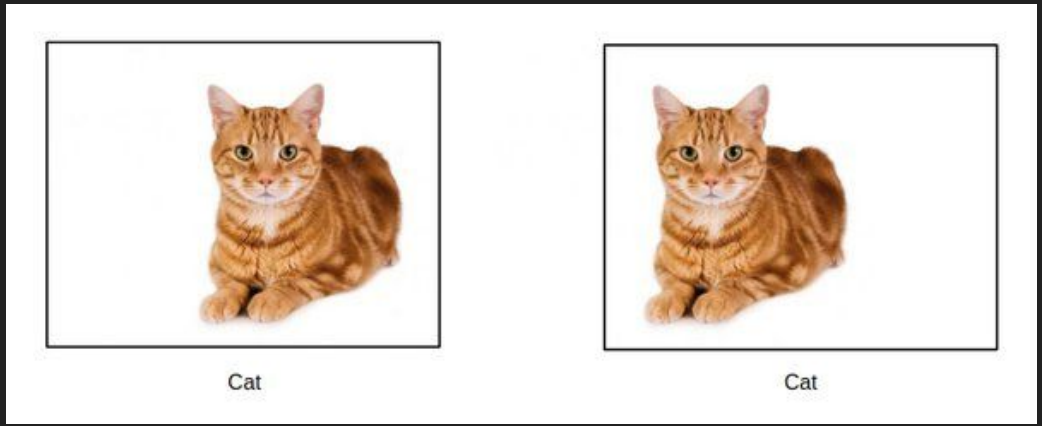
L12

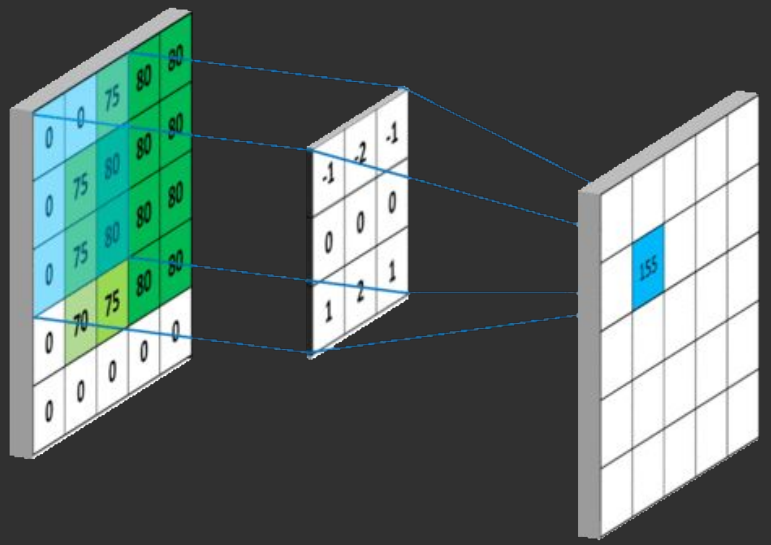
Computer Vision. Convolutions

MLP for Images

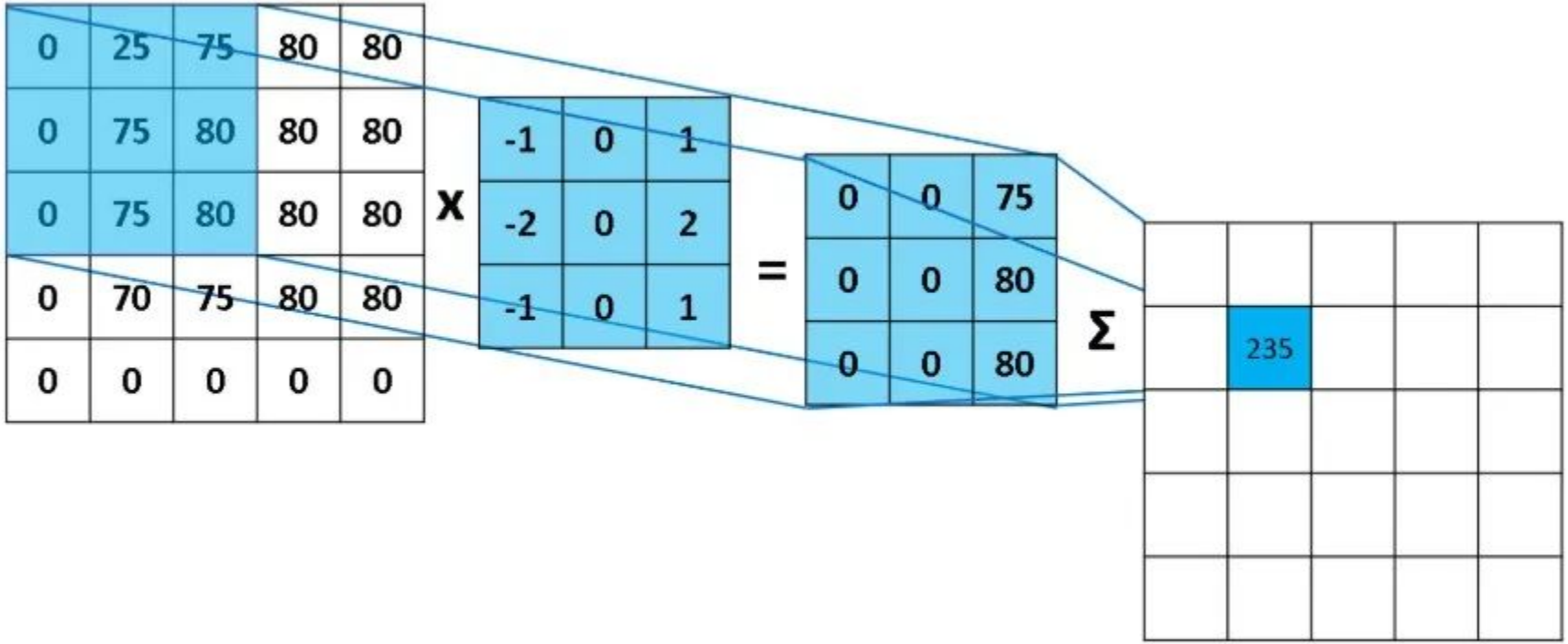
- Can approximate any function (universal approximation theorem)
- Doesn’t have local translation invariance, require a lot of data to handle that and a lot of computational resources
- Doesn’t care about local neighborhood of a pixel (for example, grass around dog, or sky around tree)



