

Chapter 1: Getting Started with Digital Image Processing

Images as multidimensional arrays and matrices

Single Channel

Black & White

0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0

1 bit per pixel.
Values 0 – 1
Height x Width

Grayscale

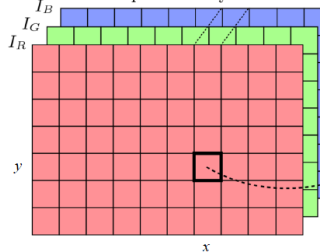
8	129	221	189	145	212	83	244
149	20	30	71	205	177	208	173
47	19	3	249	102	229	161	122
0	76	198	197	28	147	107	83
243	40	131	54	150	18	11	158
123	84	137	10	139	111	241	183
239	217	100	153	131	236	230	113
237	166	62	169	129	32	141	185

8 bits per pixel.
Values 0 – 255
Height x Width

Multi Channel

Color(RGB)

3 component arrays



$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} \leftarrow \begin{pmatrix} I_R(x, y) \\ I_G(x, y) \\ I_B(x, y) \end{pmatrix}$$

24 bits per pixel.
RGB Red/Green/Blue
Height x Width x 3:





original and the darkened image



None



none



nearest



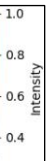
bilinear

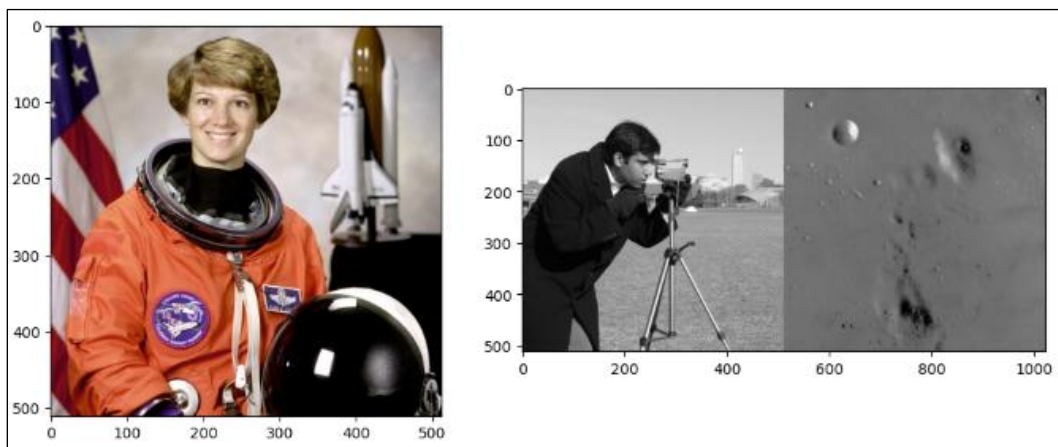
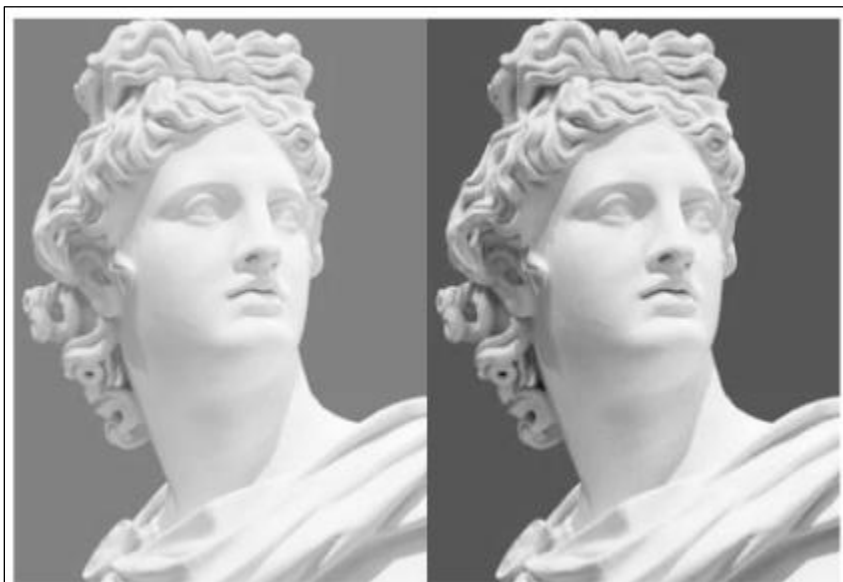


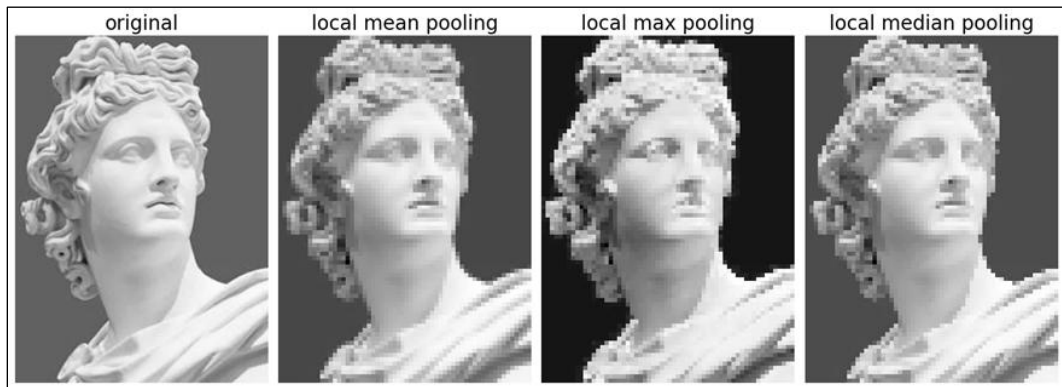
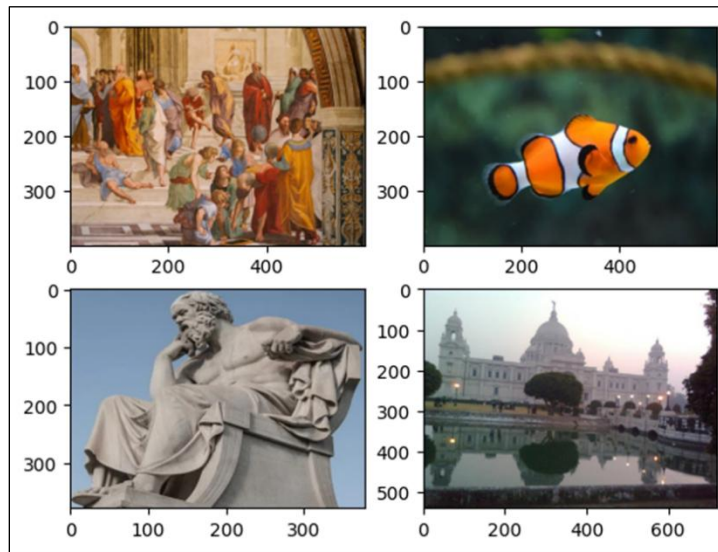
bicubic



spline16



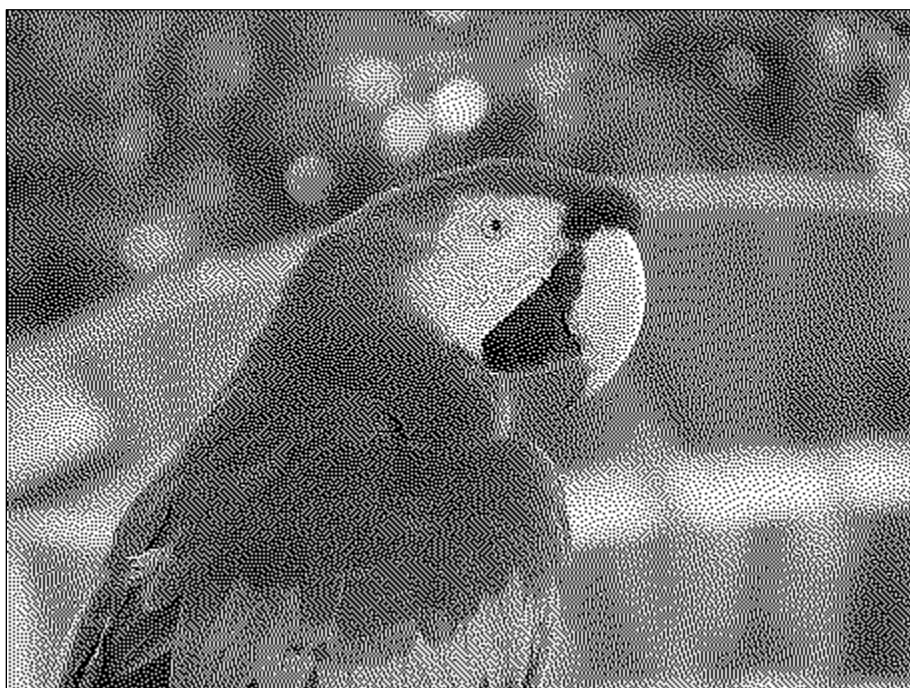


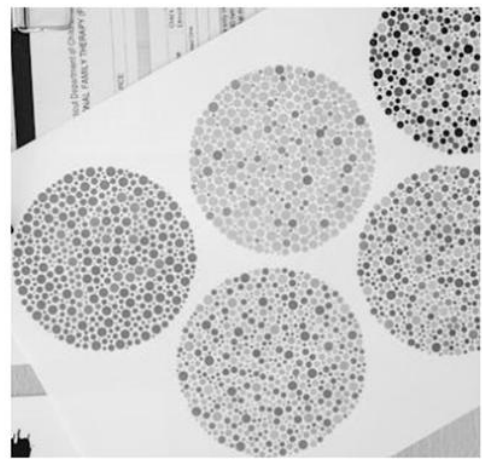
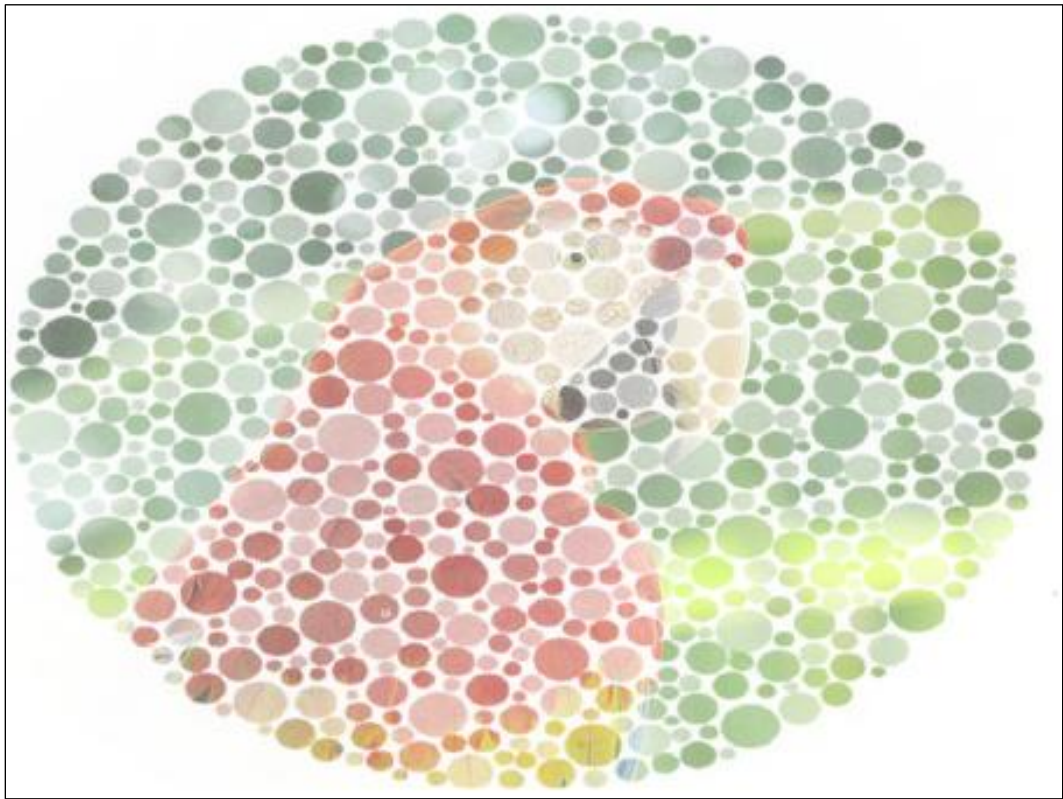




frames selected from the animated GIF







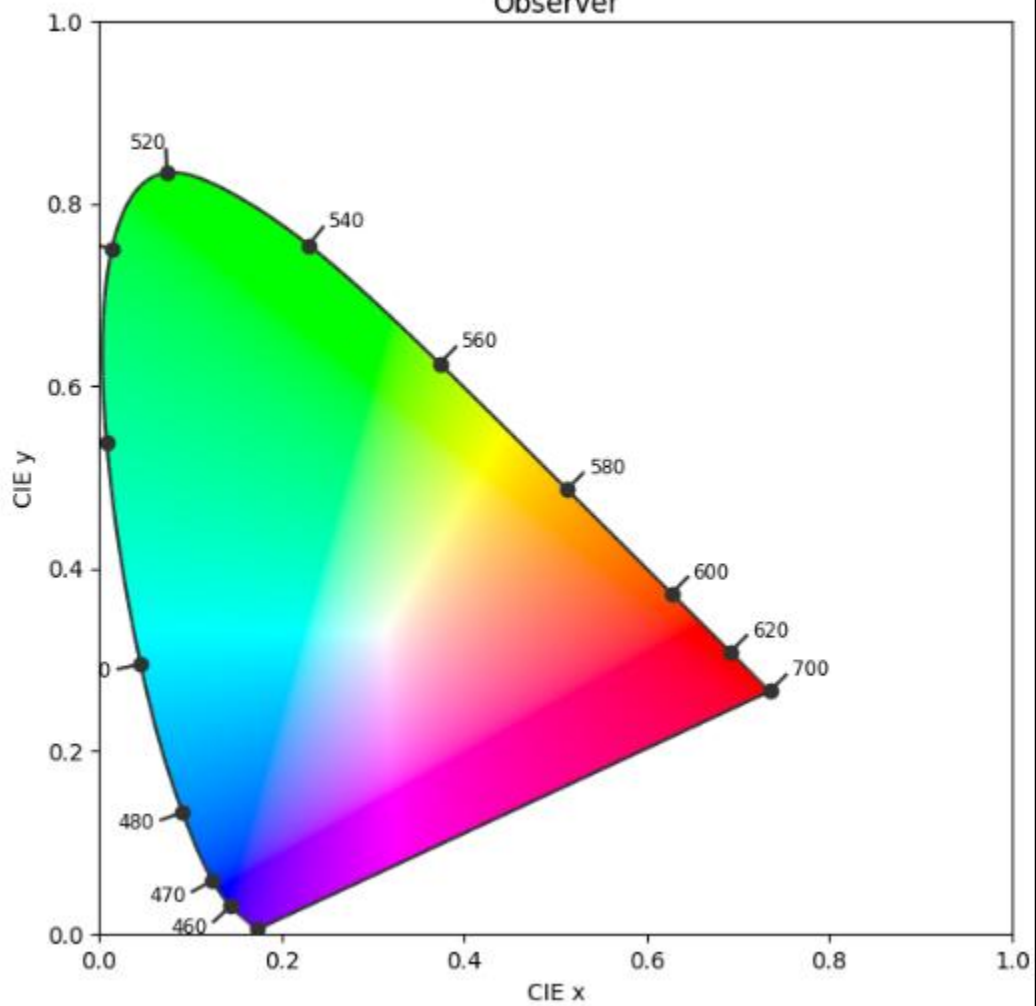
RGB to YIQ

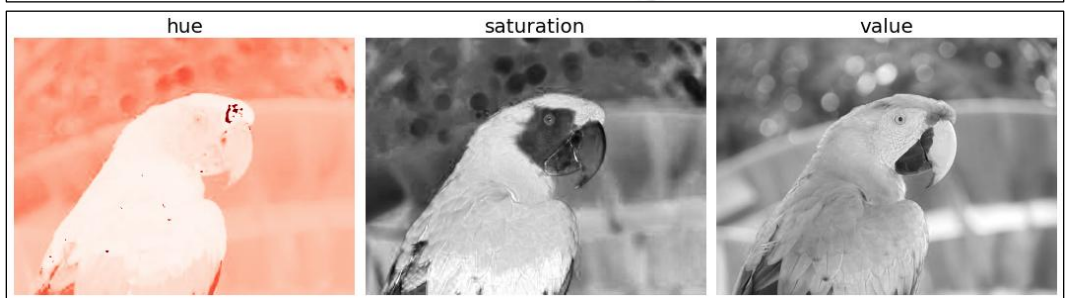
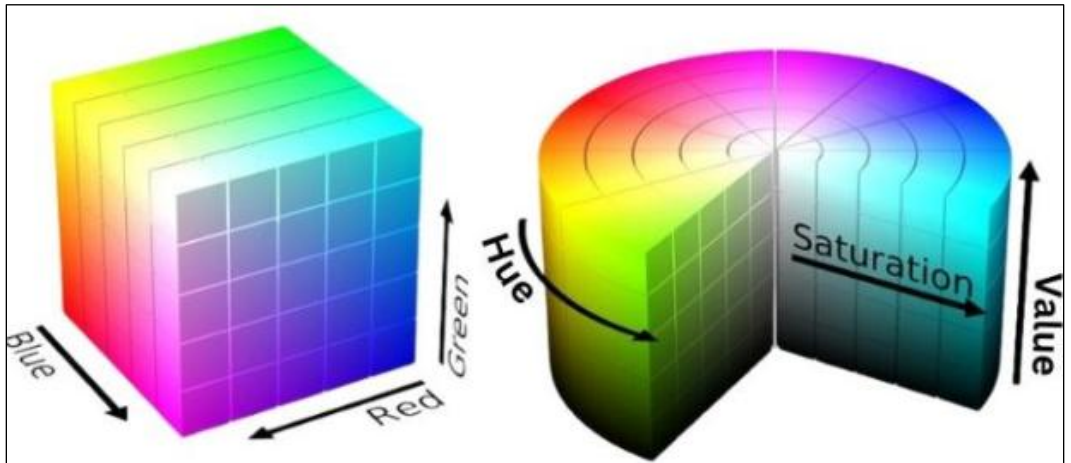
$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

YIQ to RGB

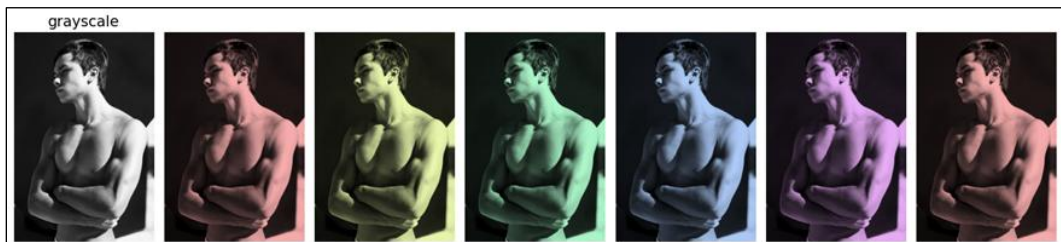
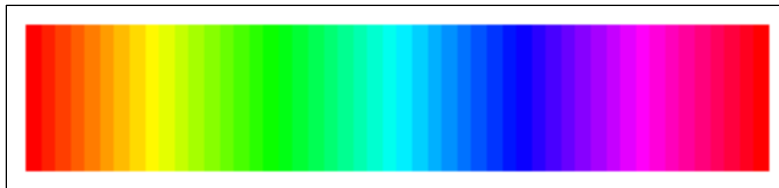
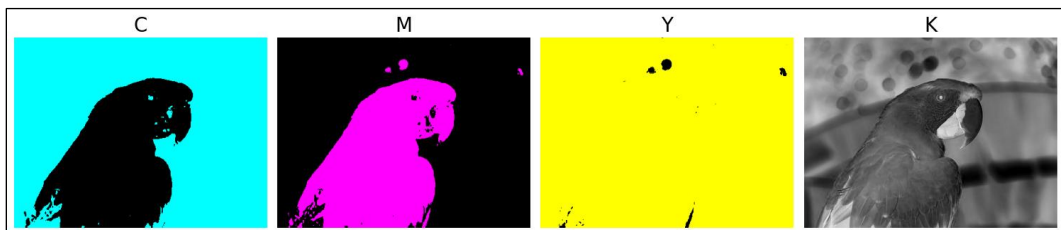
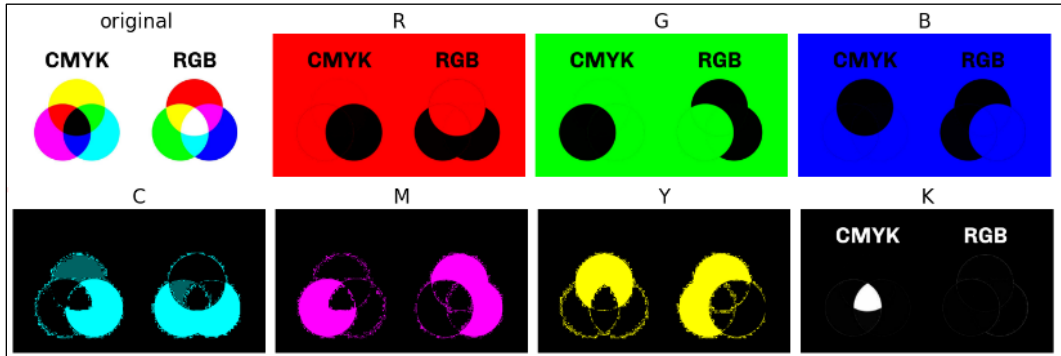
$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0.956 & 0.621 \\ 1 & -0.272 & -0.647 \\ 1 & -1.106 & 1.703 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$$

CIE 1931 Chromaticity Diagram - CIE 1931 2° Standard Observer





$$\begin{aligned} \text{Cyan} &= (1 - R_{/255} - K) / (1 - K) \\ \text{Magenta} &= (1 - G_{/255} - K) / (1 - K) \\ \text{Yellow} &= (1 - B_{/255} - K) / (1 - K) \\ \text{Black (K)} &= 1 - (\max(R, G, B) / 255) \end{aligned}$$



The original input, grayscale, and the output image

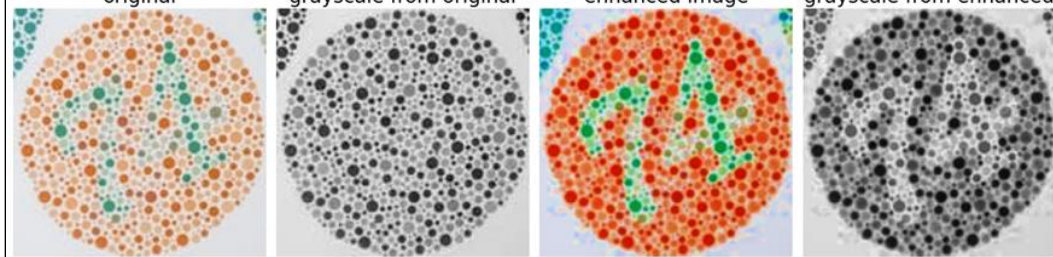


original

grayscale from original

enhanced image

grayscale from enhanced



original

luminosity changed

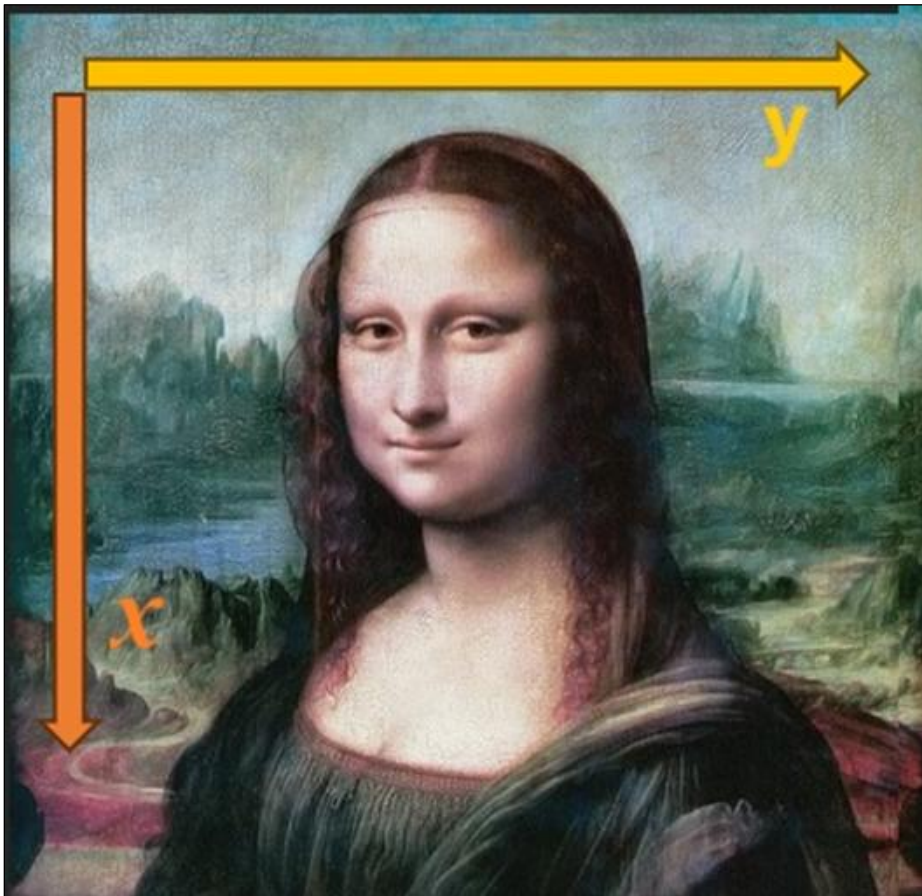
red tinted

cooled



Skin color detection in HSV, YCbCr, and mixed colorspace





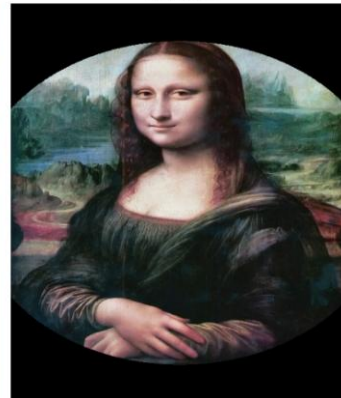
original

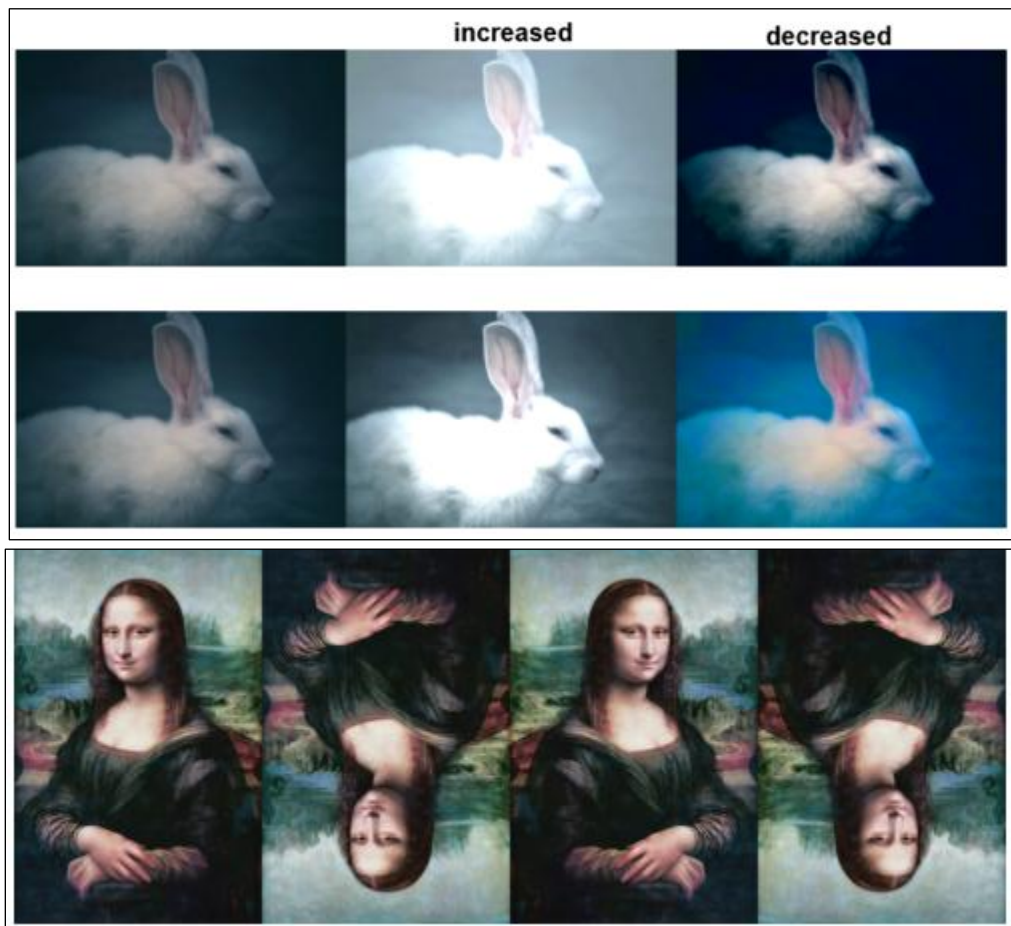


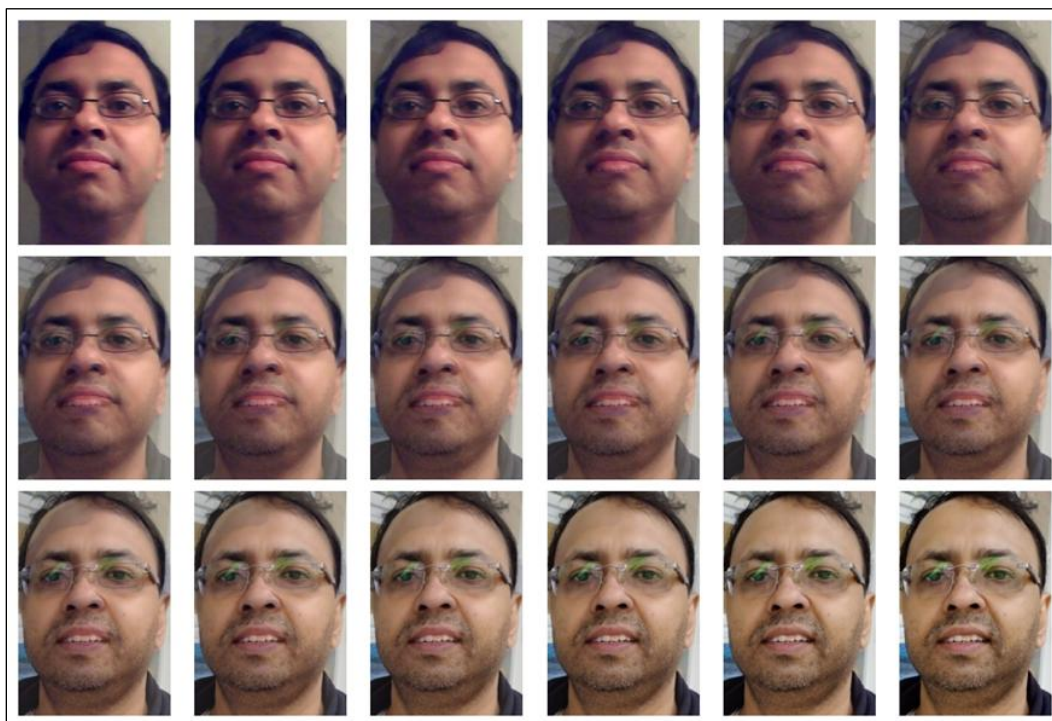
cropped



masked







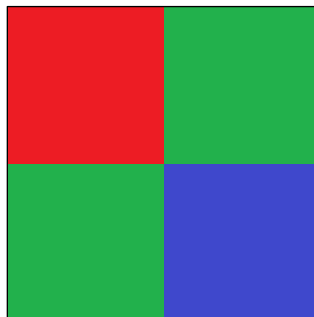
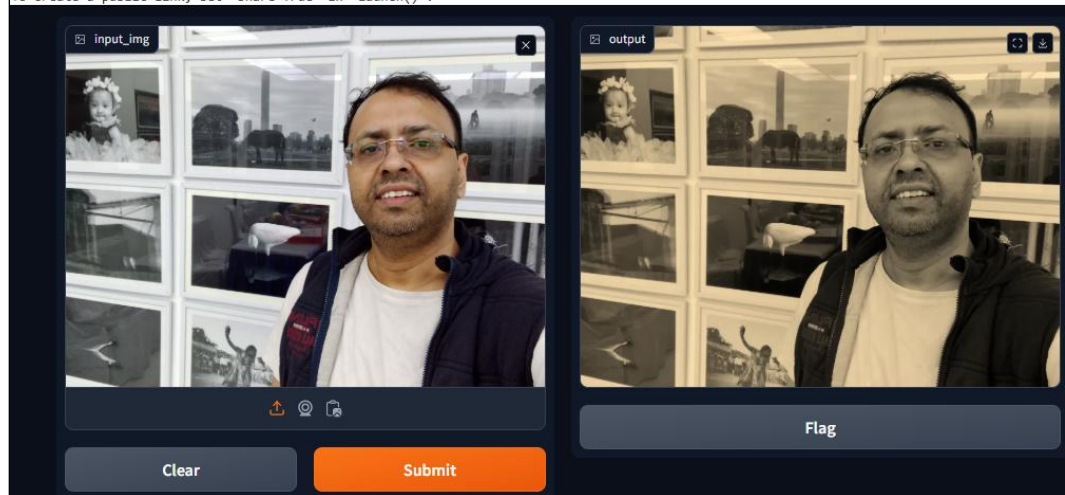
original input image, illumination pattern, and output image

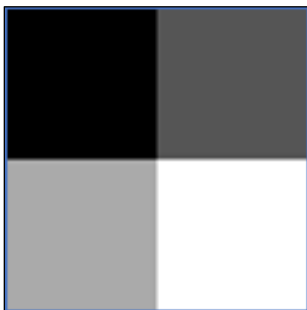




Running on local URL: <http://127.0.0.1:7860>

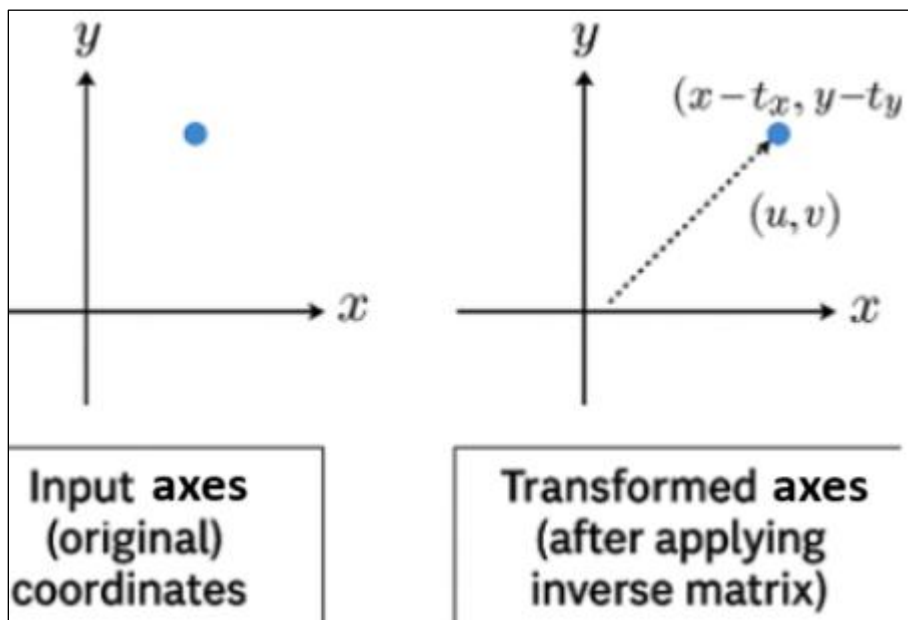
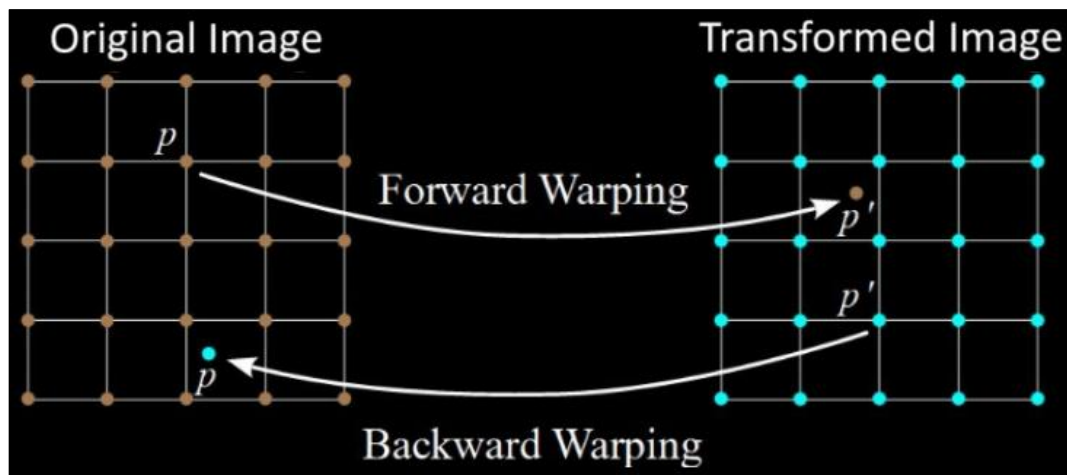
To create a public link, set `share=True` in `launch()`.

