1.

Interpolation Methods

Interpolation methods are used to estimate unknown values that fall within the range of known data points. In sound representation, interpolation can be used to reconstruct missing samples, upsample audio data, or smooth out noise.

Linear interpolation

Linear interpolation estimates the value between two known points by assuming a straight line between them.

For each interval $[x_i, x_{i+1}]$ where $i = 0, 1, ..., n - 1$, the linear spline $L_{i(x)}$ is a linear polynomial given by:

$$L_{i(x)} = y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i}(x - x_i)$$

Simplicity and Efficiency: Linear interpolation is computationally efficient and easy to implement. This is important for real-time audio processing where computational resources may be limited.

Low Latency: The simplicity of the method ensures minimal processing delay, which is crucial for applications such as live audio streaming or real-time sound synthesis.

Trade-offs: While it is simple and efficient, linear interpolation may introduce noticeable artifacts, especially in signals with high frequency content or rapid changes. This can result in a piecewise linear approximation that may sound harsh or "clicky."

Cubic spline interpolation

Cubic spline interpolation constructs a piecewise cubic polynomial that interpolates a set of data points

$(x_0, y_0), (x_1, y_1), ..., (x_n, y_n)$

For each interval $[x_i, x_{i+1}]$ the cubic spline $S_i(x)$ is given by:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

Justification:

Smoothness: Cubic spline interpolation provides a smooth transition between points, preserving the natural flow of audio signals. This smoothness is essential for avoiding artifacts that can occur with simpler methods.

Natural Sounding Interpolation: The continuity of the first and second derivatives ensures that the interpolated signal mimics the natural physical properties of sound waves, leading to more natural sounding results.

Trade-offs: While providing smoothness, cubic splines are more computationally intensive than linear interpolation, which may be a limitation in some real-time applications.

Piecewise Cubic Hermite Interpolating Polynomial

$$H_i(x) = h_{00}(x)y_i + h_{10}(x)(x - x_i)y'_i + h_{01}(x)y_{i+1} + h_{11}(x)(x - x_{i+1})y'_{i+1}$$

where the basis functions are defined as:

$$h_{00}(x) = \left(1 + 2\frac{x-x_i}{x_{i+1}-x_i}\right)\left(1 - \frac{x-x_i}{x_{i+1}-x_i}\right)^2$$

$$h_{10}(x) = (x - x_i)\left(1 - \frac{x-x_i}{x_{i+1}-x_i}\right)^2$$

$$h_{01}(x) = \left(1 + 2\frac{x_{i+1}-x}{x_{i+1}-x_i}\right)\left(1 - \frac{x_{i+1}-x}{x_{i+1}-x_i}\right)^2$$

