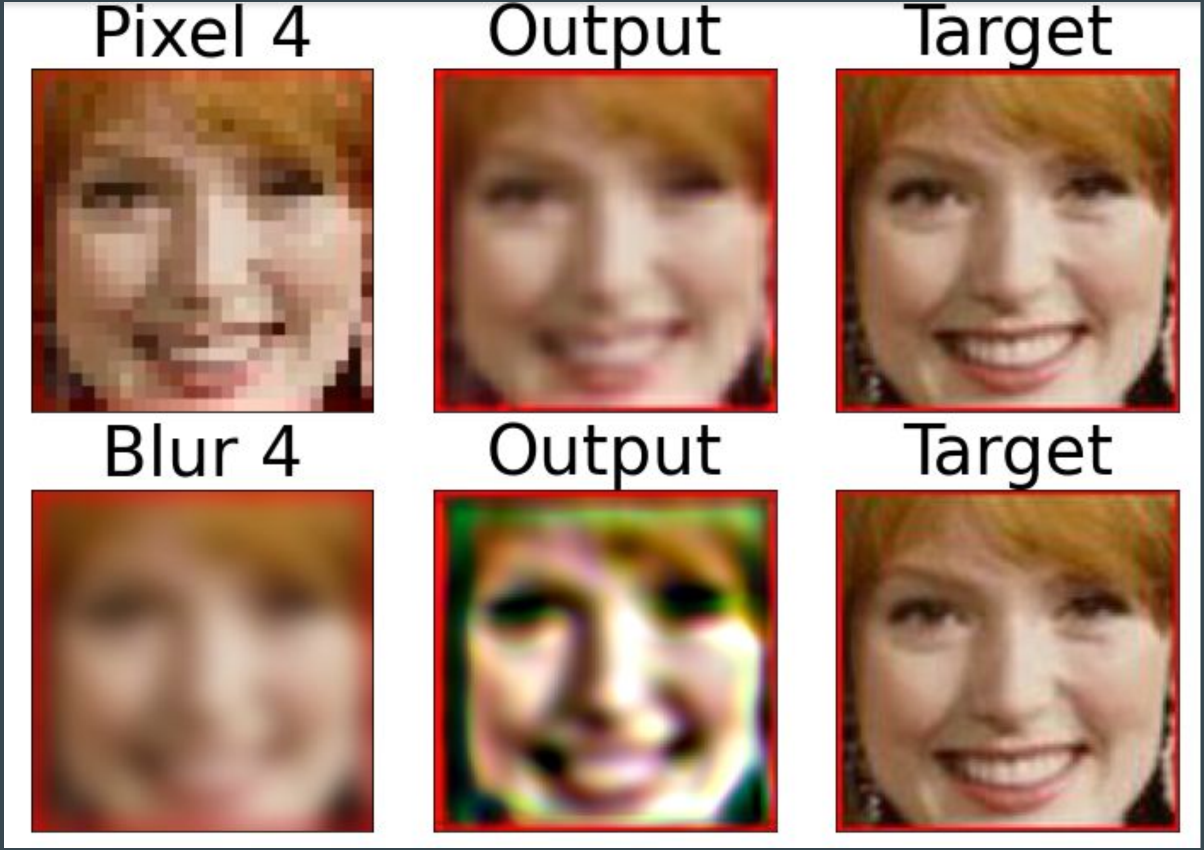


Filter type	Filter size	Avg PSNR	Avg SSIM
Blurring	6	23.37	0.64
Blurring	8	21.53	0.53
Pixelate	6	25.45	0.79
Pixelate	8	23.23	0.68

We train the model with an higher filter size, then we tested the results for smaller one. In real life in fact we do not know the size of the obfuscation.

As we can see, for this simple model we get a result, but it is not good looking.

In the blurred case the SSIM is very low, in fact we can recognize big differences in contrast and luminosity.



Upsampling - Transposed Convolution

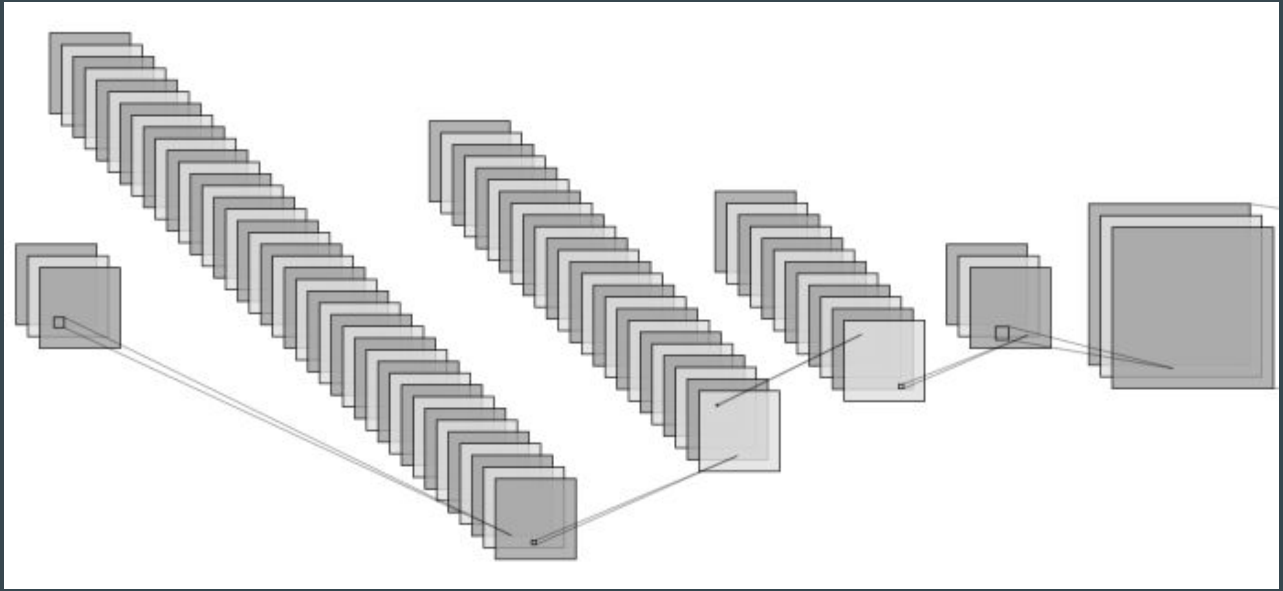
In order to increase the resolution of our final images we introduce a learnable upscaling layer (Transposed convolution) at the end of our model.