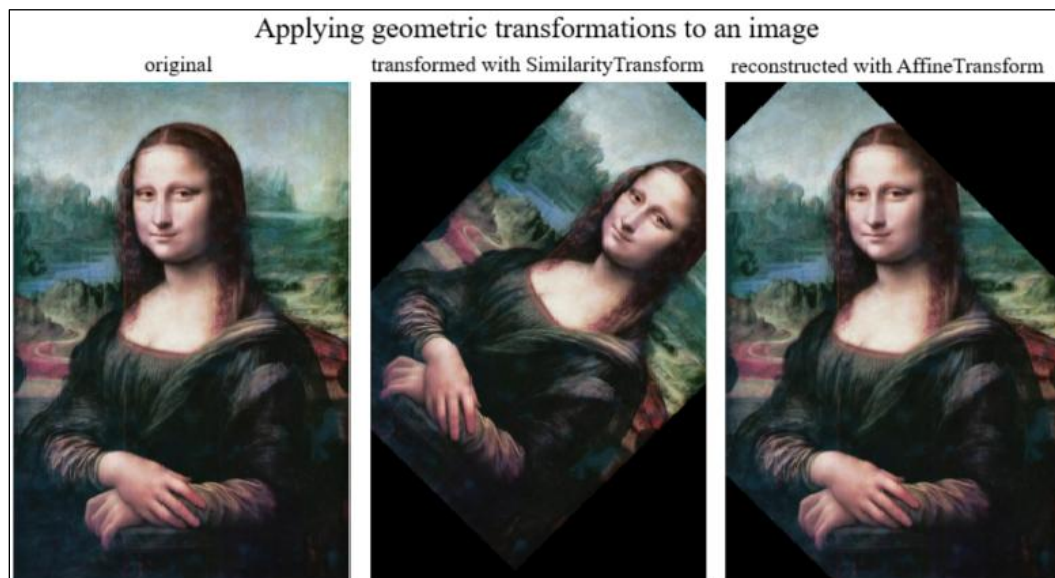




Chapter 2: Image Manipulation



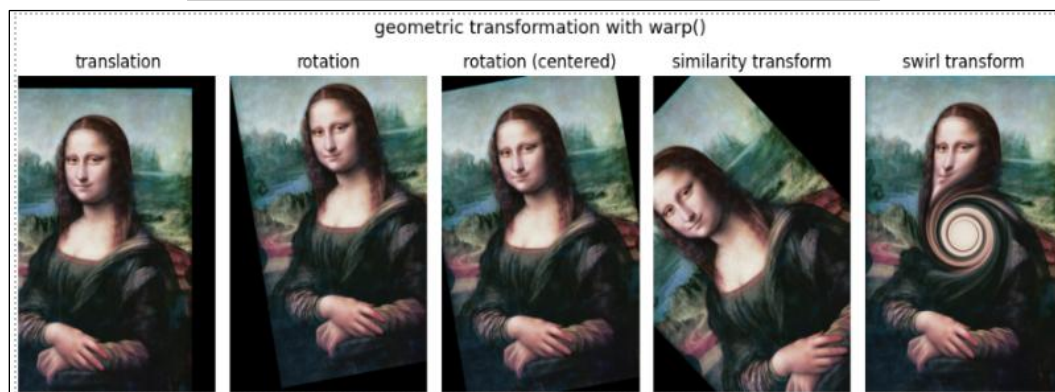




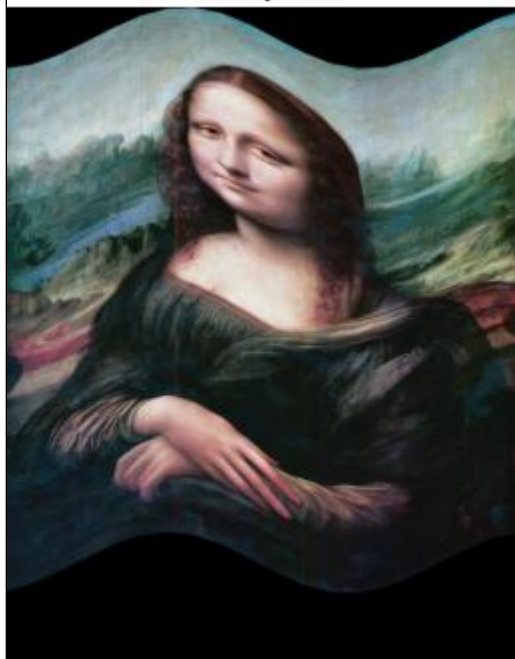
$$\theta = \arctan\left(\frac{y}{x}\right) \quad \rho = \sqrt{(x - x_0)^2 + (y - y_0)^2}$$

$$r = \ln(2) \cdot \frac{\text{radius}}{5} \quad \theta' = \psi + s \cdot e^{-\rho/r + \theta}$$

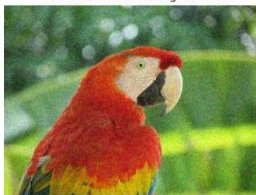
$$\phi = \text{rotation} \quad s = \text{strength}$$



warped



Gaussian noise with sigma=0.1



Gaussian noise with sigma=0.25



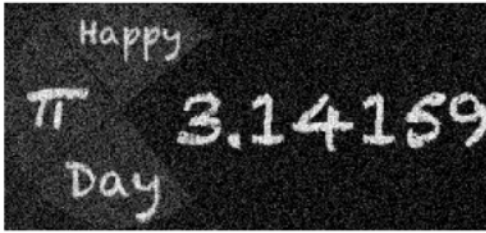
Gaussian noise with sigma=0.5



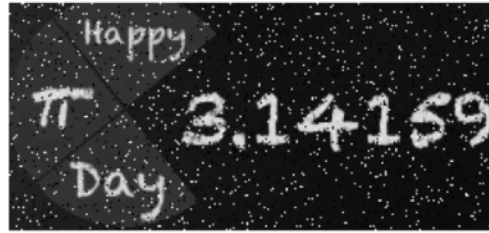
Gaussian noise with sigma=1



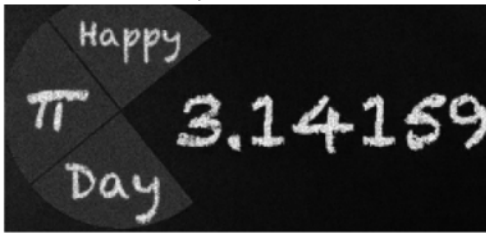
Gaussian noise



S&P noise



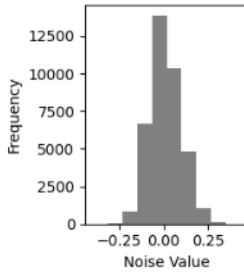
Speckle noise



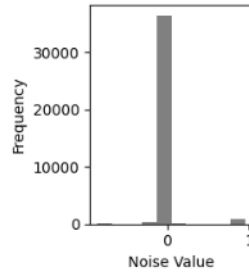
Poisson noise



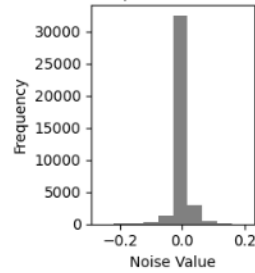
Gaussian noise



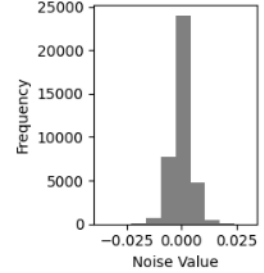
S&P noise

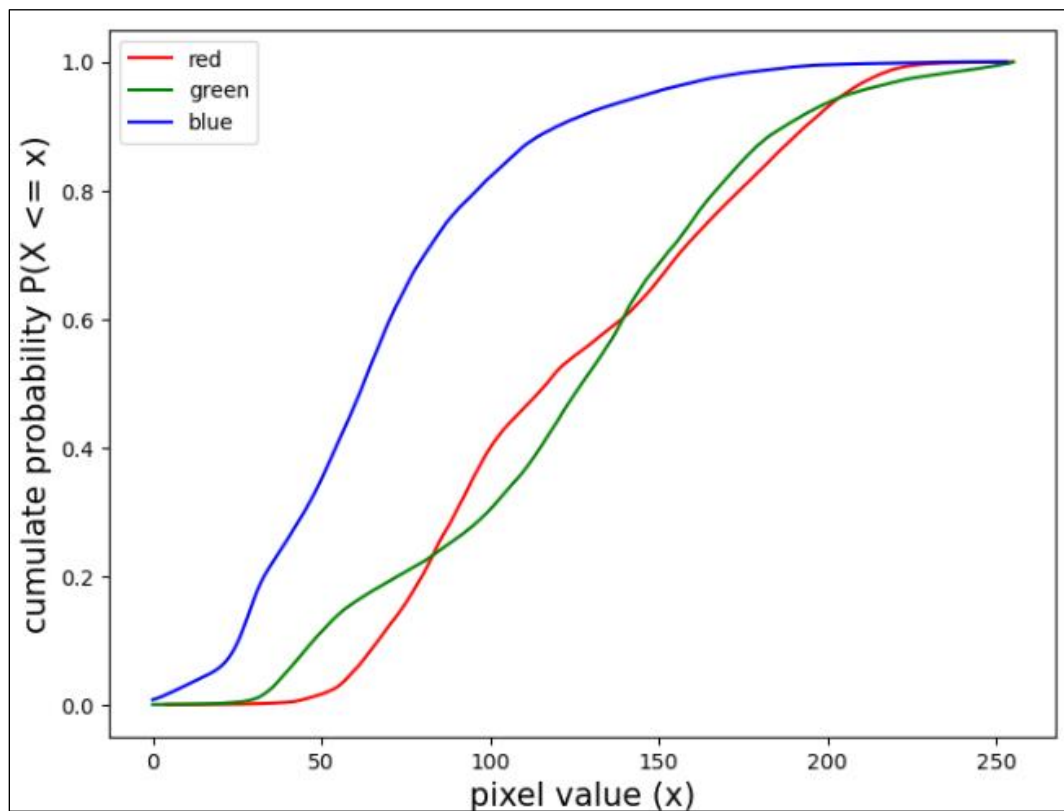


Speckle noise



Poisson noise





rescaled (aliasing), resized (no aliasing), and downsampled images



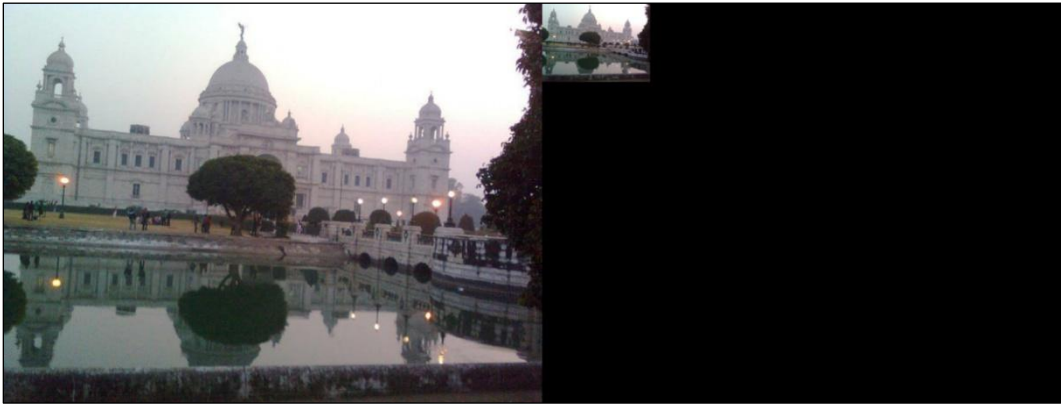
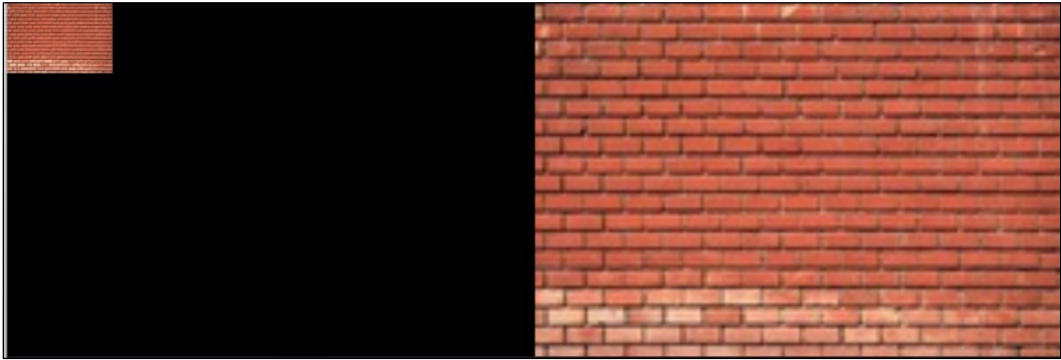
original



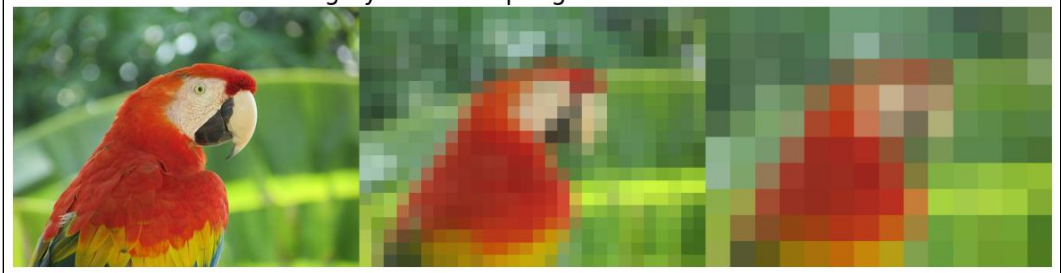
cropped



Original		Target			Interpolation Result		
100	50	100	?	50	100	75	50
70	20	?	?	?	85	60	35
		70	?	20	70	45	20
2 x 2		3 x 3			3 x 3		



Pixelating by downsampling to factor of 16 and 32

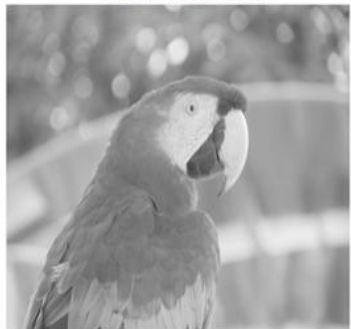




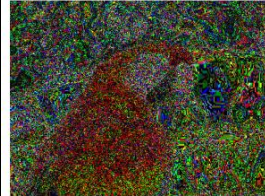
Grayscale

Log transformation

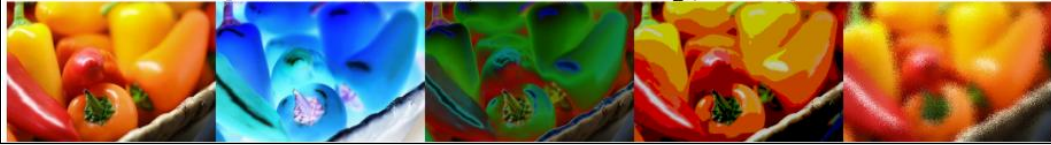
Power-law transformation



point transforms with PIL Image and ImagingCore



The original, inverted, solarized, posterized, and effect_spread images with PIL



input

reflected

rotated



Image transformations

A 2D affine transformation

$$T = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \quad T \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + c \\ dx + ey + f \\ 1 \end{bmatrix}$$

Translation

Scale

Rotation

$$\begin{bmatrix} 1 & 0 & s \\ 0 & 1 & t \\ 0 & 0 & 1 \end{bmatrix}$$

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

$$\begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Correct rotation $T_2 R T_1$

1. Translate center to origin

2. Rotate

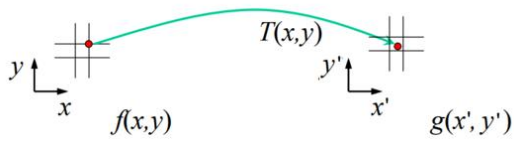
3. Translate back to center

$$T_1 = \begin{bmatrix} 1 & 0 & -w/2 \\ 0 & 1 & -h/2 \\ 0 & 0 & 1 \end{bmatrix}$$

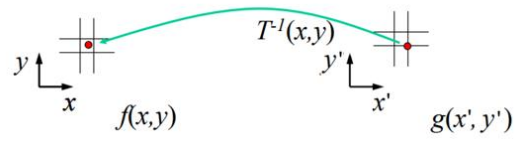
$$R = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_2 = \begin{bmatrix} 1 & 0 & w/2 \\ 0 & 1 & h/2 \\ 0 & 0 & 1 \end{bmatrix}$$

Forward warping



Inverse warping



Inverse mapping

1. Create an output image
 2. For each pixel in the output image, find the corresponding pixel in the input image
 3. Give that output pixel the same color
- Requires that we invert the mapping

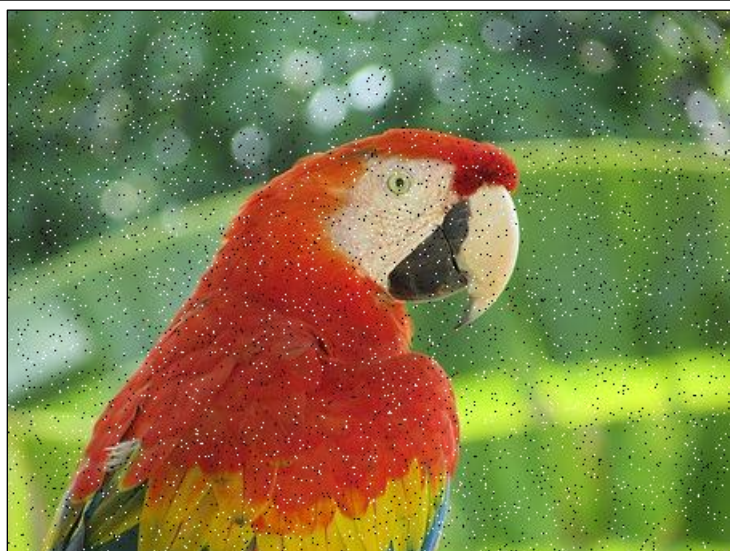
rotation with respect to origin and center



shear



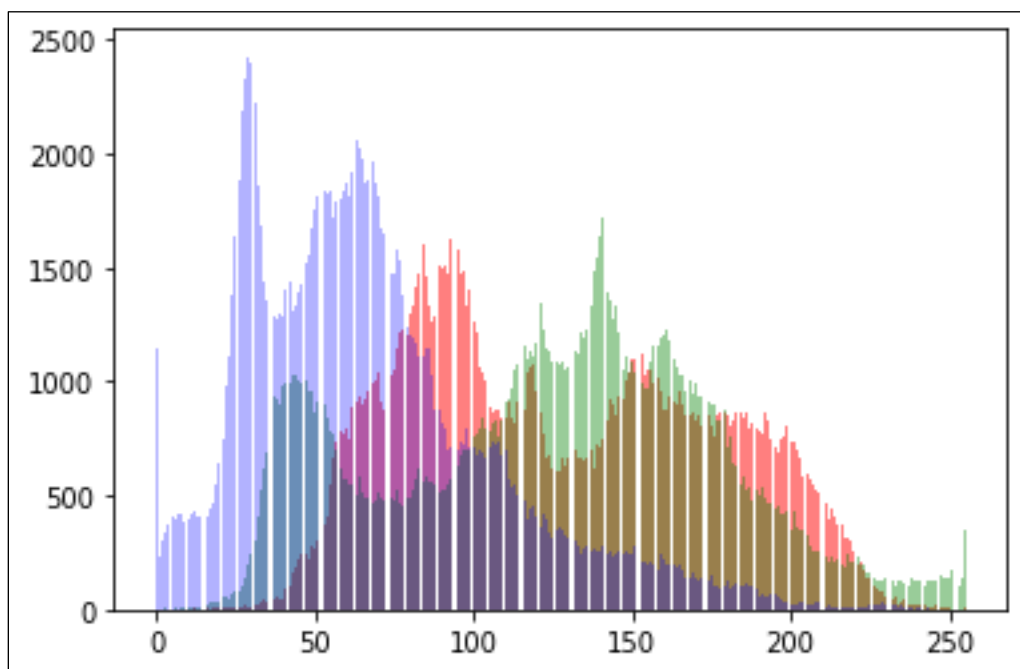
perspective

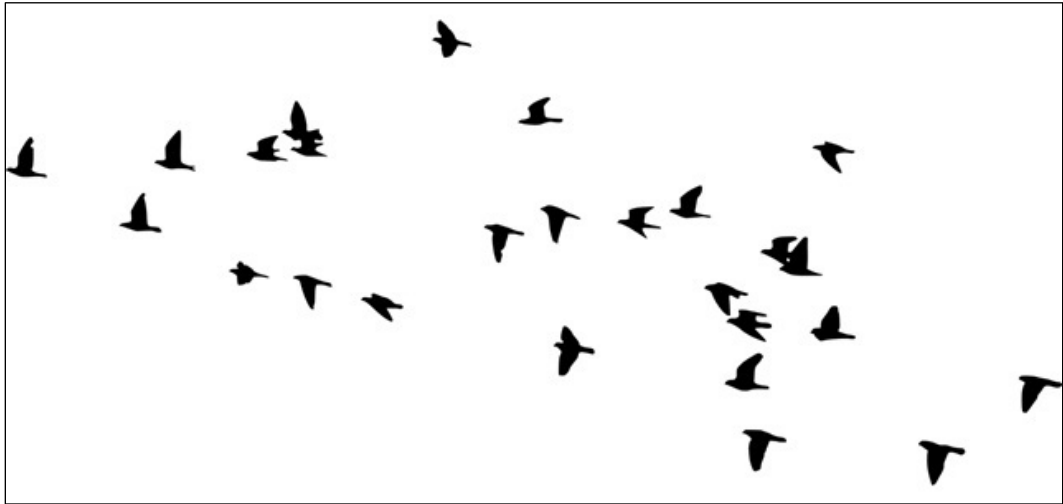




Welcome to image processing with Python







JPG

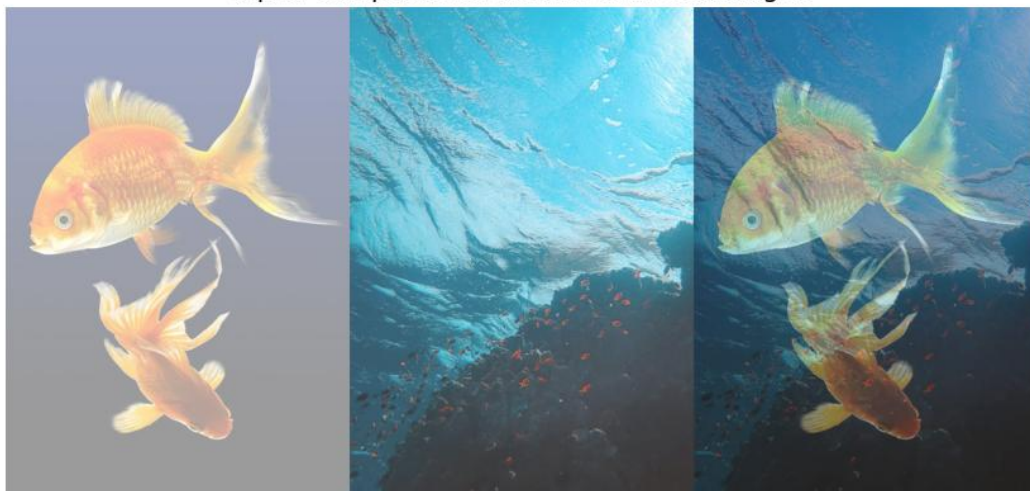


Transparent PNG





alpha composite of the fish and sea images

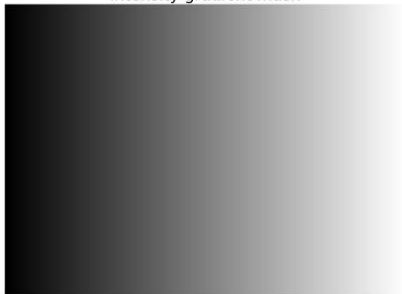




original



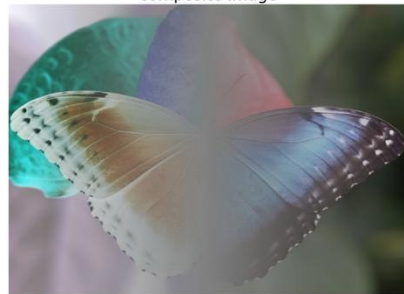
intensity gradient mask



inverted



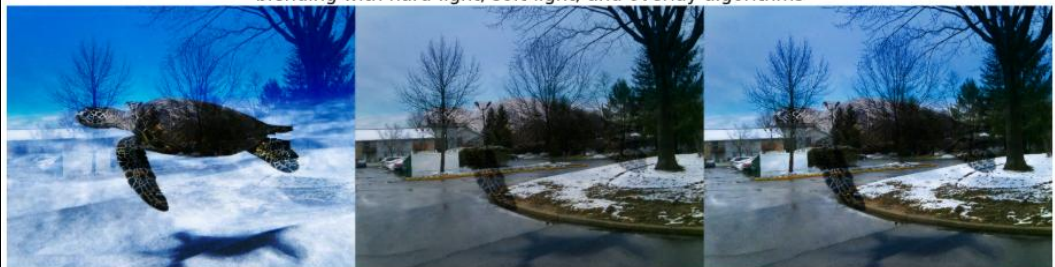
composite image



Comparison of Effects

Blend Mode	Dark Pixels Behavior	Light Pixels Behavior	Contrast Effect
Hard Light	Darker (Multiply)	Lighter (Screen)	High contrast
Soft Light	Subtly darker	Subtly lighter	Soft transitions
Overlay	Darker (Multiply)	Lighter (Screen)	Strong contrast, but base-driven

blending with hard-light, soft-light, and overlay algorithms



Blending with soft-light algorithm with blend_modes

opacity=0.3

opacity=0.5

opacity=0.8



Blending with hard-light algorithm with blend_modes

opacity=0.3

opacity=0.5

opacity=0.8



screen



lighter



Chapter 3: More Image Manipulation

Changing image quality with PIL (with qualities: [95, 20, 5, 1])



Extracted image patch with PIL (with qualities: [95, 20, 5, 1])



The original and the deformed image

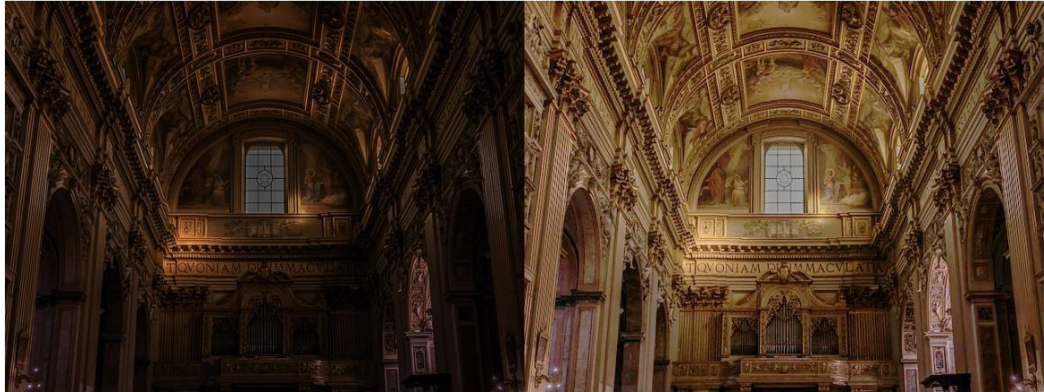


The original and the colorized image





original and γ -corrected image



original and γ -corrected image



Lens Blur with Bokeh



Palette of Image 1 before swap



Palette of Image 2 before swap

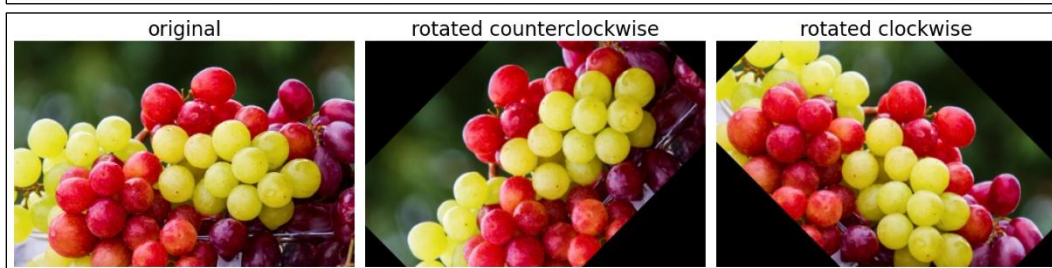
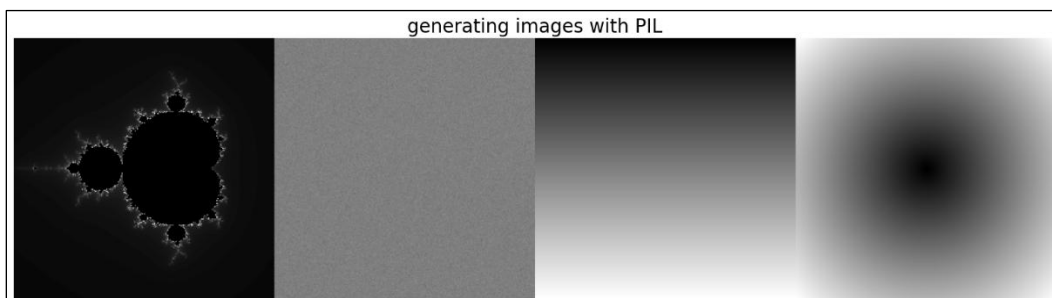


Exchange image color palettes



RGB tone changes across color temperatures





original



rotated without crop



rotated with crop



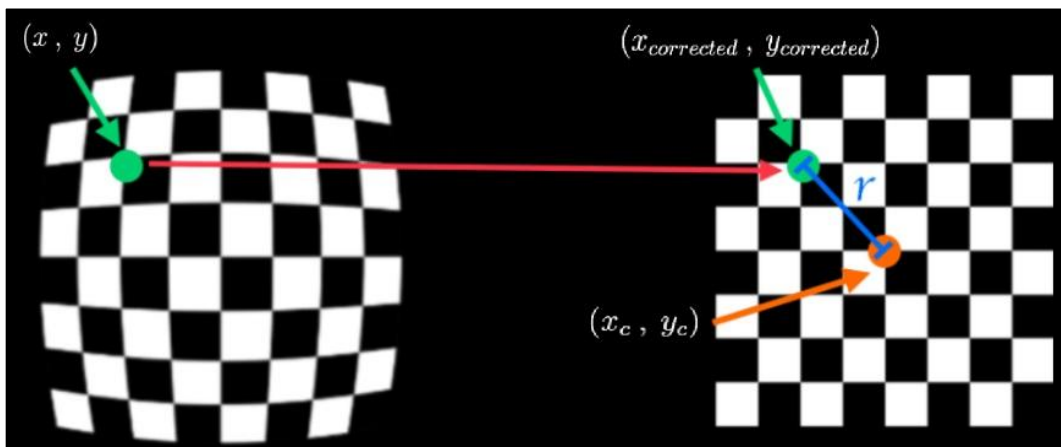
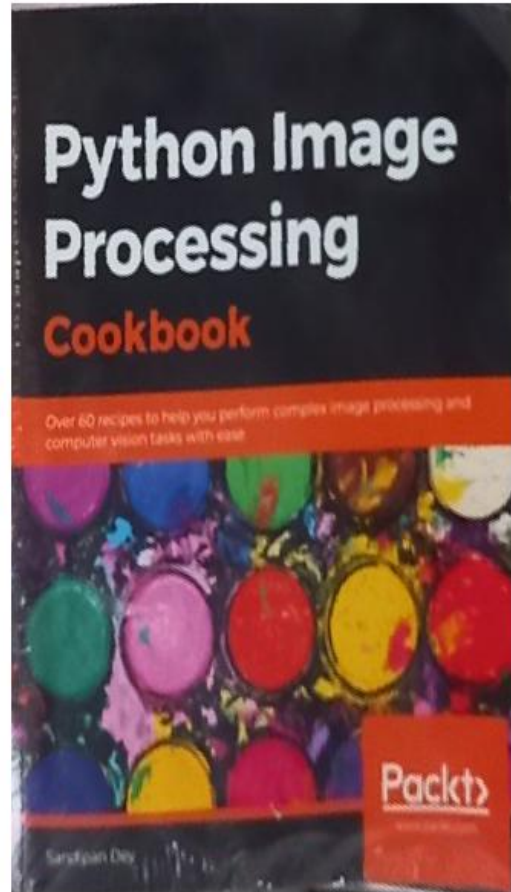
original vs. crosshatched images



Source image



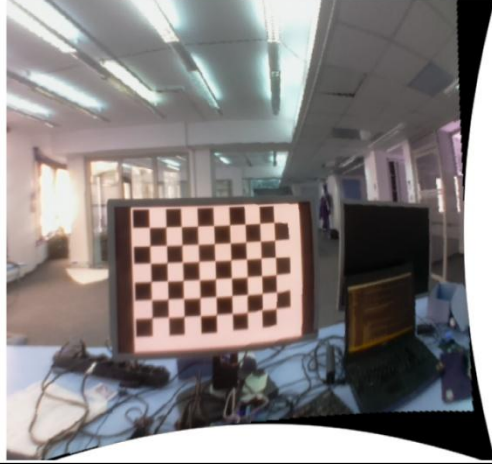
Destination image



distorted

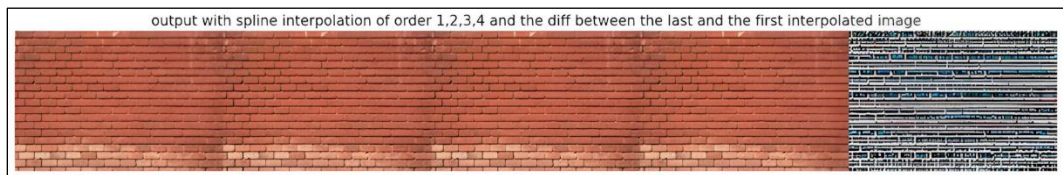
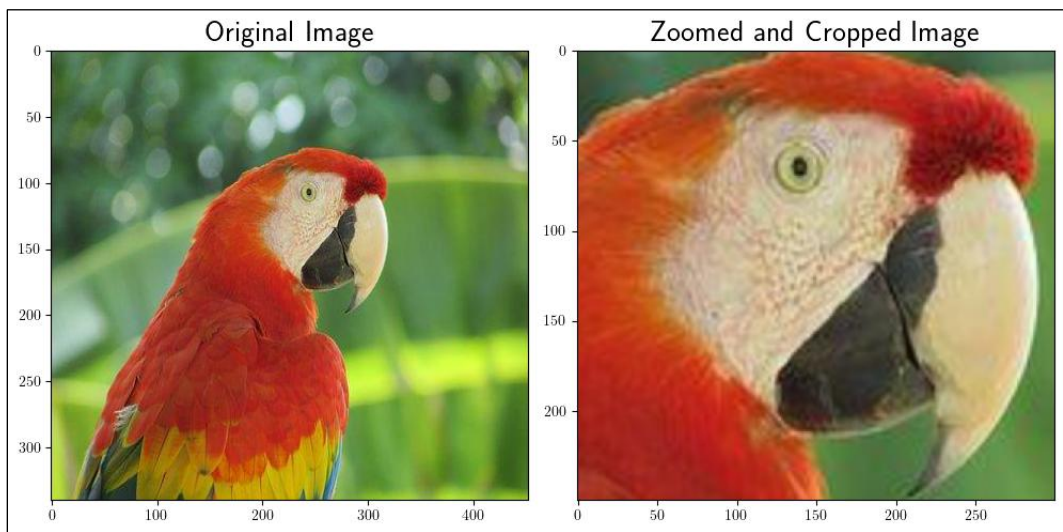


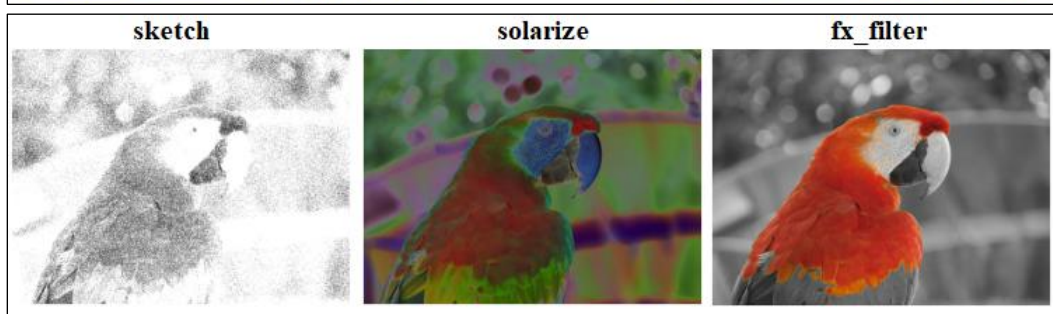
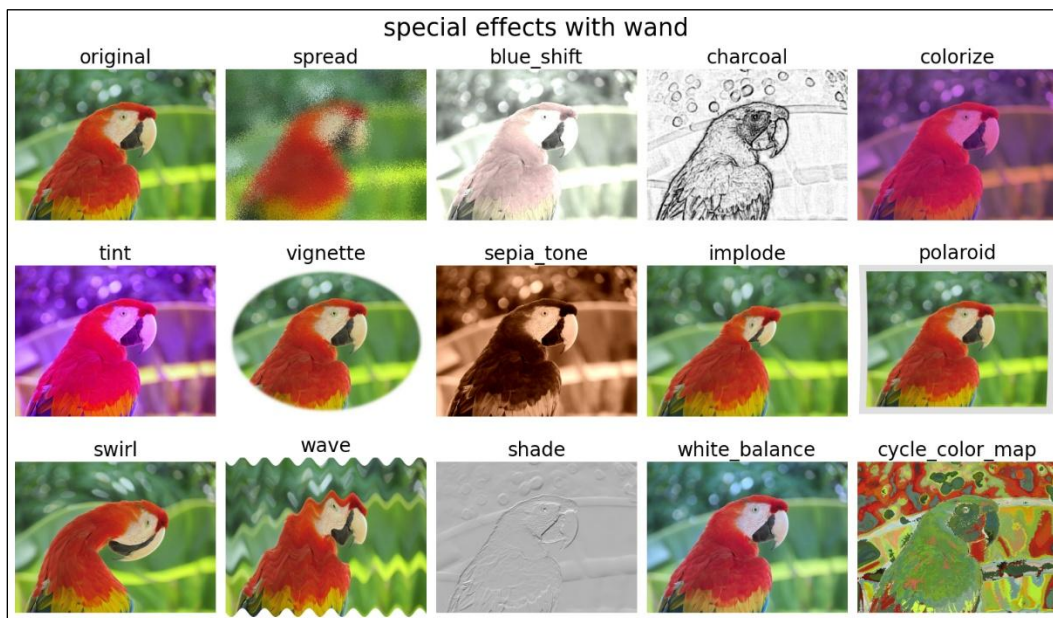
undistorted



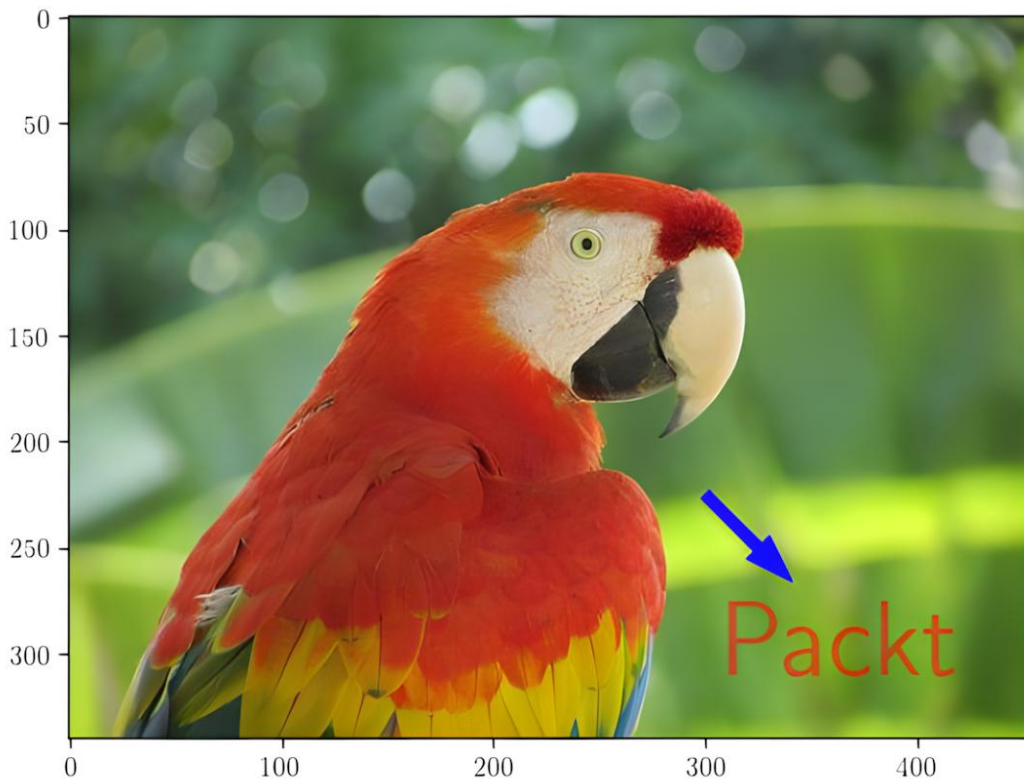
Fish eye correction with defisheye







The distorted and corrected images



Original Image

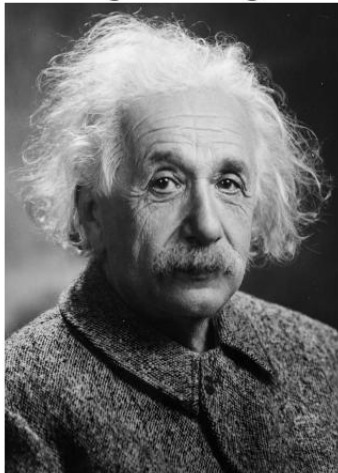
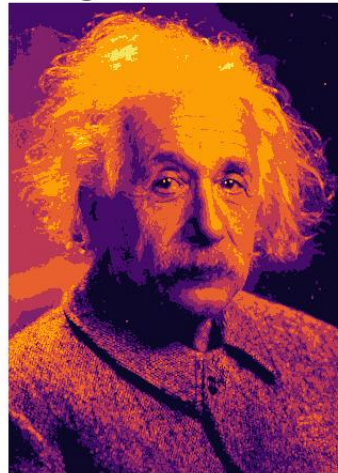


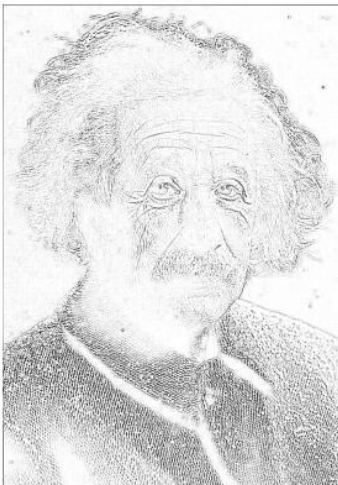
Image Contour Lines



Image Filled Contour



PIL

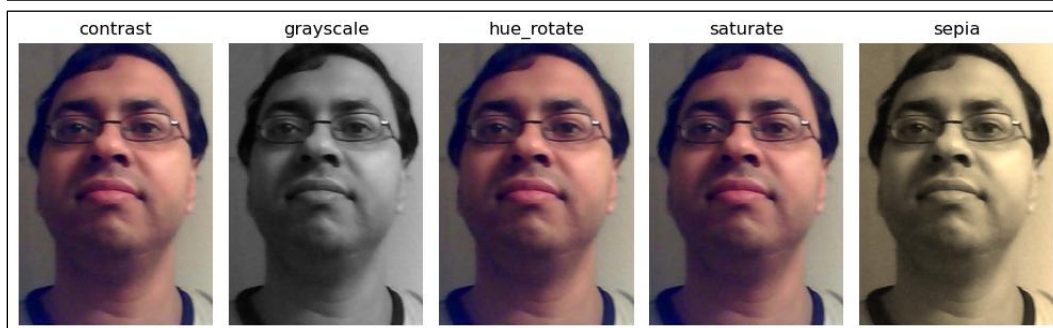


opencv-python



skimage

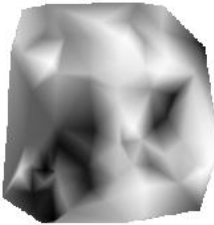




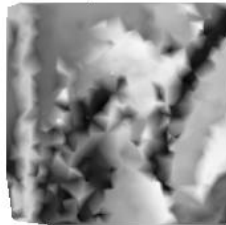
Chapter 4: Sampling and Fourier Transform

