Image Interpolations

Joan Amorós Ramírez
Faculty of Electrical Engineering and Computing
Sveučilište u Zagrebu
Zagreb, Croatia

Miguel Cid Espeso
Faculty of Electrical Engineering and Computing
Sveučilište u Zagrebu
Zagreb, Croatia

Niladri Sarkar
Faculty of Electrical Engineering and Computing
Sveučilište u Zagrebu
Zagreb, Croatia

Oleksii Chumakov
Faculty of Electrical Engineering and Computing
Sveučilište u Zagrebu
Zagreb, Croatia

Auris Prääm
Faculty of Electrical Engineering and Computing
Sveučilište u Zagrebu
Zagreb, Croatia

Alessandro Zito
Faculty of Electrical Engineering and Computing
Sveučilište u Zagrebu
Zagreb, Croatia

***Abstract* — The project will be about image interpolation, an important process when analyzing images and extracting information from images.**

Keywords — interpolation, estimation, pixel, frame, non-neural, neural

I. INTRODUCTION

Image interpolation is a basic technique in the field of image processing, whose objective is to estimate the missing information (or fill the gaps with pixels) in an image.

However, the problem of image interpolation arises when it is necessary to generate more frames for a video sequence (f.p.s), improve the resolution of an image (increase the number of pixels) or also smooth out pixel transitions.

Besides that, the motivation of this procedure is to improve the quality and the details of the image while maintaining its general coherence and realism – despite images captured in many cases by cameras or obtained from various sources may lack the desired level of resolution, resulting in pixelated edges.

II. OVERVIEW OF THE PROJECT

In this project we will cover non-neural (nearest neighbour, bilinear and bicubic interpolation) and neural networks.

III. SOLUTION IMPLEMENTED

A. *Non-neural networks (nearest neighbour, bilinear and bicubic interpolation)*

We created three functions using Python in which parameters are the images and the scaling percentages. These functions follow the next algorithm:

- Calculating the new dimensions (width and height)
- Creating a zero-padded array to hold the resized image.
- Determining the scaling factor for every dimension.

Regarding bilinear interpolation, it also calculates the integer coordinates, the fractional distances and the pixel values of the four nearest pixels. After that, the pixel values are interpolated using the fractional distances and assigned to the resized image.

In the case of performing the algorithm with bicubic interpolation, it would follow the same steps as before but with the sixteen nearest pixels in the image.

B. *Neural networks*

We have also used Python to define the algorithm of neural networks, which consists of the following:

- In the first place it is required to import several modules from Python such as *Keras* for creating the deep learning model, *Matplotlib* for plotting and *OpenCV* for image processing.
- Then the 'Cifar' dataset is loaded, which contains 32x32 color images that can be used to teach a computer how to recognize objects.
- It is important to normalize the pixel values to be between 0 and 1, because it will help the model to learn more effectively.
- After all that, the images are resized to the original size and low-resolution versions of the images are created.
- An important step in this procedure is the definition of the autoencoder model, which will be sequential.
- The model is compiled with the Adam optimizer and binary cross-entropy (as the loss function).
- The autoencoder is trained using the low-resolution images as inputs and the original high-resolution images as targets.

- Lastly, a random image is chosen from the test set and an extra dimension is added for the batch size.

IV. Experimental Results

In the case of non-neural networks, we can appreciate the original and the resized image:

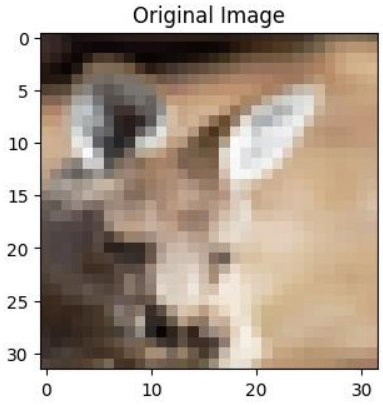

Fig. 1. Original image using non-neural networks.

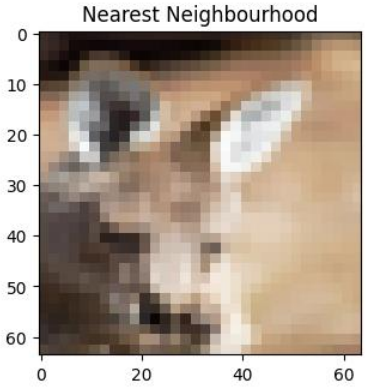

Fig. 2. Resized image using nearest neighbourhood.

