MATHEMATICAL SUPPORT OF GEOMETRIC TRANSFORMATIONS OF IMAGES DURING DATA AUGMENTATION OF NEURON NETWORK TOOLS

Liudmyla Tereikovska[1]
Ihor Tereikovskyi[2]

DOI: https://doi.org/10.30525/978-9934-26-499-3-12

Abstract. One of the key problems in the field of increasing the efficiency of neural network tools intended for the analysis of graphic materials is the formation of representative training databases. A promising way to overcome this problem is to increase the size of the training sample by applying the augmentation of training examples due to geometric transformations. However, the modern mathematical apparatus for modifying the geometric parameters of images has shortcomings that can reduce the quality of the obtained images or lead to their insufficient compliance with the tasks. **The purpose of the work** is the formation of mathematical support, which is used to implement geometric transformations of images during the augmentation of training data of neural network tools. **The research methodology** is based on the theory of digital processing of signals and images, system analysis and the theory of neural networks and involves the determination of key components of the mathematical support for affine transformations and the definition of the mathematical support for calculating the color of pixels when scaling an image using interpolation methods in various application conditions. **As a result of the conducted research**, mathematical support was formed, which is used to implement geometric transformations of images when augmenting the training data of neural network tools. The key components of the mathematical support of affine transformations related to the determination of the display dimensions

[1]Doctor of Technical Sciences, Associate Professor,
Professor of the Department of Information Technologies Design and Applied Mathematics,
Kyiv National University of Construction and Architecture, Ukraine
[2] Doctor of Technical Sciences, Professor,
Professor of the Department of System Programming and Specialized Computer Systems,
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Ukraine

of the modified image and the determination of the color of individual points of the modified image are defined. A mathematical apparatus has been defined that allows you to calculate the dimensions of the display area of the modified image, provided that a rectangular display area is provided. It is shown that in the task of augmentation of training data of neural network tools, it is advisable to perform image scaling using proven non-adaptive interpolation methods: nearest neighbor, bilinear interpolation, bicubic interpolation, Lanzosh interpolation, box filter, triangular filter. A mathematical apparatus for calculating the pixel color of a scaled image using each of the above interpolation methods is defined. The conditions that determine the expediency of using the specified methods in solving practical problems are outlined. It is shown that the method of the nearest neighbor is advisable to use in the case of the need to ensure the simplicity of implementation and high productivity of the augmentation procedure, when the noise of small details and the distortion of the shape of objects containing thin lines can be neglected. The bilinear interpolation method is recommended for use under the conditions of ensuring a balance between the quality of the scaled image and computational costs, and the bicubic interpolation method – under the condition that there are no strict limits on the computational resource intensity of the processing process, when it is necessary to achieve high image quality during scaling. The Lanzosh interpolation method is recommended for scaling images in cases where maximum preservation of quality and detail is required. The box method and the triangular filter method are appropriate to use when the speed of scaling is more important than the quality of the image obtained after scaling. Prospects for further research are related to the development of a methodology for adapting the means of integrated modification of geometry and visual characteristics of images to the conditions of augmentation of training data of neural network systems for video stream analysis.

1. Introduction

The modern world increasingly relies on neural network technologies, which play a key role in solving complex tasks in areas such as medicine, industry, transportation and security. Continuous improvement of neural network tools remains a key direction of artificial intelligence development, which contributes to increasing the accuracy and reliability of automated solutions. A special place among neural network tools is occupied by convolutional neural networks, which are the de facto standard for image and video analysis. Their effectiveness is due to the ability to extract and process the spatial characteristics of images, which makes them indispensable in tasks such as object recognition, segmentation and classification.

One of the main problems in the development and training of such models is the difficulty of forming a large and representative training sample. Collecting, annotating, and ensuring the diversity of data often requires significant resources and time, which can limit the scale and quality of the models produced. An effective way to overcome this problem is to increase the size of the training sample by applying data augmentation – the process of artificially creating additional samples based on existing data. This makes it possible to increase the model's resistance to data variations and improve its ability to generalize [4; 15; 17].

In the context of the use of convolutional neural networks, usually intended for the analysis of raster images, augmentation includes a wide range of methods that can change the geometric, color and texture properties of images. Methods that change the geometry of the image attract special attention [6; 7; 16]. In most cases, the basis of such methods are affine transformations, which include rotation, scaling, shifting, skewing and mirroring of the original image. Such transformations mimic real image changes while preserving their semantic integrity, making them one of the most effective augmentation tools for neural network image analysis tools.

It is worth noting that the problem of modifying the geometry of images by applying affine transformations has been studied for many years [1; 3; 14]. However, most of the existing approaches are focused on the general tasks of computer vision and do not provide for the adaptation of the mathematical support to the specifics of the application conditions when modifying images in the task of augmenting the training data of neural network tools [8; 13; 15]. Thus, **the goal** of this scientific work is the formation of mathematical support, which is used to implement geometric transformations of images when augmenting the training data of neural network tools.

To achieve the set goal, the following tasks should be solved:

– Determine the key components of the mathematical support of affine transformations.

– Determine the mathematical support for calculating the color of pixels when scaling an image using interpolation methods in various conditions of application.