The new layer has :

- stride=upscale factor
- kernel size = 9
- padding = 'same'

Consideration:

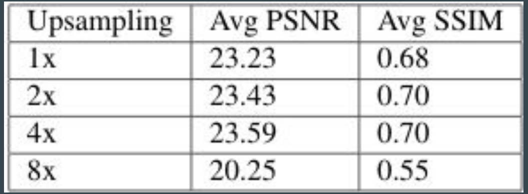
- kernel size has to be bigger than the obfuscation size



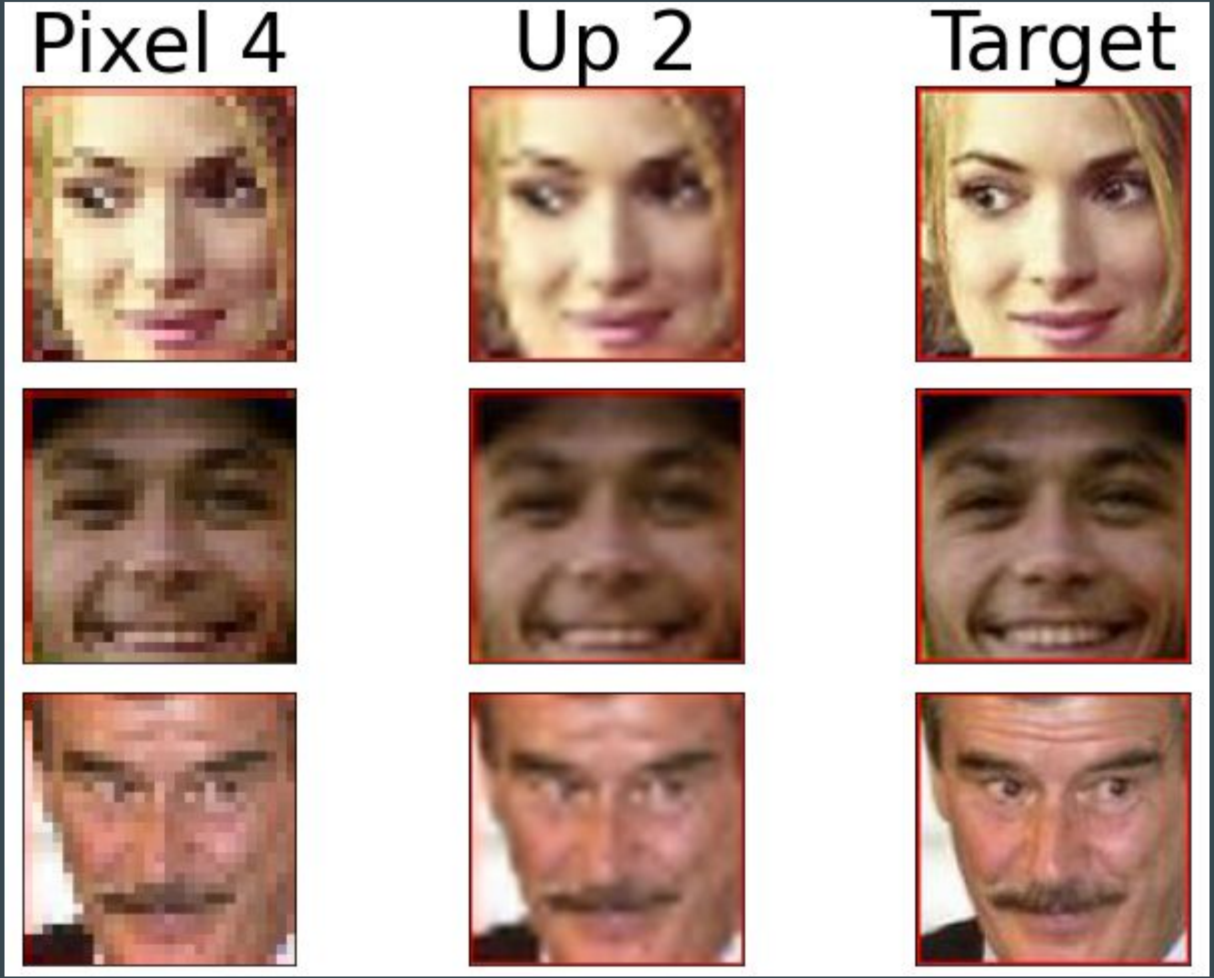
Results

From the images we can see that :

- chequered board effect
- we get better result increasing the upsampling.
- we lose some details



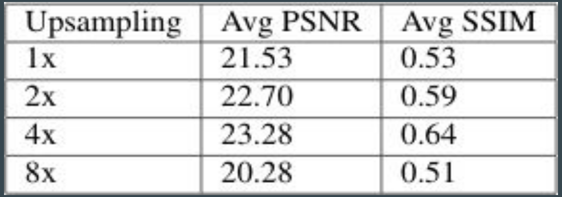
Upsampling	Avg PSNR	Avg SSIM
1x	23.23	0.68
2x	23.43	0.70
4x	23.59	0.70
8x	20.25	0.55



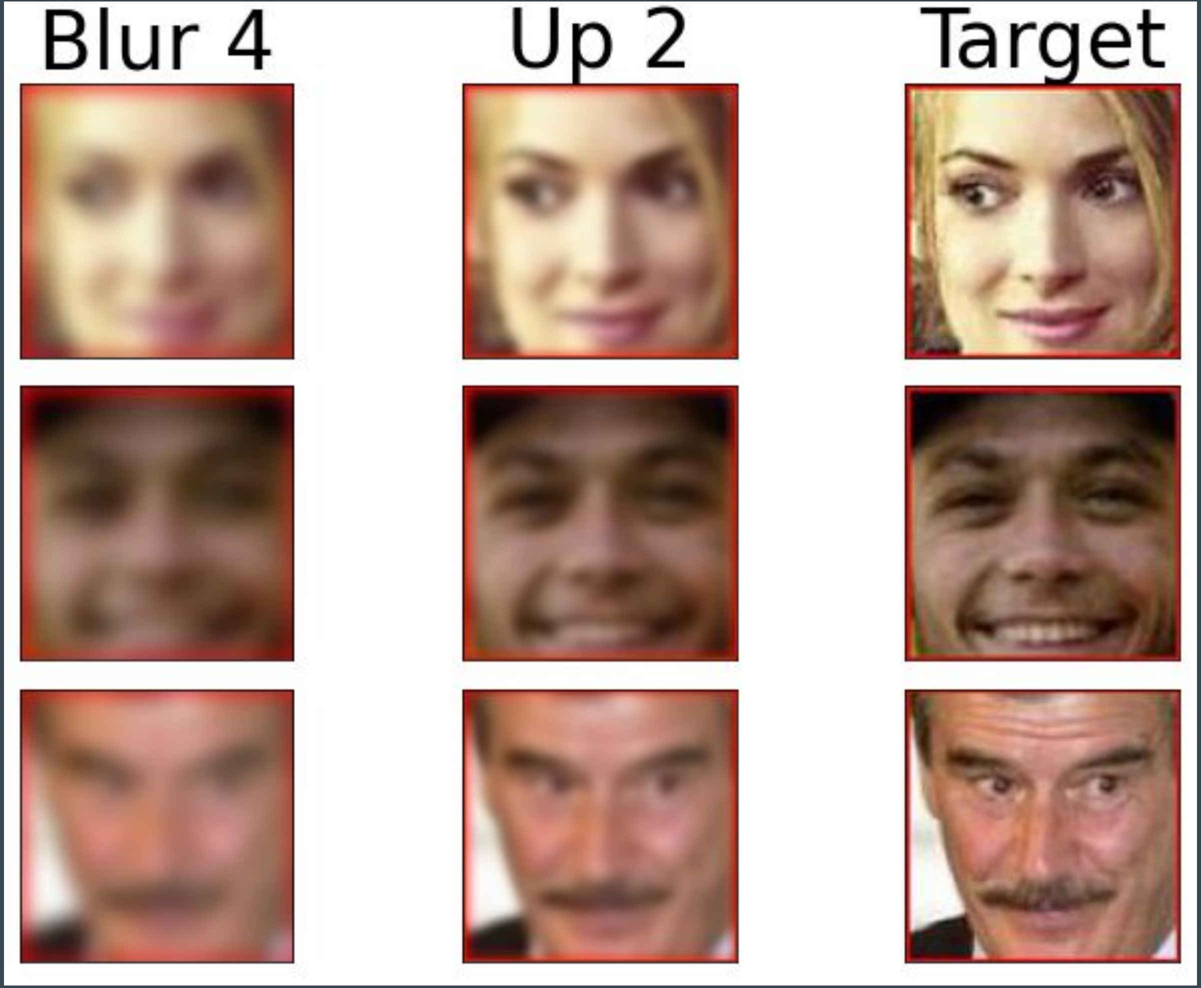
Results

From the images we can see that :

- We get better results than before
- In the reconstruction there could be some errors
- in general pixelation seem to be less effective than blurration.