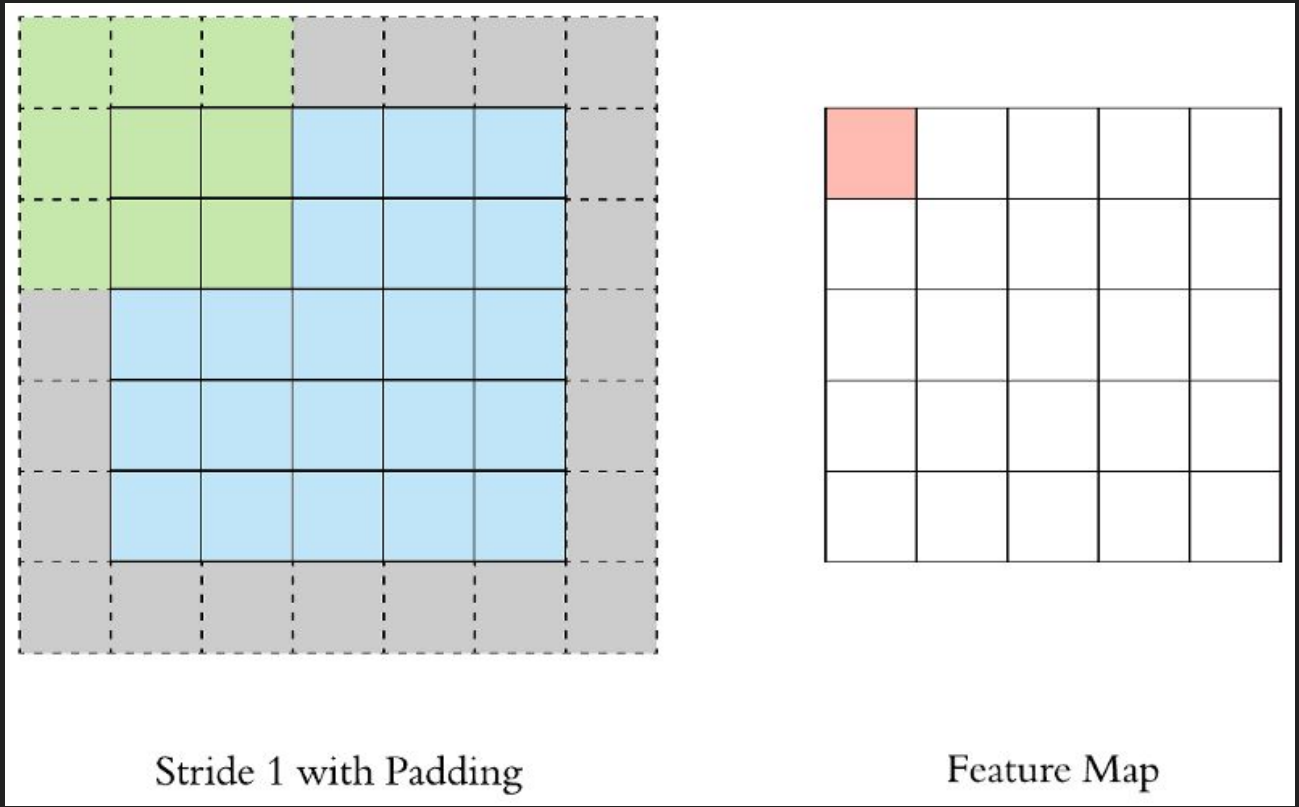
Convolution



Convolution

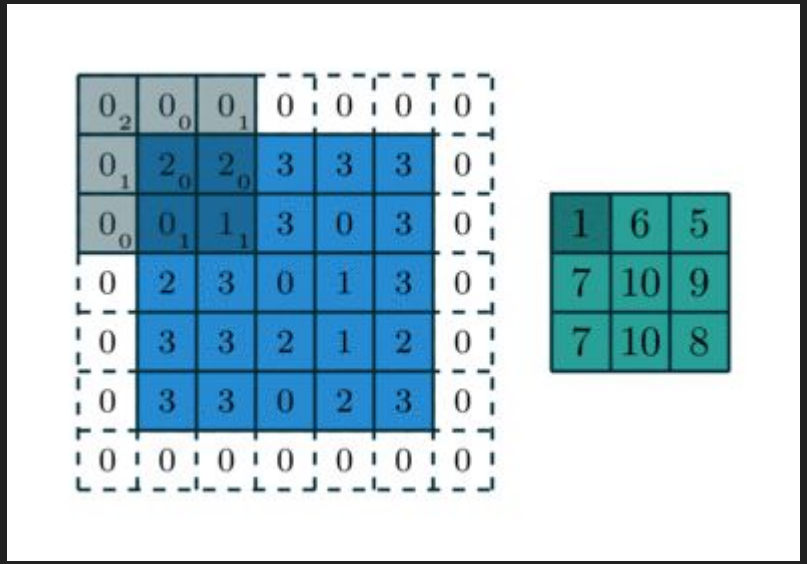


Convolution + Padding

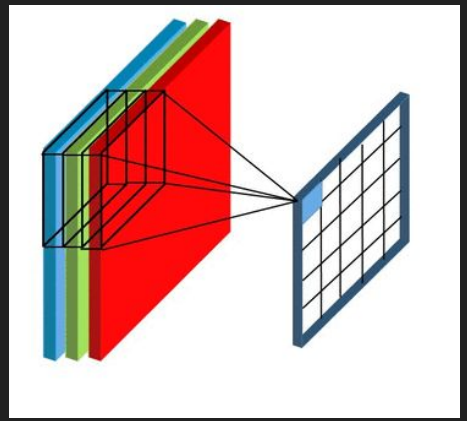


Convolution + Padding + Stride

Kernel: 3x3
Stride: 2
Padding: 1

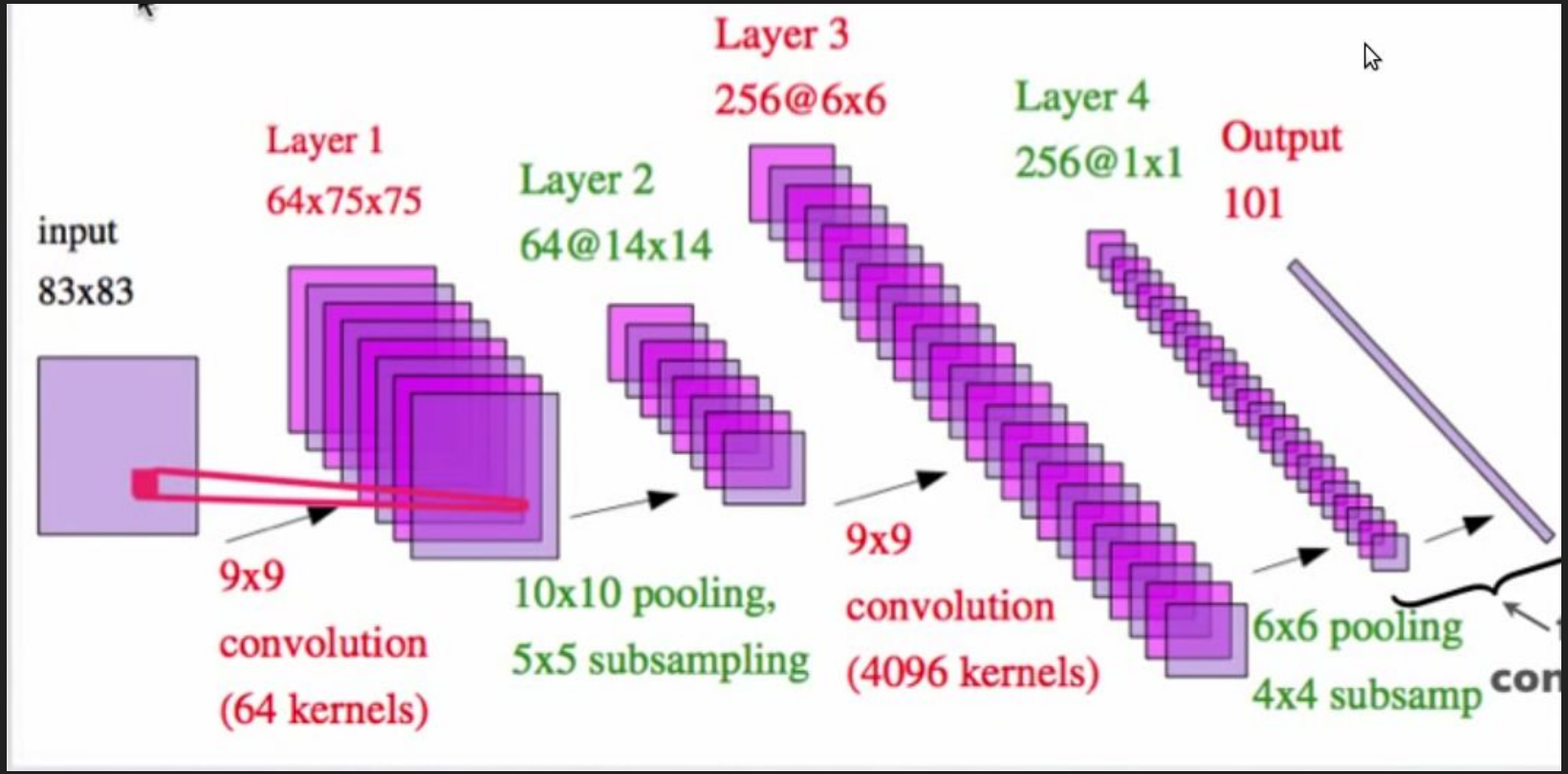