Reconstructing image from small number of samples with interpolation

#samples = 100



#samples = 1000



#samples = 10000



#samples = 100000



RBF

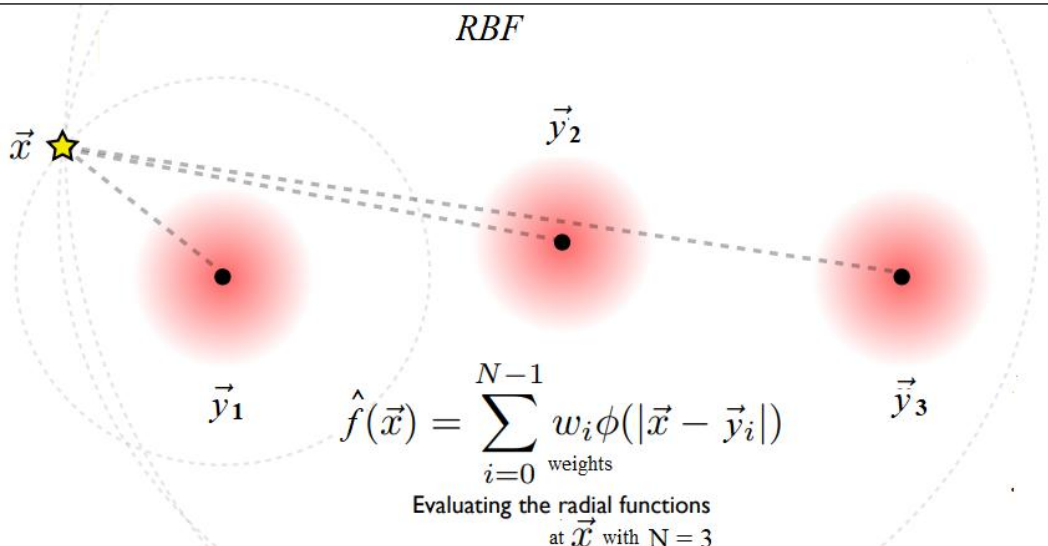


Image inpainting with different interpolation methods with scipy



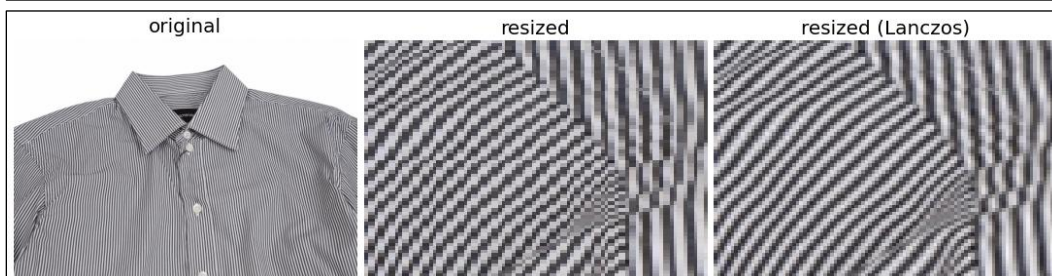
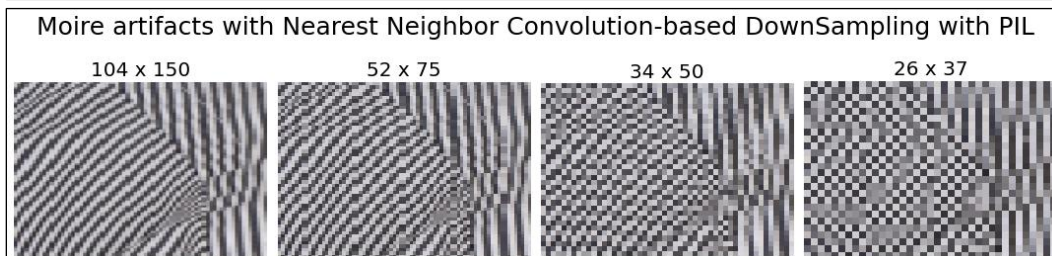
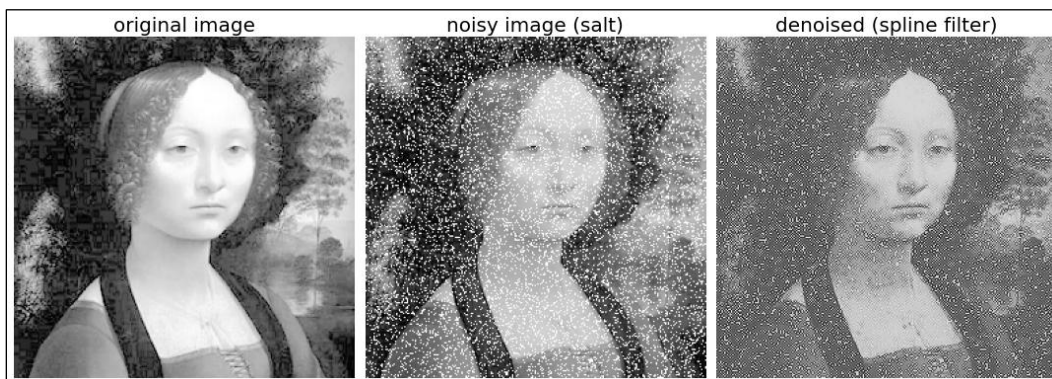


image size = 243 x 162 antialiasing=False



image size = 243 x 162 antialiasing=True



image size = 73 x 49 antialiasing=False



image size = 73 x 49 antialiasing=True



Original (Float)

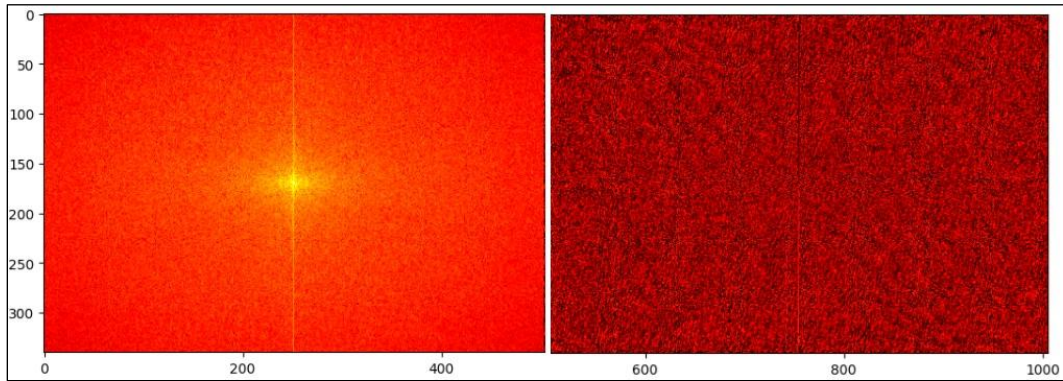


8-bit Quantization (16 levels)



4-bit Quantization (4 levels)



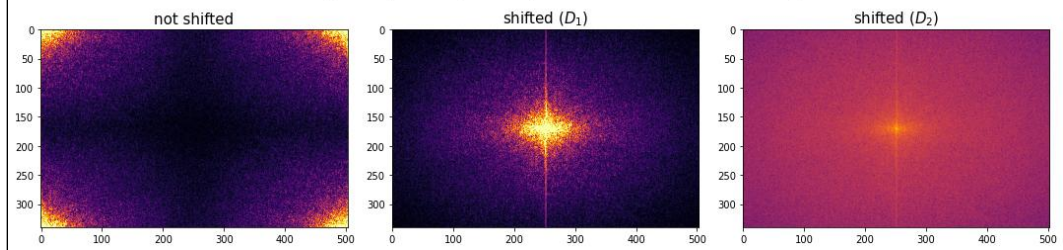


Original Image

Image obtained after reconstruction



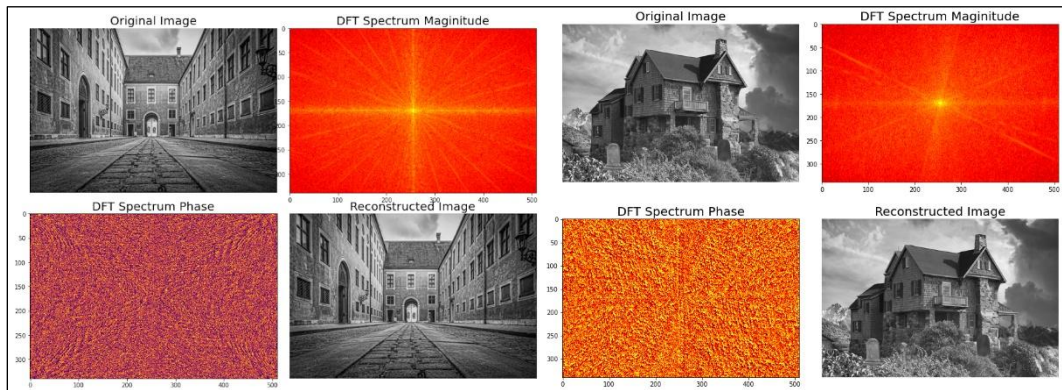
Plotting the (shifted) DFT Spectrum of the rhinos image



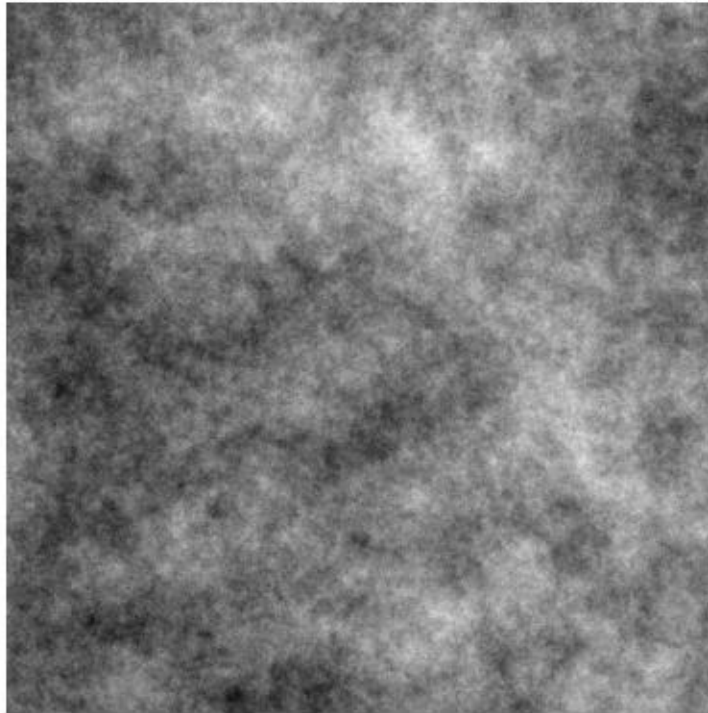
$$F(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-j2\pi(ux+vy)} dx dy$$

Separability Theorem

$$f(x, y) = f(x)g(y) \xrightarrow{\mathcal{F}} F(u, v) = F(u)G(v)$$



randomly generated texture
(Fractional Brownian field)



Reconstructed Images with IDFT

with $|F_1|, \Phi(F_2)$



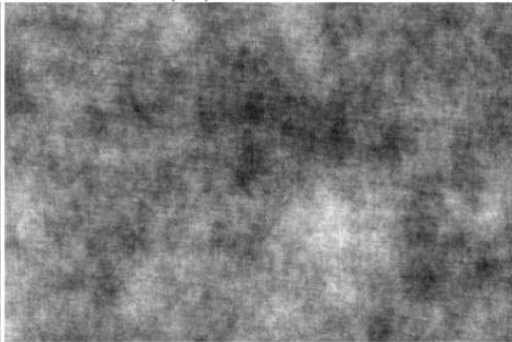
with $|F_2|, \Phi(F_1)$



with Random Magnitude, $\Phi(F_2)$



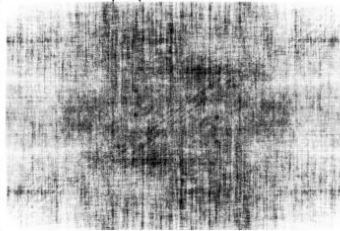
with $|F_2|$, Random phase



$|F_1|, \Phi(F_1)$

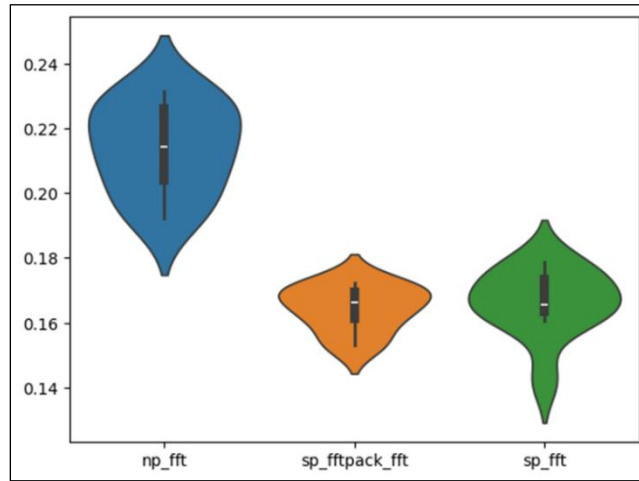


$|F_1|, \Phi(F_1) = 0$



$|F_1| = 1, \Phi(F_1)$





Separability of the 2D DFT

$$F(k, \ell) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} \left[\frac{1}{\sqrt{M}} \sum_{m=0}^{M-1} f(m, n) e^{-j2\pi(k\frac{m}{M})} \right] e^{-j2\pi(\ell\frac{n}{N})}$$

matrix notation

2D DFT **forward transform** $F = (W_M f) W_N$
 $M \times M \quad N \times N$

$$\begin{matrix} F \\ M \times N \end{matrix} = \begin{matrix} W_M \\ M \times M \end{matrix} \begin{matrix} f \\ M \times N \end{matrix} \begin{matrix} W_N \\ N \times N \end{matrix}$$

$$f(m, n) = \frac{1}{\sqrt{N}} \sum_{\ell=0}^{N-1} \left[\frac{1}{\sqrt{M}} \sum_{k=0}^{M-1} \hat{F}(k, \ell) e^{j2\pi(k\frac{m}{M})} \right] e^{j2\pi(\ell\frac{n}{N})}$$

matrix notation

2D DFT **inverse transform** $f = (W_M^* F) W_N^*$
 $M \times M \quad N \times N$

$$W_M(m, k) = \frac{1}{\sqrt{M}} e^{-j2\pi(k\frac{m}{M})}$$

$$W_N(n, \ell) = \frac{1}{\sqrt{N}} e^{-j2\pi(\ell\frac{n}{N})}$$

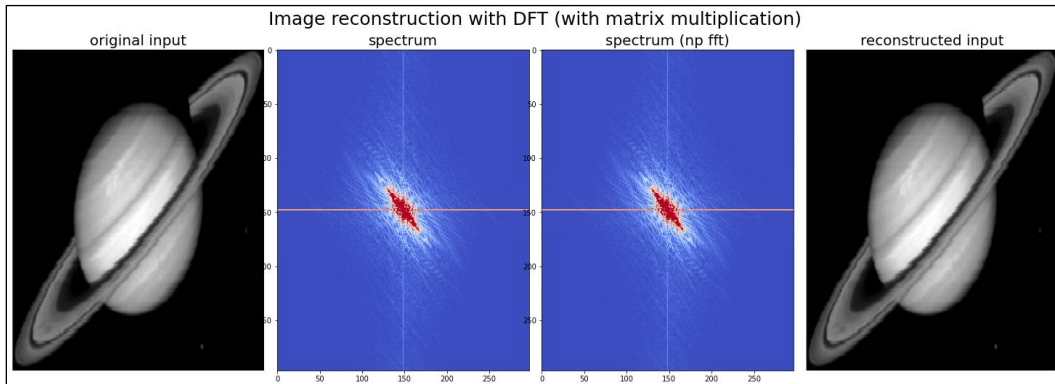


Image transforms

forward transformation

$$T(u, v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) g(x, y, u, v)$$

Forward transformation kernel.

$$g(x, y, u, v) = g_1(x, u) g_2(y, v) \text{ separable}$$

$$= g_1(x, u) g_1(y, v) \text{ Symmetric form}$$

inverse transformation

$$f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T(u, v) h(x, y, u, v)$$

Inverse transformation kernel

$$T(u, v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} g_1(x, u) f(x, y) g_2(y, v)$$

Matrix-based transforms

For orthonormal basis vectors

$$T = G f G^T$$

$$f = G^{*T} T G^*$$

for complex vectors

$$f = G^T T G$$

for real vectors

DFT

$$g_1(x, u) = \frac{1}{\sqrt{N}} e^{-j \frac{2\pi}{N} ux}$$

$$g_1(y, v) = \frac{1}{\sqrt{N}} e^{-j \frac{2\pi}{N} vy}$$

$$g(x, y, u, v) = \alpha(u) \alpha(v) \cos \left[\frac{(2x+1)u\pi}{2N} \right] \cos \left[\frac{(2y+1)v\pi}{2N} \right]$$

$$\alpha(u) = \frac{1}{\sqrt{N}}, \text{ when } u = 0$$

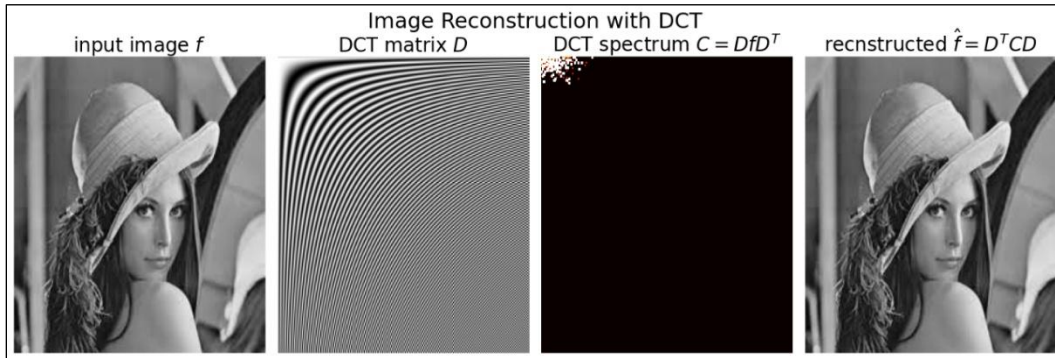
$$= \sqrt{\frac{2}{N}} \quad \forall u = 1, 2, \dots, N-1$$

DCT

$$C(u, v) = \alpha(u) \alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \cos \left[\frac{(2x+1)u\pi}{2N} \right] \cos \left[\frac{(2y+1)v\pi}{2N} \right]$$

Inverse DCT

$$f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \alpha(u) \alpha(v) C(u, v) \cos \left[\frac{(2x+1)u\pi}{2N} \right] \cos \left[\frac{(2y+1)v\pi}{2N} \right]$$



Hadamard matrix

$$H = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \text{ at } N = 2 \text{ and } H_{2N} = \begin{bmatrix} H_N & H_N \\ H_N & -H_N \end{bmatrix}$$

Forward transform

$$g(x, y, u, v) = \frac{1}{N} (-1)^{\sum_{i=0}^{n-1} b_i(x)b_i(u) + b_i(y)b_i(v)}$$

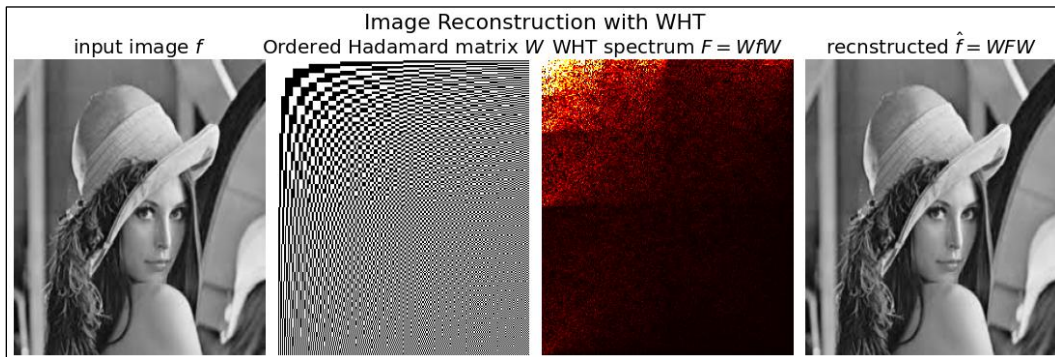
Inverse transform

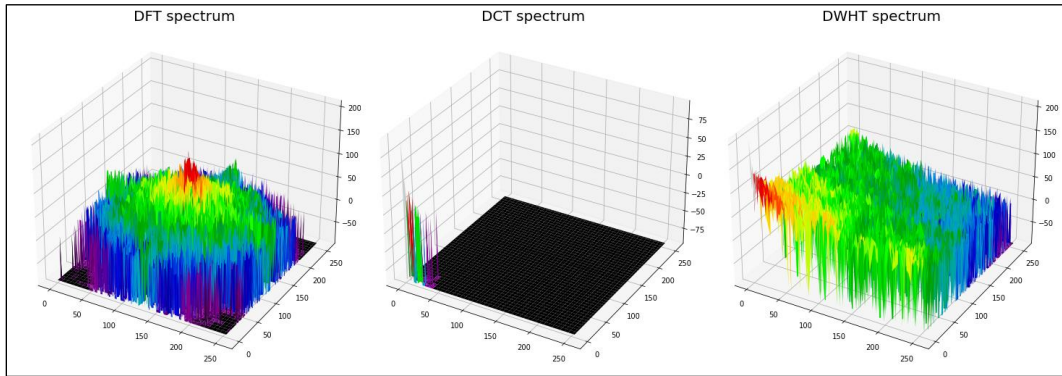
$$h(x, y, u, v) = \frac{1}{N} (-1)^{\sum_{i=0}^{n-1} b_i(x)b_i(u) + b_i(y)b_i(v)}$$

$N \rightarrow$ Total number of samples

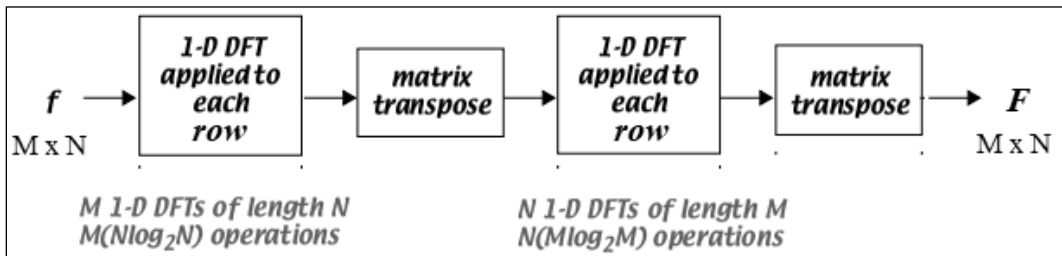
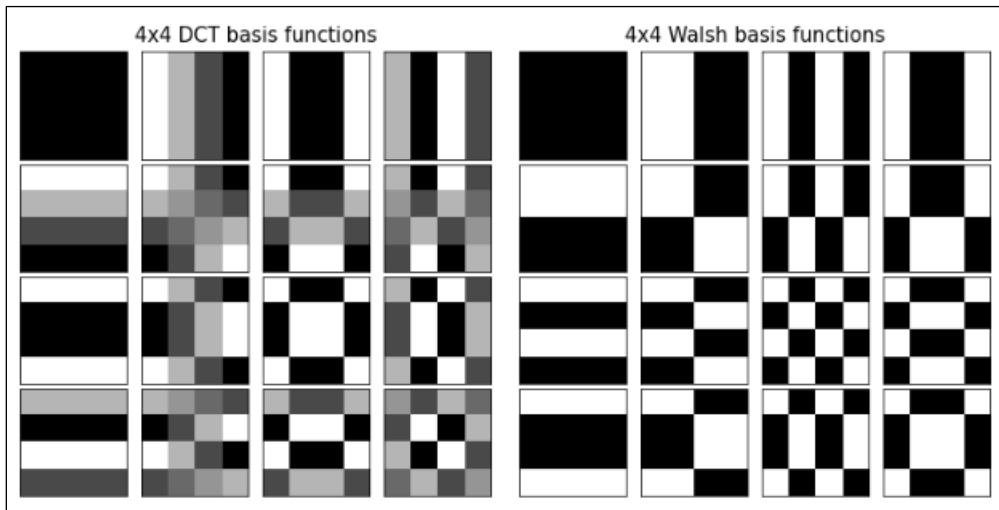
$n \rightarrow$ Number of bits x/u

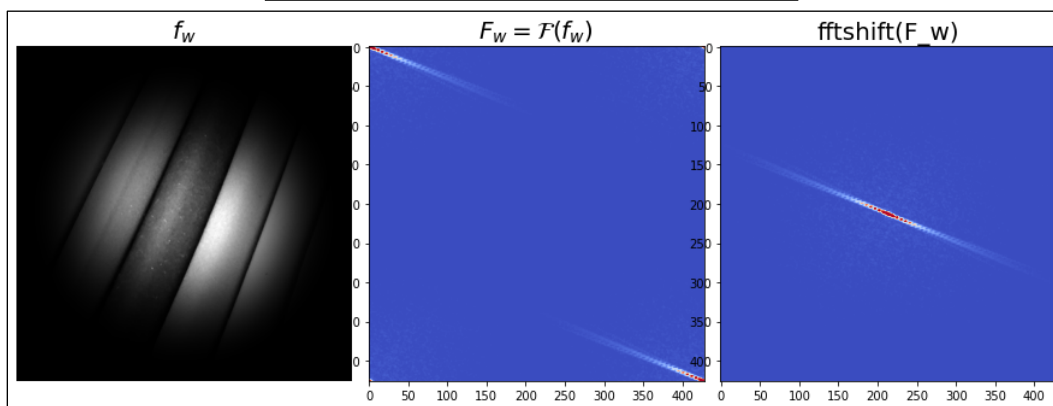
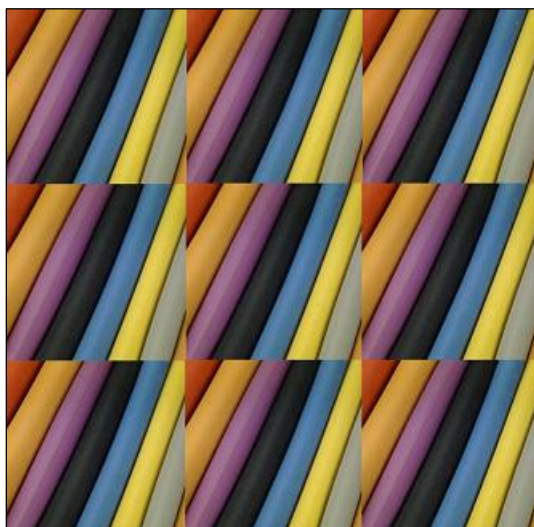
$b_k(z) \rightarrow k^{th}$ bit in digital/binary representation of z .

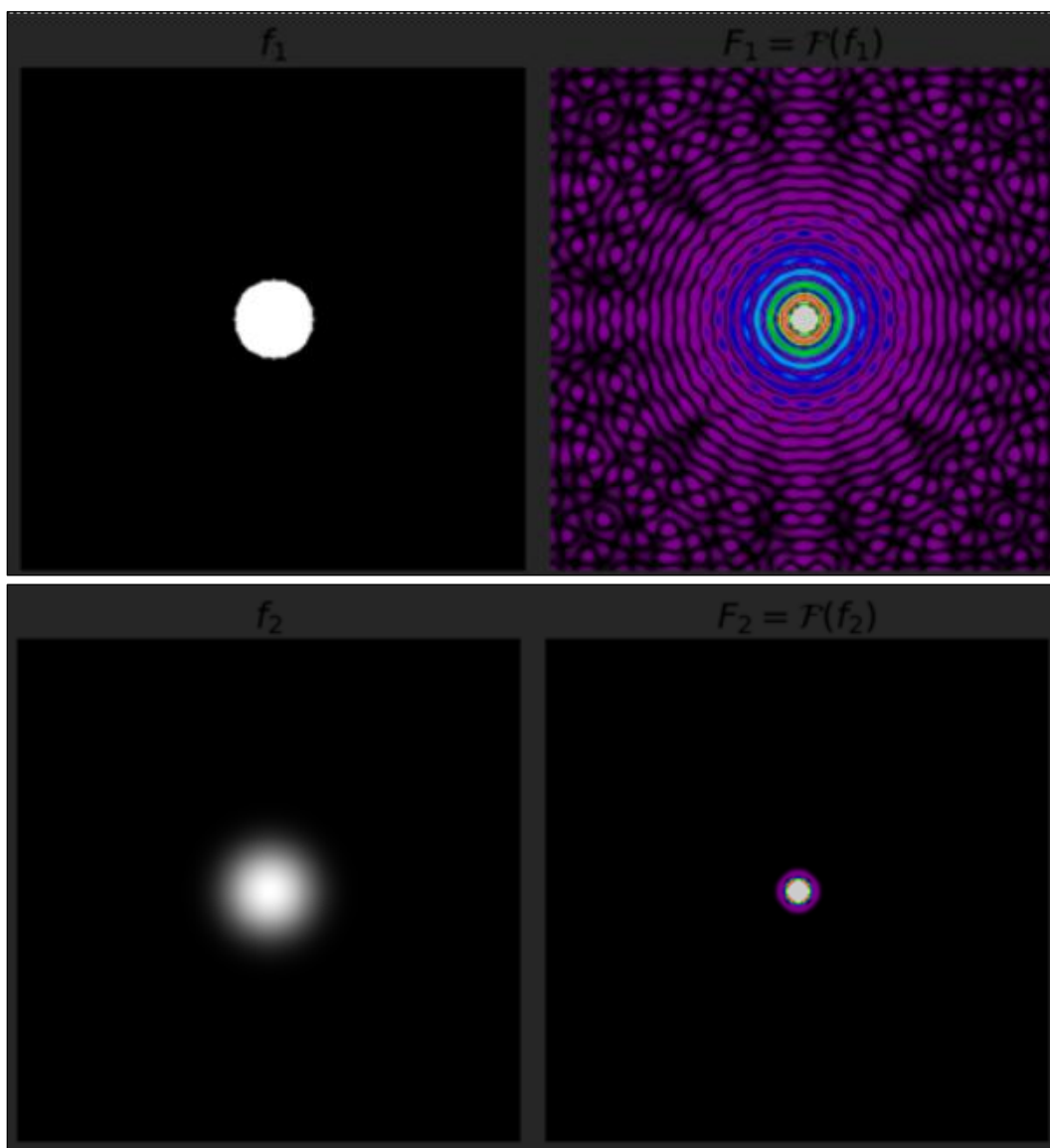




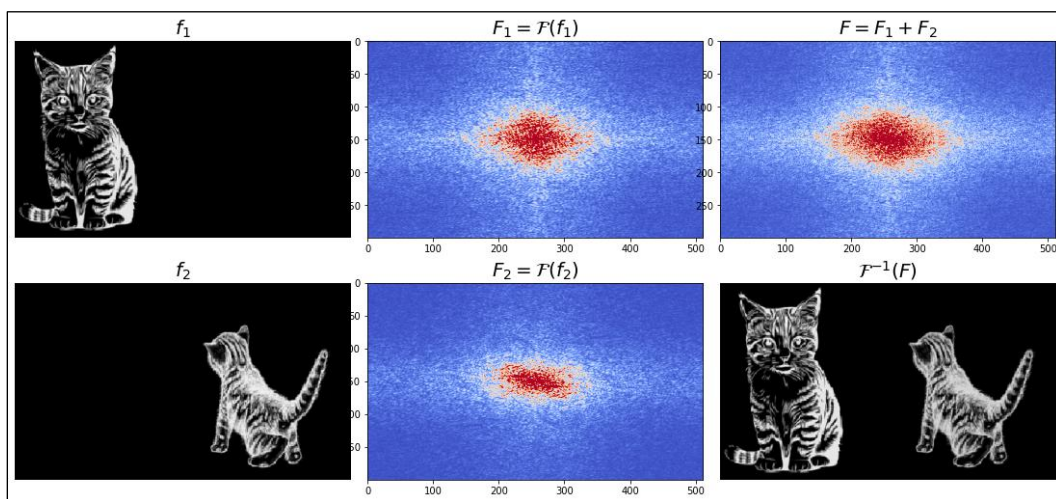
transformation			forward	Inverse
	$g_1(x, u)$	$g_1(y, v)$	$g(x, y, u, v)$	$h(x, y, u, v)$
DFT	$\frac{1}{\sqrt{N}} e^{-j\frac{2\pi}{N}ux}$	$\frac{1}{\sqrt{N}} e^{-j\frac{2\pi}{N}vy}$	$\frac{1}{N} e^{-j\frac{2\pi}{N}(ux+vy)}$	$\frac{1}{N} e^{j\frac{2\pi}{N}(ux+vy)}$
DCT	$\alpha(u) \cos \left[\frac{(2x+1)u\pi}{2N} \right]$	$\alpha(v) \cos \left[\frac{(2y+1)v\pi}{2N} \right]$	$\alpha(u)\alpha(v) \cos \left[\frac{(2x+1)u\pi}{2N} \right] \cos \left[\frac{(2y+1)v\pi}{2N} \right]$	$\alpha(u)\alpha(v) \cos \left[\frac{(2x+1)u\pi}{2N} \right] \cos \left[\frac{(2y+1)v\pi}{2N} \right]$
Walsh	$\frac{1}{N} \prod_{i=0}^{n-1} (-1)^{b_i(x)b_{n-1-i}(u)}$	$\frac{1}{N} \prod_{i=0}^{n-1} (-1)^{b_i(y)b_{n-1-i}(v)}$	$\frac{1}{N} \prod_{i=0}^{n-1} (-1)^{b_i(x)b_{n-1-i}(u)+b_i(y)b_{n-1-i}(v)}$	$\frac{1}{N} \prod_{i=0}^{n-1} (-1)^{b_i(x)b_{n-1-i}(u)+b_i(y)b_{n-1-i}(v)}$
Hadamard	$\frac{1}{N} \prod_{i=0}^{n-1} (-1)^{b_i(x)b_i(u)}$	$\frac{1}{N} \prod_{i=0}^{n-1} (-1)^{b_i(y)b_i(v)}$	$\frac{1}{N} \prod_{i=0}^{n-1} (-1)^{b_i(x)b_i(u)+b_i(y)b_i(v)}$	$\frac{1}{N} \prod_{i=0}^{n-1} (-1)^{b_i(x)b_i(u)+b_i(y)b_i(v)}$







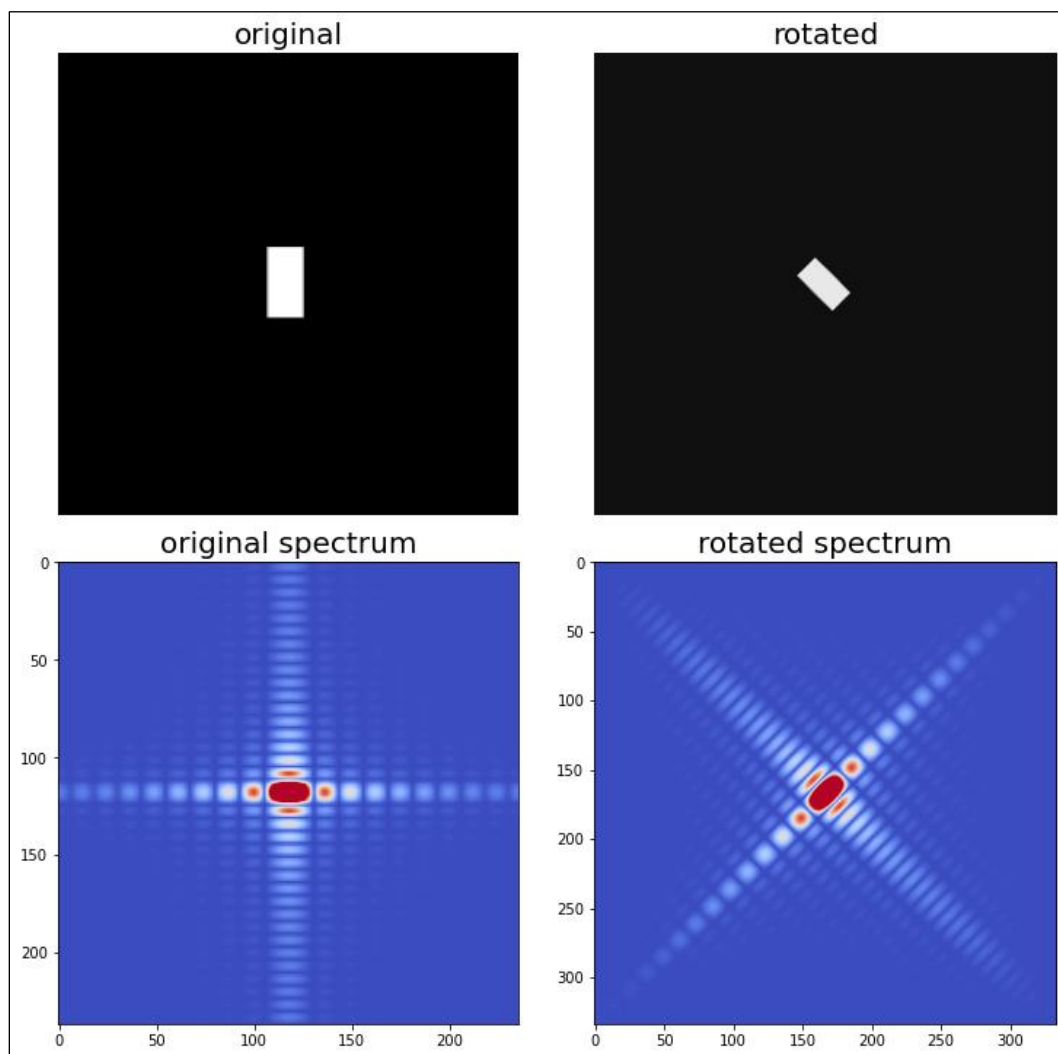
$$\begin{aligned}\mathcal{F}(f + g) &= \mathcal{F}(f) + \mathcal{F}(g) \\ \mathcal{F}(kf) &= k\mathcal{F}(f) \quad \text{\textcolor{red}{k is a scalar}}\end{aligned}$$



$$f(r, \theta) \xleftrightarrow{\mathcal{F}} F(\omega, \phi)$$

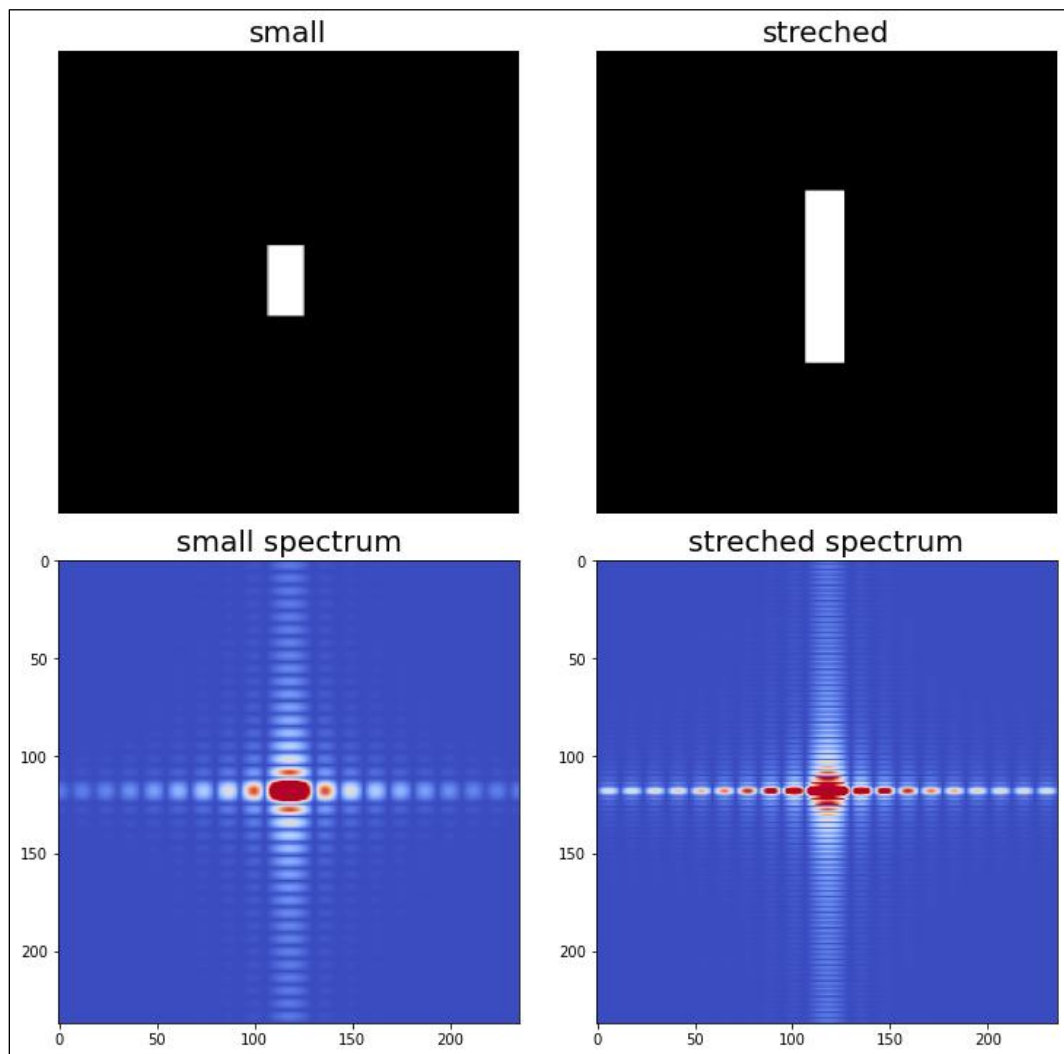
$$f(r, \theta + \alpha) \xleftrightarrow{\mathcal{F}} F(\omega, \phi + \alpha)$$

rotated by an angle α



$$f(x, y) \xleftrightarrow{\mathcal{F}} F(u, v).$$

$$f(\alpha x, \beta y) \xleftrightarrow{\mathcal{F}} \frac{1}{|\alpha\beta|} F\left(\frac{u}{\alpha}, \frac{v}{\beta}\right)$$



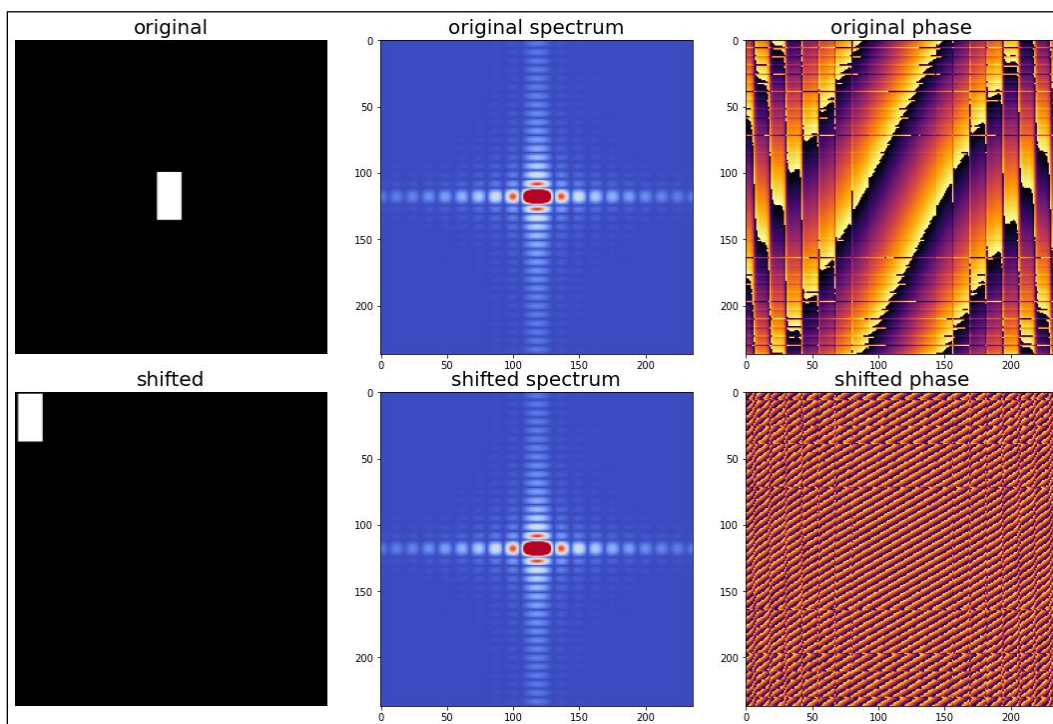
$$f(x, y) \xleftrightarrow{\mathcal{F}} F(u, v).$$

$$f(x - \alpha, y - \beta) \xleftrightarrow{\mathcal{F}} F(u, v) e^{-j(u\alpha + v\beta)}$$