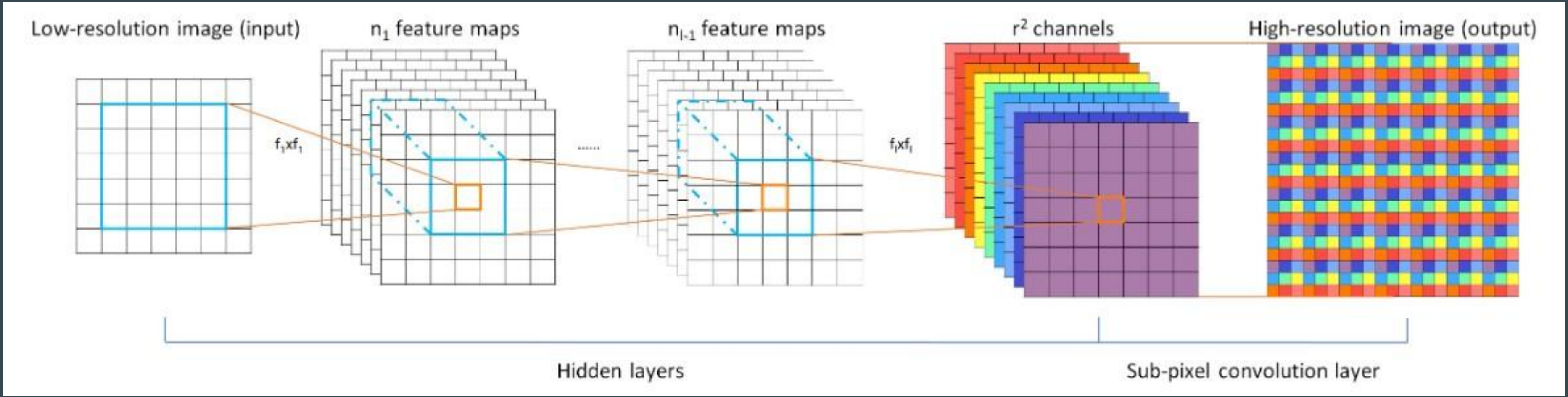


Upsampling	Avg PSNR	Avg SSIM
1x	21.53	0.53
2x	22.70	0.59
4x	23.28	0.64
8x	20.28	0.51



ESPCN model

- Efficient
- Sub-Pixel
- Convolutional
- Neural Network



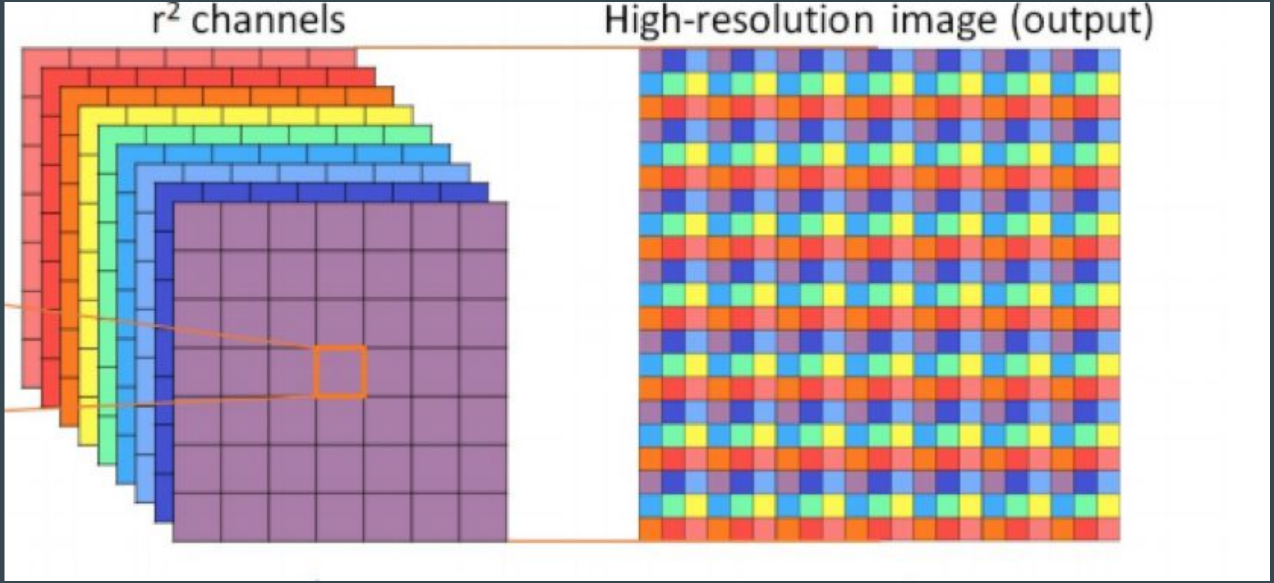
ESPCN model

- Efficient
- Sub-Pixel
- Convolutional
- Neural Network

It works well and doesn’t waste resources thanks to subpixel layer

ESPCN model

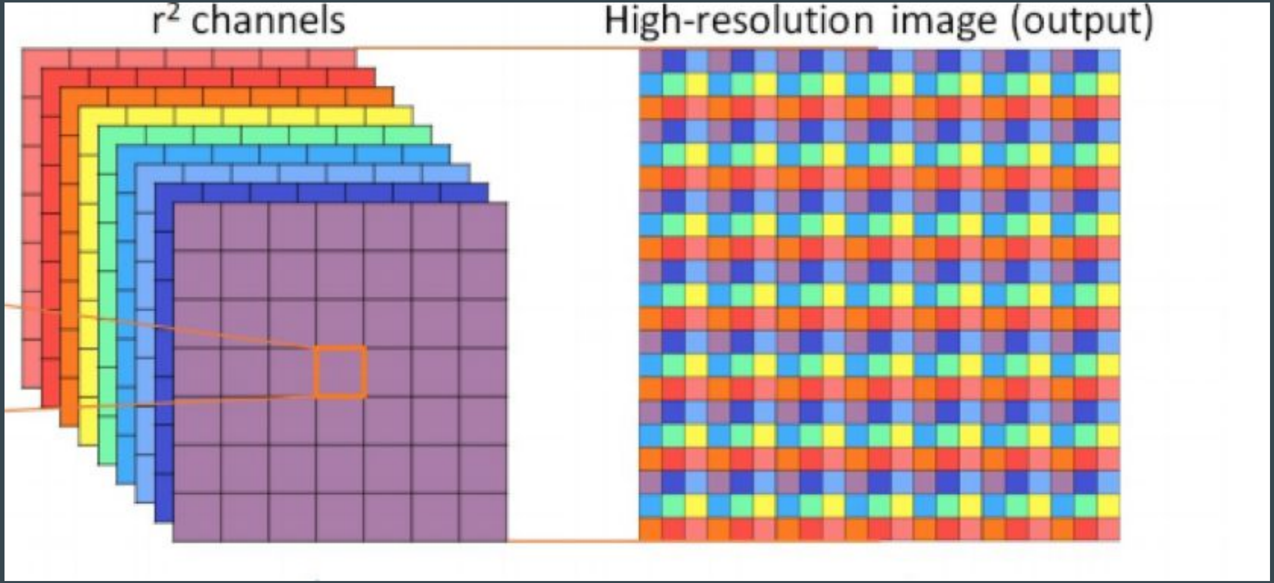
- Efficient
- Sub-Pixel
- Convolutional
- Neural Network