$$h_{11}(x) = (x - x_{i+1})\left(1 - \frac{x_{i+1}-x}{x_{i+1}-x_i}\right)^2$$

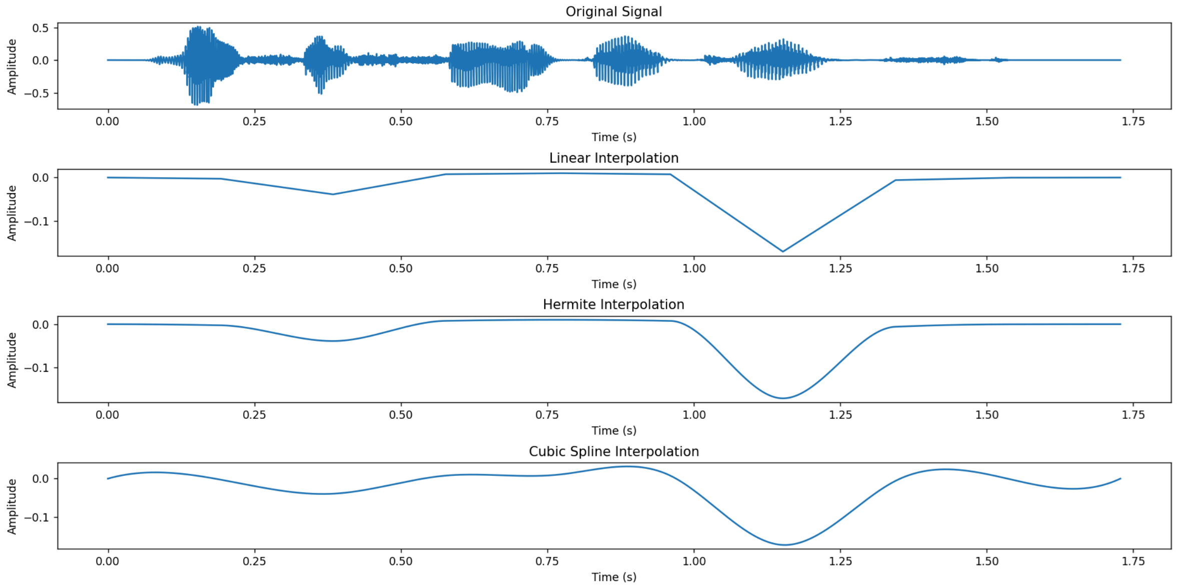
Justification:

Local Control: PCHIP provides better local control compared to cubic splines, as it uses the derivatives at each point. This can be particularly useful in sound applications where local features and transient details are important.

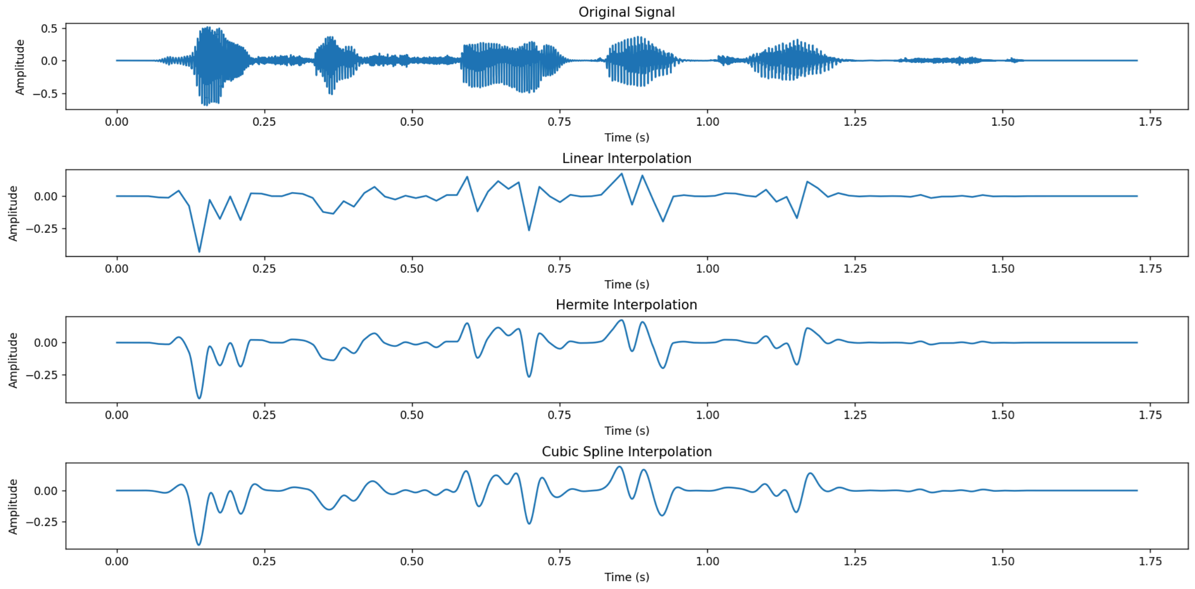
Monotonicity Preservation: PCHIP can preserve the shape of the data, avoiding the oscillations that can sometimes occur with cubic splines. This is beneficial for maintaining the integrity of the original audio signal.