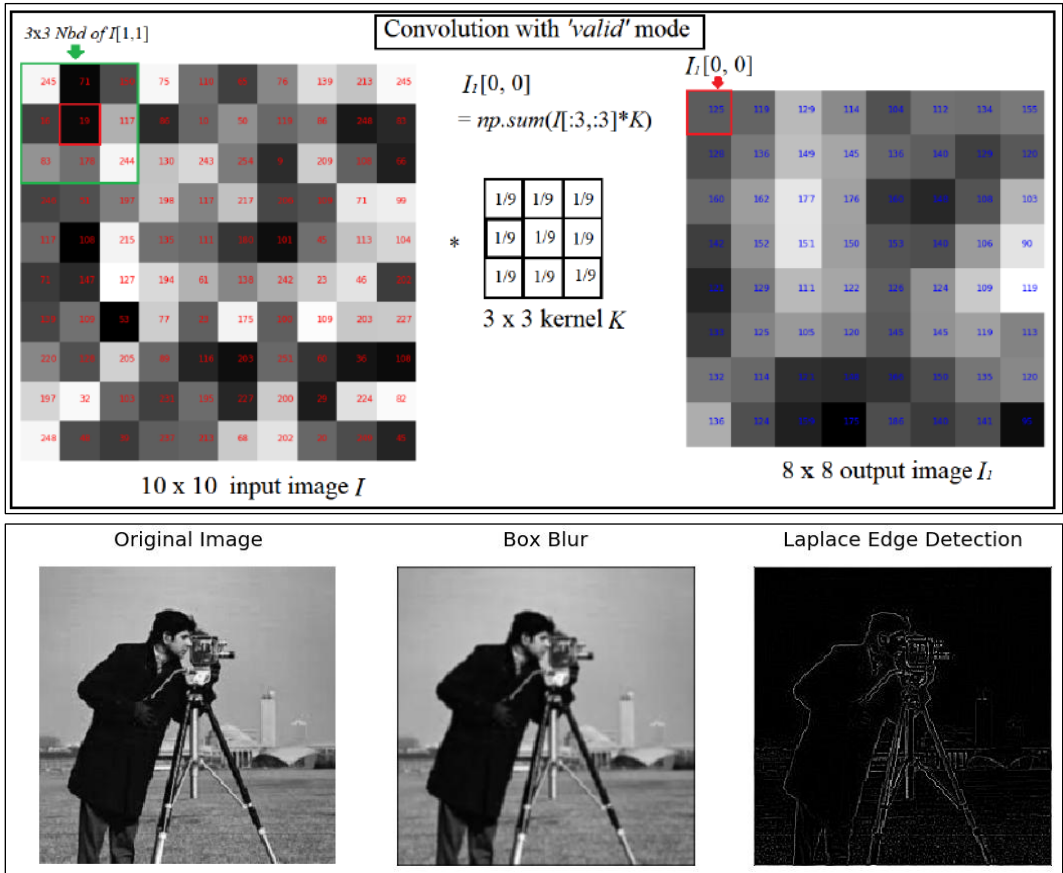
translation

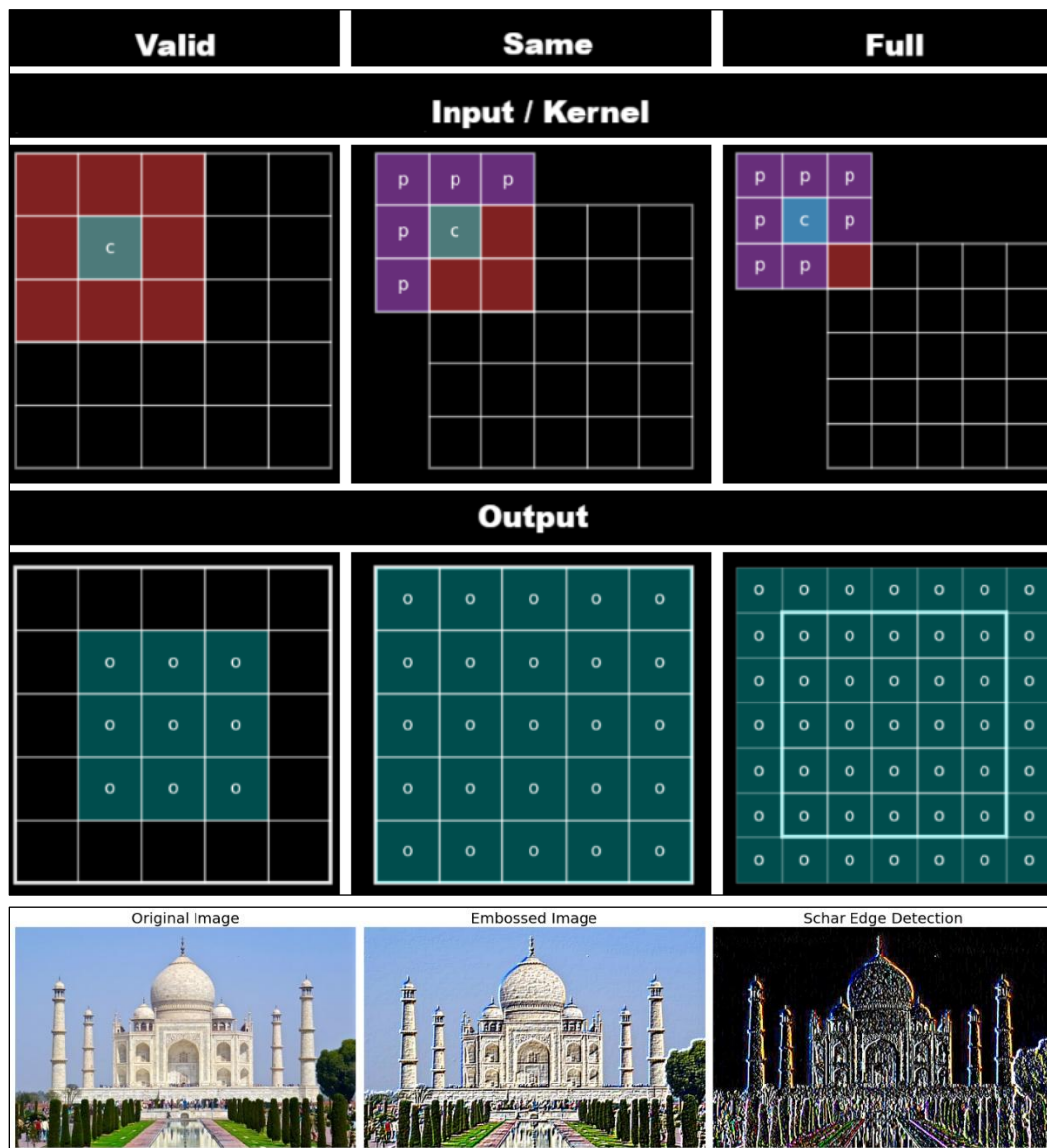
$$|F(u, v)| = |F(u, v) e^{-j(u\alpha + v\beta)}|$$

magnitude before translation magnitude after translation



Chapter 5: Convolution and Spatial/Frequency Domain Filtering





Original Image



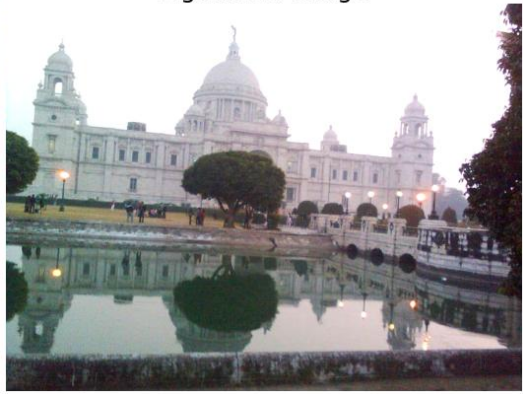
Sharpened Image



Embossed Image



Lightened Image





$$I = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

Input image

$$K = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Kernel

'valid' mode $\Rightarrow$ output is $(4 - 2 + 1) \times (4 - 2 + 1) = 3 \times 3$

2D Correlation (no flipping)

Top-left output (0,0):

$$\begin{bmatrix} 1 & 2 \\ 5 & 6 \end{bmatrix} \odot \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = 1 \times 1 + 2 \times 2 + 5 \times 3 + 6 \times 4 = 1 + 4 + 15 + 24 = \boxed{44}$$

Next output (0,1):

$$\begin{bmatrix} 2 & 3 \\ 6 & 7 \end{bmatrix} \odot \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = 2 + 6 + 18 + 28 = \boxed{54}$$

Next output (0,2):

$$3 \times 1 + 4 \times 2 + 7 \times 3 + 8 \times 4 = 3 + 8 + 21 + 32 = \boxed{64}$$

Final correlation output: = $\begin{bmatrix} 44 & 54 & 64 \\ 84 & 94 & 104 \\ 124 & 134 & 144 \end{bmatrix}$

2D Convolution (kernel is flipped)

Top-left output (0,0):

$$\begin{bmatrix} 1 & 2 \\ 5 & 6 \end{bmatrix} \odot \begin{bmatrix} 4 & 3 \\ 2 & 1 \end{bmatrix} = 1 \times 4 + 2 \times 3 + 5 \times 2 + 6 \times 1 = 4 + 6 + 10 + 6 = \boxed{26}$$

Next output (0,1):

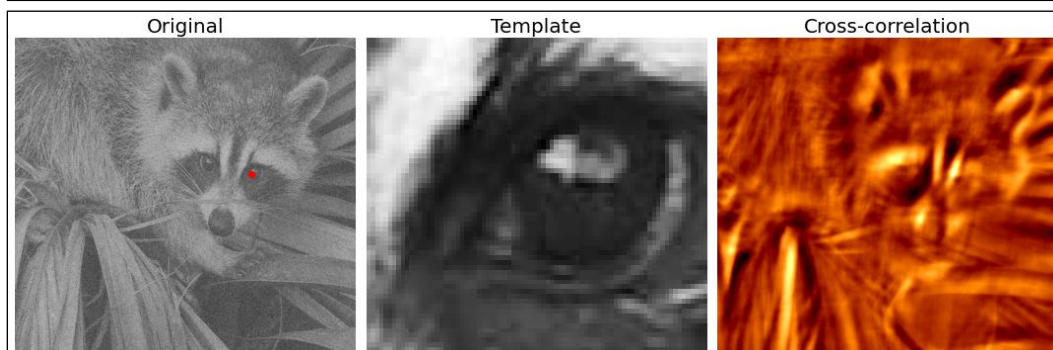
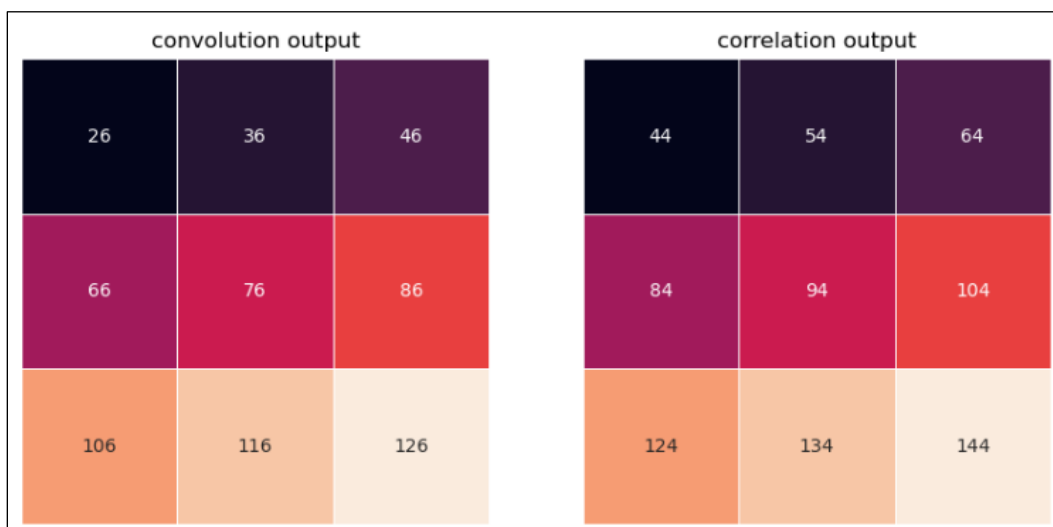
$$2 \times 4 + 3 \times 3 + 6 \times 2 + 7 \times 1 = 8 + 9 + 12 + 7 = \boxed{36}$$

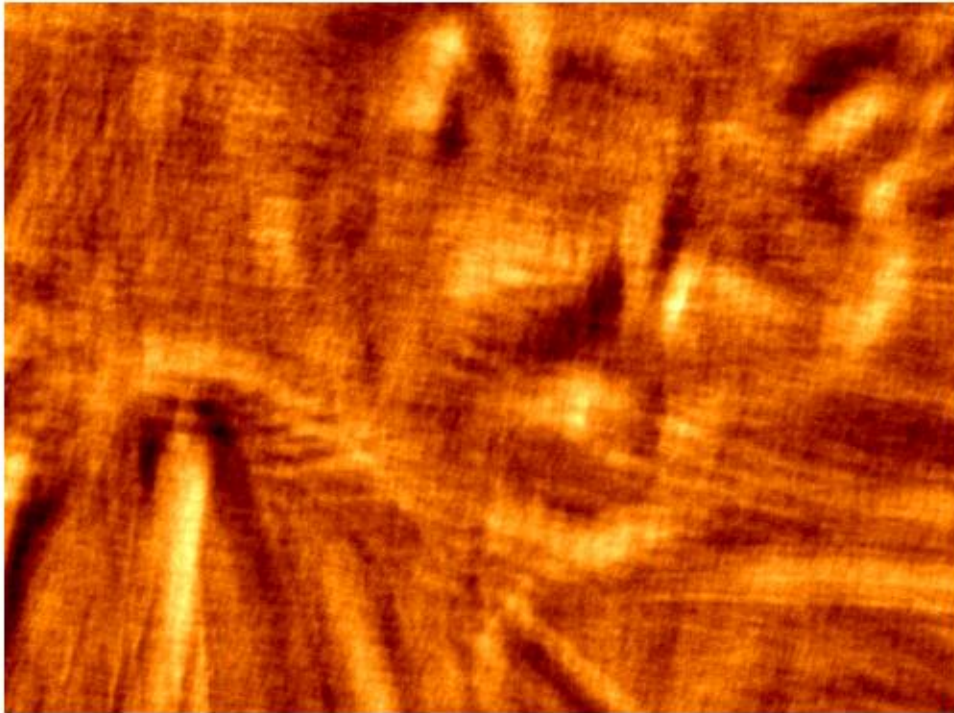
Next output (0,2):

$$3 \times 4 + 4 \times 3 + 7 \times 2 + 8 \times 1 = 12 + 12 + 14 + 8 = \boxed{46}$$

Final convolution output: = $\begin{bmatrix} 26 & 36 & 46 \\ 66 & 76 & 86 \\ 106 & 116 & 126 \end{bmatrix}$

	Original Kernel	Flipped Kernel (Convolution)	
Kernel	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$	$\begin{bmatrix} 4 & 3 \\ 2 & 1 \end{bmatrix}$	
	Input Image (4×4)	Result: Correlation	Result: Convolution
	$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$	$\begin{bmatrix} 44 & 54 & 64 \\ 84 & 94 & 104 \\ 124 & 134 & 144 \end{bmatrix}$	$\begin{bmatrix} 26 & 36 & 46 \\ 66 & 76 & 86 \\ 106 & 116 & 126 \end{bmatrix}$





Convolution theorem

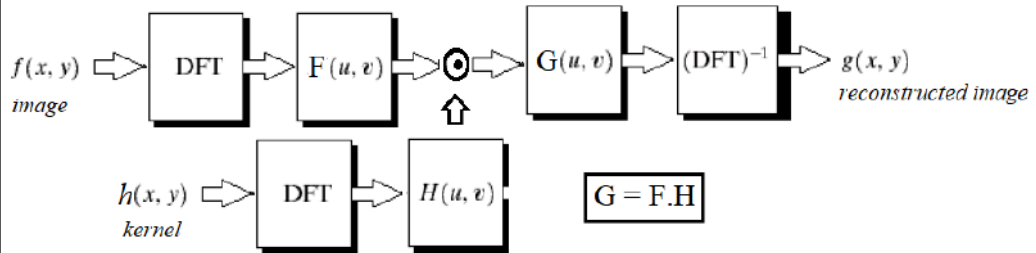
$$f(x, y) * h(x, y) \Leftrightarrow F(u, v) H(u, v)$$

Space convolution = frequency multiplication

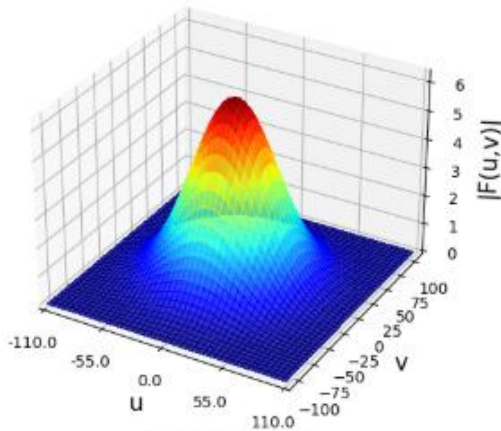
$$F(u, v) * H(u, v) \Leftrightarrow f(x, y) h(x, y)$$

frequency convolution = Space multiplication

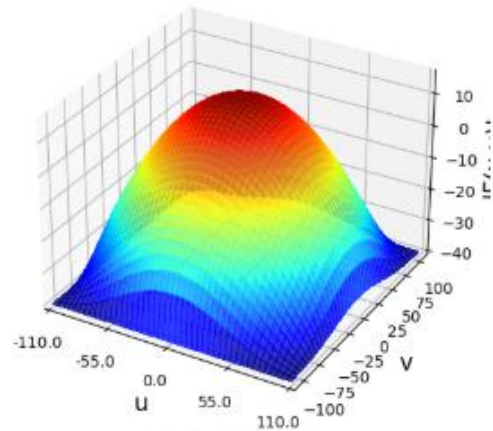
Convolution in the Frequency Domain



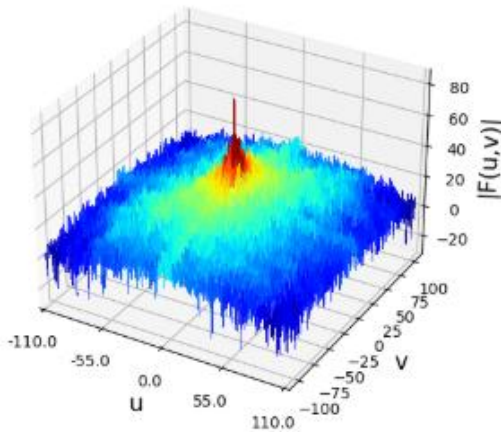
$|H(u, v)|$



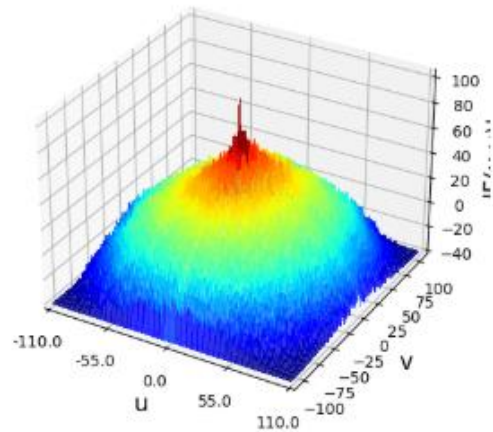
$\log |H(u, v)|$

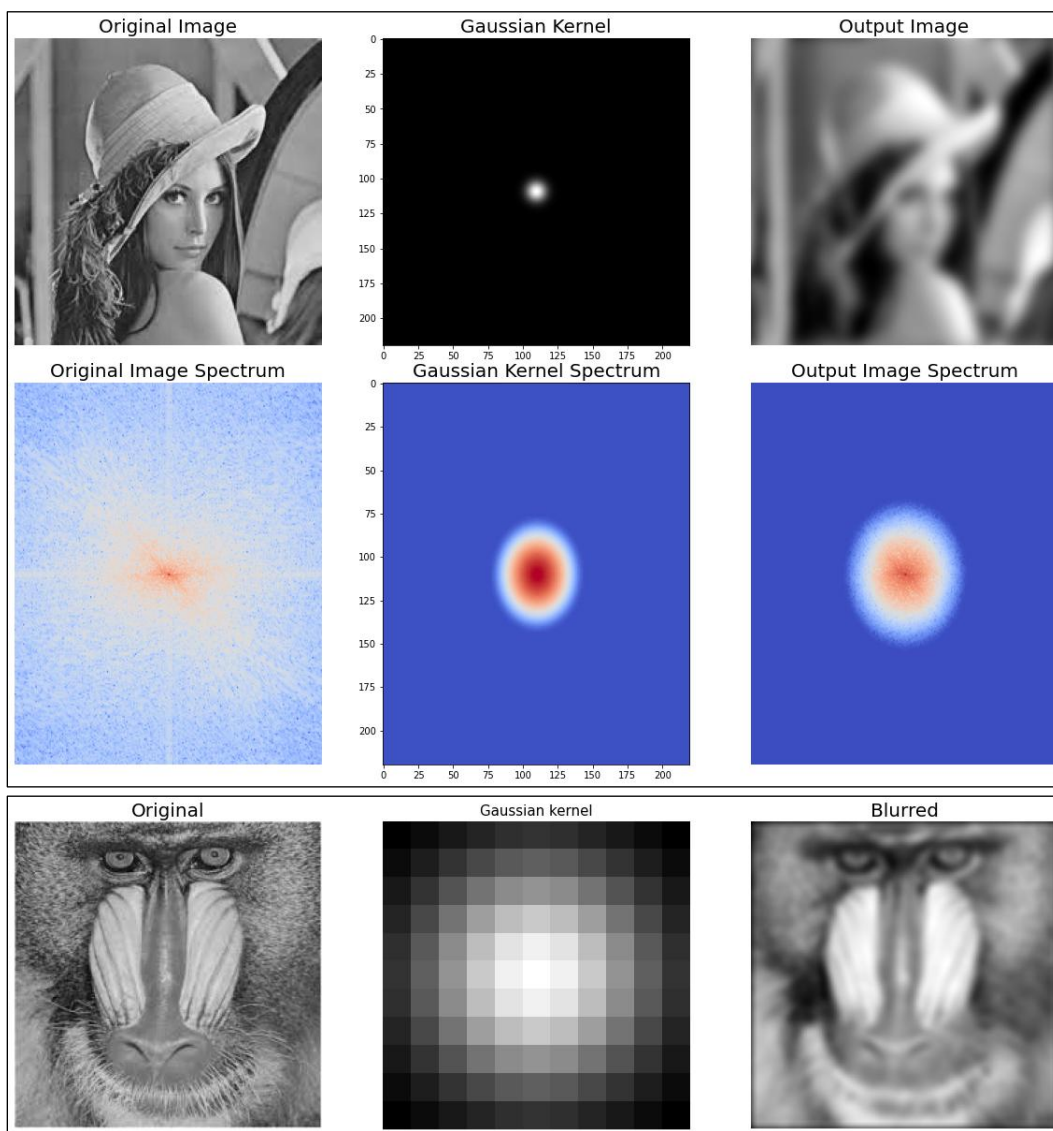


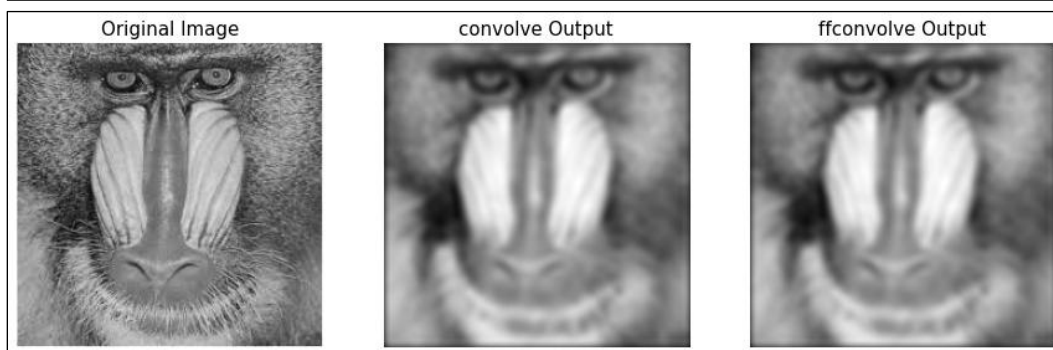
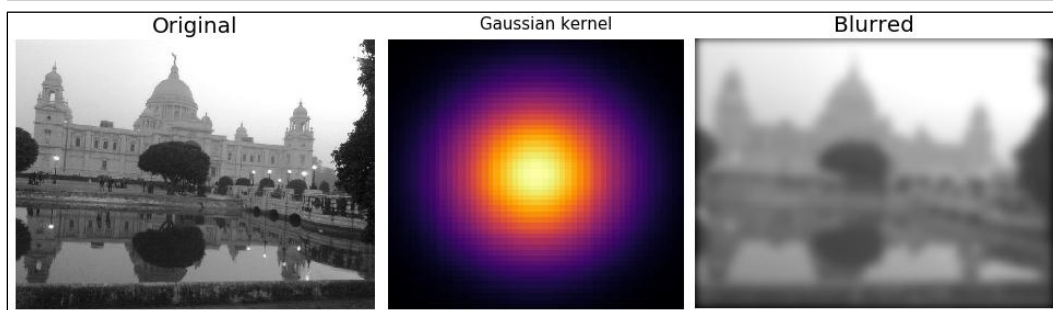
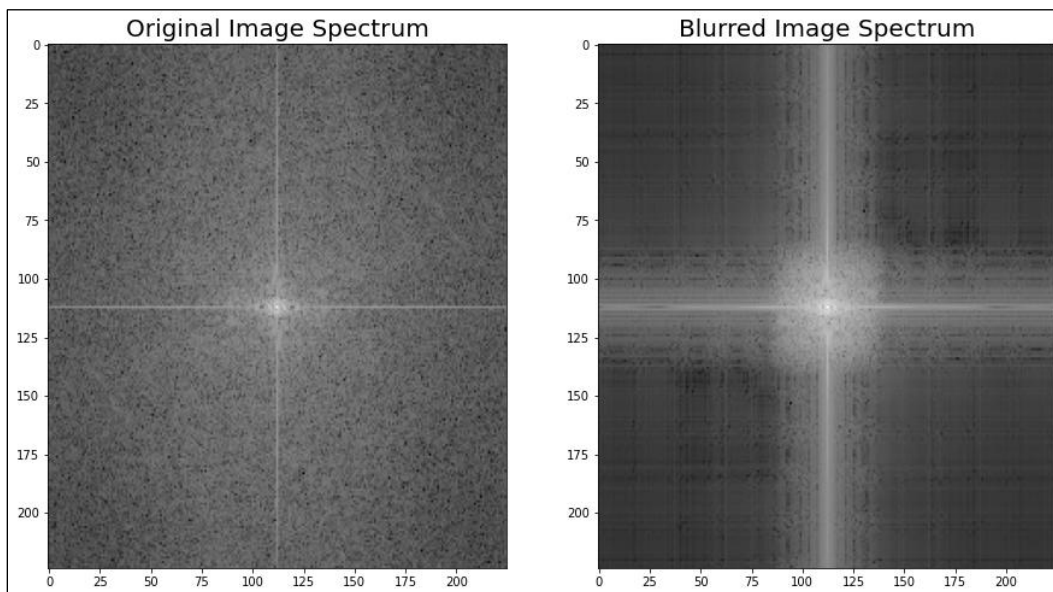
$\log |F(u, v)|$



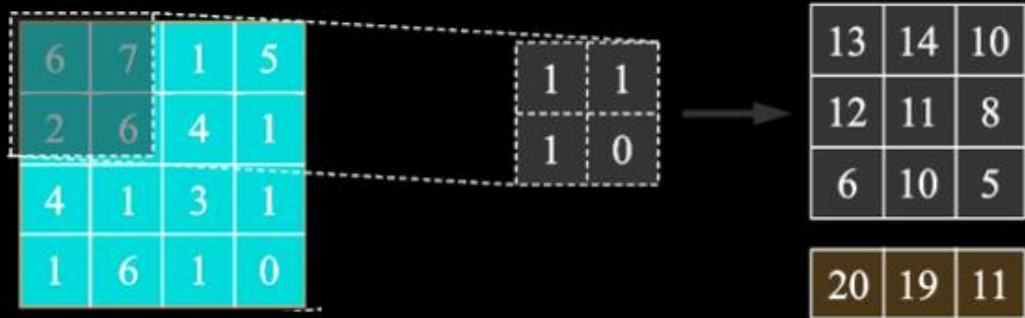
$\log |G(u, v)|$



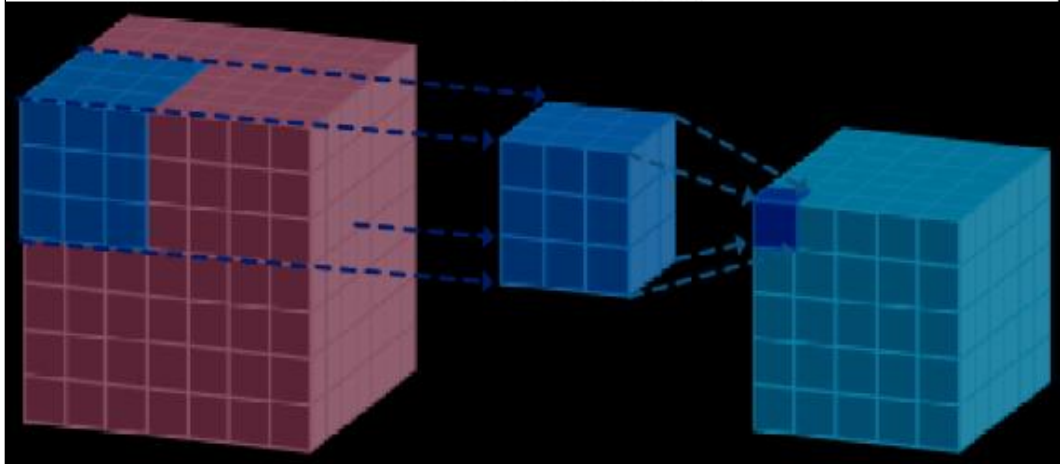


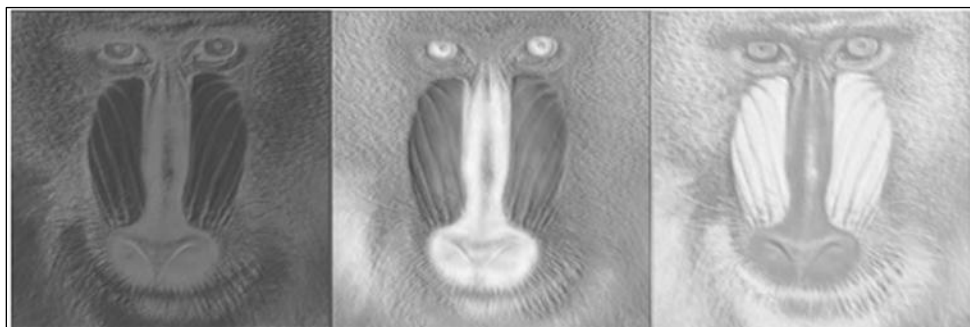
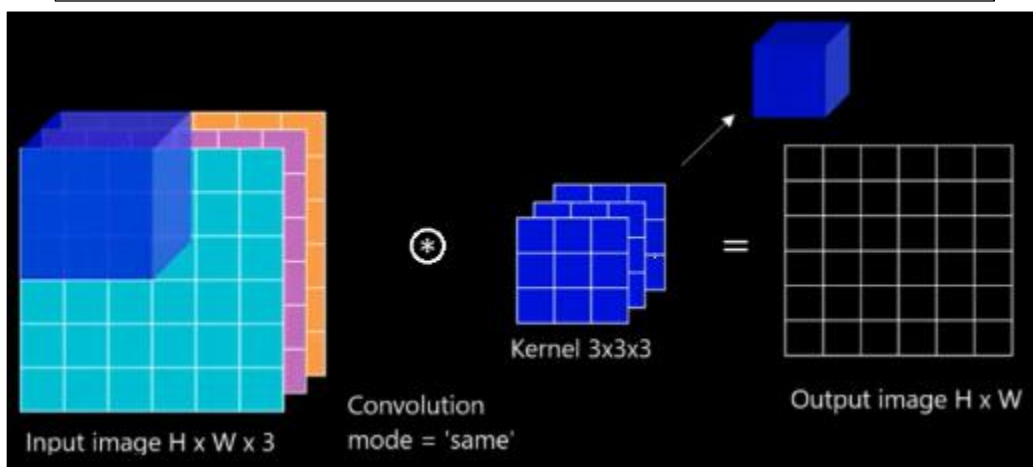
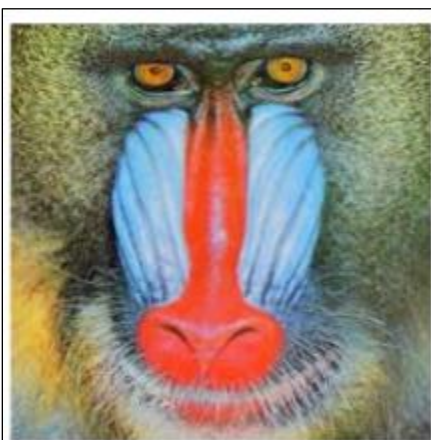


2D Convolution

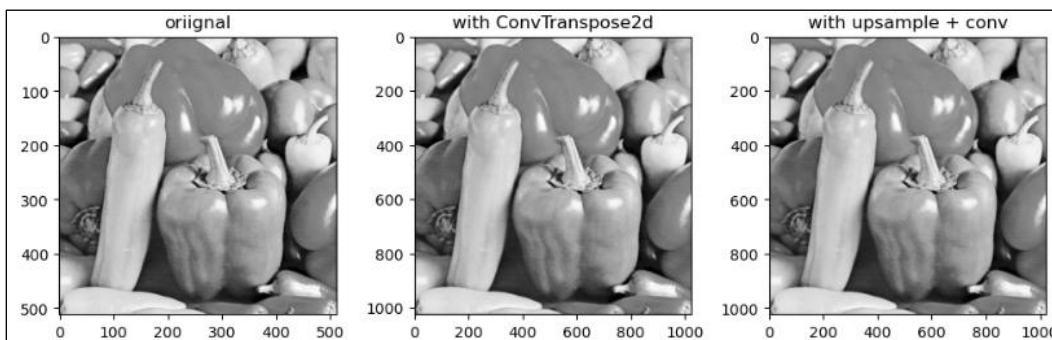


3D Convolution





Input					
conv2d	batch	in_height	in_width	in_channels	
conv3d	batch	in_depth	in_height	in_width	in_channels
Output					
conv2d	batch	filter_height	filter_width	out_channels	
conv3d	batch	filter_depth	filter_height	filter_width	out_channels
	Conv Direction	Input	Filter	Output	
conv2d	2-direction 	4-dim	4-dim	3-dim	
conv3d	3-direction 	5-dim	5-dim	4-dim	

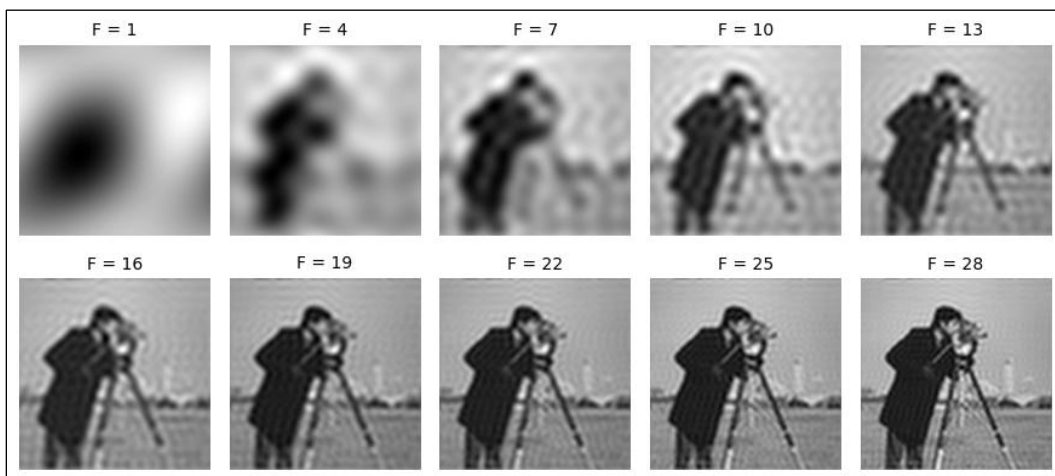
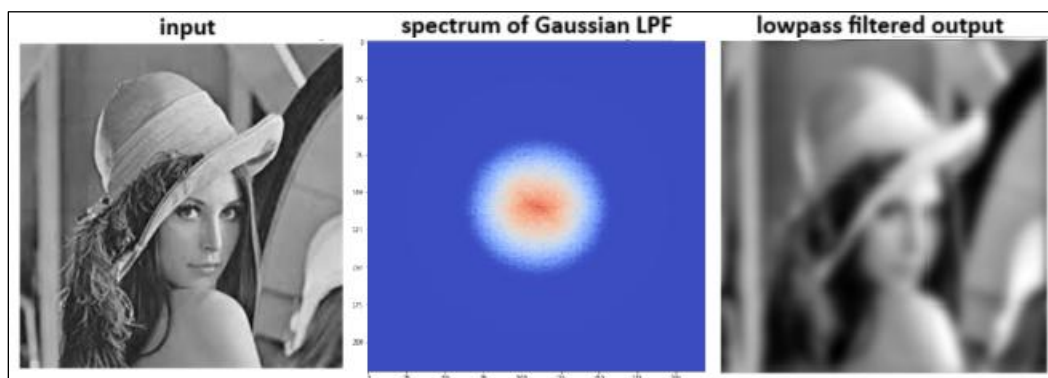


Chapter 6: Frequency Domain Filtering

Filter Type	Mathematical Expression	Characteristics	Pros	Cons
Ideal LPF (ILPF)	$H(u, v) = \begin{cases} 1, & D(u, v) \leq D_0 \\ 0, & D(u, v) > D_0 \end{cases}$	Sharp transition, removes all high frequencies above D_0	Strong smoothing effect, simple to implement	Causes blurring artifacts due to sudden cutoff
Butterworth LPF (BLPF)	$H(u, v) = \frac{1}{1 + \left(\frac{D(u, v)}{D_0}\right)^{2n}}$	Smooth transition controlled by order n	Less blurring compared to Ideal LPF, adjustable smoothness	Higher computational cost compared to Ideal LPF
Gaussian LPF (GLPF)	$H(u, v) = e^{-\frac{D^2(u, v)}{2D_0^2}}$	Very smooth transition, no ringing artifacts	No ringing artifacts, best for natural image smoothing	Weaker attenuation of high frequencies compared to ILPF and BLPF

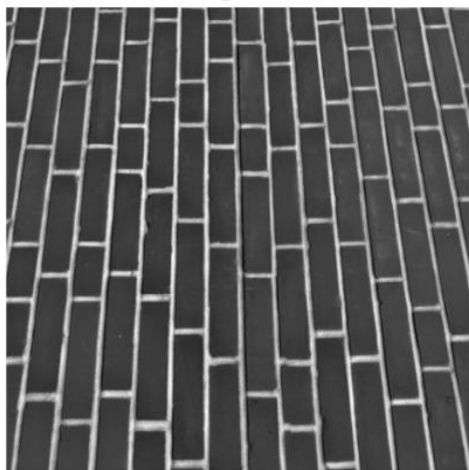
Parameter	Type	Description
<code>f</code>	<code>numpy.ndarray</code>	Input image (grayscale, stored as a 2D array).
<code>func</code>	<code>function</code>	A filtering function (e.g., <code>ideal</code> , <code>gaussian</code> , or <code>butterworth</code>).
<code>arg_dict</code>	<code>dict</code>	Keyword arguments (<code>kwargs</code>) passed to the filter function.

Output	Description
<code>F</code>	Original frequency spectrum of the image (before filtering).
<code>H</code>	The applied filter in the frequency domain.
<code>G</code>	Filtered frequency spectrum (after applying <code>H</code>).
<code>g</code>	Final processed image after inverse FFT (low-pass filtered image).