Many Deep learning models implement transposed convolution for upsampling. This technique, with strided convolution gradients, adds zero values to upscale the image and it doesn't have information to backpropagate.

ESPCN model

- Efficient
- Sub-Pixel
- Convolutional
- Neural Network



ESPCN model uses a subpixel CNN layer for upscaling: regular convolutional layers are followed by a specific type of image reshaping called **phase shift**. Instead of putting zeros in between pixels, we calculate more convolutions in lower resolution and resize the resulting map into an upscaled image.

ESPCN model - loss

We used two kinds of loss:

- Mean Squared Error (MSE) loss, or L2 norm
- VGG loss + MSE loss. In this case, the losses are simply added together after multiplying the VGG loss by 10^{-3}. The perceptual loss is computed on the 35th layer of the pretrained VGG-19 model from Pytorch library.

After having observed the first results, we decide to focalize more on using VGG + MSE loss.

ESPCN model - VGG loss

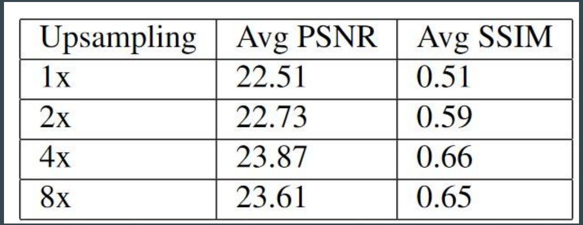
VGG loss is one type of **Perceptual loss**. It is very common in style transfer and super-resolution. In our experiments, we used only one layer from a pretrained VGG-19 model.

We can define the VGG loss as the **euclidean distance** between the **feature representations** of the **reconstructed** image and the **reference** image. In formulas:

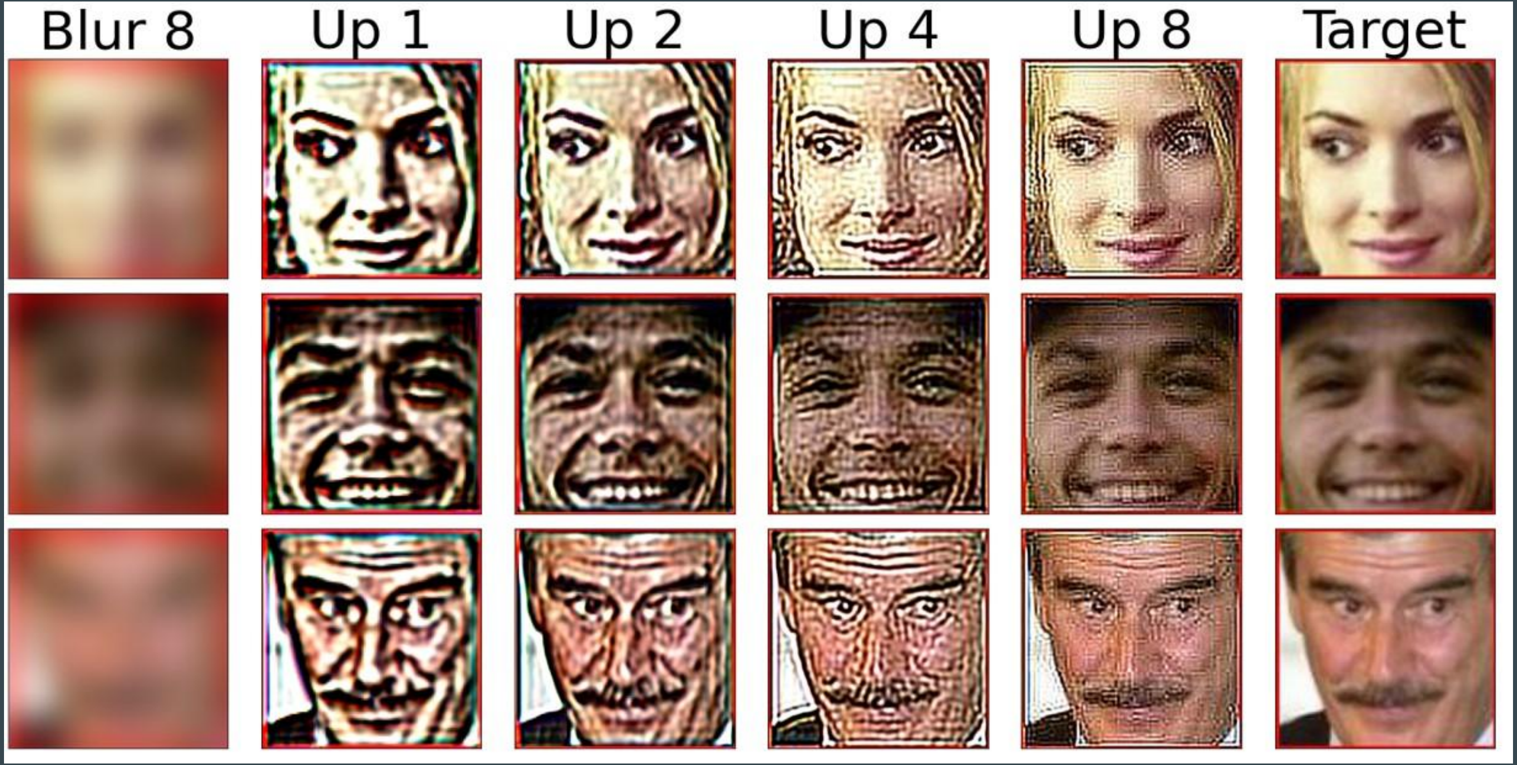
$$\mathcal{L}_{VGG/i,j} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}\left(I^{HR}\right)_{x,y} \\ - \phi_{i,j}\left(G_{\theta_G}\left(I^{LR}\right)\right)_{x,y})^2$$

- $\phi_{i,j}$ is the feature map obtained by the $j-th$ convolution after the activation, before the $i-th$ maxpooling layer within the network
- $G_{\theta_G}\left(I^{LR}\right)$ is the reconstructed image
- I^{HR} is the reference image
- $W_{i,j}$ and $H_{i,j}$ describe the dimensions of the respective feature maps within the VGG network