Trade-offs: Similar to cubic splines, PCHIP is more computationally demanding than linear interpolation. The need to compute and use derivatives adds to the complexity.

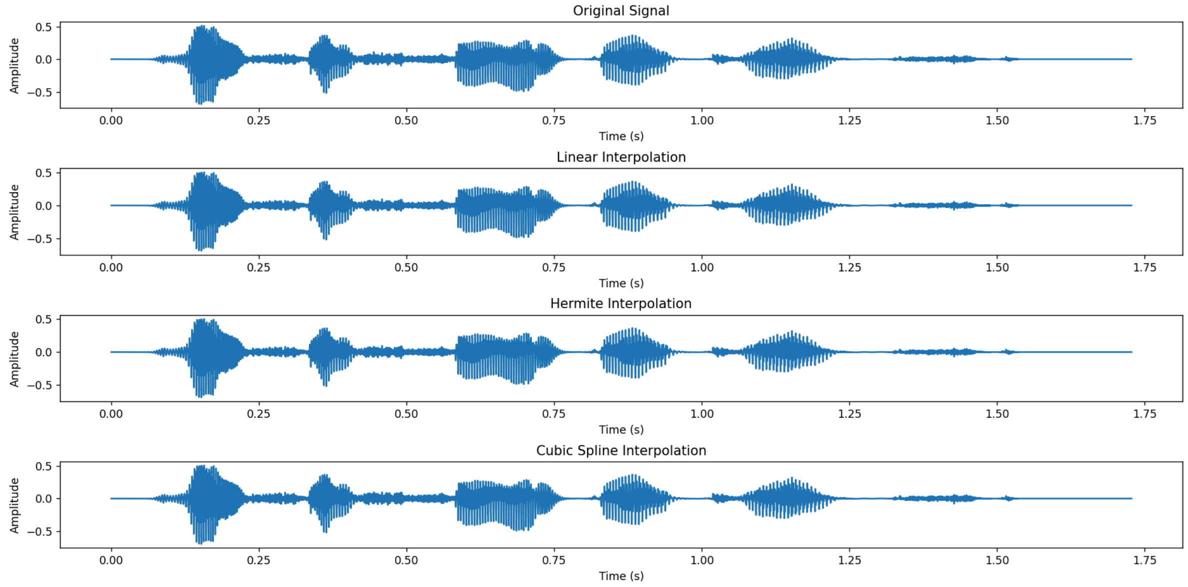
Interpolating 10 points



Interpolating 100 points



Interpolating 100000 points



2.
Radial Basis Functions (RBFs)

Radial Basis Functions are a popular choice for interpolation and function approximation due to their simplicity and efficiency. When applied to image restoration, RBFs help in reconstructing missing or corrupted parts of an image by fitting a surface that passes through known pixel values. The general form of an RBF interpolation is given by:

$f(x) = \sum_{i=1}^{N} \lambda_i \Phi(\|x - x_i\|)$

where

λ_i are the weights.

Φ is the RBF

$\|x - x_i\ \|$ is the Euclidean distance between the point x and the data point x_i

$$\begin{pmatrix} \varphi(\|x_0 - x_0\|) & \cdots & \varphi(\|x_n - x_0\|) \\ \vdots & \vdots & \vdots \\ \varphi(\|x_0 - x_n\|) & \cdots & \varphi(\|x_n - x_n\|) \end{pmatrix} \begin{pmatrix} w_0 \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} f(x_0) \\ \vdots \\ f(x_n) \end{pmatrix}$$

Gaussian RBF

Method: The Gaussian RBF is known for its smoothness and is particularly useful when a smooth reconstruction is desired. The shape parameter σ determines the width of the Gaussian curve.