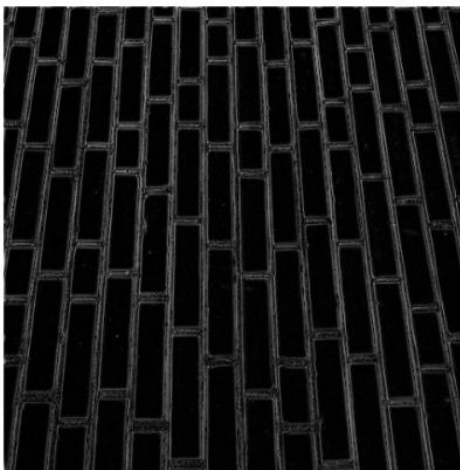


Filter Type	Mathematical Expression	Characteristics	Pros	Cons
Ideal HPF (IHPF)	$H(u, v) = \begin{cases} 0, & D(u, v) \leq D_0 \\ 1, & D(u, v) > D_0 \end{cases}$	Sharp transition, removes all low frequencies below D_0	Strong filtering effect, simple to implement	Causes ringing artifacts (Gibbs effect) due to the sudden cutoff
Butterworth HPF (BHPF)	$H(u, v) = \frac{1}{1 + \left(\frac{D_0}{D(u, v)}\right)^{2n}}$	Smooth transition controlled by order n	Less ringing compared to Ideal HPF, adjustable sharpness	Higher computational cost compared to Ideal HPF
Gaussian HPF (GHPF)	$H(u, v) = 1 - e^{-\frac{D^2(u, v)}{2D_0^2}}$	Very smooth transition, no ringing artifacts	No ringing artifacts, best for natural image processing	Weaker attenuation of low frequencies compared to IHPF and BHPF

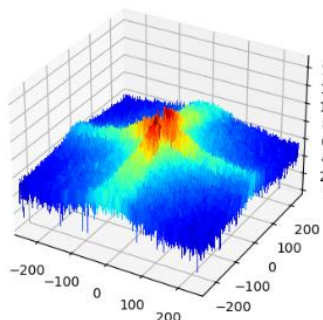
Original



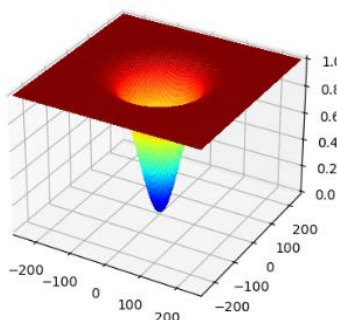
GHPF filtered



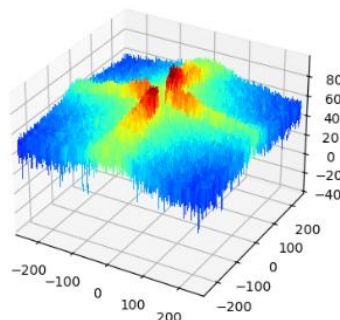
$\log F(u,v)$

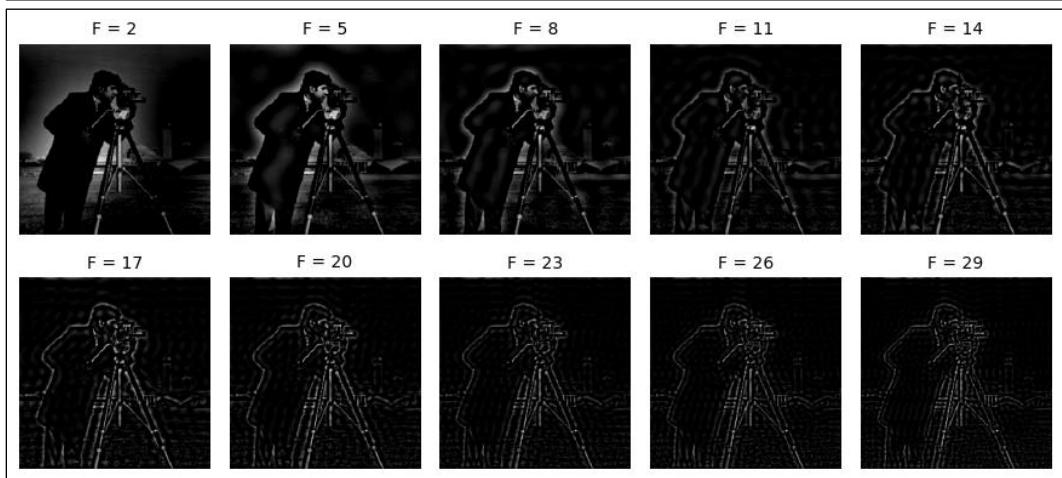
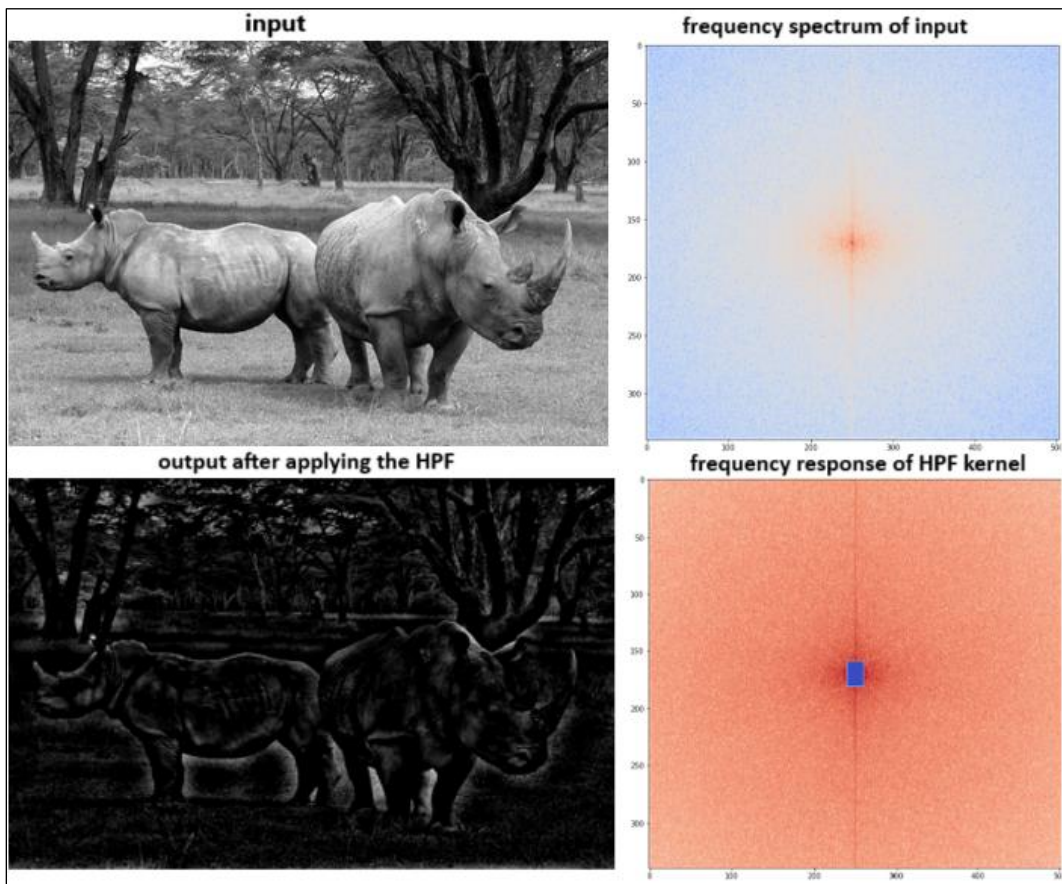


$H(u,v)$



$\log G(u,v)$





Parameter	Description
<code>image</code>	Input grayscale or color image
<code>cutoff_frequency_ratio</code>	Cutoff frequency as a fraction of the Nyquist frequency (0.0 to 1.0)
<code>order</code>	Order of the Butterworth filter (higher values create a sharper cutoff)
<code>high_pass</code>	If <code>False</code> , applies low-pass filtering ; if <code>True</code> , applies high-pass filtering
<code>squared_butterworth</code>	If <code>True</code> , applies the squared form of the filter for stronger effect
<code>npad</code>	Number of padding pixels before FFT (to reduce edge effects)
<code>channel_axis</code>	Specifies the axis for multi-channel images (RGB)



Step	Operation	Effect
1	Compute FFT	Converts image to frequency domain
2	Define frequency grid	Creates a frequency coordinate system
3	Compute distance D^2	Measures how far each frequency is from the center
4	Create Gaussian bandpass filter	Allows mid-range frequencies, removes very low & high frequencies
5	Apply filter	Multiplies filter with Fourier-transformed image
6	Compute inverse FFT	Converts filtered image back to spatial domain
7	Extract real part	Removes complex values

image with periodic noise



noisy image spectrum

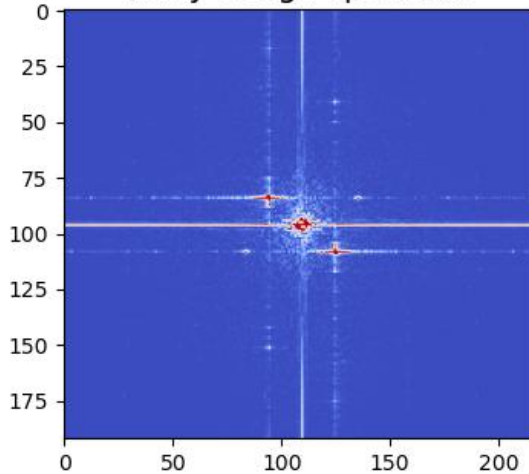
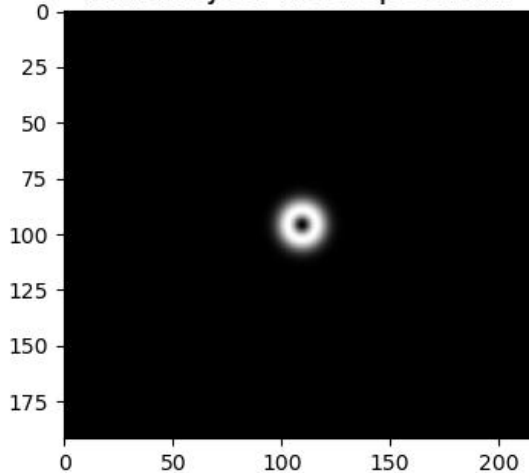


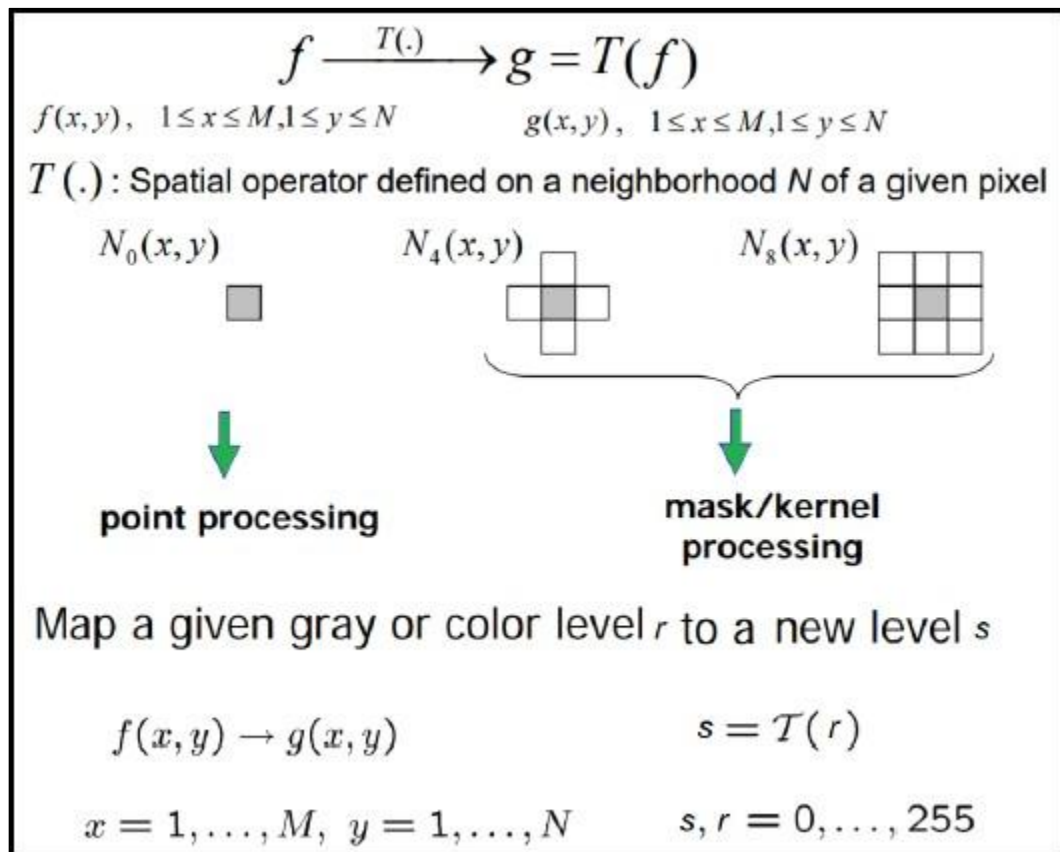
image with band-reject filter

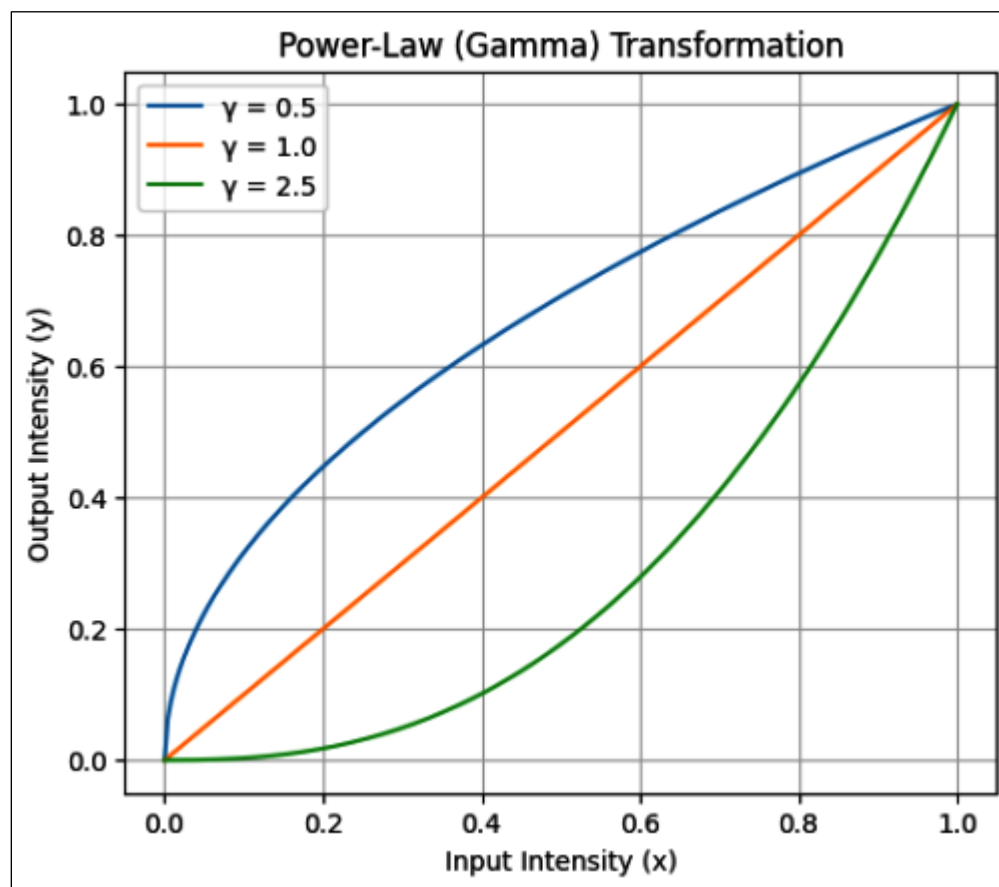


band reject filter spectrum



Chapter 7: Image Enhancement





original, increased brightness with $\alpha = 2$ and $\beta = 50$



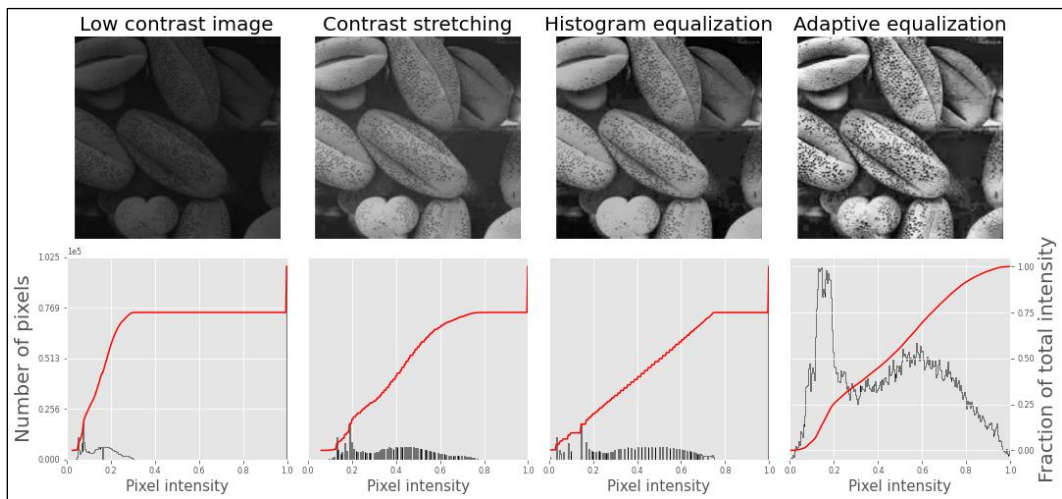
$$\begin{bmatrix} & & * & \frac{7}{16} & \cdots \\ \cdots & \frac{3}{16} & \frac{5}{16} & \frac{1}{16} & \cdots \end{bmatrix}$$

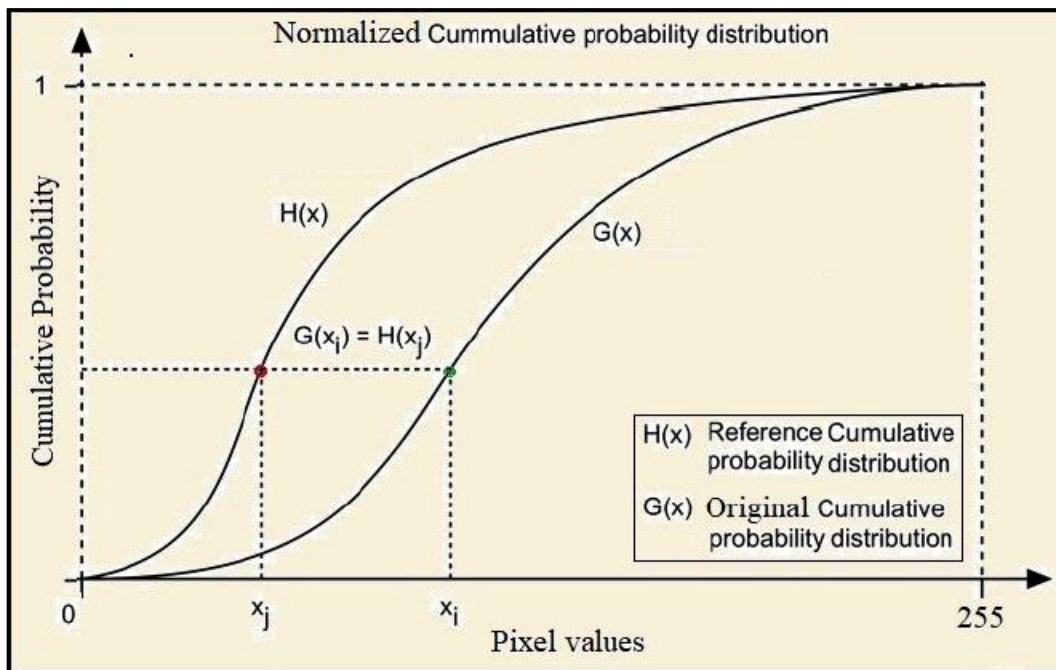
$$s_k = T(r_k) = \sum_{j=0}^k P_r(r_j) = \sum_{j=0}^k n_j / N$$

$$0 \leq r_k \leq 1, \quad k = 0, 1, 2, \dots, 255$$

N: total number of pixels

n_j : frequency of pixel with gray-level j





original, color enhanced and desaturated images

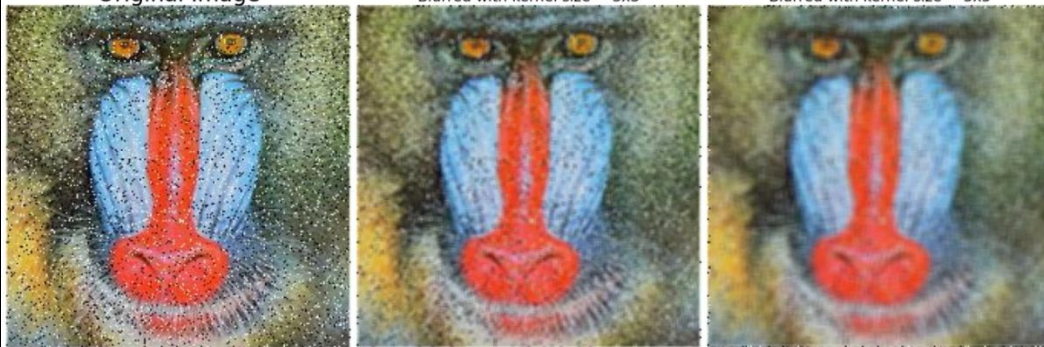


PIL Mean Filter (Box Blur) with different Kernel size

Original Image

Blurred with kernel size = 3x3

Blurred with kernel size = 5x5



PIL Gaussian Blur with different Radius

radius = 1



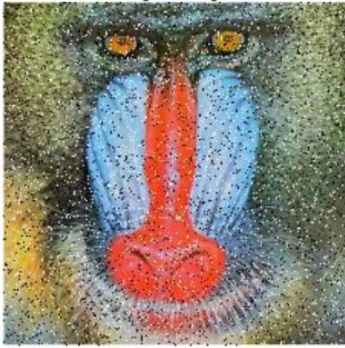
radius = 2



radius = 3



original image



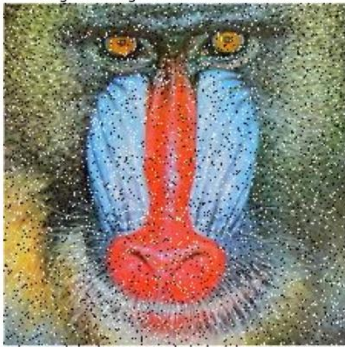
with the box filter



with the gaussian filter



Original Image with 10% added noise

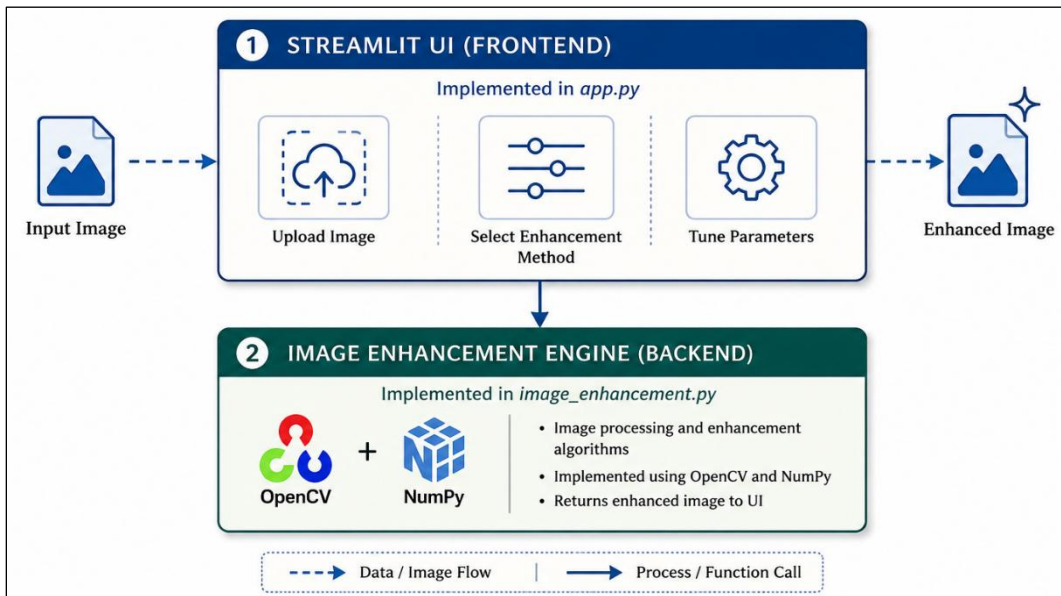


Output (Max Filter size=3)

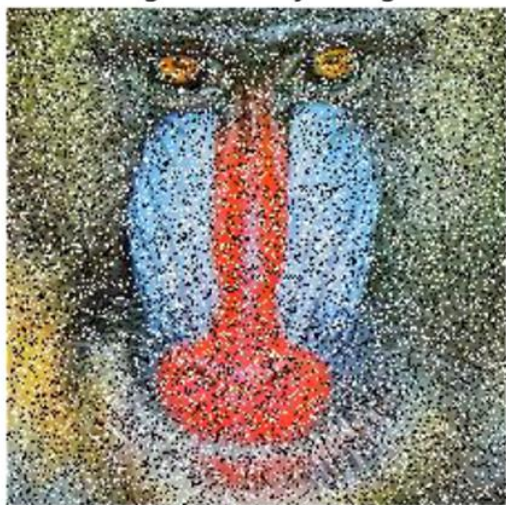


Output (Min Filter size=3)





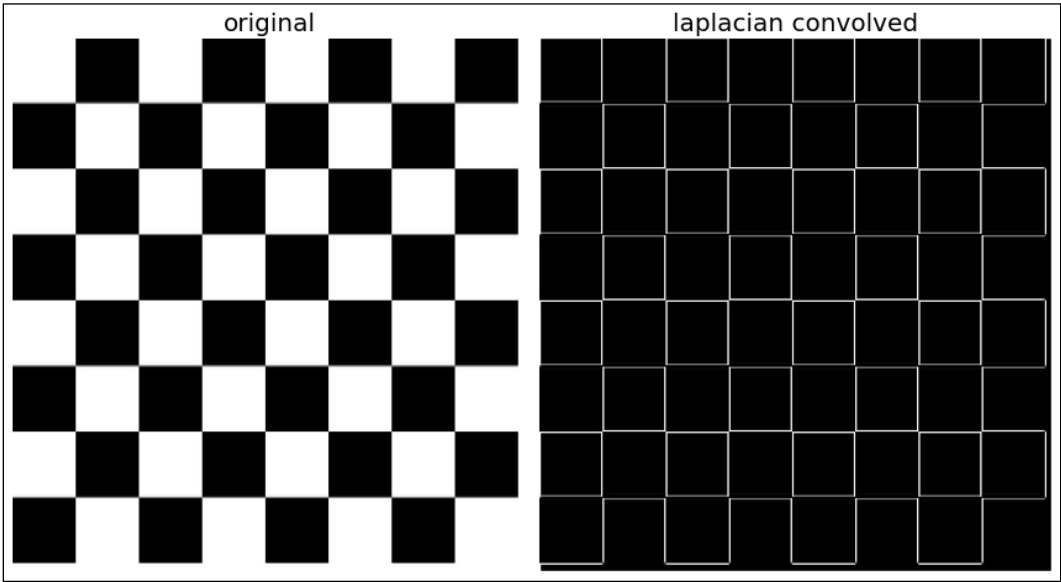
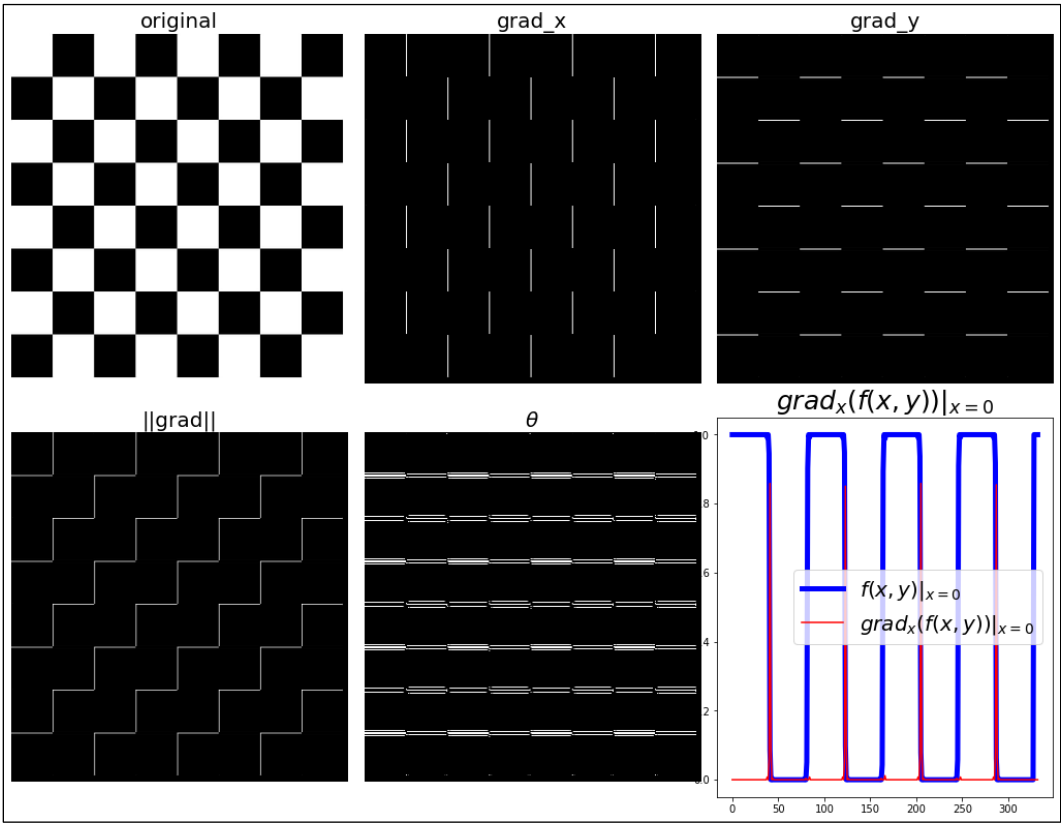
Original Noisy Image

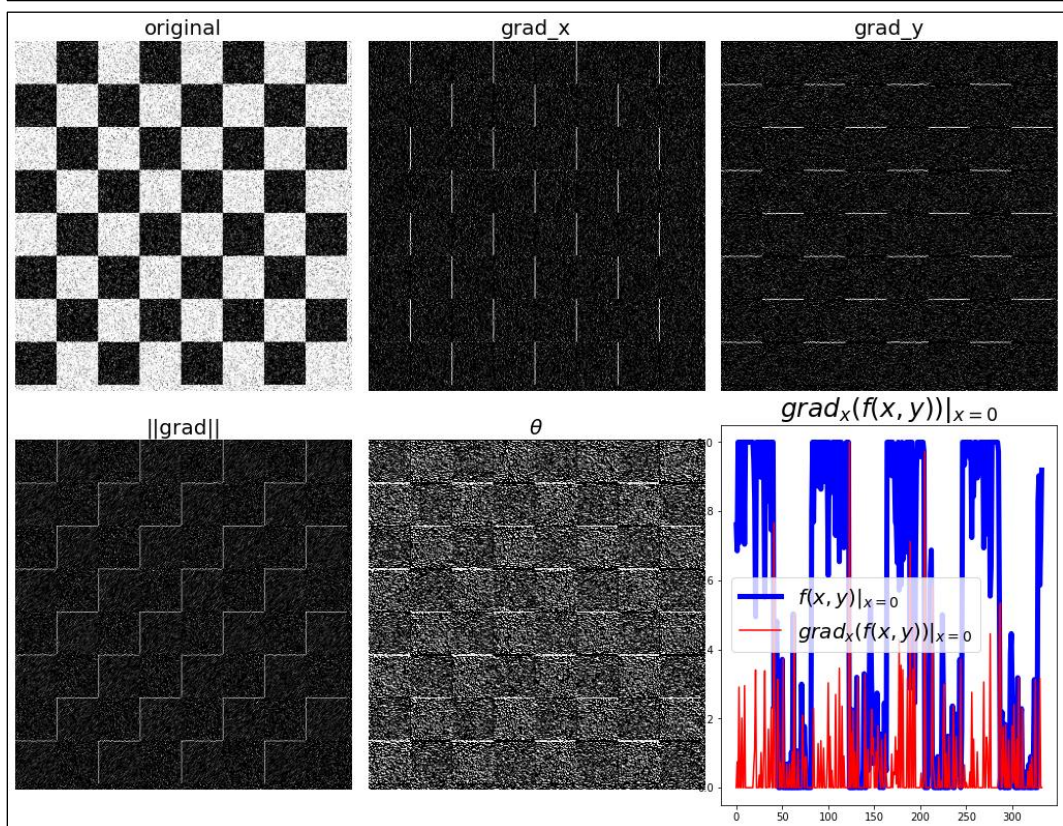
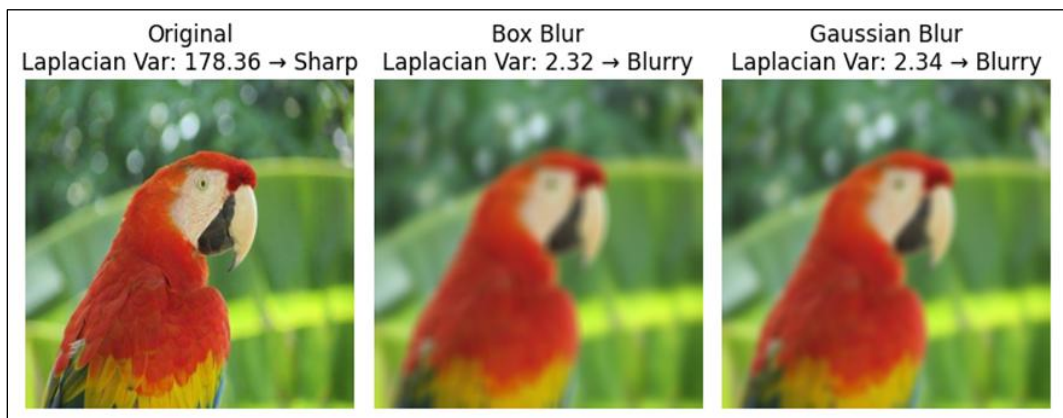


Denoised with BM3D



Chapter 8: Image Enhancements Using Derivatives





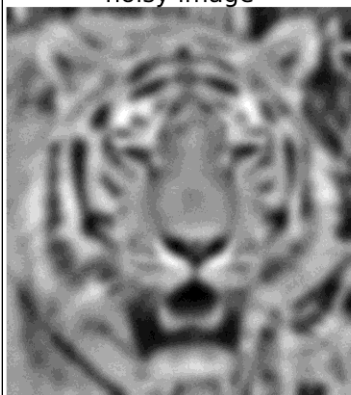
Original Image



Kirsch Compass Edges



noisy image



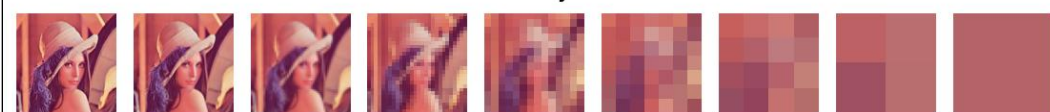
Canny filter, $\sigma = 1$



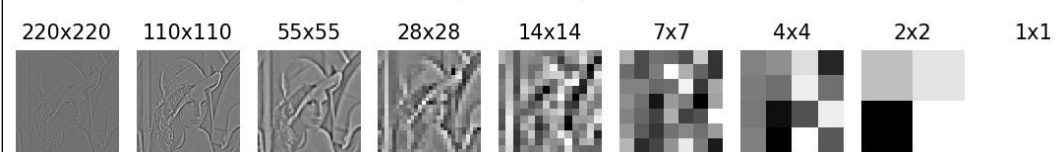
Canny filter, $\sigma = 3$

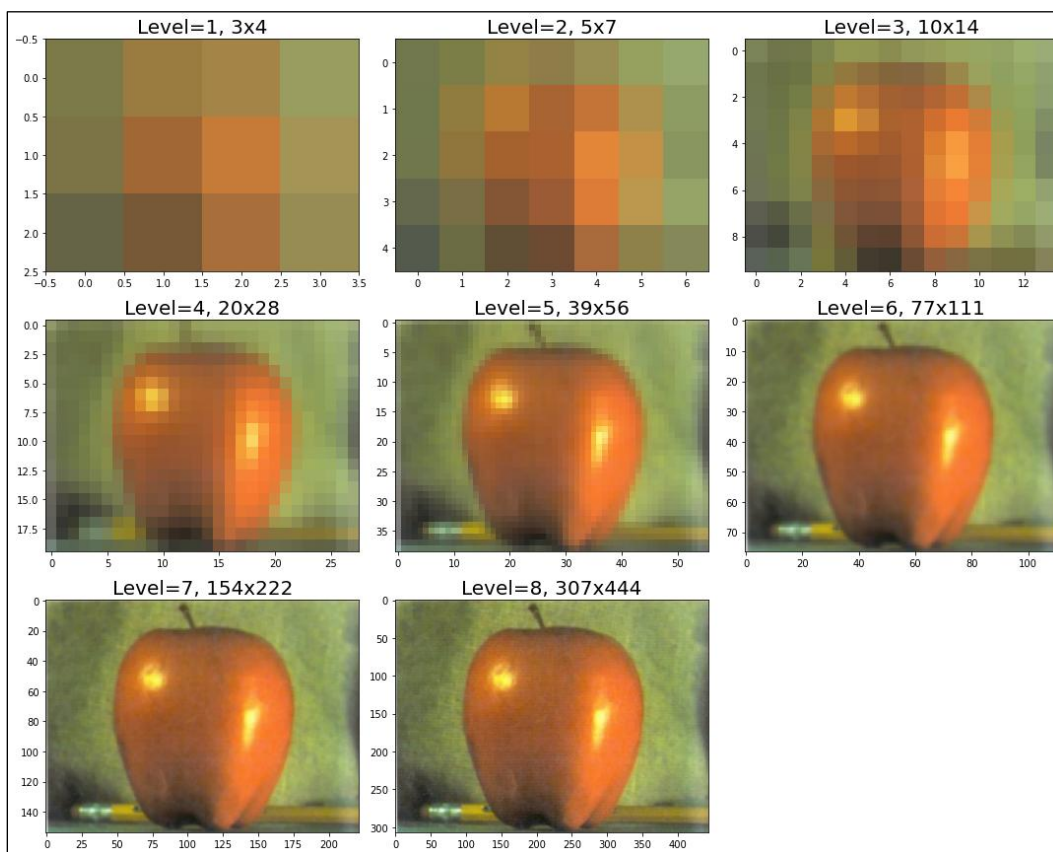


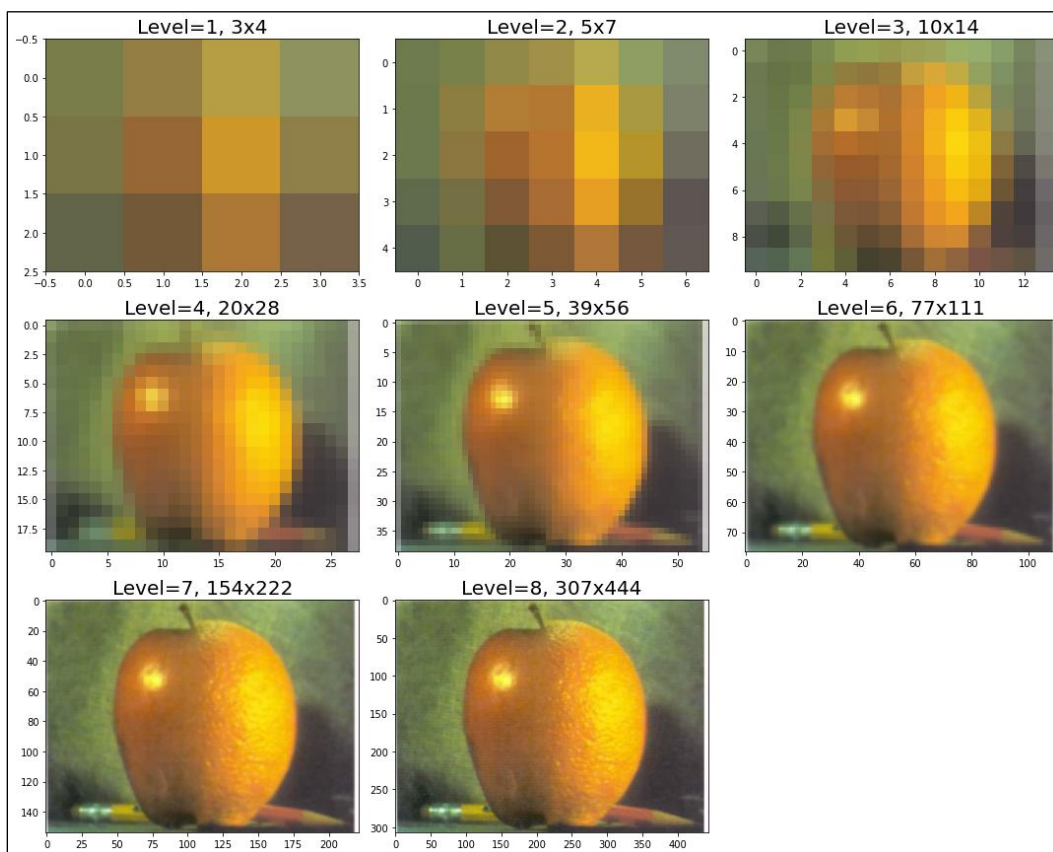
Gaussian Pyramid



Laplacian Pyramid







Input Apple Image (A)



Input Orange Image (B)