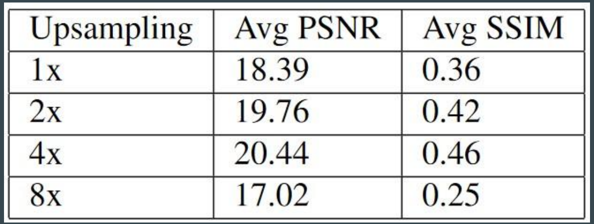
ESPCN model - results: Gaussian blur, size 8



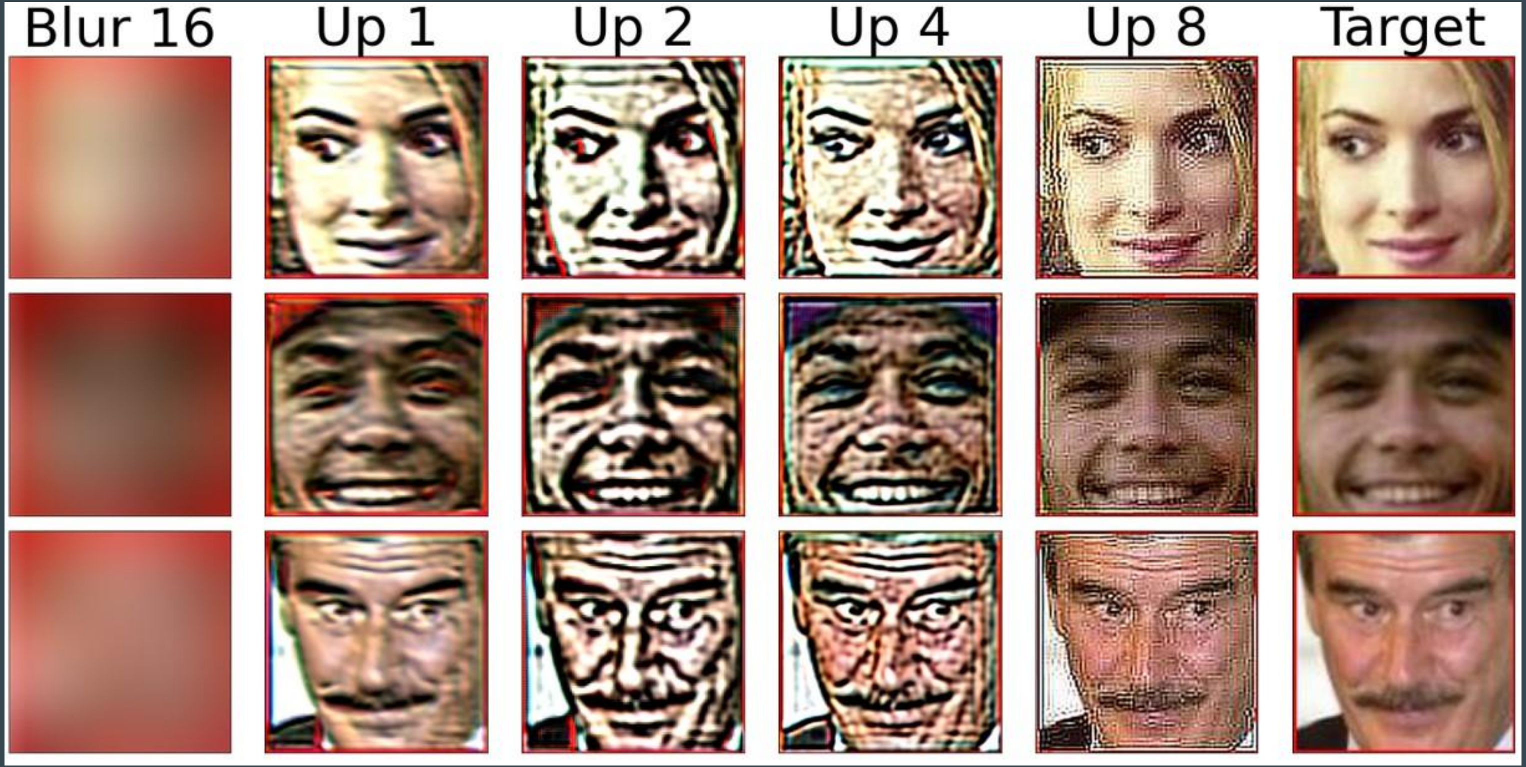
Upsampling	Avg PSNR	Avg SSIM
1x	22.51	0.51
2x	22.73	0.59
4x	23.87	0.66
8x	23.61	0.65



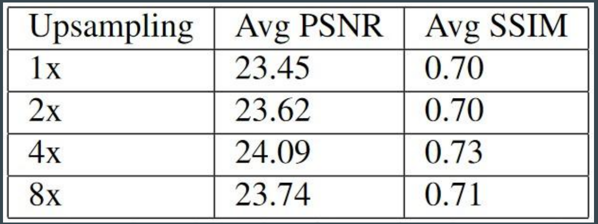
ESPCN model - results: Gaussian blur, size 16



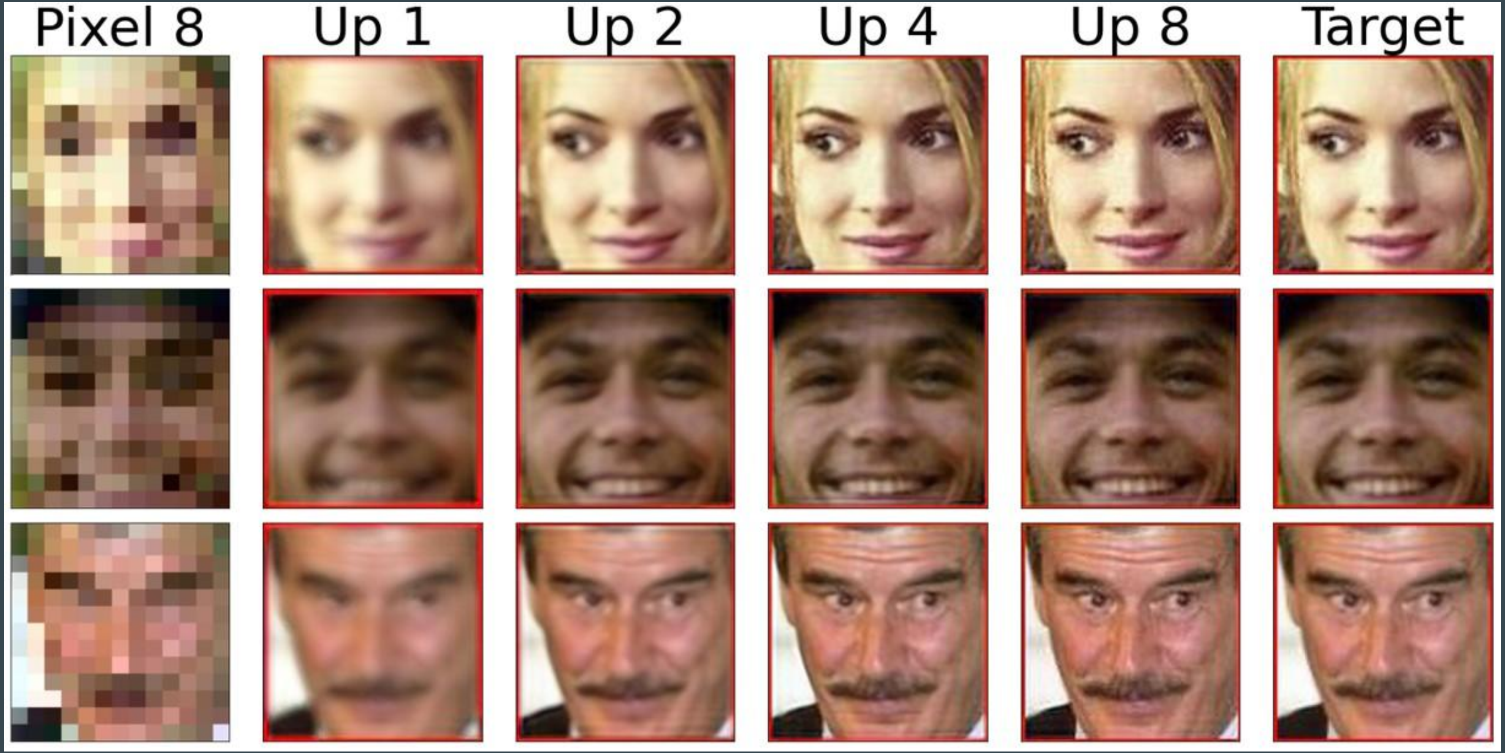
Upsampling	Avg PSNR	Avg SSIM
1x	18.39	0.36
2x	19.76	0.42
4x	20.44	0.46
8x	17.02	0.25