Justification: Gaussian RBFs provide excellent smoothness and are less sensitive to noise, making them suitable for image restoration where gradual transitions are necessary. By adjusting σ, we can control the level of smoothness, ensuring that the restored image retains its natural appearance without introducing artifacts.

Gaussian RBF: $\Phi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right)$ where σ is a shape parameter

Multiquadratic RBF

Method: Multiquadratic RBFs provide a balance between local detail preservation and global smoothness. The shape parameter σ affects the balance between these two aspects.

Justification: Multiquadratic RBFs are flexible and can handle both smooth areas and areas with more detail effectively. By adjusting σ, we can fine-tune the interpolation to either emphasize local details or ensure a smoother global reconstruction, making them versatile for different types of image restoration tasks.

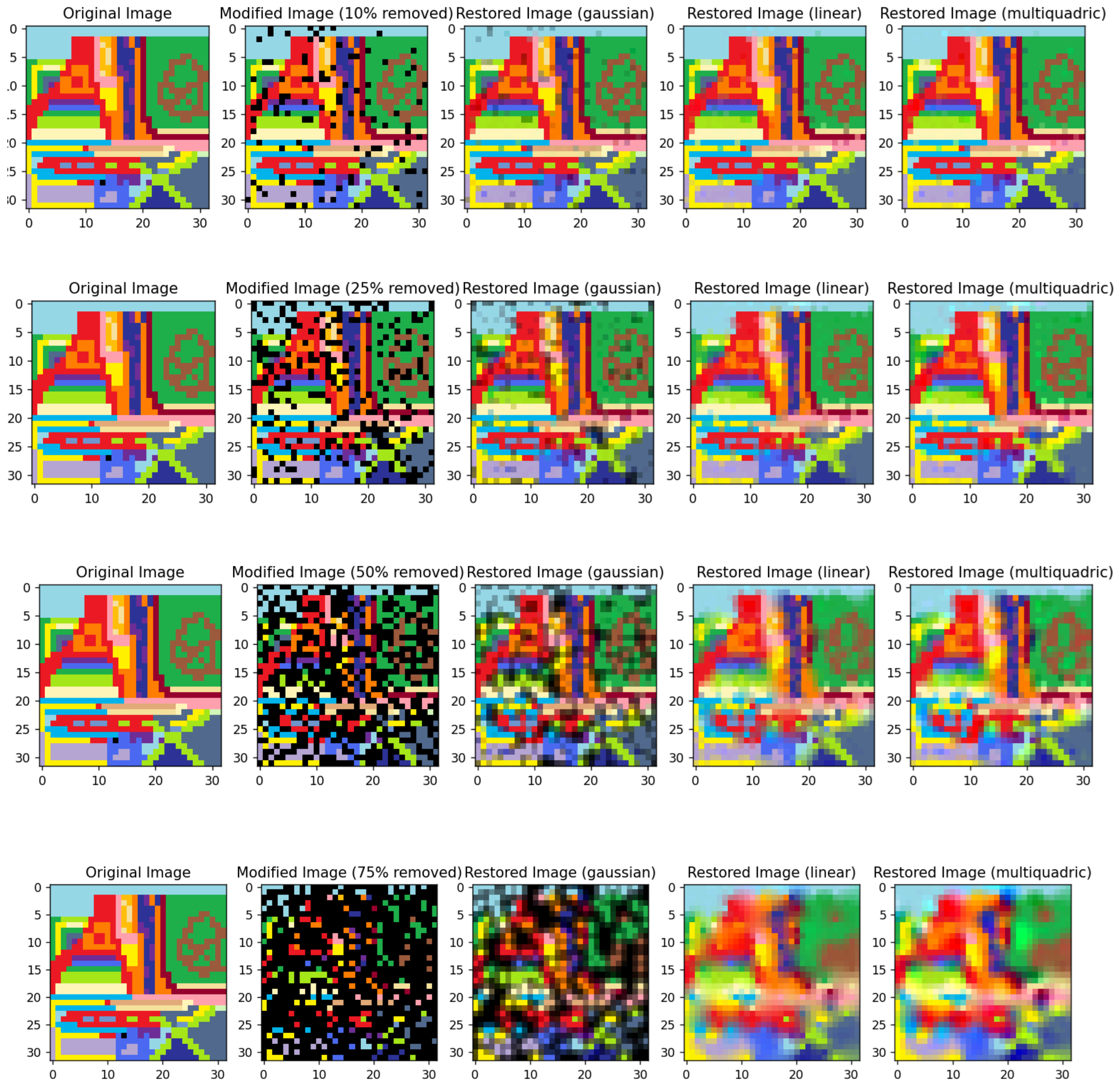
Multiquadratic RBF: $\Phi(r) = \sqrt{(1 + (c^2 r^2))}$ where c is a non-negative shape parameter