Convolution for each channel

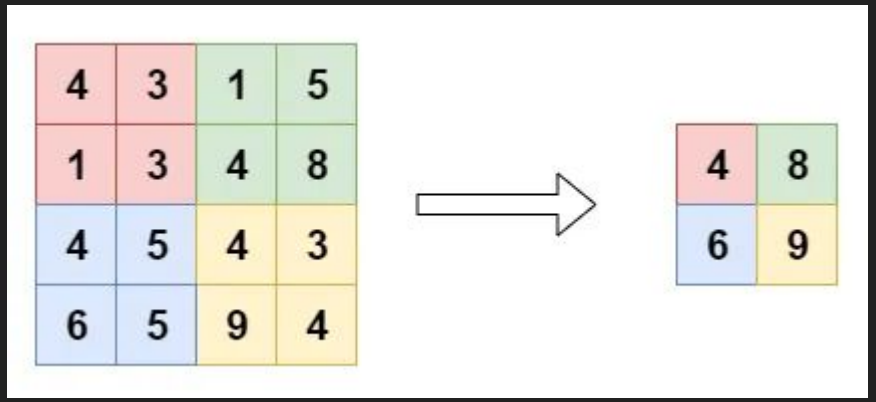


	Input					Kernel			Intermediate Output			Output		
	1	0	1	0	2									
	1	1	3	2	1	0	1	0	7	5	3			
	1	1	0	1	1	0	0	2	4	7	5			
	2	3	2	1	3	0	1	0	7	2	8			
	0	2	0	1	0									
	1	0	0	1	0									
	2	0	1	2	0	2	1	0	5	3	10	19	13	15