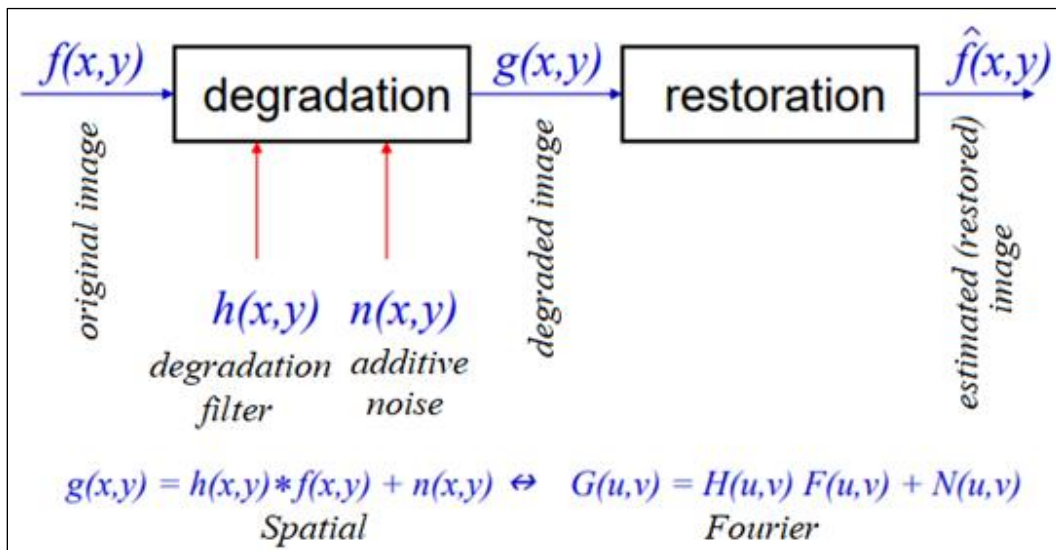
Input Mask Image (M)



Output Image with pyramid blending (I)



Chapter 9: Image Restoration: Inverse Problems in Imaging



Original image



Blurred image with motion blur kernel



Restored image with inverse filter



Diff restored - original image



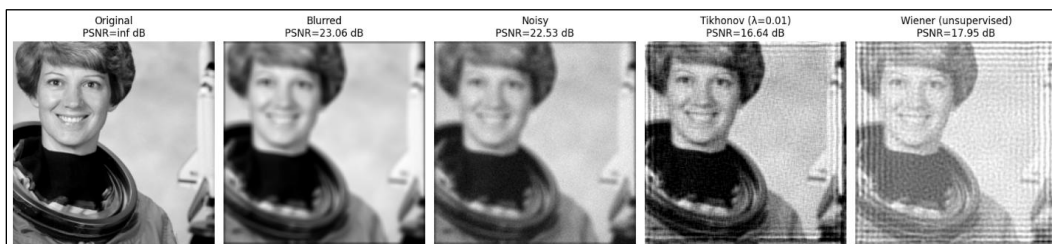
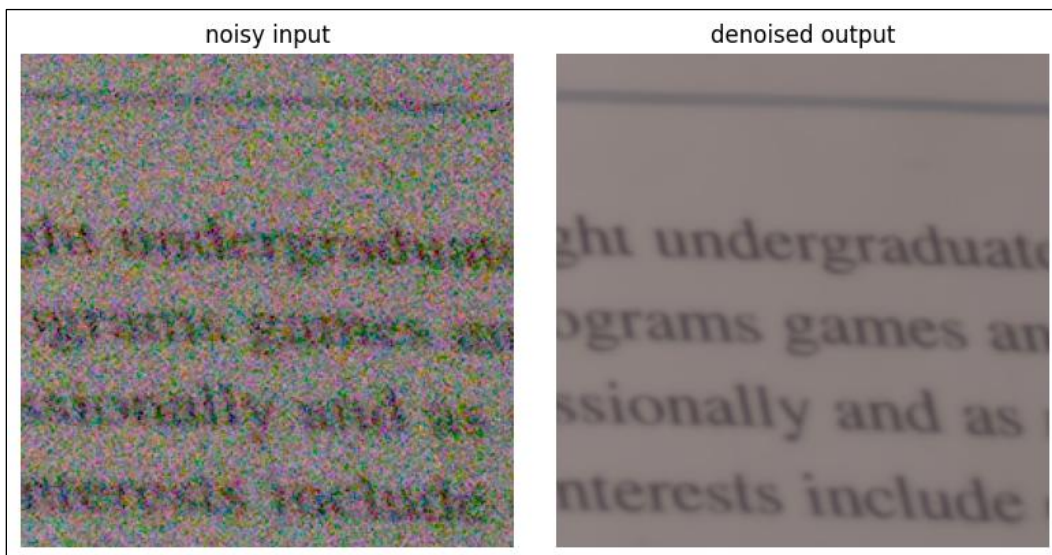
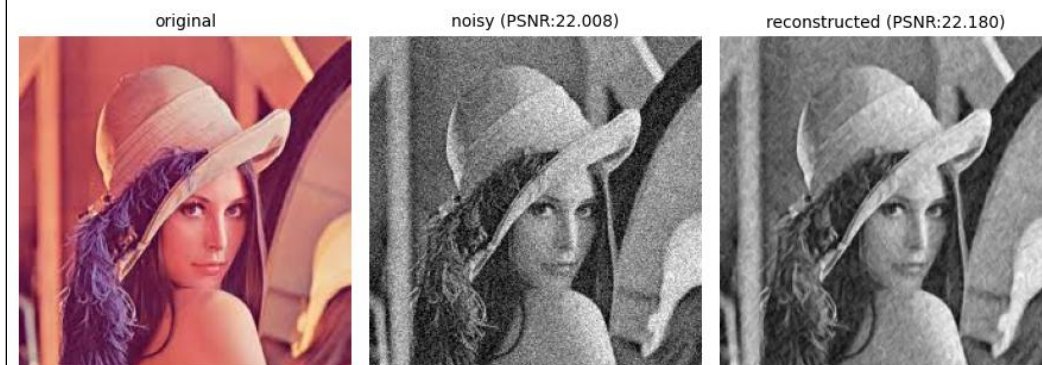


Image Restoration with Sparse Representations with scikit-learn OMP



Before



After



Before



After



damaged input



masked input



inpainted output



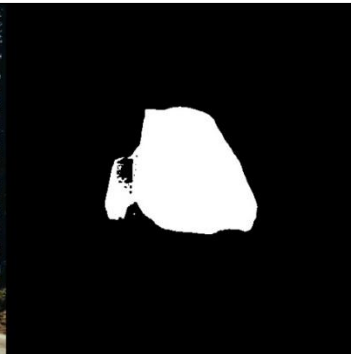
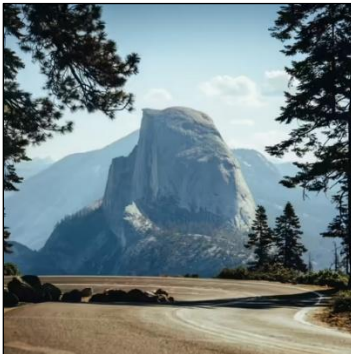
damaged input



masked input



inpainted output



input



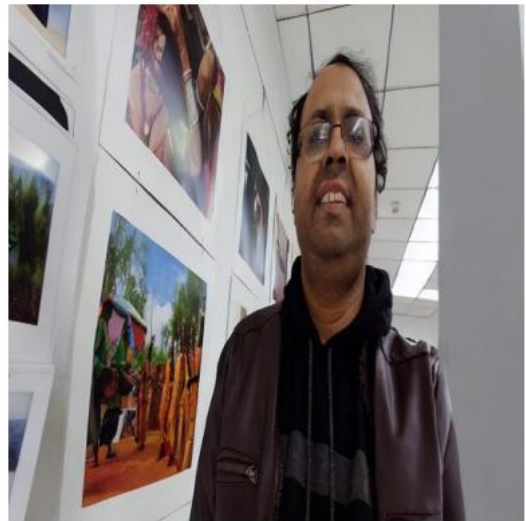
retouched



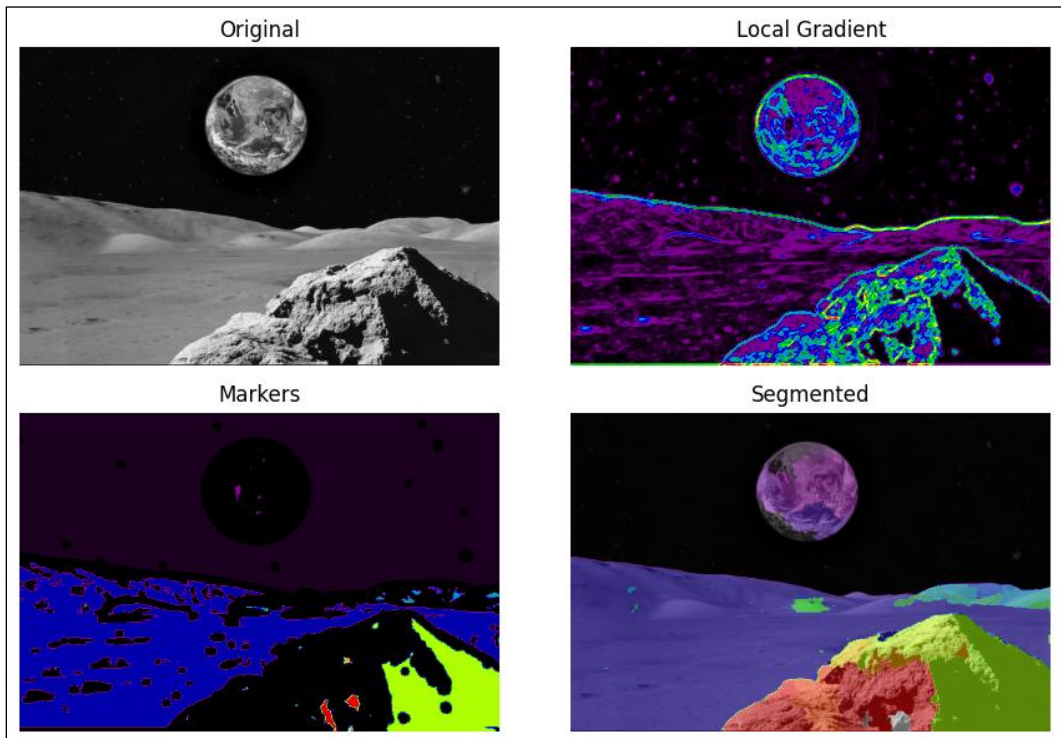
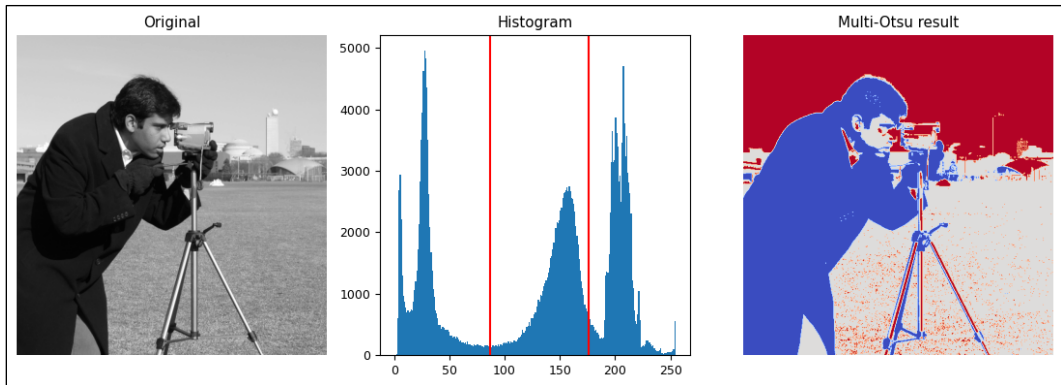
input

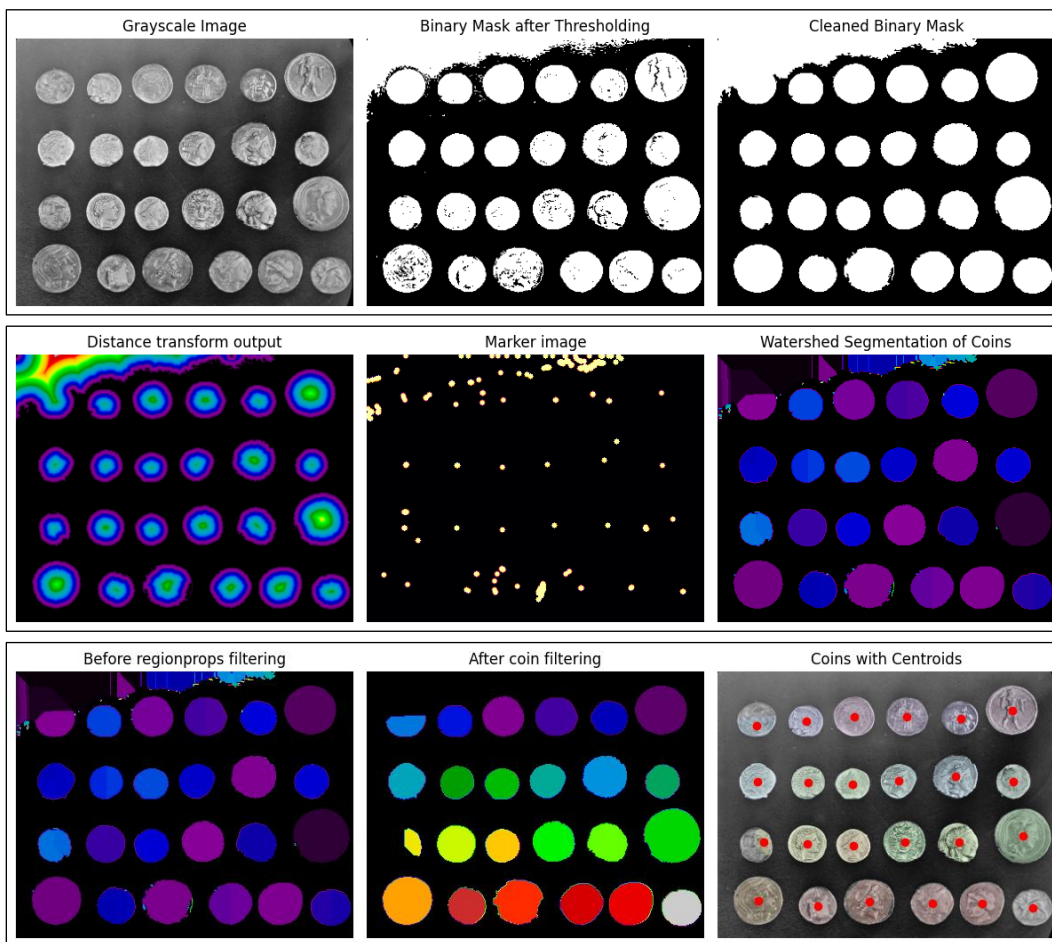


outpainted output



Chapter 10: Image Segmentation: From Classical Methods to Deep Learning

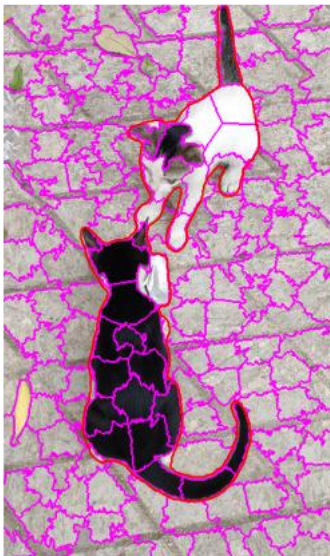




Original image



SLIC



maskSLIC



Mask



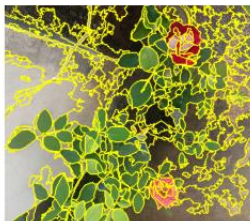
SLIC + RAG



maskSLIC + RAG



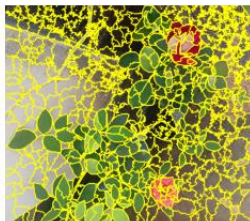
Felzenszwalb's method



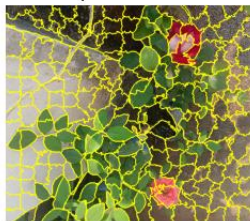
SLIC



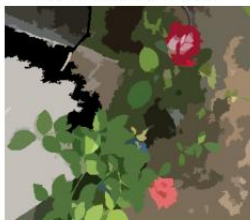
Quickshift



Compact watershed



Felzenszwalb + RAG



SLIC + RAG



Quickshift + RAG



Watershed + RAG



Original Image



Chan-Vese segmentation - 200 iterations



Final Level Set



Evolution of energy over iterations

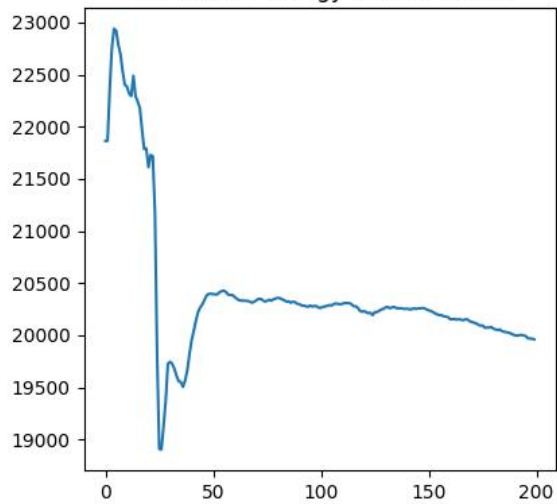


Image Segmentation with Calculus of Variations

MUMFORD-SHAH (MS) FUNCTIONAL

functional minimization

$$E_{MS}(u, \Gamma) = \beta \int_{\Omega} (u - f)^2 dx + \alpha \int_{\Omega \setminus \Gamma} |\nabla u|^2 dx + \text{length}(\Gamma)$$

$\Omega \subset \mathbb{R}^2$ is connected, bounded, open subset

$\Gamma \subset \Omega$

edge set segmenting Ω

AMBROSIO-TORTORELLI (AT) APPROXIMATION OF MS FUNCTIONAL

functional minimization

$$E_{AT}(u, v) = \int_{\Omega} \left(\beta(u - f)^2 + \alpha(v^2 |\nabla u|^2) + \frac{1}{2} \left(\rho |\nabla v|^2 + \frac{(1-v)^2}{\rho} \right) \right) dx$$

$\rho \rightarrow 0, \quad v \rightarrow 1 - \chi_{\Gamma}$

edge indicator

$v(x) \approx 0$ if $x \in \Gamma$ and $v(x) \approx 1$

Euler-Lagrange Equations & system of coupled elliptic PDEs

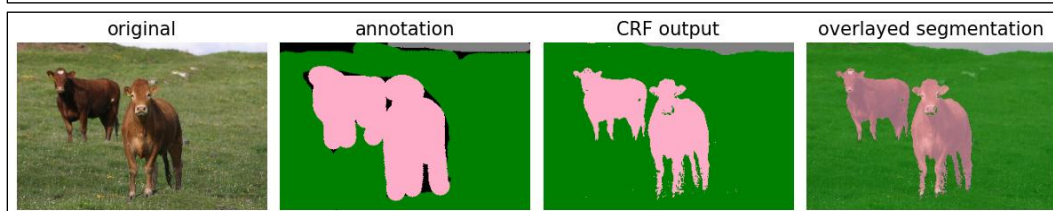
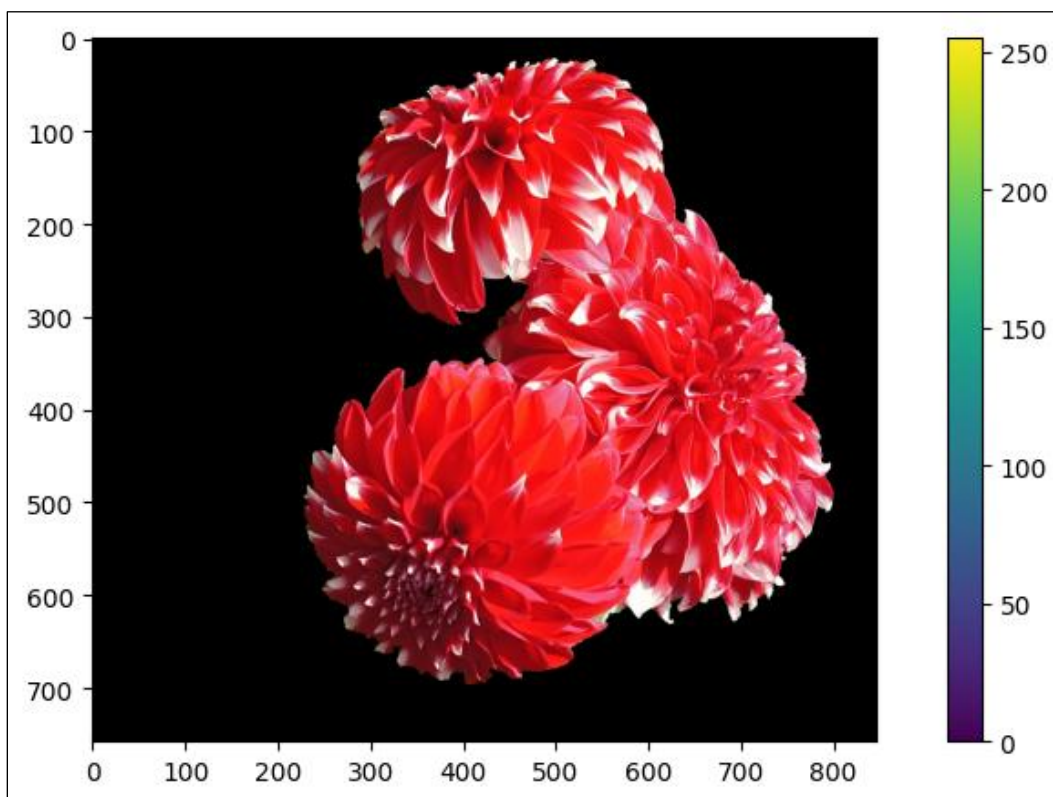
for Gradient-descent updates

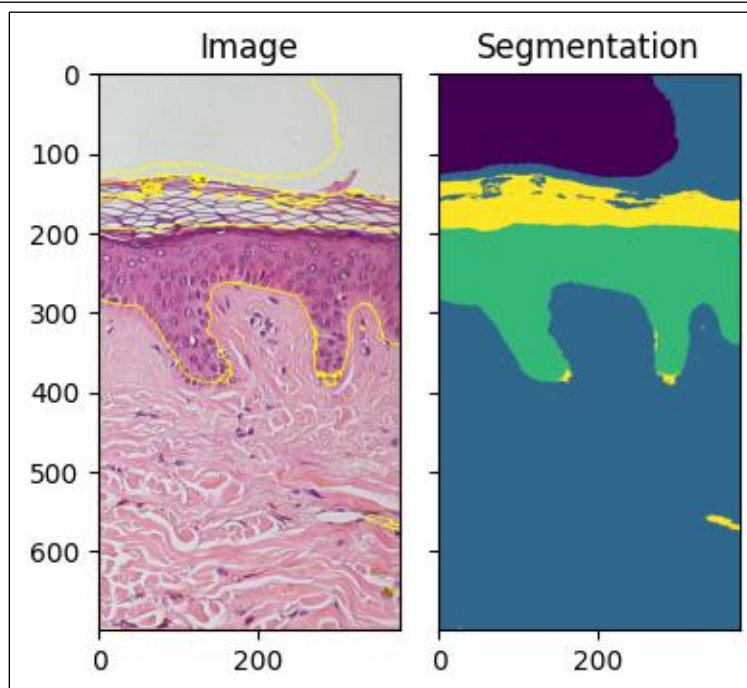
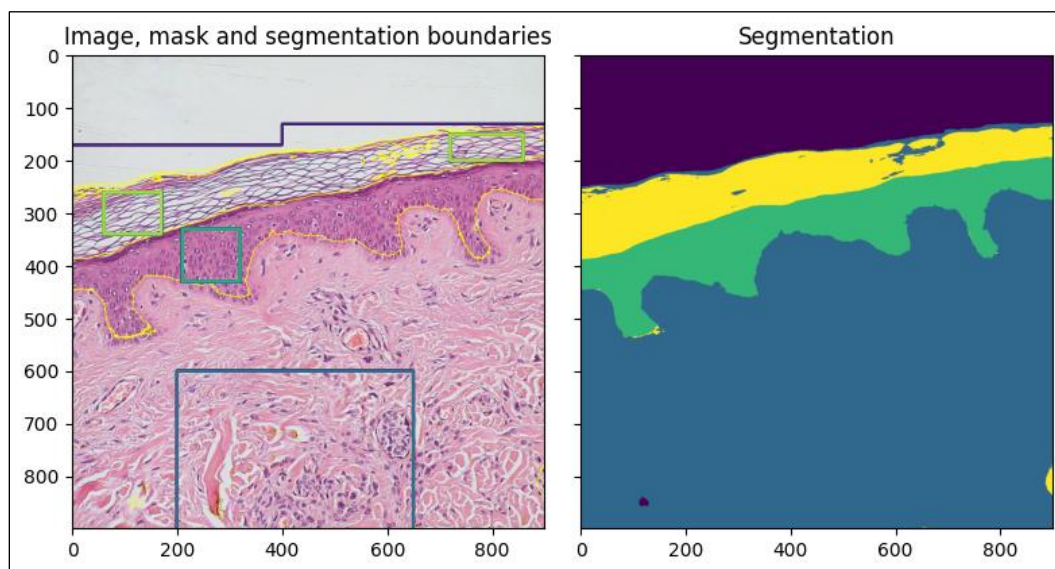
$$\begin{aligned} \frac{\partial u}{\partial t} &= \nabla \cdot (v^2 \nabla u) - \frac{\beta}{\alpha} (u - f); & \frac{\partial u}{\partial n} \Big|_{\partial \Omega} &= 0, & \text{Neumann boundary} \\ \frac{\partial v}{\partial t} &= \nabla^2 v - \frac{2\alpha |\nabla u|^2 v}{\rho} - \frac{(v-1)}{\rho^2}; & \frac{\partial v}{\partial n} \Big|_{\partial \Omega} &= 0 & \text{conditions} \end{aligned}$$

Ref. https://web.cs.hacettepe.edu.tr/~erkut/bil717.s12/v06-ms_tv.pdf

Image Segmentation with Ambrosio-Tortorelli approximation of Mumford-Shah functional
original image segmented image regularized edge set







Image



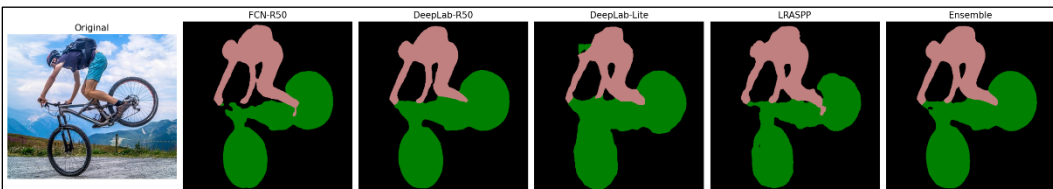
Sobel+Watershed

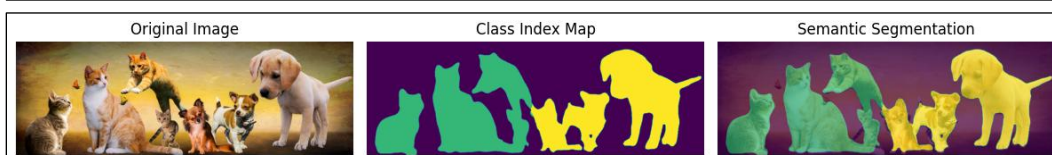
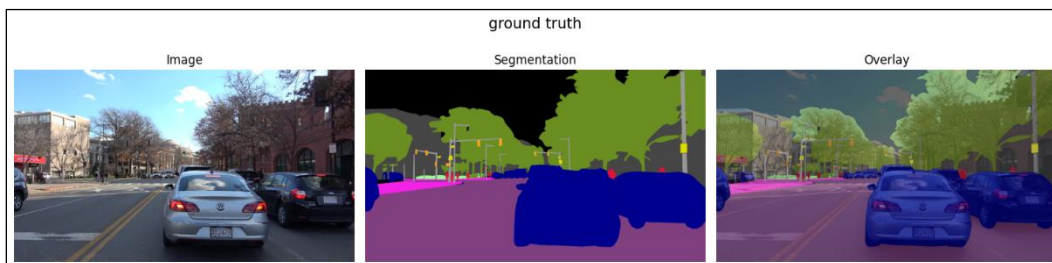
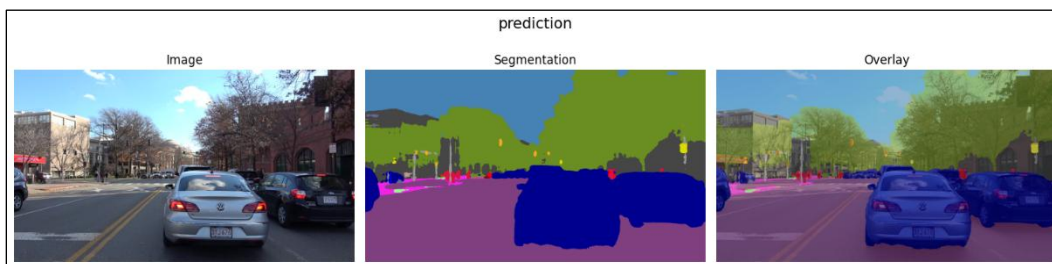


SLIC superpixels



Join

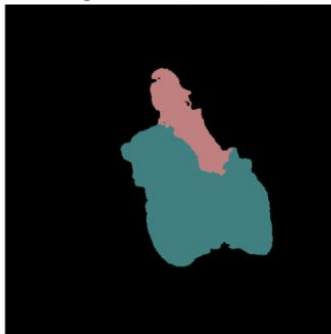




Input Image



Segmentation (VOC colors)



Overlay

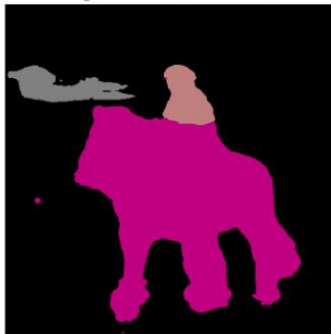


1/1 — 0s 131ms/step

Input Image

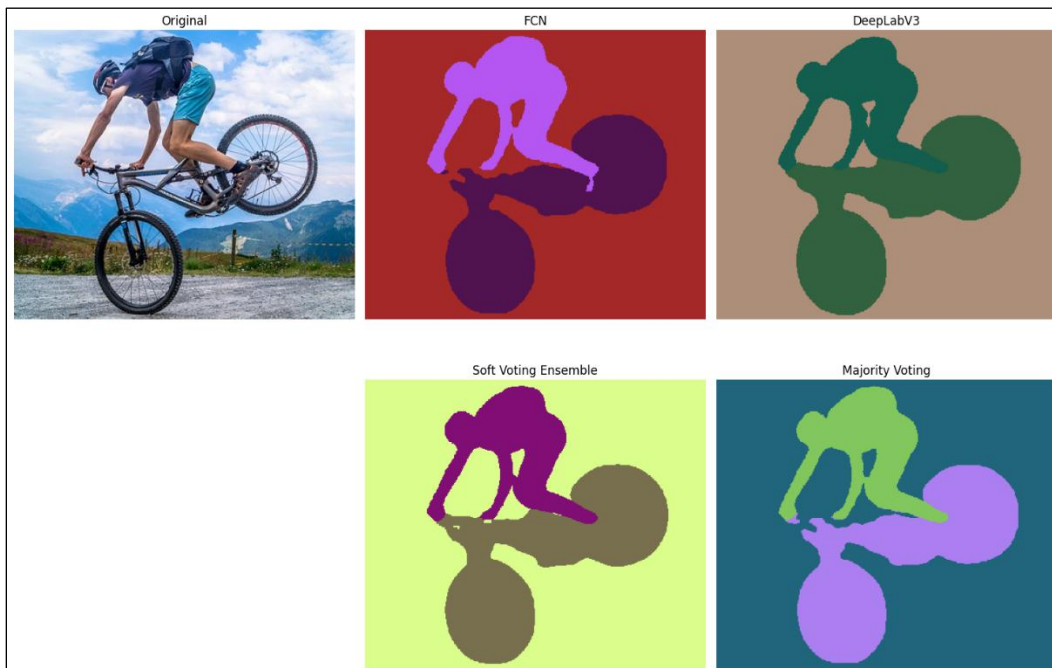
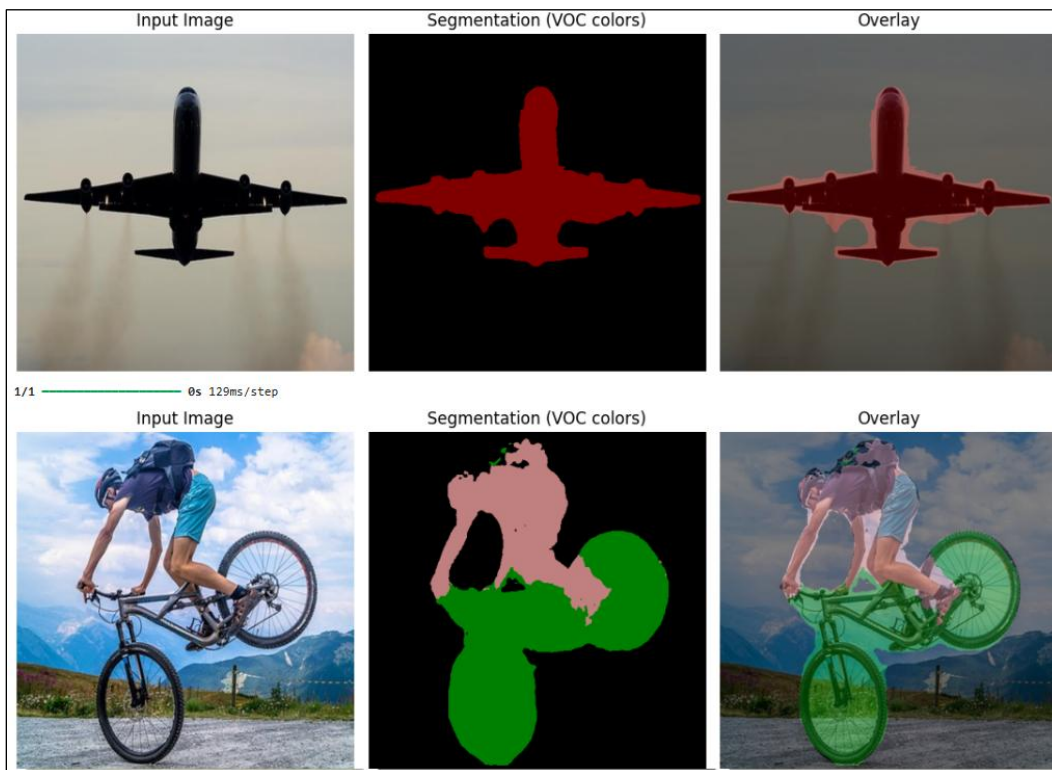


Segmentation (VOC colors)



Overlay





Mask R-CNN: Instance Segmentation

Original Image



Vibrant Masks



Combined Overlay



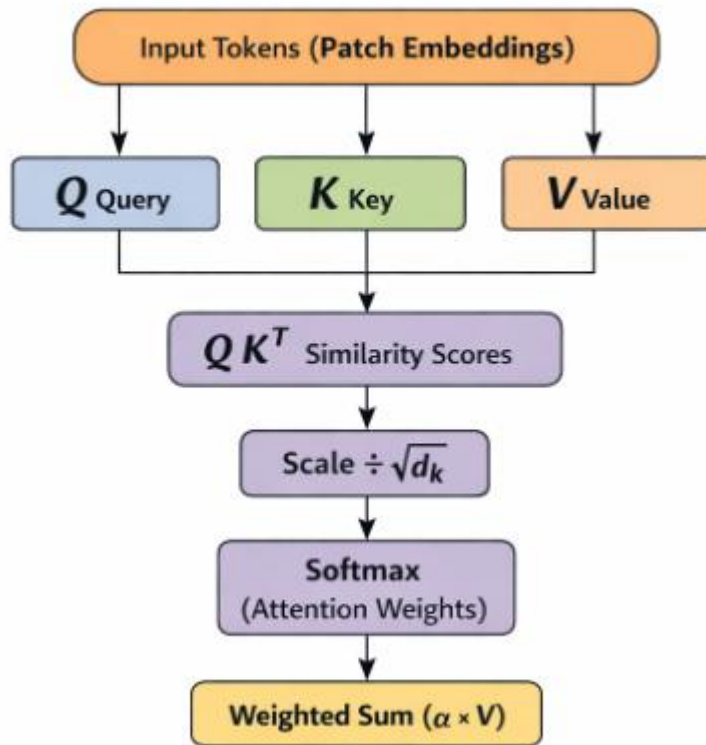
Original Image



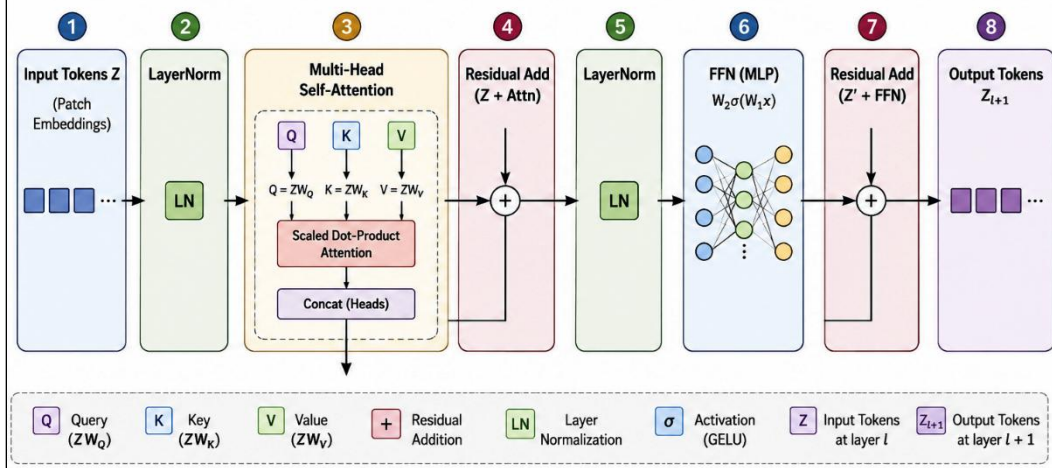
YOLO11 Instance Segmentation

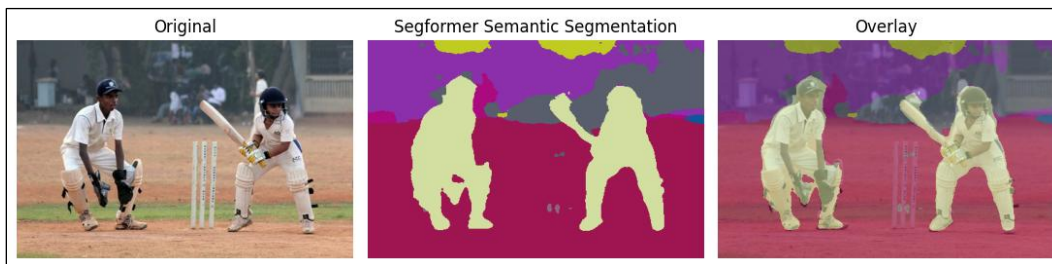
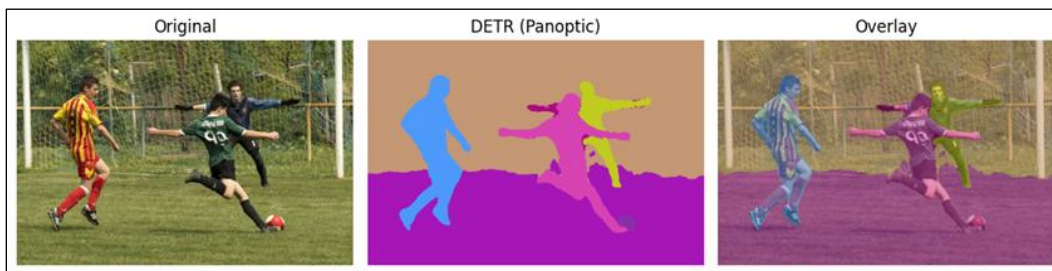
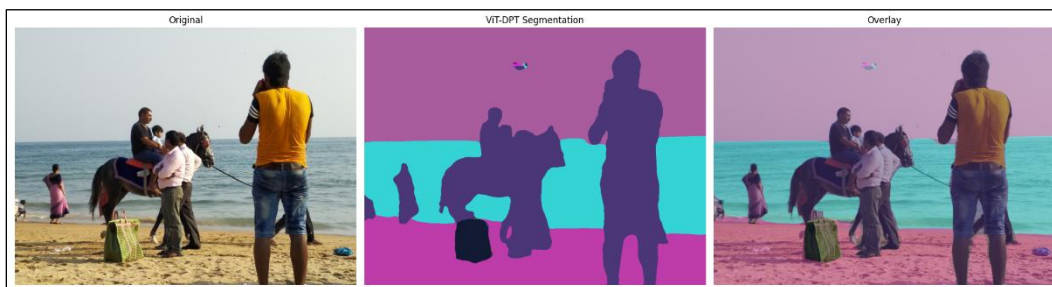