Linear RBF

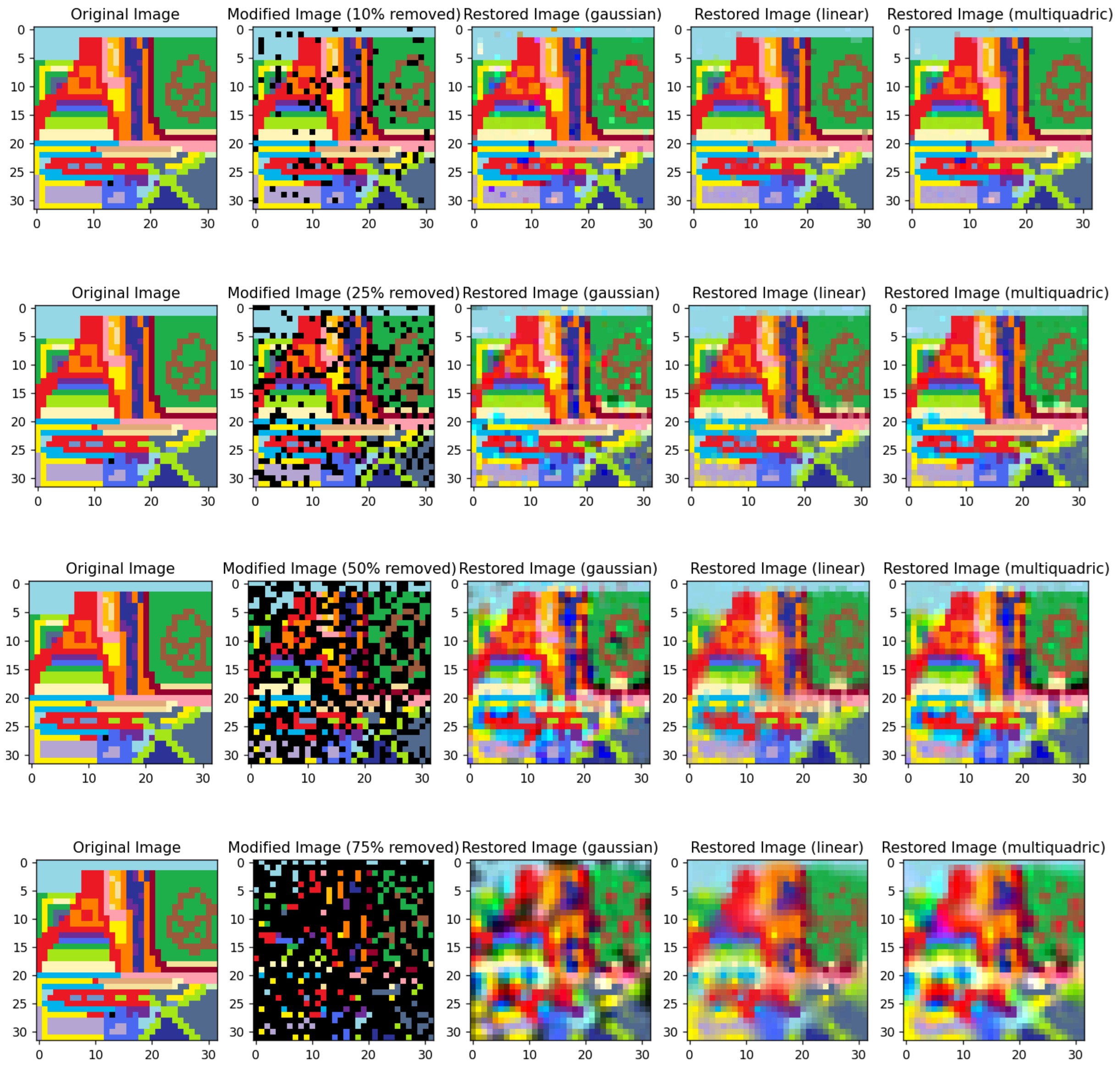
Method: Linear RBFs are simple and computationally efficient. They linearly interpolate between known data points.