	3	1	1	3	0	0	0	0	13	1	13	28	16	20
	0	3	0	3	2	0	3	0	7	12	11	23	18	25
	1	0	3	2	1									
	2	0	1	2	1									
	3	3	1	3	2	1	0	0	7	5	2			
	2	1	1	1	0	1	0	0	11	8	2			
	3	1	3	2	0	0	0	2	9	4	6			
	1	1	2	1	1									

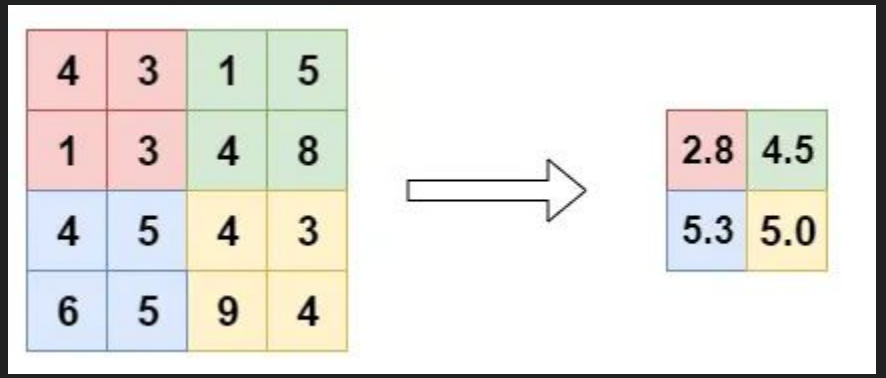
#Filters (kernels)