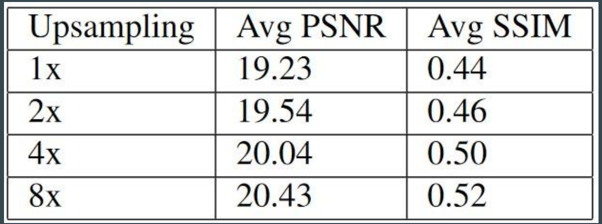
ESPCN model - results: Pixelation, size 8



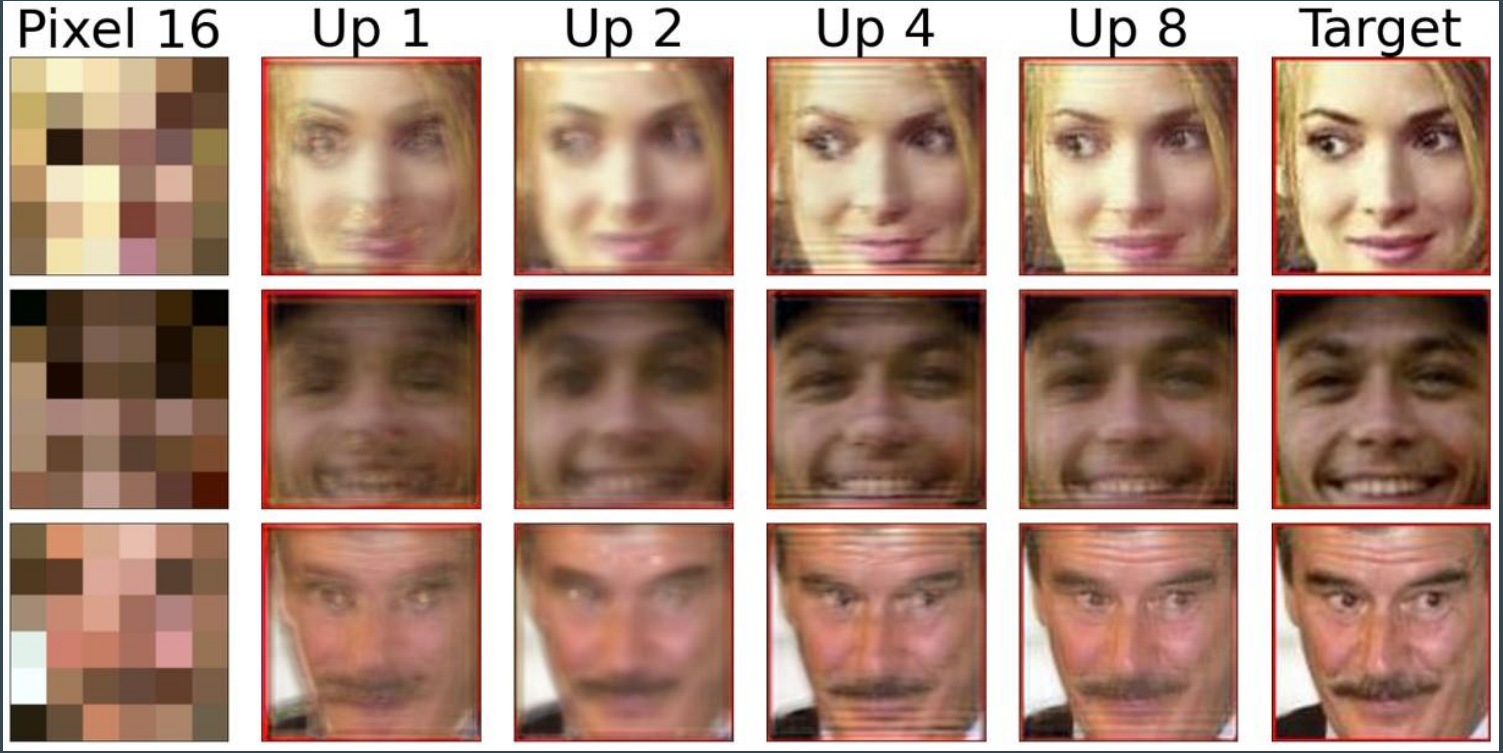
Upsampling	Avg PSNR	Avg SSIM
1x	23.45	0.70
2x	23.62	0.70
4x	24.09	0.73
8x	23.74	0.71



ESPCN model - results: Pixelation, size 16

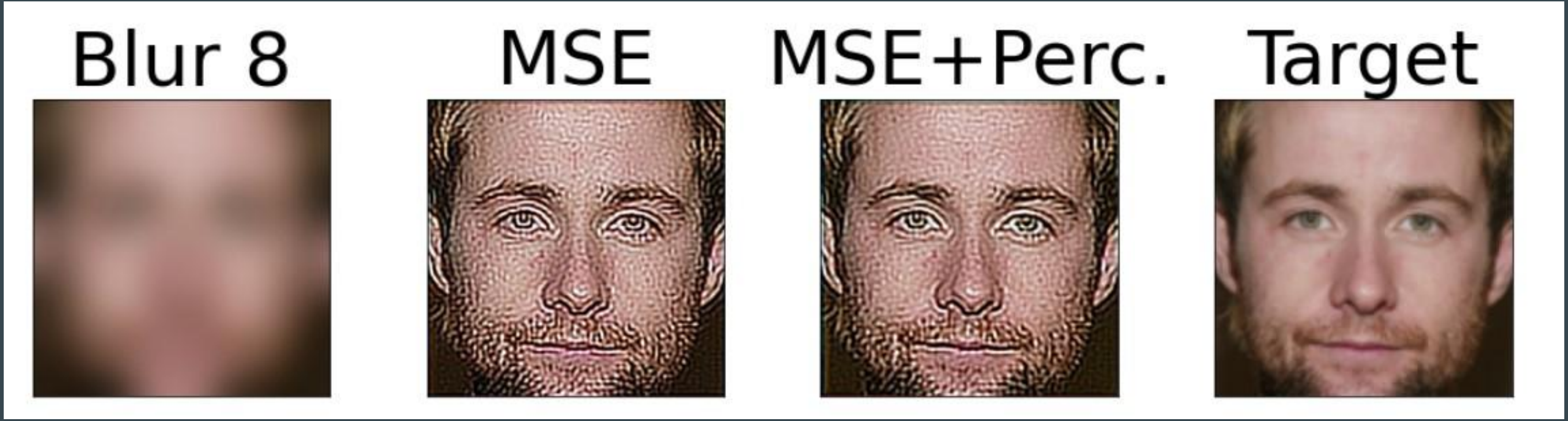


Upsampling	Avg PSNR	Avg SSIM
1x	19.23	0.44
2x	19.54	0.46
4x	20.04	0.50
8x	20.43	0.52