Justification: Linear RBFs are straightforward and work well for restoring images with linear trends or when high computational efficiency is needed. However, they may not perform as well in capturing non-linear features or fine details. Adjusting the shape parameter is less relevant here, as the linear RBF does not have a shape parameter like Gaussian or Multiquadratic RBFs. Linear RBF: $\Phi(r) = r$

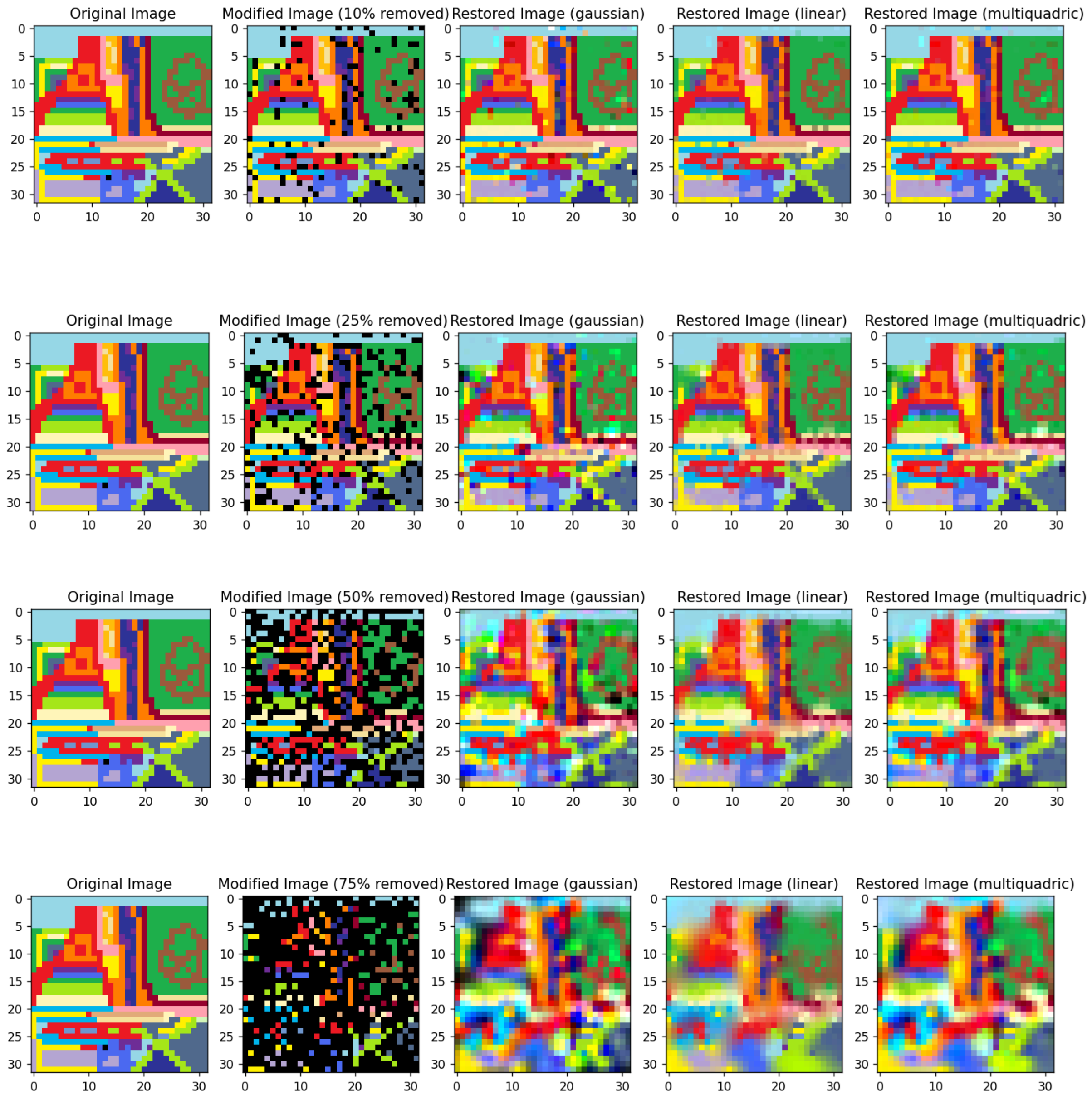
$\sigma = 1$


Original Image
Modified Image (10% removed)
Restored Image (gaussian)
Restored Image (linear)
Restored Image (multiquadric)
Original Image
Modified Image (25% removed)
Restored Image (gaussian)
Restored Image (linear)
Restored Image (multiquadric)
Original Image
Modified Image (50% removed)
Restored Image (gaussian)
Restored Image (linear)
Restored Image (multiquadric)
Original Image
Modified Image (75% removed)
Restored Image (gaussian)
Restored Image (linear)
Restored Image (multiquadric)