MaxPool, AvgPool



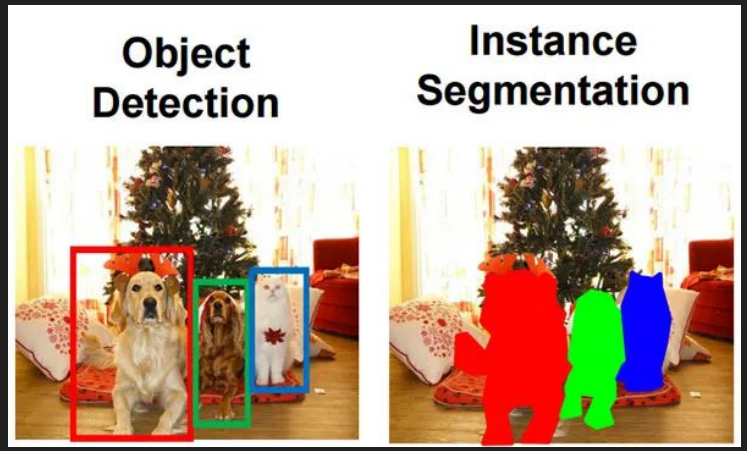
Kernel: 2x2
Stride: 1
Padding: 0



CV Tasks

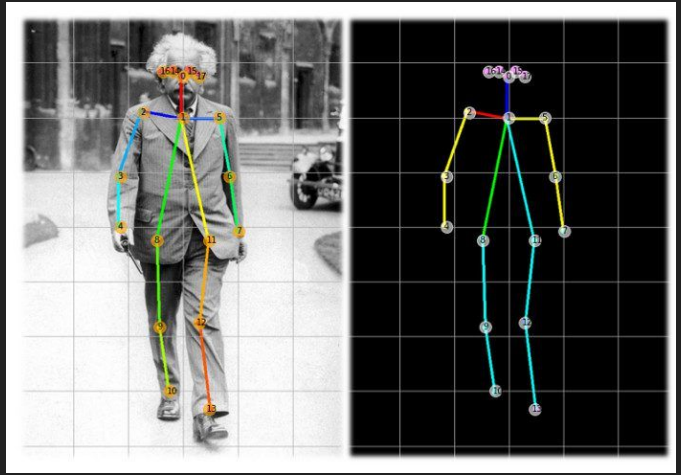
- Object detection
- Inpainting
- Segmentation