ESPCN model - Comparison between MSE and VGG + MSE loss

● MSE:	PSNR = 21.10,	SSIM = 0.54
● VGG + MSE:	PSNR = 23.61,	SSIM = 0.65



ESPCN model - Observations

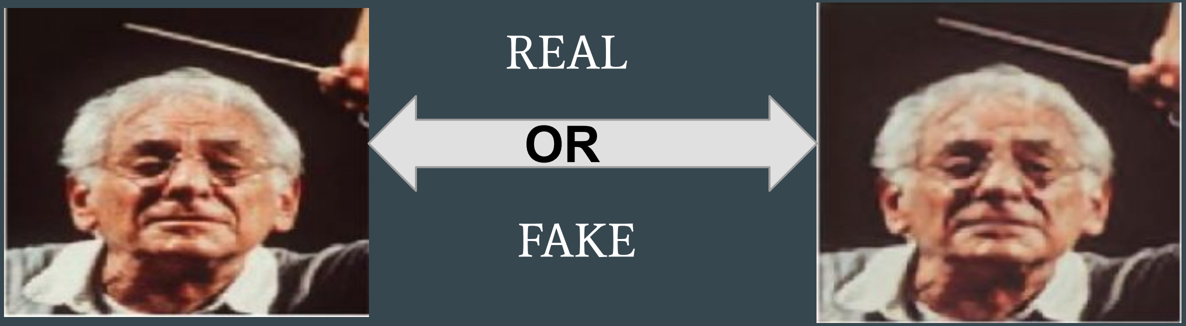
- ➔ The choice of loss affects the training, but the most important parameter seems to be the number of **upsamplings**.
- ➔ For a fixed size, results starting from **pixelated** images are **better** than results obtained from blurred ones.
- ➔ The coefficient for perceptual loss needs to be **very low**, because it is typically much higher than MSE loss. Using a weight larger than 10^{-3} for the perceptual loss, or not using MSE loss at all, doesn't give good results, because the model trains too much on structural features and disregards important properties like the color tonality of the face.
- ➔ Training with perceptual loss **doubles** the time needed. Seeing the results, it may not be convenient.

SRGAN MODEL

- **S**uper **R**esolution **G**enerative **A**dversarial **N**etwork is the third model proved.
- It's an upgrade of **GAN** model where a discriminator D and a generator G play a min-max game following different and opposites aims :

  - G tries to produce "fake" images such that D isn't able to recognize and distinguish it from "real" one.

  - D,on the contrary,tries not to be fooled by G



loss functions

- *Content loss* :

$$l^{SR}_{MSE} = \frac{1}{r^2WH}\sum_{x=1}^{rW}\sum_{y=1}^{rH}\left(I^{HR}_{x,y} - G_{\theta_G}(I^{LR})_{x,y}\right)^2$$

- where W and H in this case stand for width and height of images during training procedure, *r* is a downsampling factor, I^{HR} and I^{LR} are respectively high and low resolution images.
- θ_G that takes into account weights and biases of L-layer

Adversarial loss :

$$l_{Gen}^{SR} = \sum_{n=1}^{N} -\log D_{\theta_D}(G_{\theta_G}(I^{LR}))$$

- where $D_{\theta_D}(G_{\theta_G}(I^{LR}))$ is the probability that the reconstructed image $G_{\theta_G}(I^{LR})$ is a natural HR image.

Adding together this two loss function we can define the *perceptual loss* :

$$l^{SR} = \underbrace{l_X^{SR}}_{\text{content loss}} + \underbrace{10^{-3} l_{Gen}^{SR}}_{\text{adversarial loss}}$$

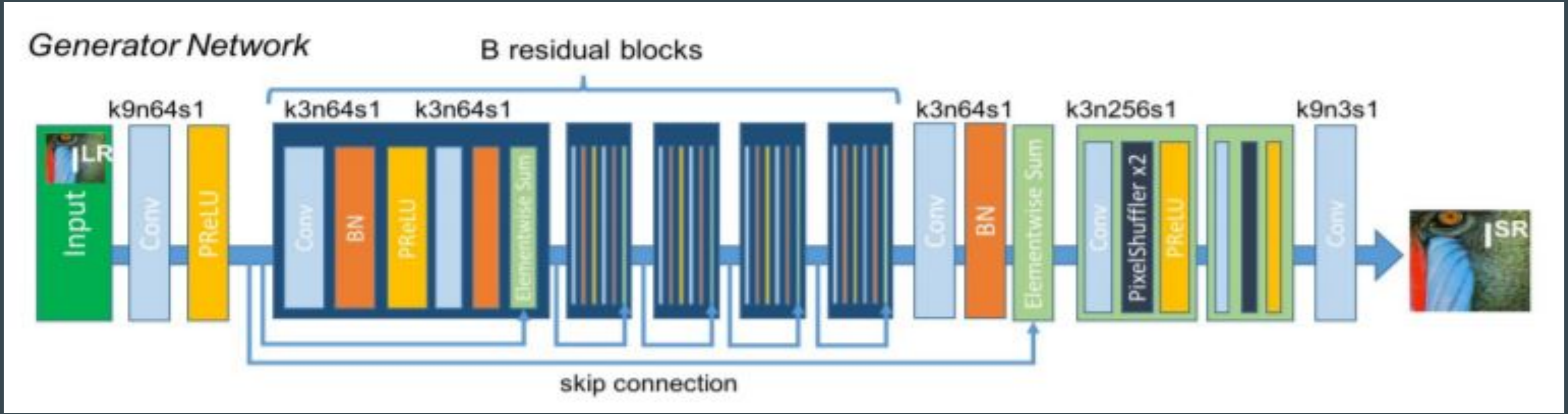
MIN-MAX GAME

- This min-max game in SRGAN's architecture is expressed as :

$$\min_{\theta_G} \max_{\theta_D} E_{I^{HR} \sim p_{train(I^{HR})}} [log D_{\theta_D}(I^{HR})] + E_{I^{LR} \sim p_{G(I^{LR})}} [log(1 - D_{\theta_D}(G_{\theta_G}(I^{LR}))]$$