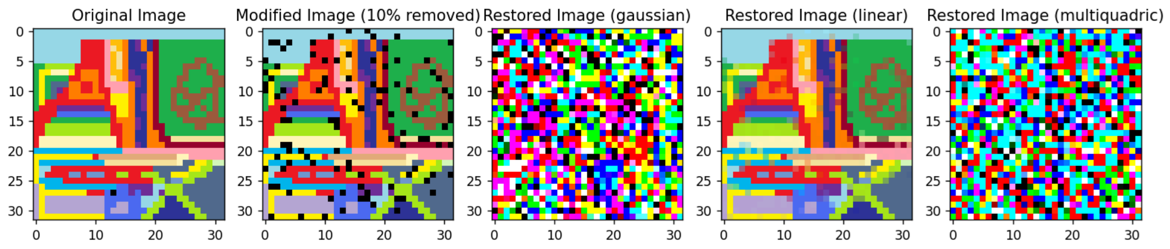
$\sigma = 1.5$

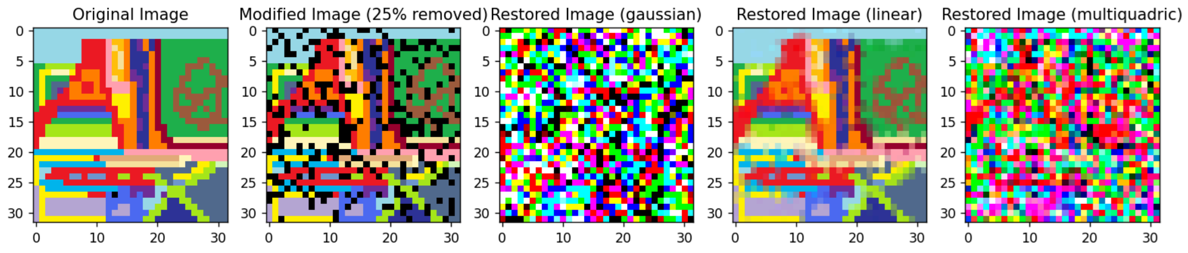

Original Image
Modified Image (10% removed)
Restored Image (gaussian)
Restored Image (linear)
Restored Image (multiquadric)
Original Image
Modified Image (25% removed)
Restored Image (gaussian)
Restored Image (linear)
Restored Image (multiquadric)
Original Image
Modified Image (50% removed)
Restored Image (gaussian)
Restored Image (linear)
Restored Image (multiquadric)
Original Image
Modified Image (75% removed)
Restored Image (gaussian)
Restored Image (linear)
Restored Image (multiquadric)

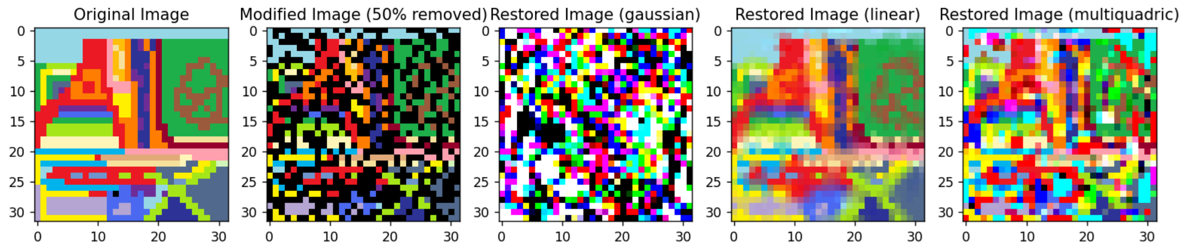
$\sigma = 2$

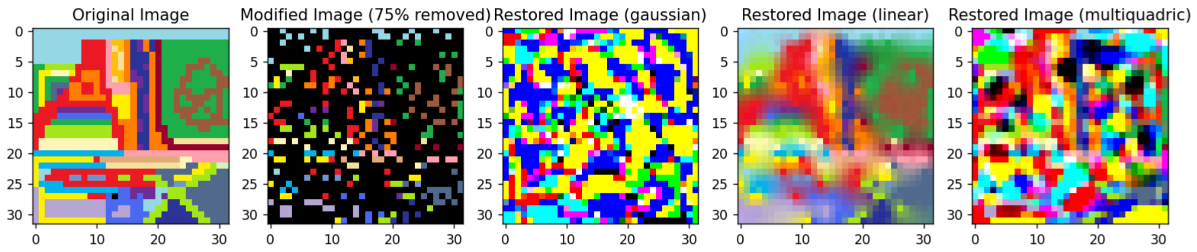

Original Image
Modified Image (10% removed)
Restored Image (gaussian)
Restored Image (linear)
Restored Image (multiquadric)
Original Image
Modified Image (25% removed)
Restored Image (gaussian)
Restored Image (linear)
Restored Image (multiquadric)
Original Image
Modified Image (50% removed)
Restored Image (gaussian)
Restored Image (linear)
Restored Image (multiquadric)
Original Image
Modified Image (75% removed)
Restored Image (gaussian)
Restored Image (linear)
Restored Image (multiquadric)

$\sigma = 10$


Original Image
Modified Image (10% removed)
Restored Image (gaussian)
Restored Image (linear)
Restored Image (multiquadric)


Original Image
Modified Image (25% removed)
Restored Image (gaussian)
Restored Image (linear)
Restored Image (multiquadric)


Original Image
Modified Image (50% removed)
Restored Image (gaussian)
Restored Image (linear)
Restored Image (multiquadric)


Original Image
Modified Image (75% removed)
Restored Image (gaussian)
Restored Image (linear)
Restored Image (multiquadric)

as we can see -> chosing σ has big impact on output for gaussian and multiquadratic rbf.