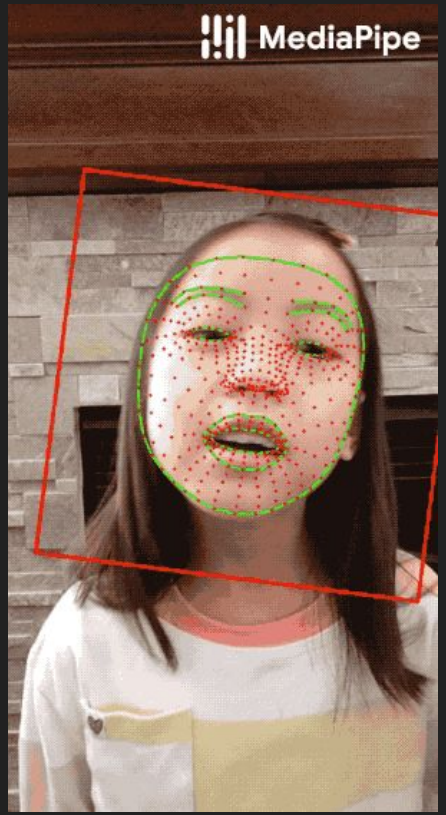
https://huggingface.co/tasks



CV Tasks

- Superresolution
- Pose estimation
- Face recognition and authentication





Pre-Convolution Era

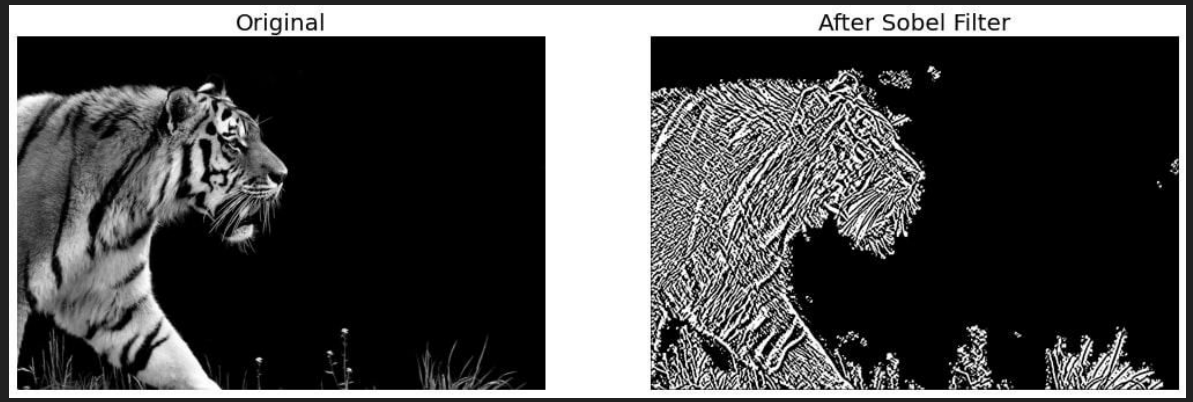
Sobel Edge Detection:

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

X-Direction Kernel

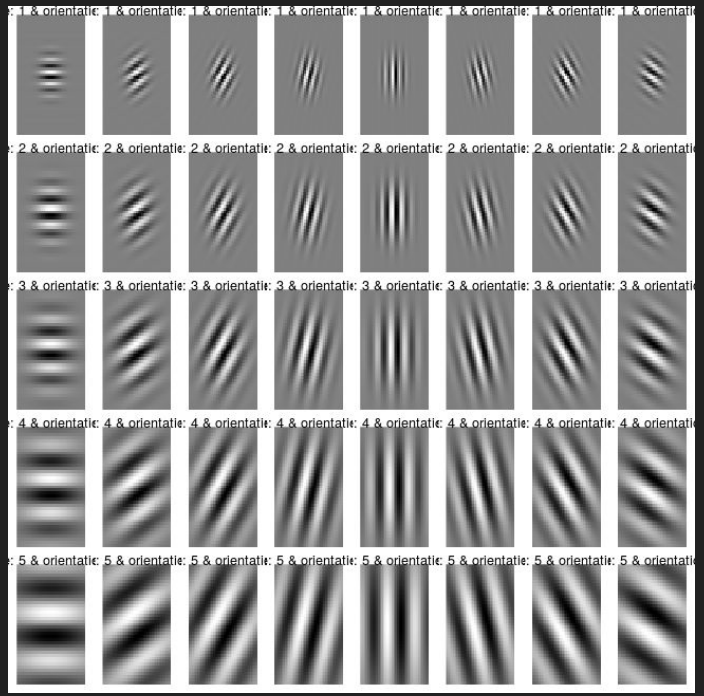
$$\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