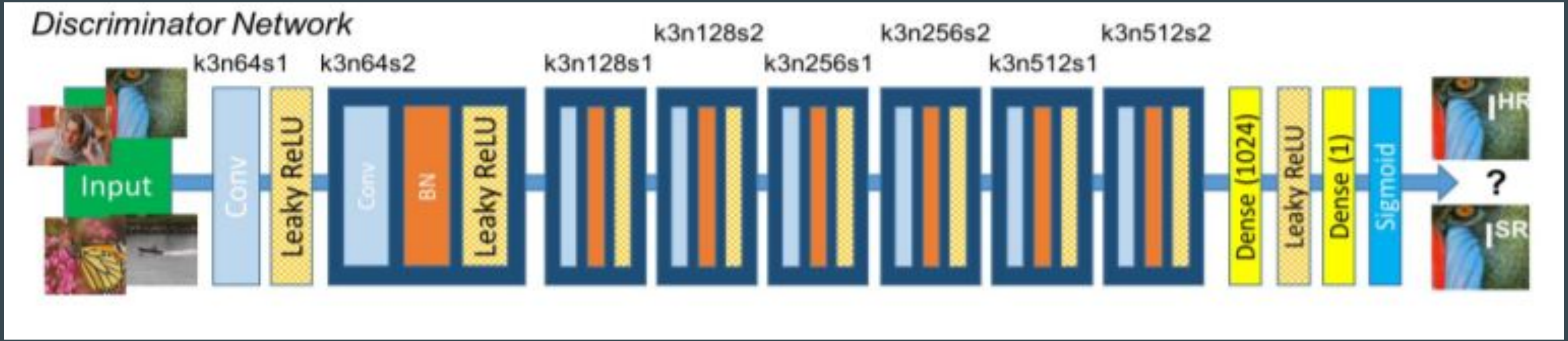
- E is the expected value ,p_{train} and $p_G(I^{LR})$ are respectively the distribution of train set and the distribution of reconstructed images

model architecture: generator



- It starts with LR images obtained downsampling HR ones
- At the core we find B residual blocks with identical layout :
  - **P**arametric**ReLU** :which instead uses a scaled tanh to ensure that the output image has pixels in the range [0, 255]
  - **B**atch **N**ormalization : useful to counteract the internal covariate shift in deep learning

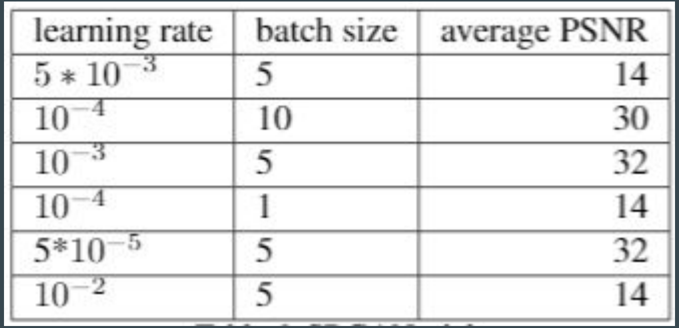
Model architecture : Discriminator



- HR images are passed as input to discriminator network such that it is able to distinguish them to SR ones.
- Less layers in architecture than generator one.

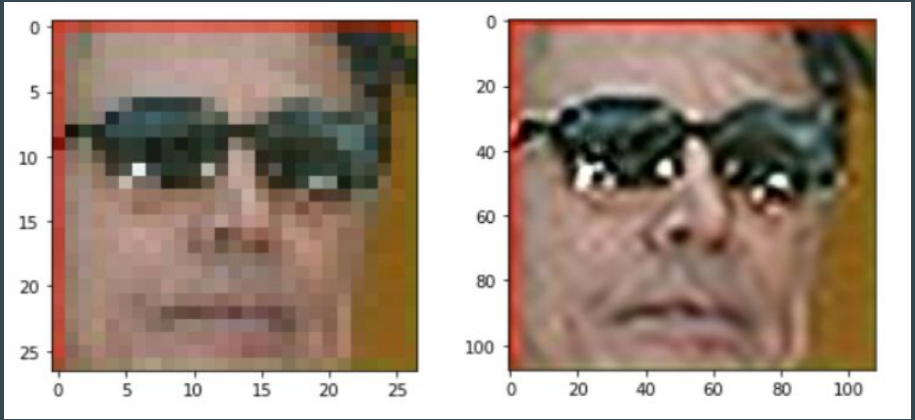
RESULTS

- Different experiments are also done on two datasets : the best is for 'labeled face in the wild' using an upscaling x4 and it has brought to PSNR ~ 33 dB and SSIM~ 0.956 with lr = 0.001 and batch size = 2.

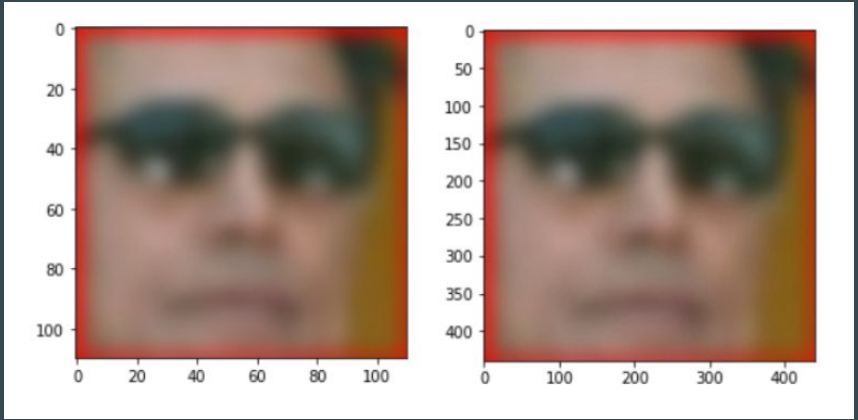


learning rate	batch size	average PSNR
$5 * 10^{-3}$	5	14
10^{-4}	10	30
10^{-3}	5	32
10^{-4}	1	14
$5*10^{-5}$	5	32
10^{-2}	5	14

PIXELATE



BLURRED



TARGET

Note and observations:

- During the training procedure was applied a sort of “early stopping” based on the PSNR and SSIM’s behaviour : if after 5 epochs at least one of them didn’t increase we stopped the training procedure and save the parameter of the model.

- Using blurred images or pixelate ones the capability of the network to reconstruction images does not improve.

- about ”Facesgrub” dataset : the high-quality of the images was not be very useful to improve SRGAN model results.

CONCLUSION

- In this project we explored the potentialities offered by deep learning to face a common problem in computer vision: images reconstruction.
- Starting from uncorrupted images we trained three different models, simple CNN,ESPCN and SRGAN, obtaining different results
- These results depend both by starting images with different resolutions and by task fixed as aim: unblurration or unpixelation.

- For both of them we can state that ESPCN is better than SRGAN in reconstruction task: improving the ”upsample” from 1 to 8, very blurred/pixelated images recover their original form

- ESPCN and CNN have very similar results

- This is true even if PSNR and SSIM’s value are not excellent if compared with results obtained training SRGAN: that is another proof that shows how these parameters don’t affect uniquely the results

- Indeed this model is able to reconstruct quite fine the starting images when those are pixelated but it does almost nothing when input ones are blurred. This is true for both datasets without difference if the input images are LR or HR

PROSPECTIVES

- A possible improvement could be using HD dataset with a larger resize photos as inputs: using Colab GPU we have limited use of time and size and so we are forced to limit our trials both for the number of photos in training/test and also their shape

- In this hypothesis SRGAN could improve its performance because it is a more complex architecture than ESPCN and so more weights must be properly setted

- We would like to notice that in this work we do not use special attention loss able to consider the facial structures, it would be another possible development of our study.