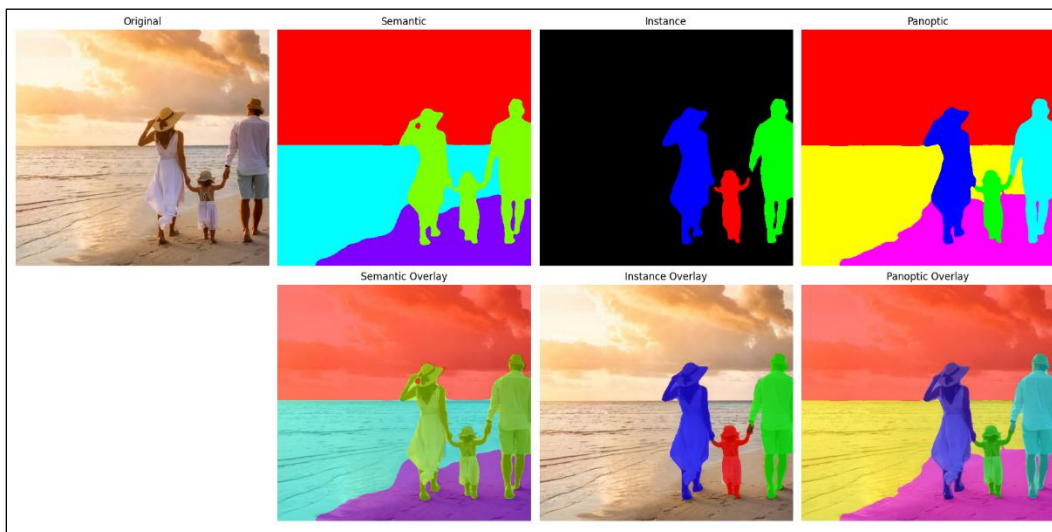
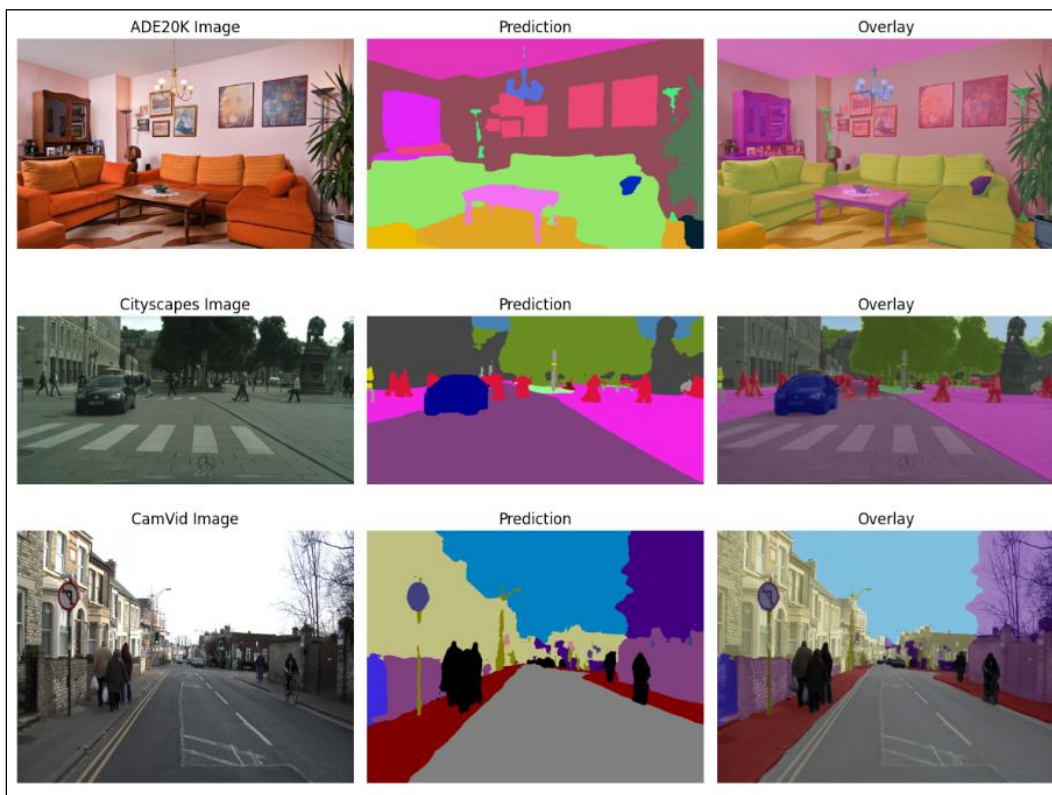
Scaled Dot-Product Attention Diagram



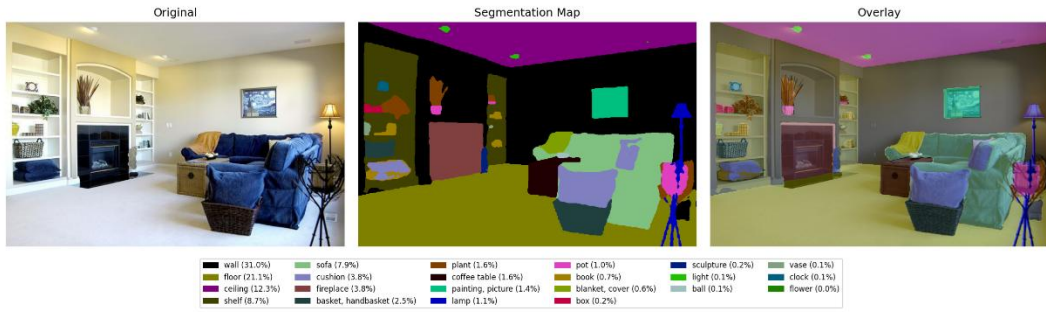
Transformer block architecture



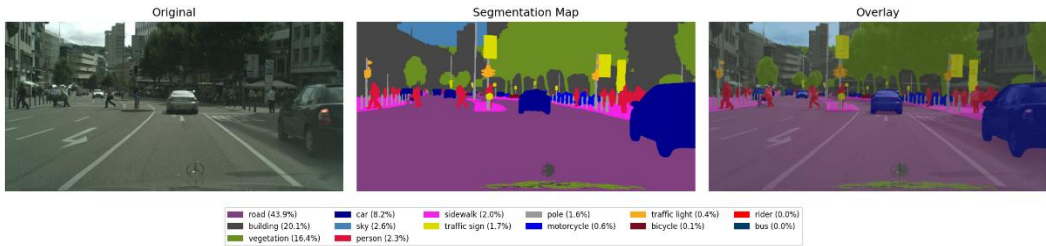




ADE20K — ADE_val_00000519.jpg



Cityscapes — stuttgart_00_000000_000113_leftimg8bit.png



Chapter 11: More Deep Learning Methods for Image Segmentation



original



depth map

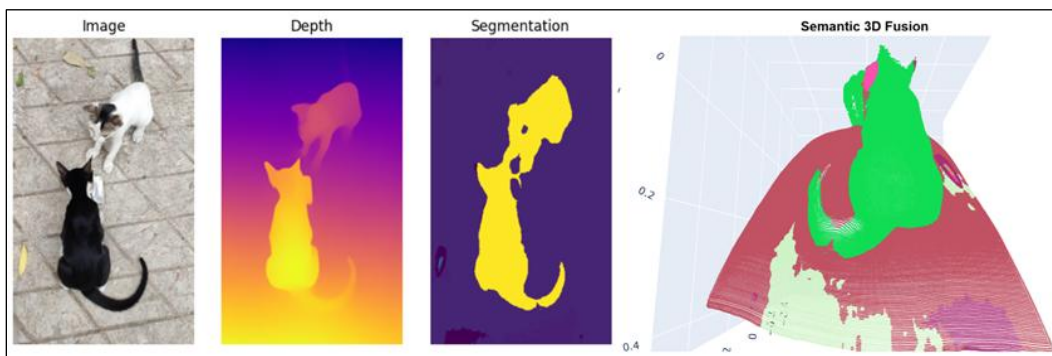


input images

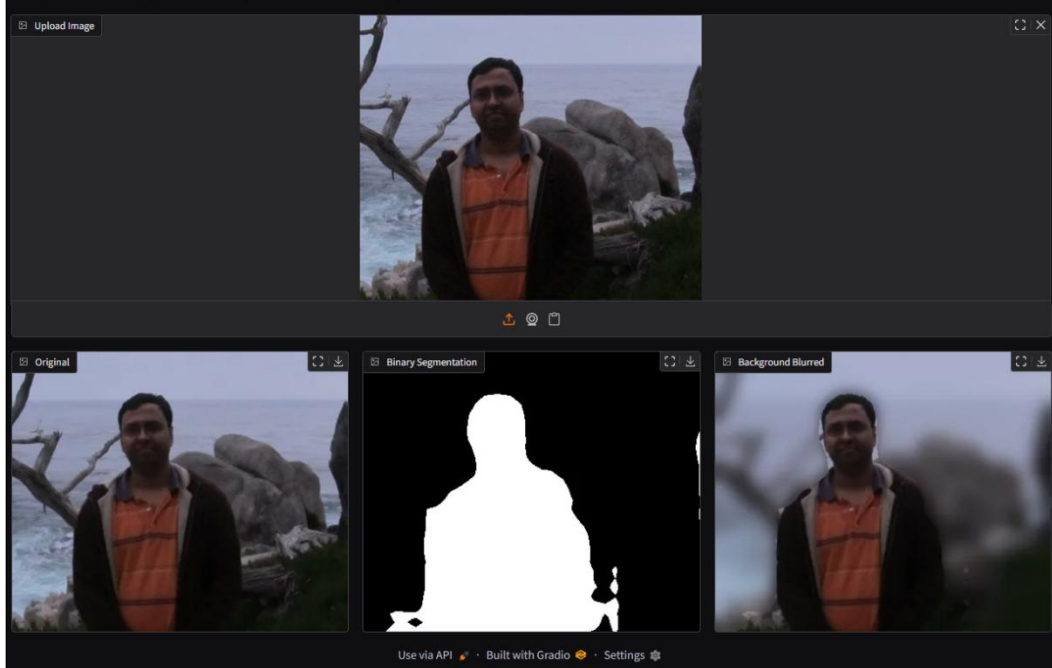


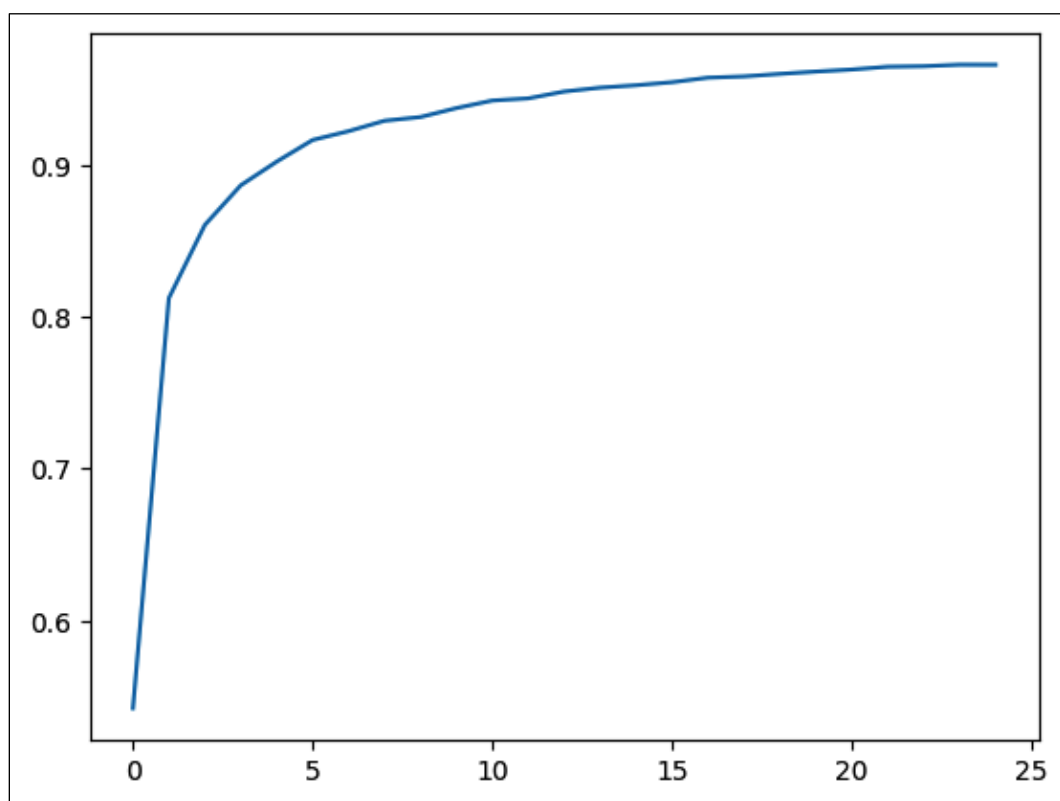
3D point cloud

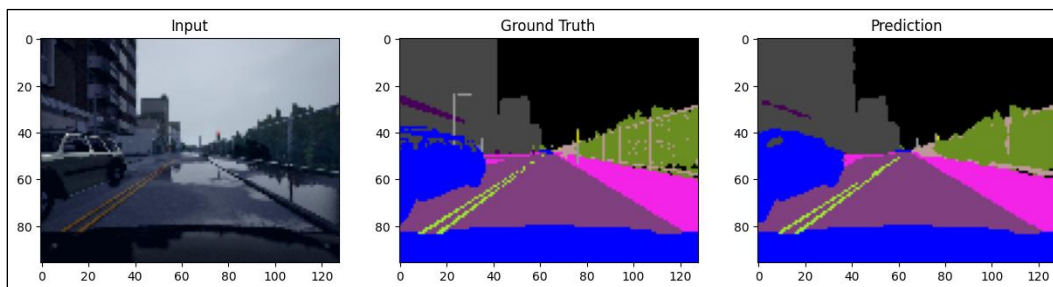
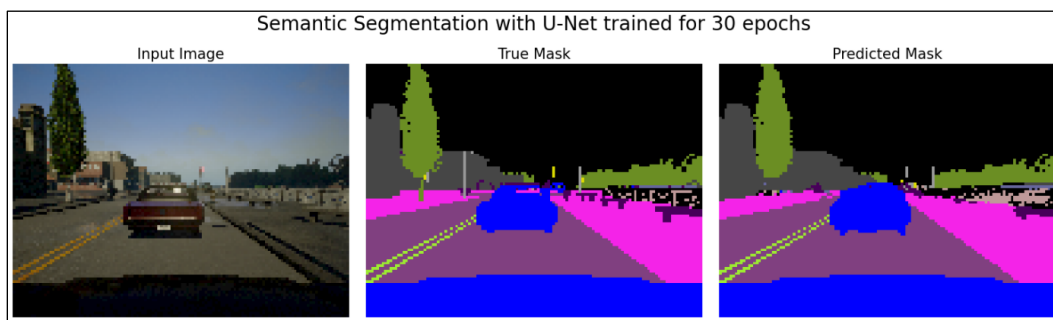
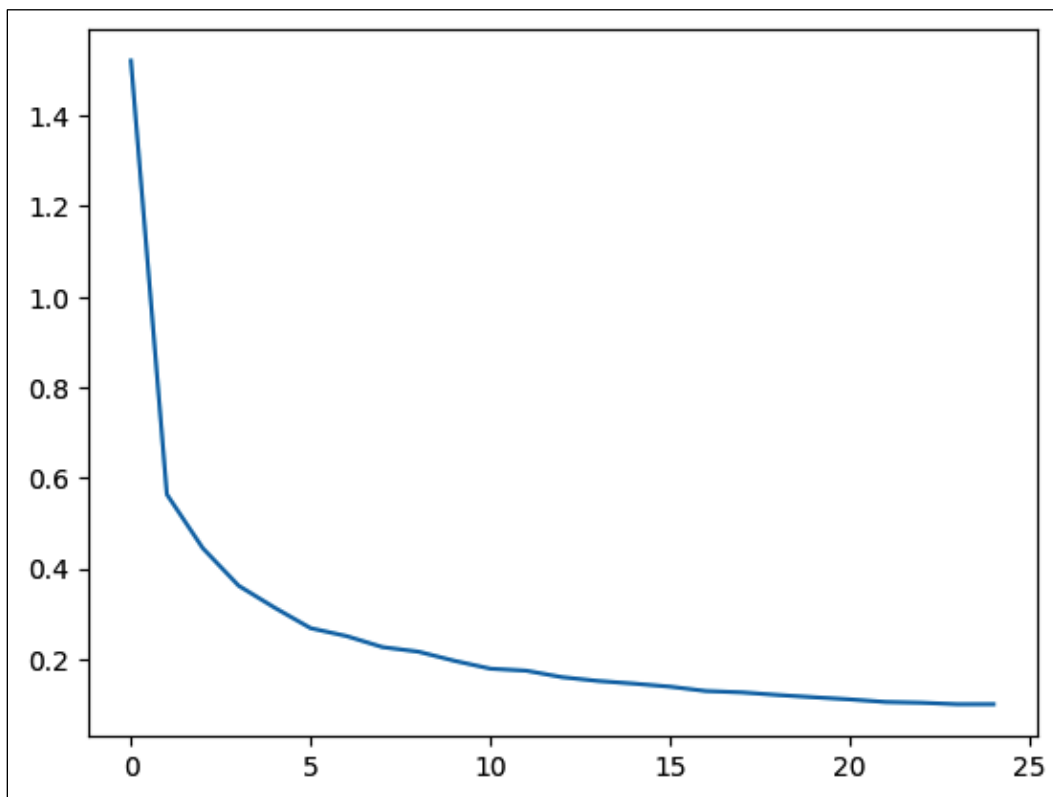


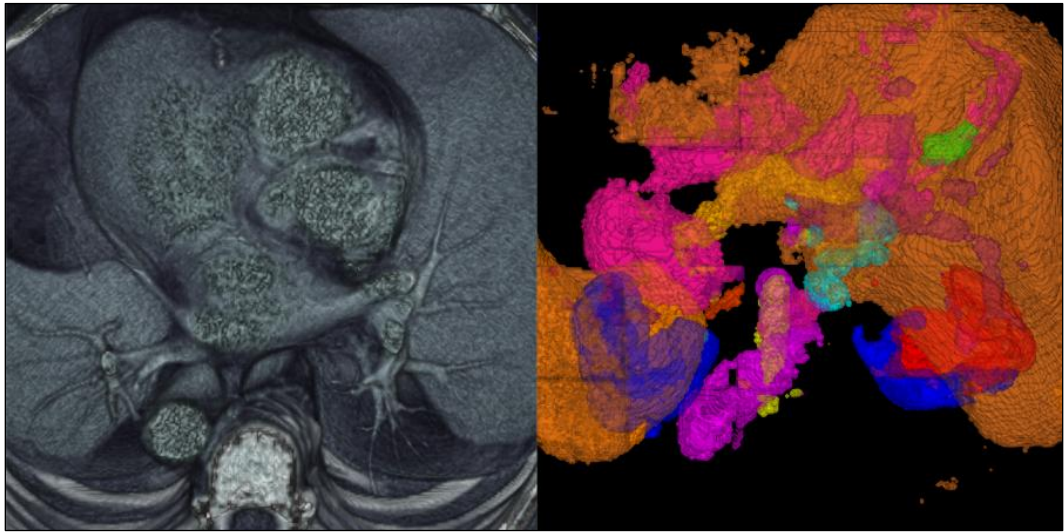
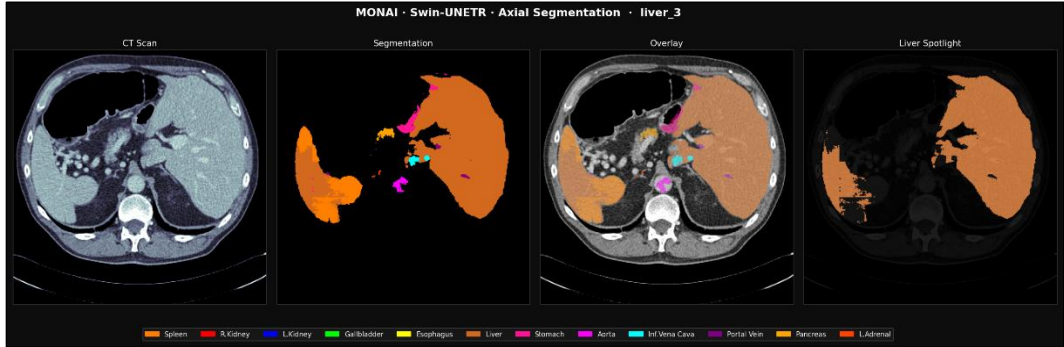
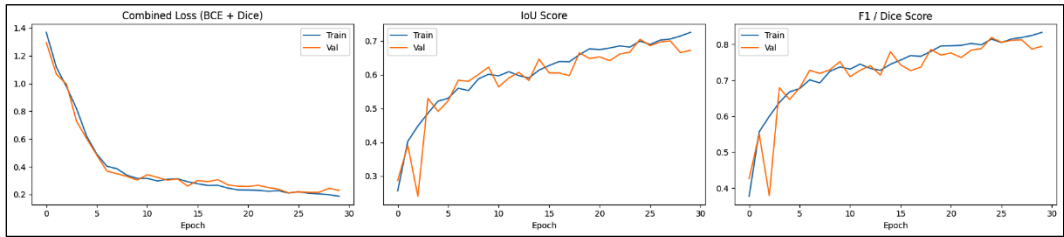


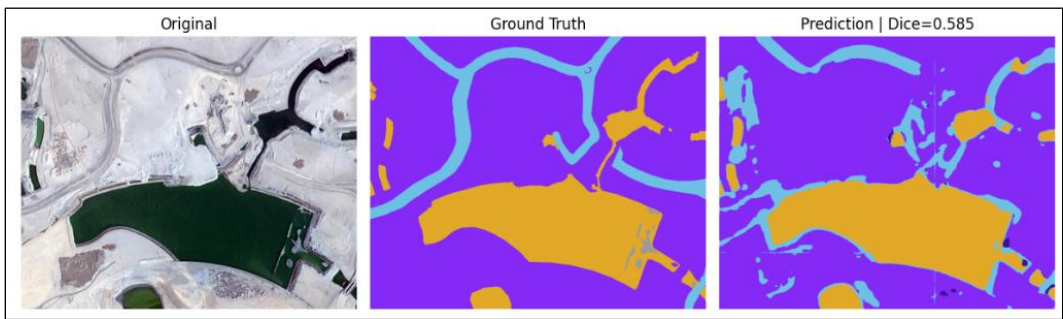
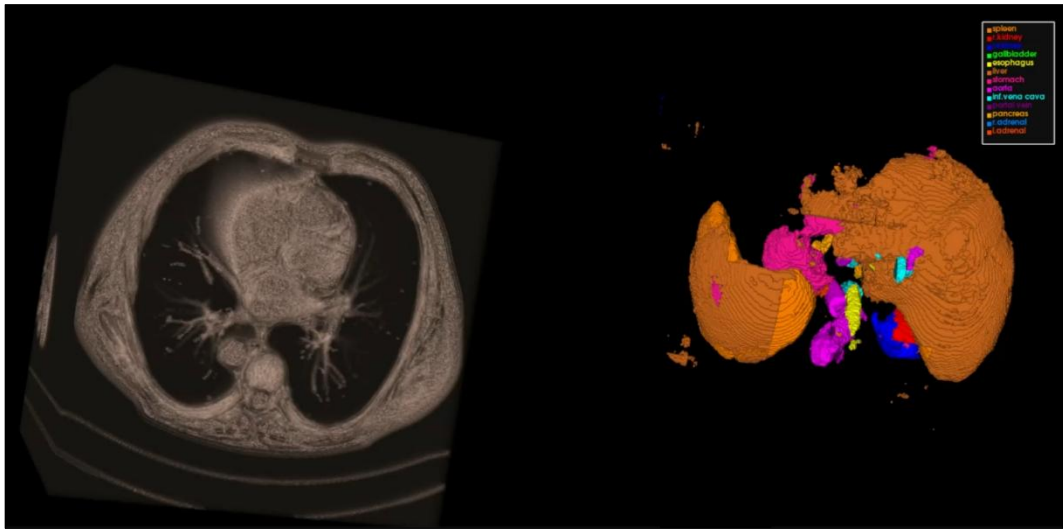
DeepLabV3 Background Blur (TFLite + MediaPipe)



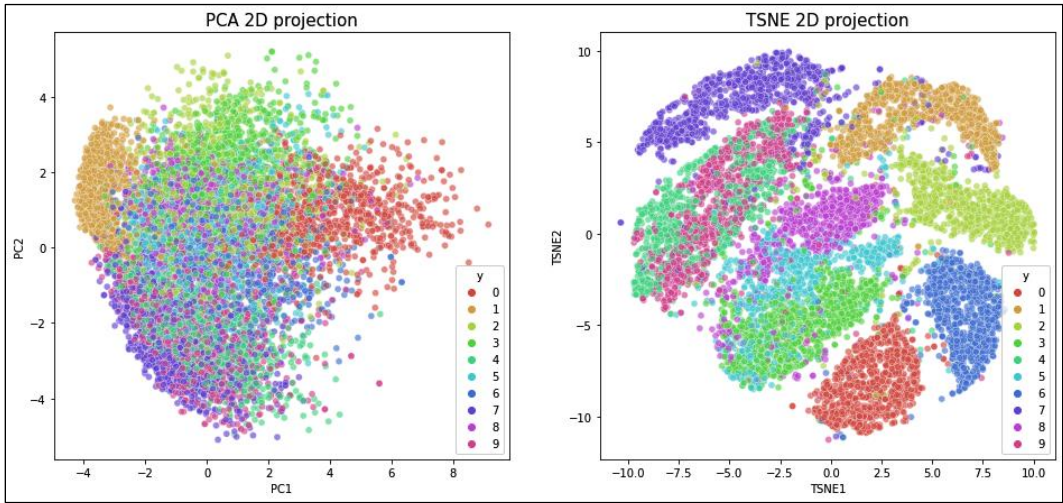








Chapter 12: Image Classification and Object Detection



CNN Brain Viewer (ResNet18)

Visualize how CNNs transform images from edges → textures → semantic object parts

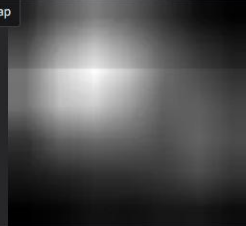
 Upload Image



 Input



 Activation Heatmap



 Overlay (Model Attention)



CNN Layer (0=Early → 4=Deep)

4



0



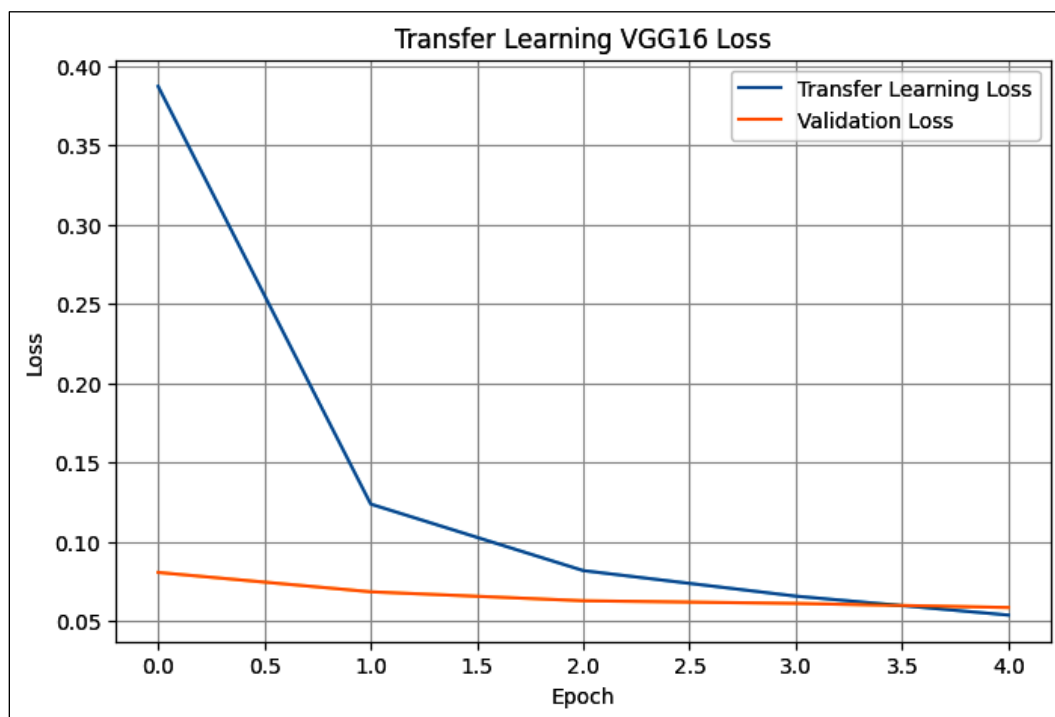
4

Clear

Submit

Flag

Use via API  · Built with Gradio  · Settings 

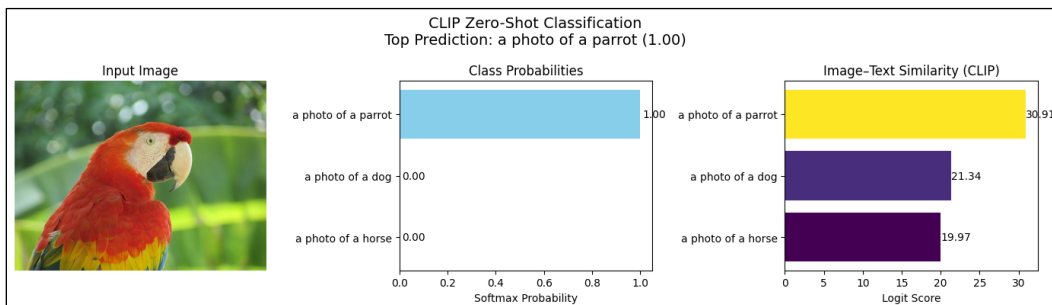
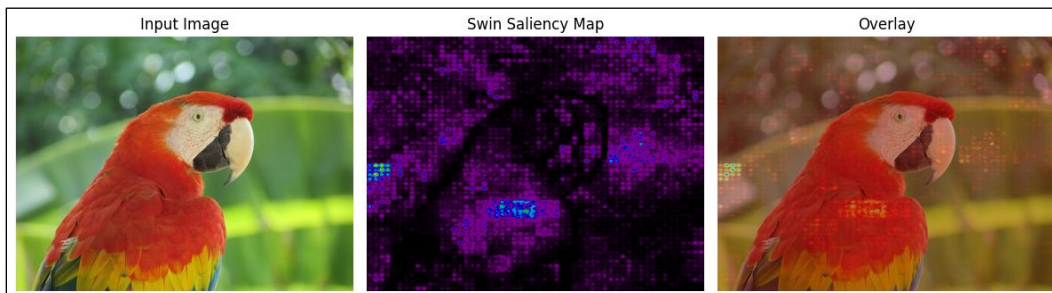
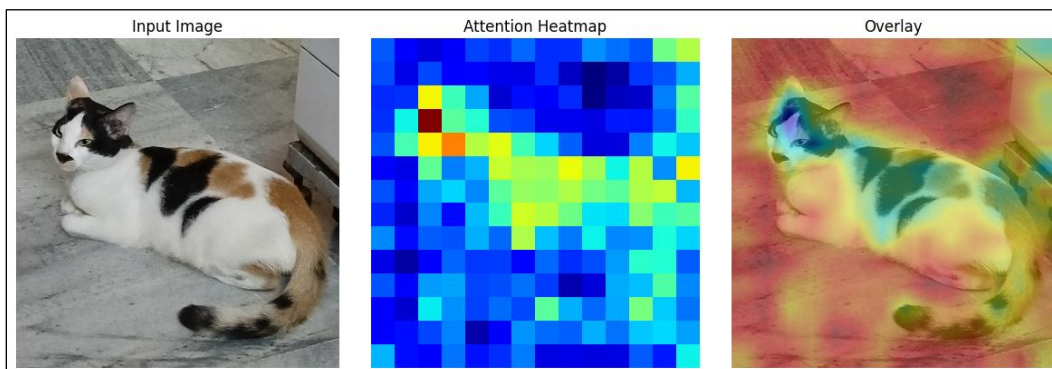


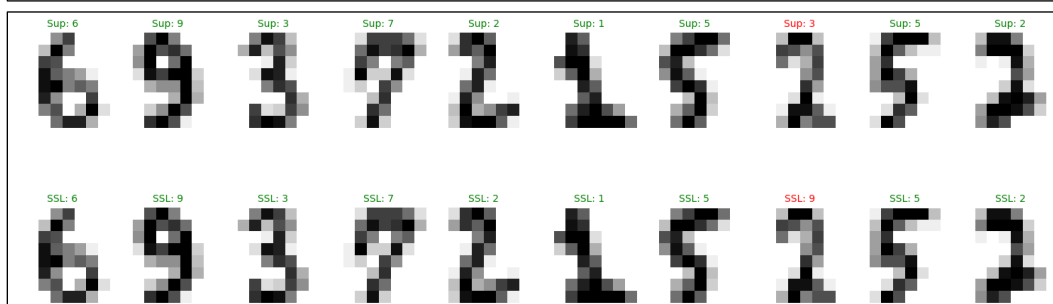
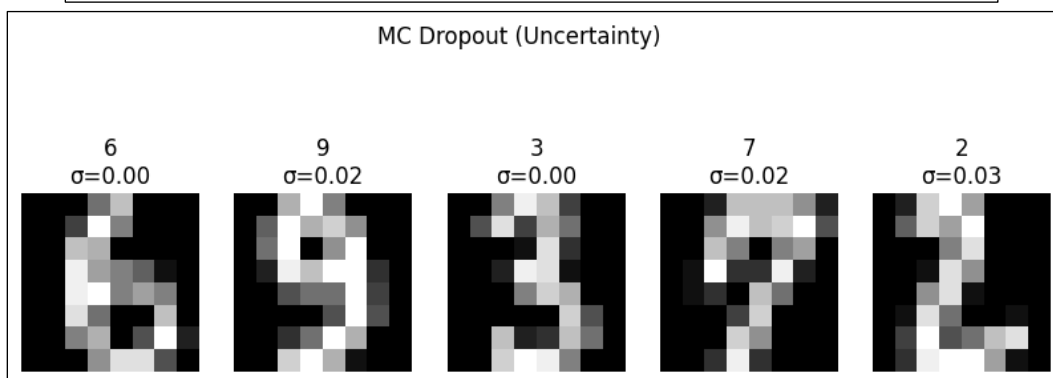
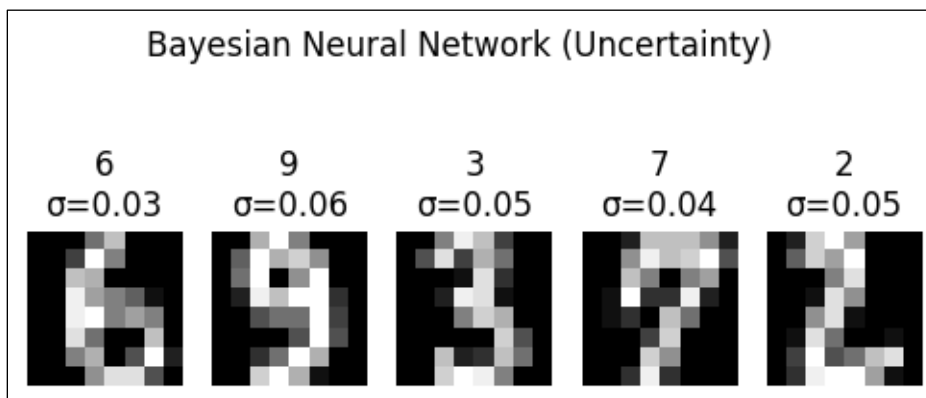
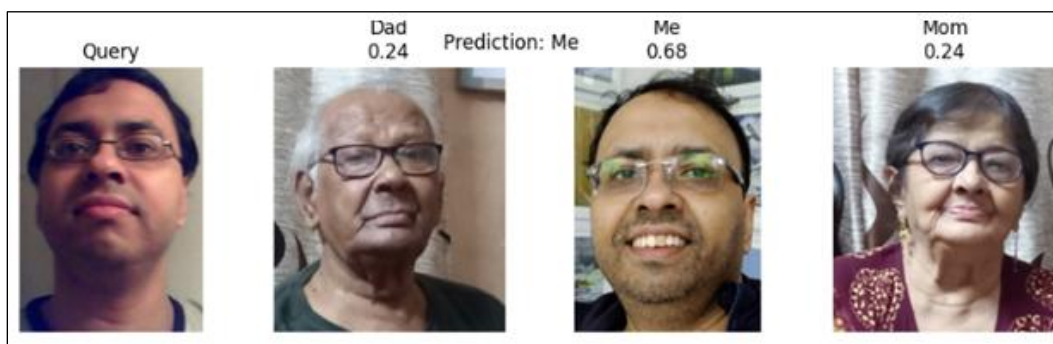
Pretrained VGG16: cats
ImageNet Label: Persian_cat
Transfer Learning: cats

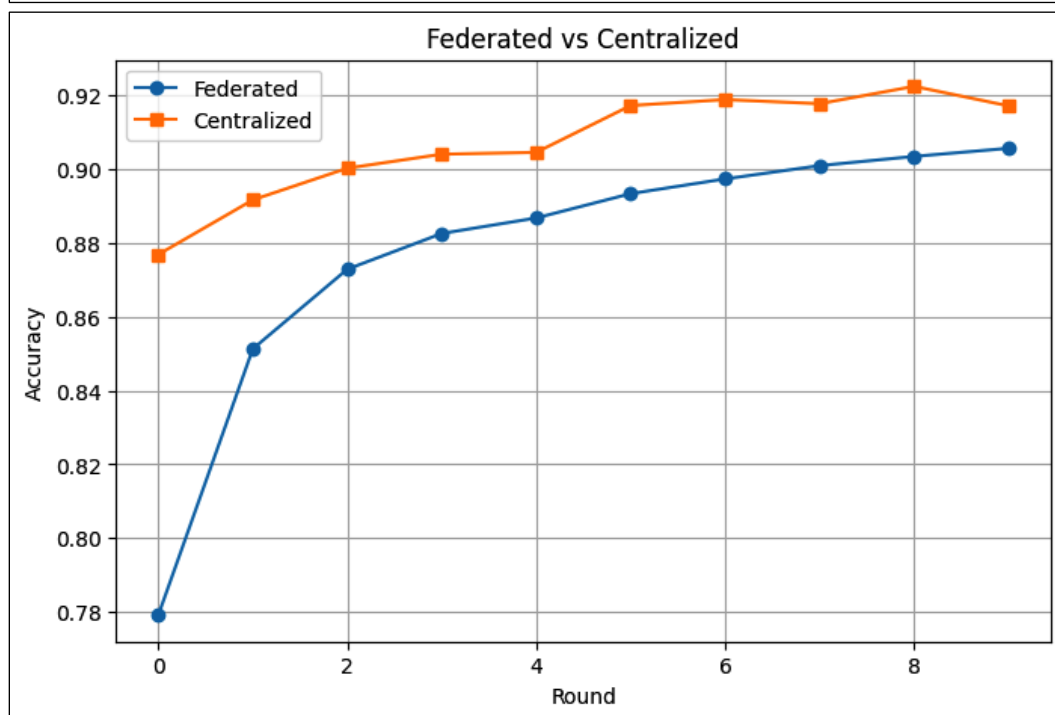
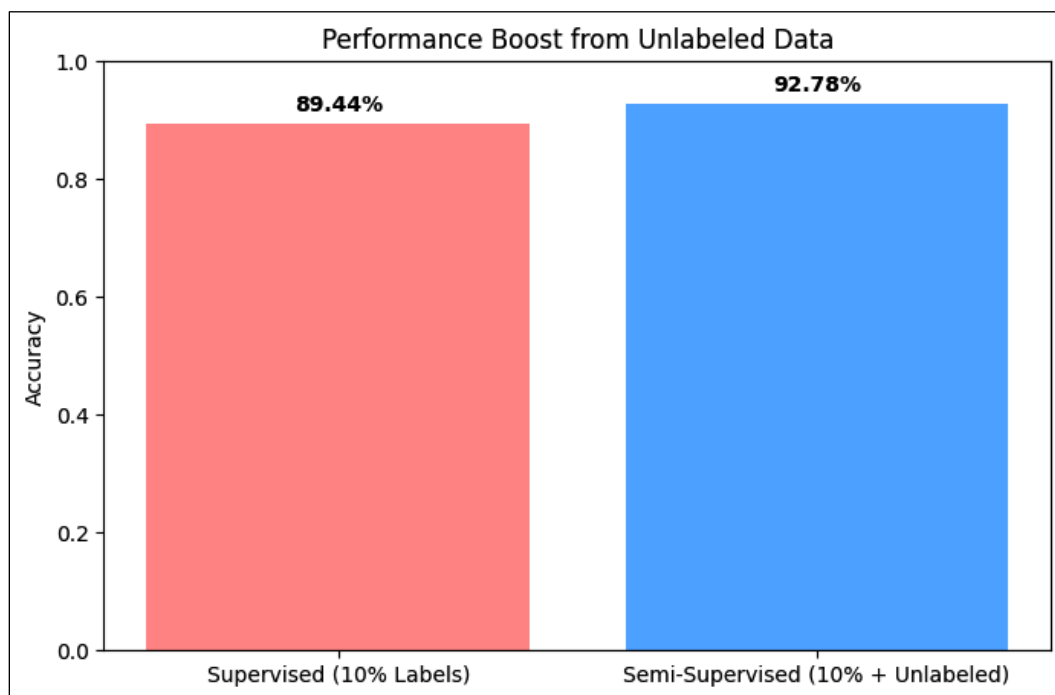