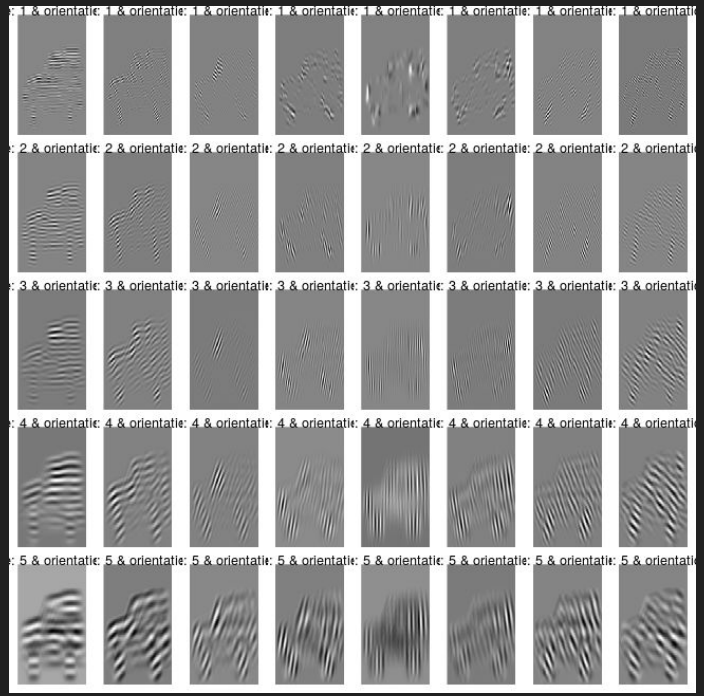
Y-Direction Kernel



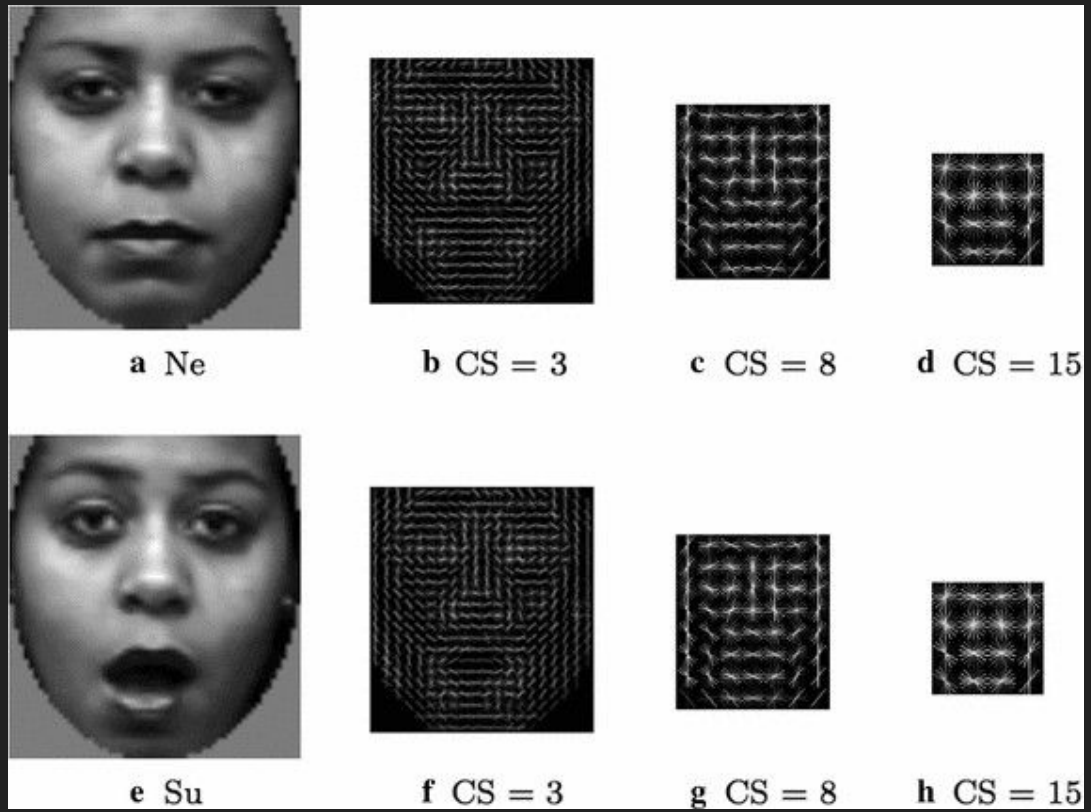
Gabor Filters

Ref: link, link





HOG Filters



References

- https://cs231n.github.io/convolutional-networks/
- https://madebyollin.github.io/convnet-calculator/
- https://scikit-image.org/docs/stable/auto_examples/index.html
- https://www.kaggle.com/code/isbhargav/guide-to-pytorch-learning-rate-scheduling
- https://google.github.io/mediapipe/solutions/face_mesh.html
- https://towardsdatascience.com/hog-histogram-of-oriented-gradients-67ecd887675f

HW

1. Calculate number of weights on each layer
2. Calculate shape of tensors before and after each layer
3. Make model overfit the data. Show loss curves with overfit
4. Reduce model complexity (number parameters) with keeping accuracy