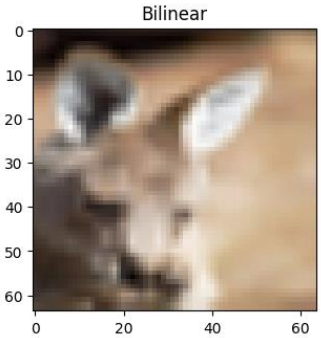

Fig. 3. Resized image using bilinear interpolation.

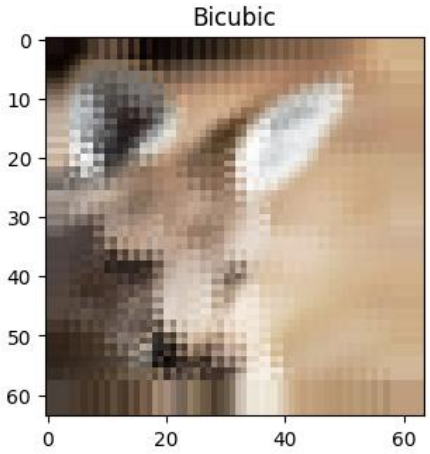

Fig. 4. Resized image using bicubic interpolation.

Regarding the neural ones, we can distinguish between the original low-resolution image, the upscaled and the original high-resolution images.

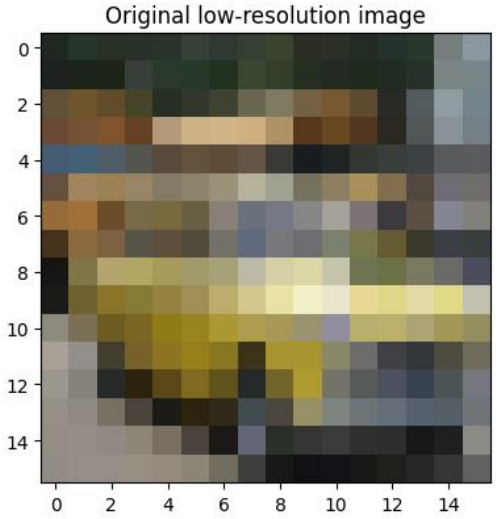

Fig. 5. Original low-resolution image using neural networks.

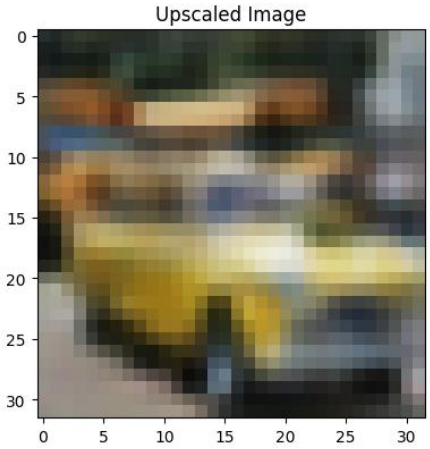

Fig. 6. Upscaled image using neural networks.

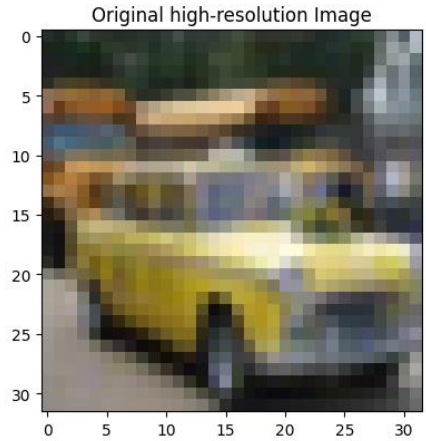

Fig. 7. Original high-resolution image using neural networks.

V. Conclusion

In conclusion, image interpolation is a fundamental method used to estimate pixel values in an image at locations where they are not explicitly defined. It plays a crucial role in various applications such as image resizing, zooming, rotation, and restoration.

For non-neural networks we can say that they can be interpreted easy and efficiently and can often perform well even with limited data. Despite that, their flexibility and adaptability are limited, and conventional image processing algorithms can struggle with recognizing and managing complex patterns.

In the case of neural networks, the advantages are feature learning (extract meaningful representations from an image), flexibility and generalization (can be adapted to various image processing tasks) and scalability. Nevertheless, its disadvantages are the complexity and lack of interpretability, data requirements (a large amount of labelled training data is needed) and computationally demanded.

VI. References

[1] numpy.org (for numerical computations)
[2] keras.io (deep learning model)
[3] matplotlib.org (plotting)
[4] opencv.org (image processing)
[5] pypi.org (cv2 module)
[6] scikit-image.org (skimage.io module)