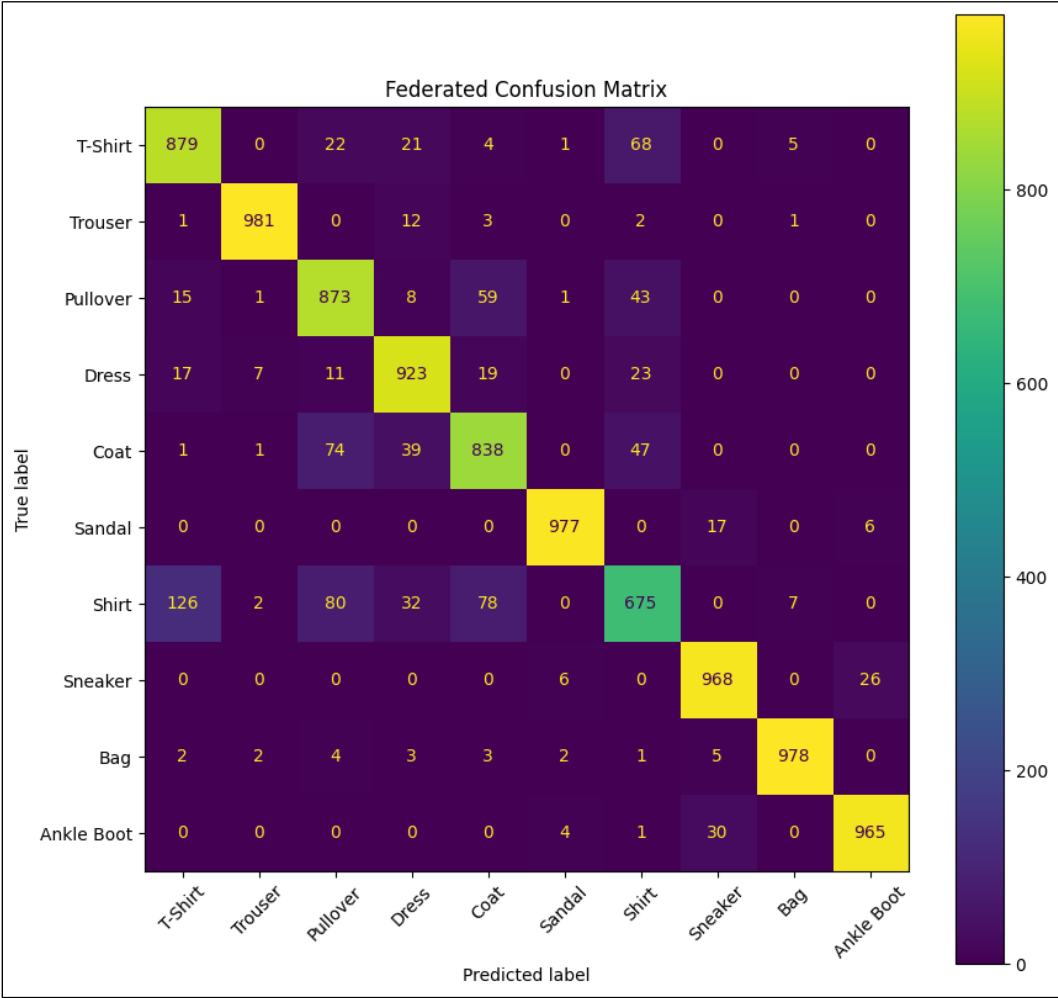
Pretrained VGG16: dogs
ImageNet Label: golden_retriever
Transfer Learning: dogs

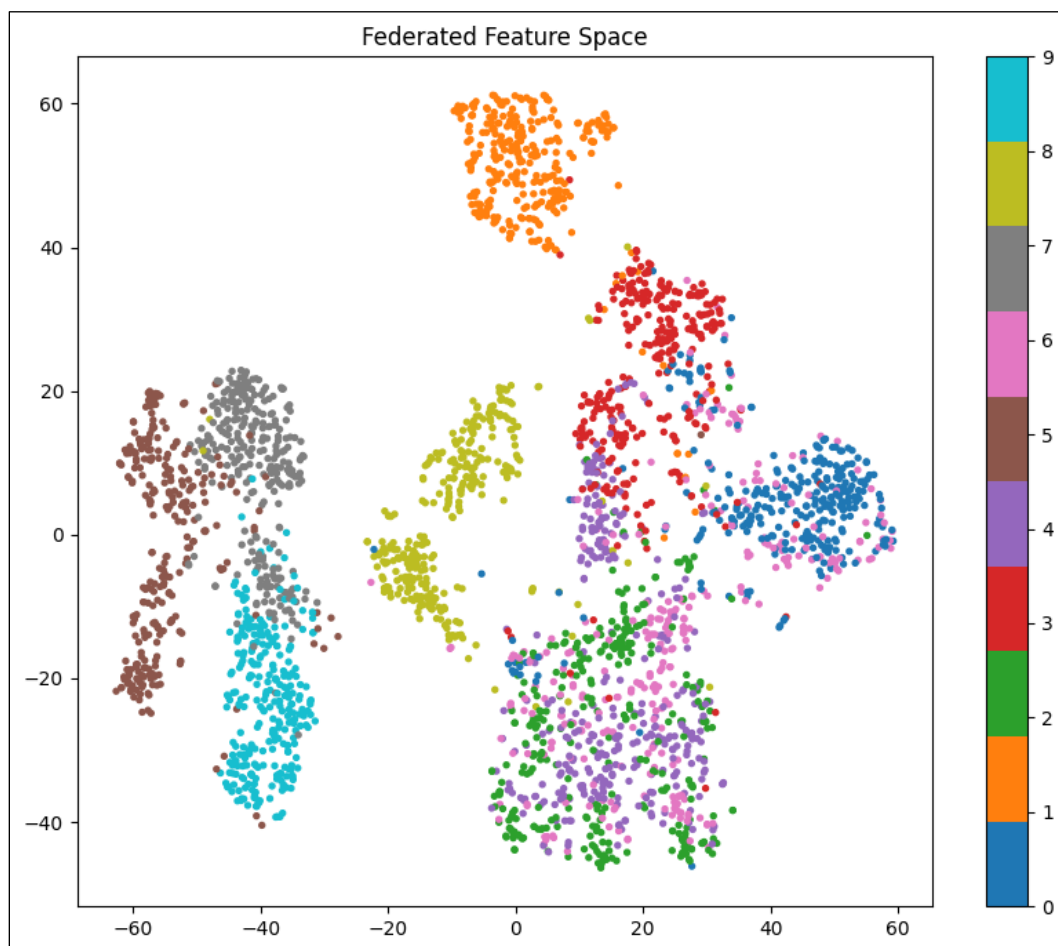


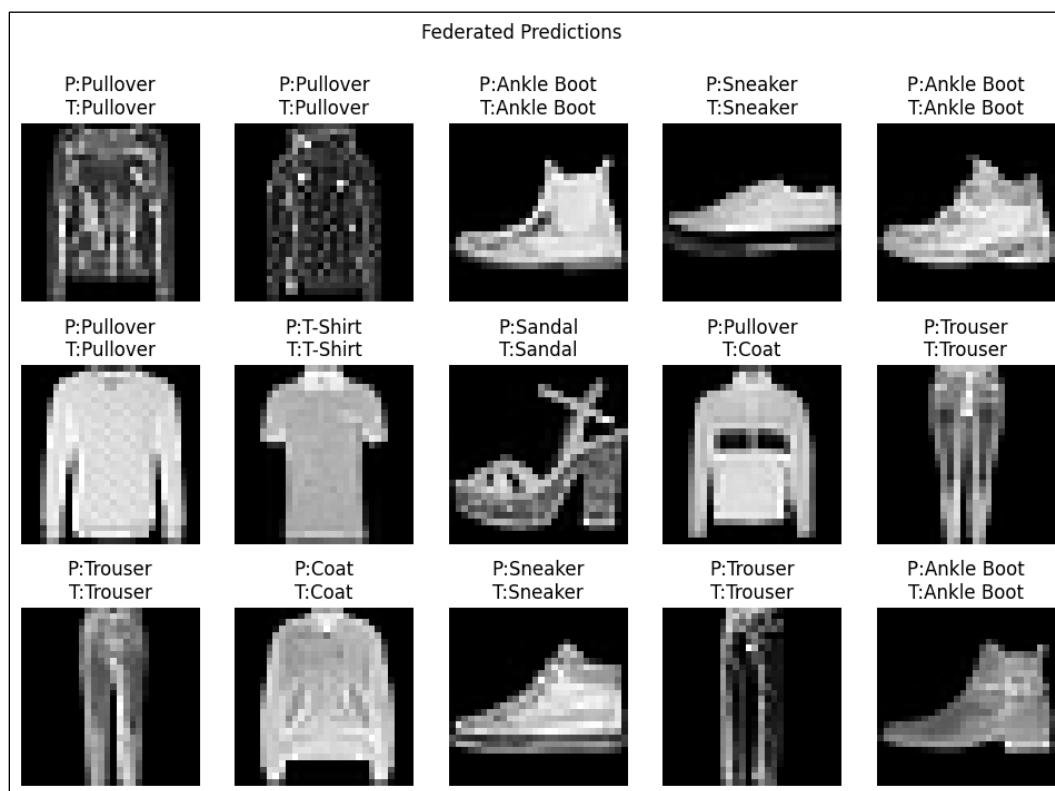
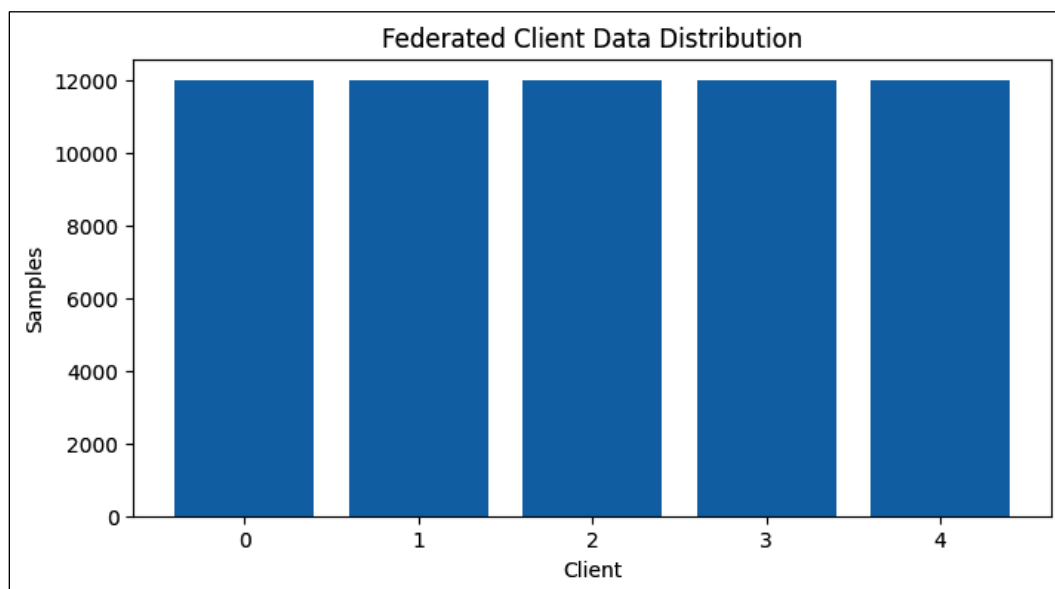




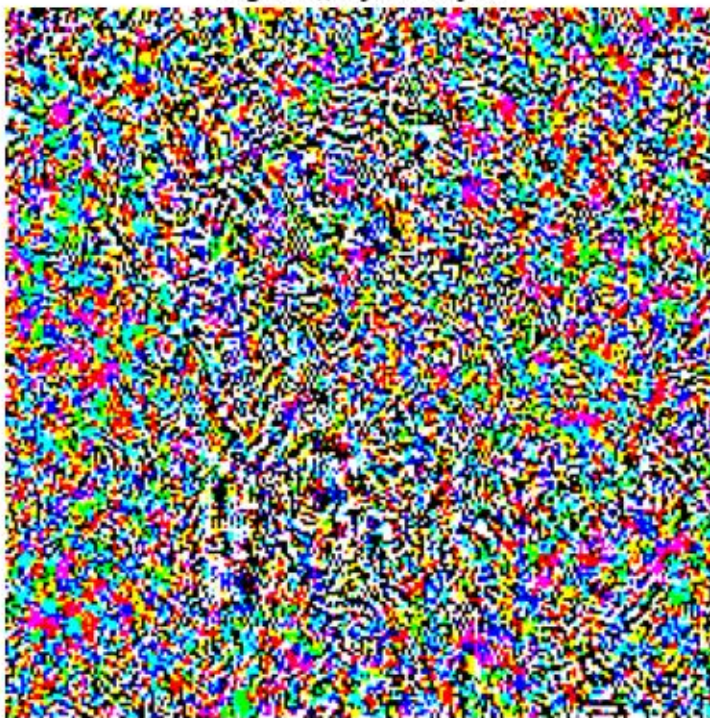




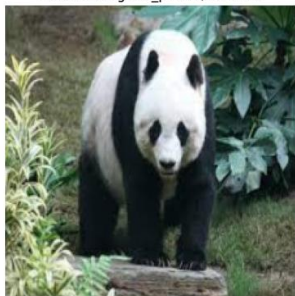




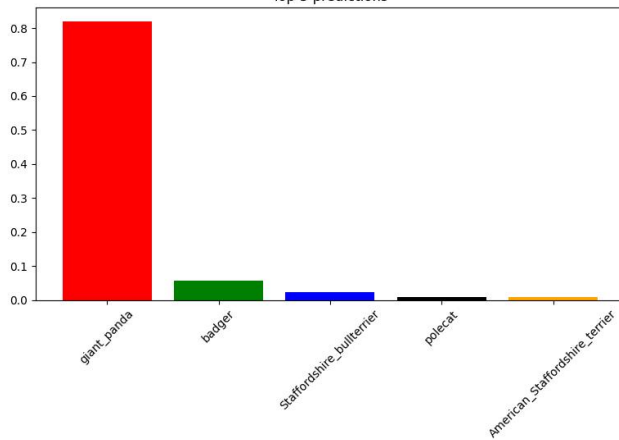
$$\text{sign}(\nabla_x \cdot J(\theta, x, y))$$



Original Input (x)
Prediction: giant_panda, 81.91%



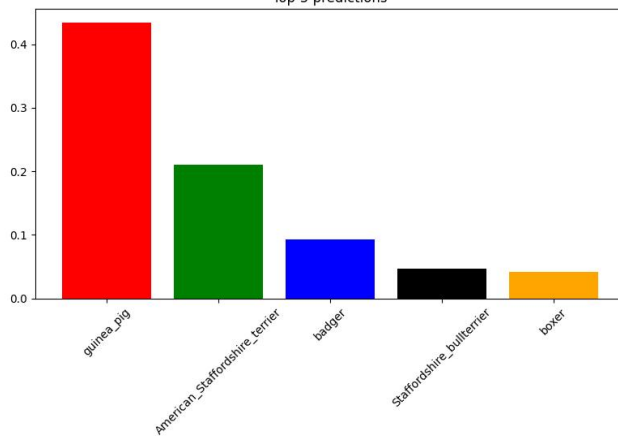
Top-5 predictions



$x + \epsilon \cdot \text{sign}(\nabla_x J(\theta, x, y))$, $\epsilon = 0.100$
Prediction: guinea_pig, 43.39%



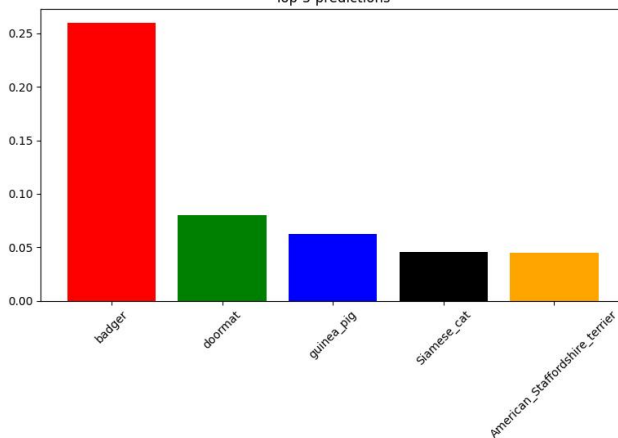
Top-5 predictions

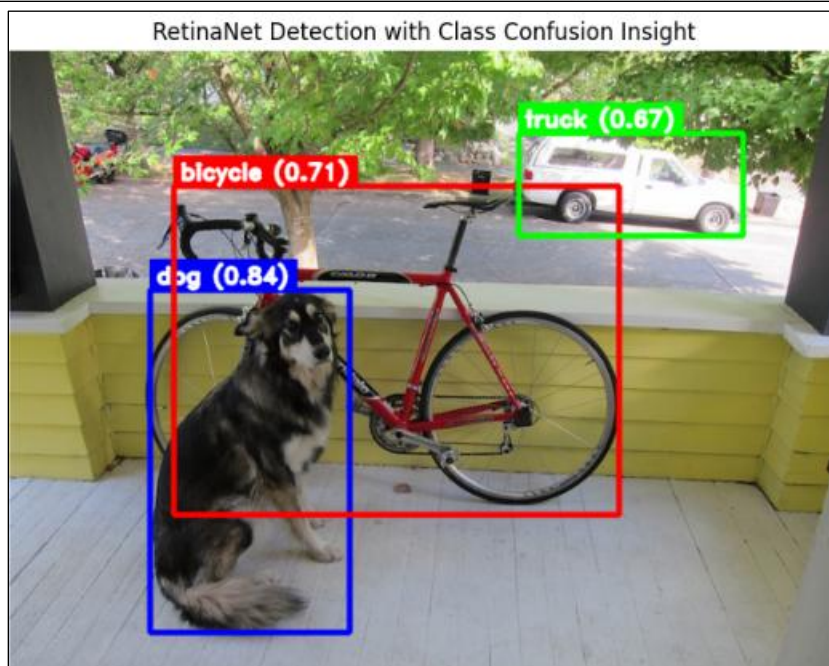
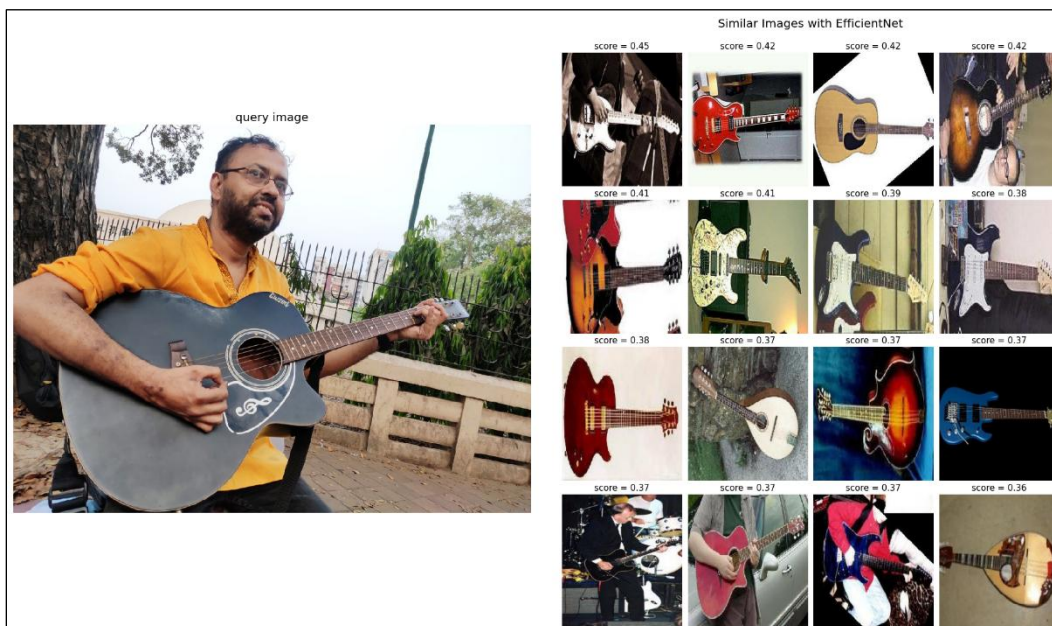


$x + \epsilon \cdot \text{sign}(\nabla_x J(\theta, x, y))$, $\epsilon = 0.200$
Prediction: badger, 25.96%



Top-5 predictions





RESTAURANT

FOOD BILL

Invoice #: 1304
Date 24/04/2024

DESCRIPTION	QTY	RATE	AMOUNT
Paneer Butter Masala	1	₹180.00	₹180.00
Veg Biryani	1	₹150.00	₹150.00
Tandoori Roti	3	₹20.00	₹60.00
Gulab Jamun	2	₹100.00	₹100.00
CGST @ 2.5%			₹12.25
SGST @ 2.5%			₹12.25
TOTAL			₹ 514.50

Extracted text:
RESTAURANT
FOOD BILL

Invoice #: 1304
Date 24/04/2024

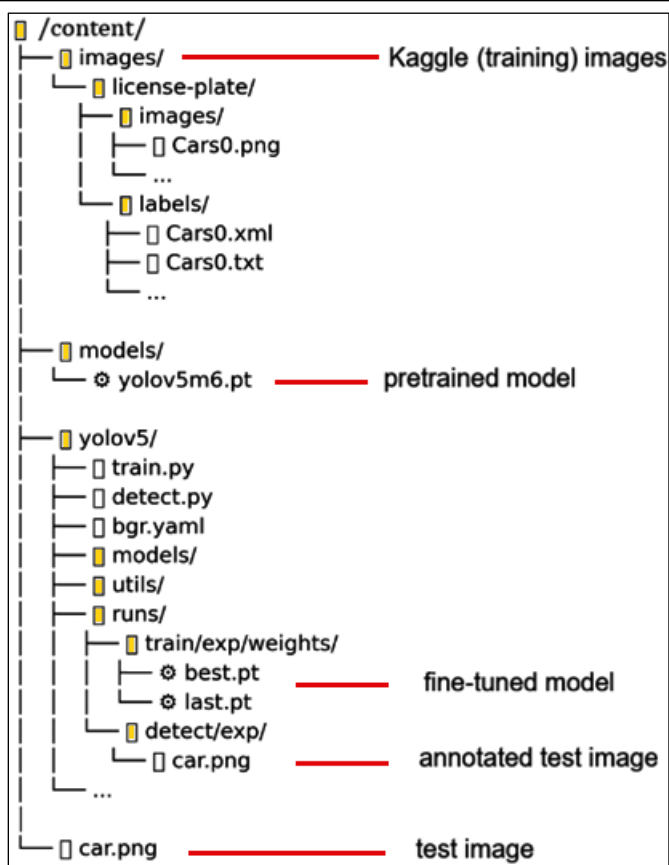
DESCRIPTION	QTY	RATE	AMOUNT
Paneer Butter Masala	1	₹180.00	2180.00
Veg Biryani	1	₹150.00	150.00
Tandoori Roti	3	220.00	260.00
Gulab Jamun	2	100.00	7100.00
CGST @ 2.5%			212.25
SGST @ 2.5%			212.25
TOTAL			7514.50



Detected Text:
YOU ARE UNDER N
SCTV. PARKING
আপনি সিসিটিভি নজরদারি করছেন পার্কিং নিষেধ

PaddleOCR High-Quality Visualization

1: Cheque No.876543 0.889
 2: ABC Bank 0.981
 3: Date25052024 0.978
 4: John Doe 0.933
 5: DDMMYYYY 0.968
 6: Pay 0.999
 7: Rupees Twenty Five Thousand Only 0.946
 8: 25,000/- 0.937
 9: A/cNo.12345678901234 0.939
 10: Signature 0.994
 11: Huol 0.836
 12: 234561234567890123412345629 0.966



License plate detection with Yolo-V5 + Tesseract



EasyOCR Plate Detection



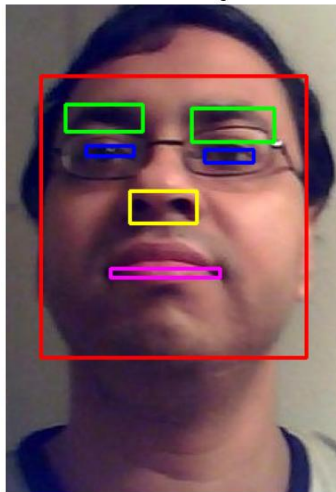
License Plate Recognition with TrOCR



Original Image



Detected Face Regions



Snapchat-style Filters Applied



MediaPipe Hand Tracking (Tasks API)

Original Image



Hand Landmarks



Original Image



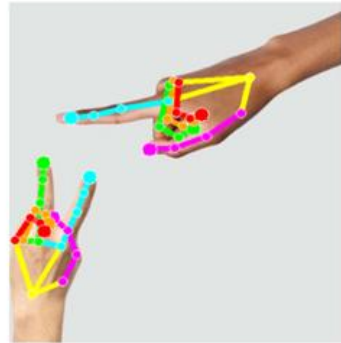
Hand Landmarks



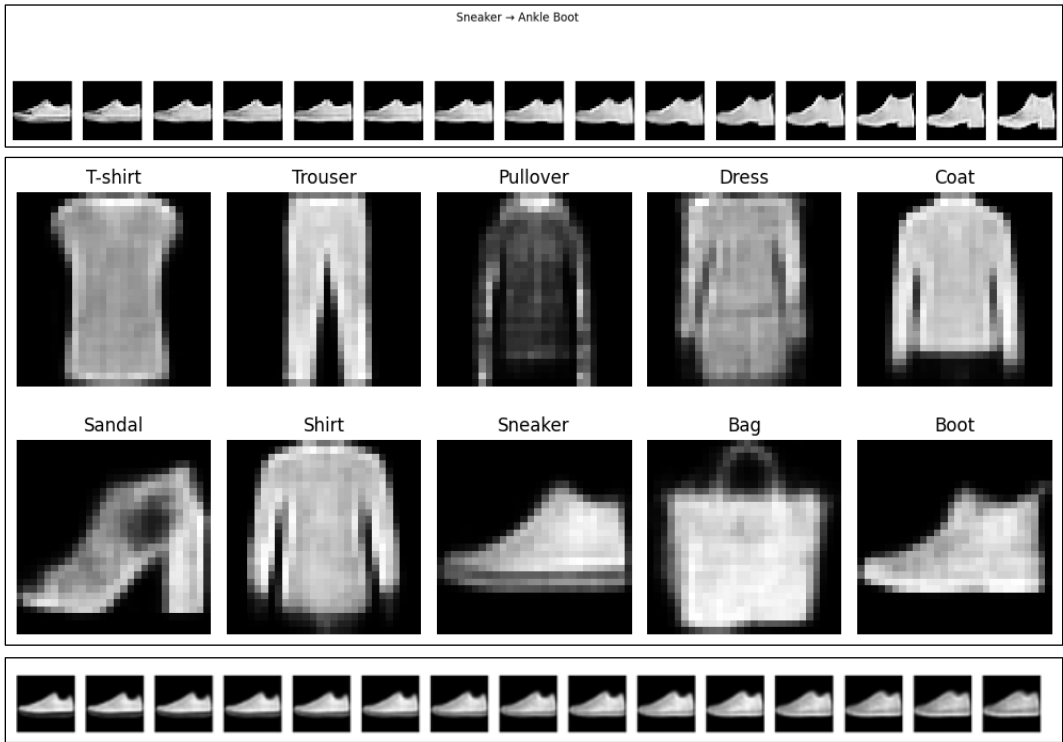
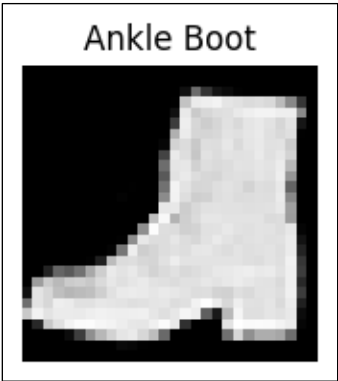
Original Image



Hand Landmarks



Chapter 13: Generative AI in Image Processing and Computer Vision



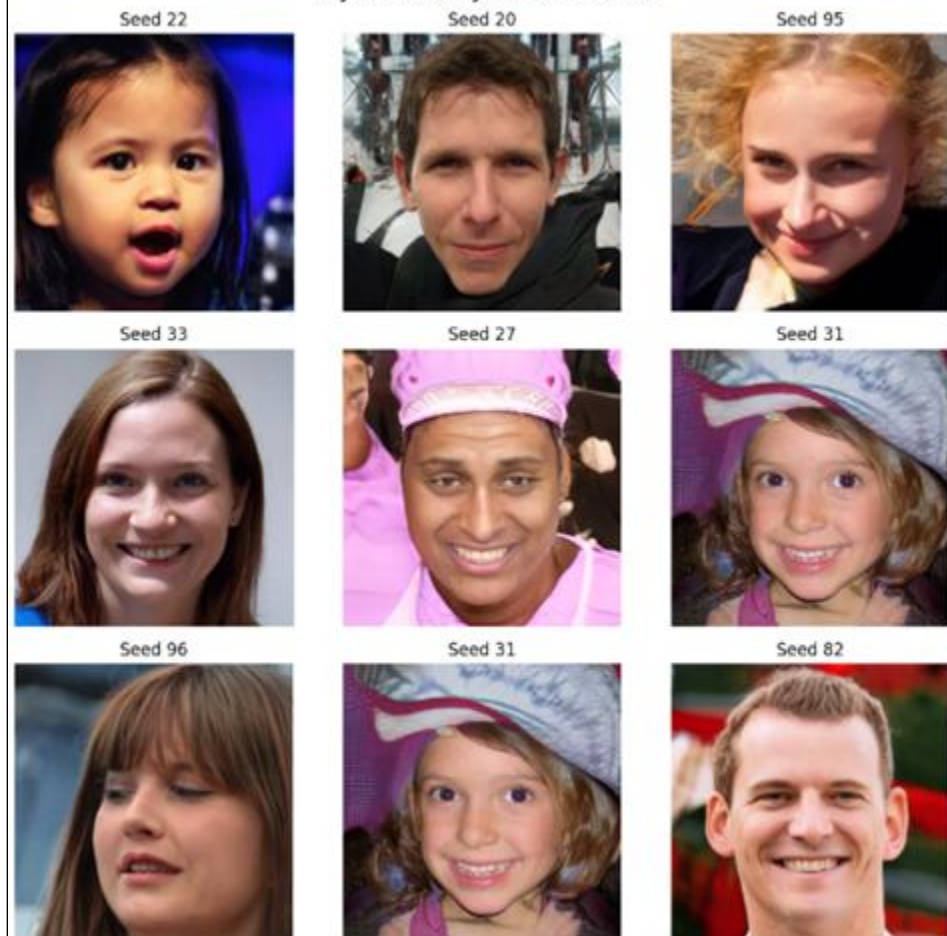
Input Horse



Generated Zebra



StyleGAN3 Synthetic Faces

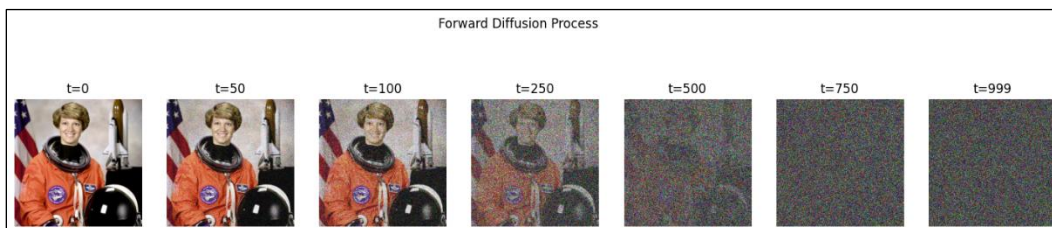
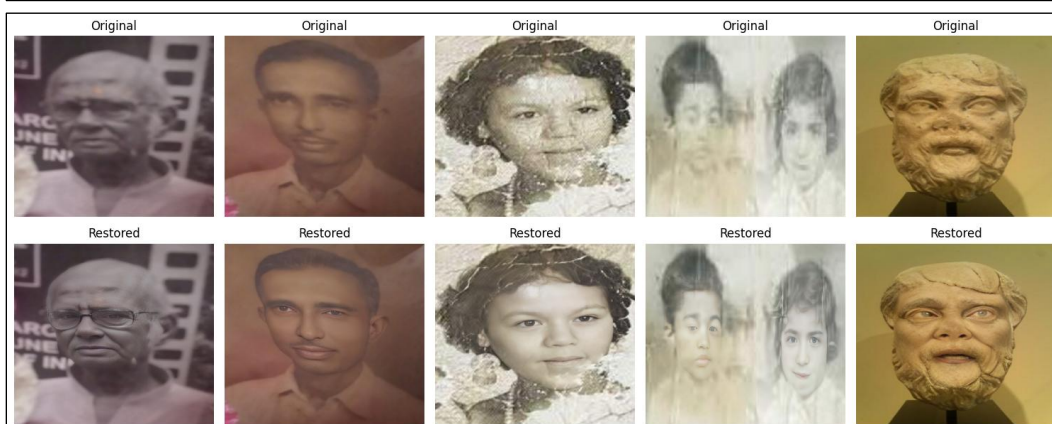


Latent Space Interpolation



StyleGAN Morph Progression





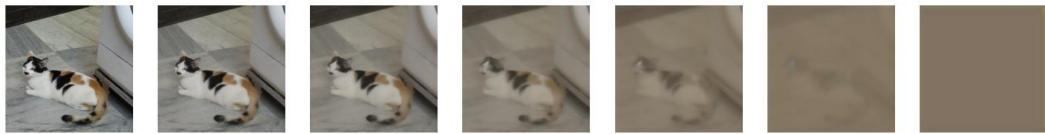
Original



Reconstructed



VAE Decoding of Scaled Latents



A futuristic city



A medieval castle



A cyberpunk street



A photorealistic tiger



Input: Sketch



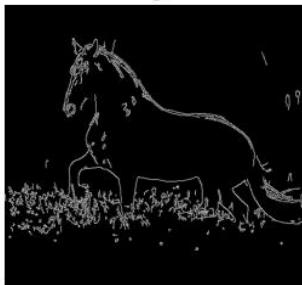
Output: Photorealistic



Horse



edges



Controlnet + StableDiffusion





A cat



A realistic cat



A realistic Persian cat

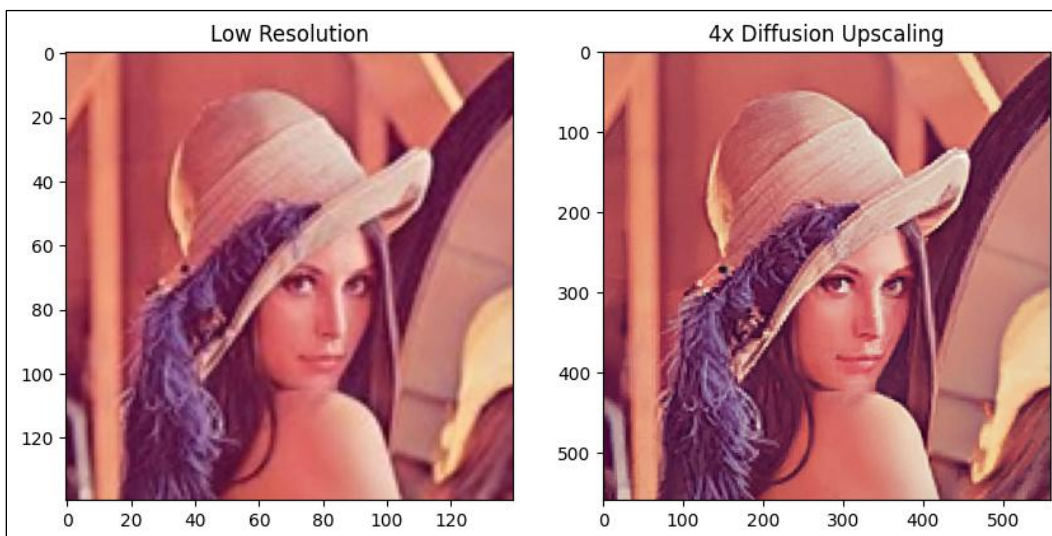
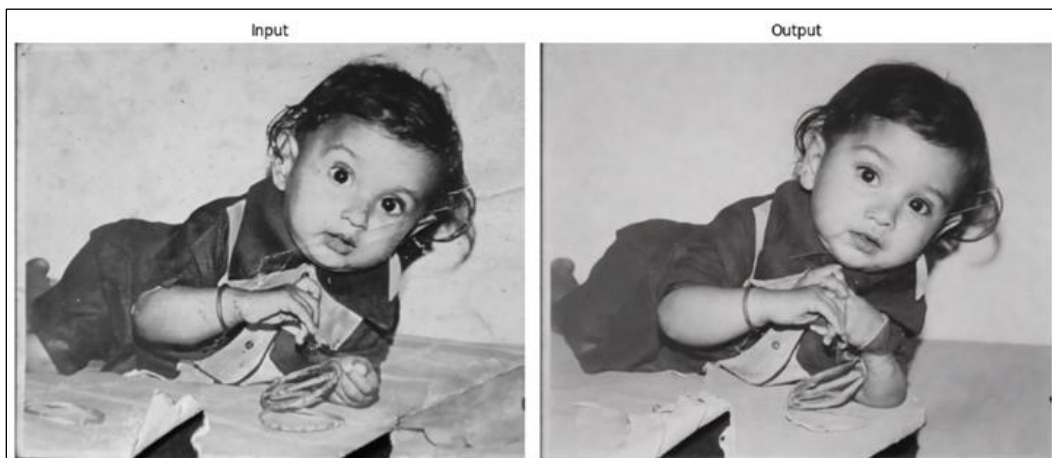
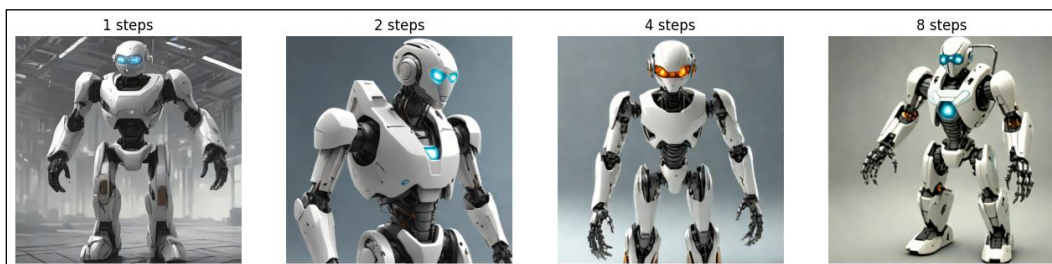


A realistic Persian cat sitting on a sofa



A realistic Persian cat sitting on a sofa, cinematic lighting





Original



Mask



Inpainted



Prompt 1: Photo of Victoria Memorial Hall in the winter. A yellow taxi is standing in front with snows over it. On the right, a boy is taking selfie.

DALL-E 2 (Base)



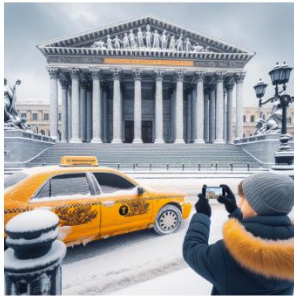
DALL-E 2 (Var 1)



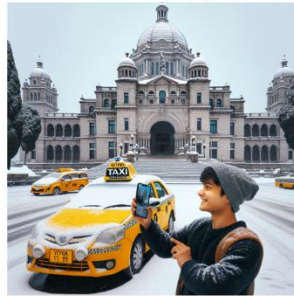
DALL-E 2 (Var 2)



DALL-E 3 (Base)



DALL-E 3 (Var 1)



DALL-E 3 (Var 2)





original image



edited image



Original

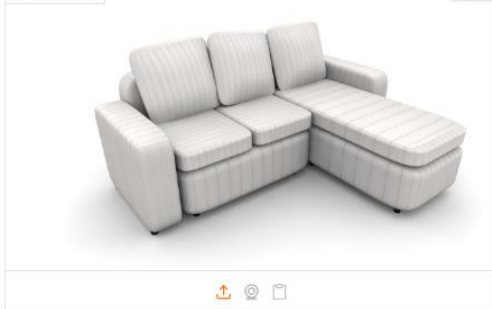


GPT Image Restoration



E-commerce Product Generator

image_path



Clear

Submit

output



Flag

Use via API - Built with Gradio - Settings

Input Grayscale



Colorized



Input sketch



photorealistic



Input painting



photorealistic



Input



Output



original



person removed



original



replaced background



night



day



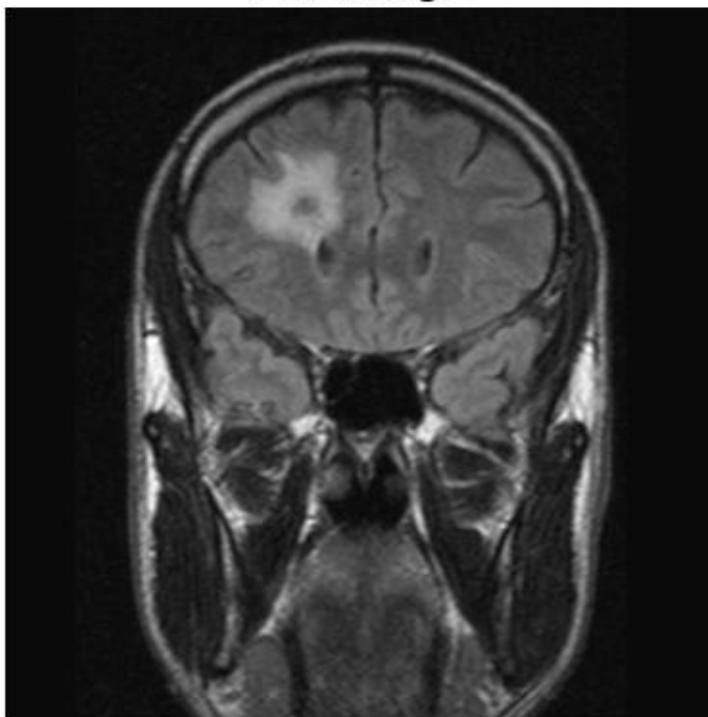
summer



winter



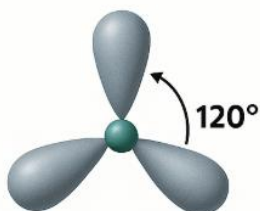
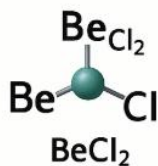
MRI image



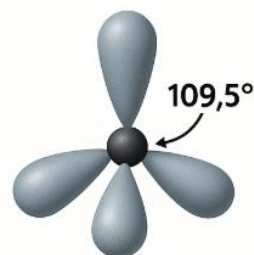
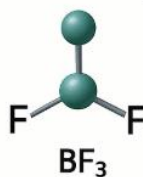
HYBRIDIZATION



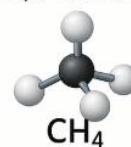
sp
HYBRIDIZATION
Linear geometry
180° bond angle



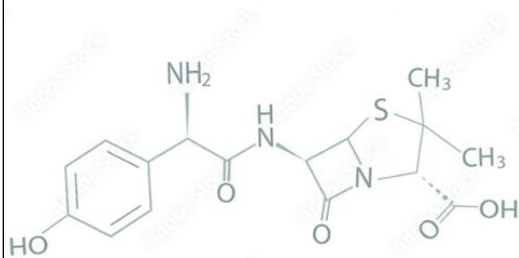
sp²
HYBRIDIZATION
Trigonal planar geometry
120° bond angle



sp³
HYBRIDIZATION
Tetrahedral geometry
109,5° bond angle



input drug (AMOXICILLIN) molecule



New drug-like molecule generated