2. Key components of the mathematical support of affine transformations

When modifying the geometry of images using affine transformations, the dimensions, angles, orientation, and position of the image in space may change, but the straightness and parallelism of the lines are preserved. In the case of affine transformations, the ratio of the lengths of segments lying on the same or on parallel lines is preserved. The ratio of the areas of the figures is preserved. Any affine transformation can be matched with an inverse affine transformation, the result of which leads to the reproduction of the original image. Affine transformations include scaling, rotation, shift, skew, and mirroring [5; 9; 10].

Scaling – changing the size of the image horizontally and vertically.

Rotation – rotation of the image by a certain angle around a given point. At the same time, the size and shape of the image remain unchanged.

Shift – moving the image by a certain distance in a given direction without changing its shape, size and spatial orientation.

Skew is a change in the shape of the image in compliance with the requirements for collinearity and parallelism.

Mirroring – creating a mirror image by flipping it around a horizontal or vertical axis.

Affine transformations can be described using a matrix that is applied to all points of the image, allowing flexible and accurate control of its geometric characteristics [5; 11]. In a general form, the expression that determines the change in the coordinates of a point of a two-dimensional digital image after applying an affine transformation can be written as:

$$\begin{pmatrix} x_{AT} \\ y_{AT} \end{pmatrix} = \begin{pmatrix} \alpha_1 & \alpha_2 & \alpha_3 \\ \alpha_4 & \alpha_5 & \alpha_6 \end{pmatrix} \times \begin{pmatrix} x_B \\ y_B \\ 1 \end{pmatrix}, \qquad (1)$$

where x_B, y_B are the initial coordinates of the point; x_{AT}, y_{AT} – point coordinates after implementation of affine transformation; α_1 – scaling factor along the horizontal X axis; α_2 – image rotation factor relative to the X axis; α_3 – amount of displacement along the X axis; α_4 – coefficient of rotation relative to the vertical axis Y; α_5 – scaling factor along the Y axis; α_6 is the amount of displacement along the Y axis.

The implementation of scaling, rotation, shift, skew, and mirroring procedures can be described using expressions (2-7), respectively, obtained by adapting expression (1) to the specifics of the implementation of the corresponding procedure [5; 12]. Note that the mirroring procedure corresponds to expressions (6, 7), which determine the rotation around the X axis and the Y axis.

$$\begin{pmatrix} x_m \\ y_m \end{pmatrix} = \begin{pmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \end{pmatrix} \times \begin{pmatrix} x_B \\ y_B \\ 1 \end{pmatrix}, \tag{2}$$

$$\begin{pmatrix} x_r \\ y_r \end{pmatrix} = \begin{pmatrix} \cos(\varphi) & -\sin(\varphi) & 0 \\ \sin(\varphi) & \cos(\varphi) & 0 \end{pmatrix} \times \begin{pmatrix} x_B \\ y_B \\ 1 \end{pmatrix}, \tag{3}$$

$$\begin{pmatrix} x_s \\ y_s \end{pmatrix} = \begin{pmatrix} 1 & 0 & s_x \\ 0 & 1 & s_y \end{pmatrix} \times \begin{pmatrix} x_B \\ y_B \\ 1 \end{pmatrix}, \tag{4}$$

$$\begin{pmatrix} x_{sk} \\ y_{sk} \end{pmatrix} = \begin{pmatrix} 1 & \tan(\varphi_x) & 0 \\ \tan(\varphi_y) & 1 & 0 \end{pmatrix} \times \begin{pmatrix} x_B \\ y_B \\ 1 \end{pmatrix}, \tag{5}$$

$$\begin{pmatrix} x_{rx} \\ y_{rx} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \end{pmatrix} \times \begin{pmatrix} x_B \\ y_B \\ 1 \end{pmatrix}, \tag{6}$$

$$\begin{pmatrix} x_{rx} \\ y_{ry} \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} x_B \\ y_B \\ 1 \end{pmatrix}, \tag{7}$$

where x_B, y_B are the initial coordinates of the point; x_m, y_m – coordinates after scaling; x_r, y_r – coordinates after rotation; x_s, y_s – coordinates after displacement; x_{sk}, y_{sk} – coordinates after skewing; x_{rx}, y_{rx} – oordinates after rotation around X; x_{ry}, y_{ry} – coordinates after rotation around Y;

k_x, k_y – scaling factors along the X and Y axes; φ – rotation angle; s_x, s_y – shift along the X and Y axes; φ_x, φ_y – skew angles along the X and Y axes.

The results of applying the indicated procedures for modifying the fingerprint image are shown in Figure 1–4.

Analysis of Figure 1–4 and expressions (2–7) indicates that the display on the computer screen of a digital image modified by affine transformations leads to the need to solve the following problems:

Determining the dimensions of the screen area on which the modified image should be displayed.

Determining the color of individual points (pixels) of the image.

As for the first of the mentioned tasks, the need to take into account a possible change in the size of the screen area intended for displaying the modified image arises in the following affine transformations and is explained as follows:

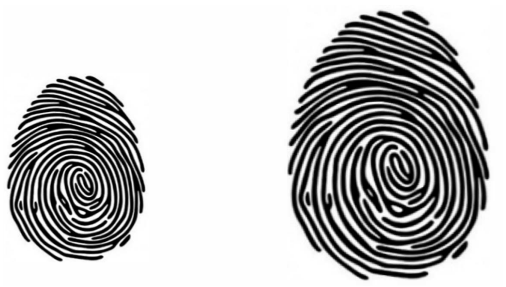
Figure 1. Scaling the image

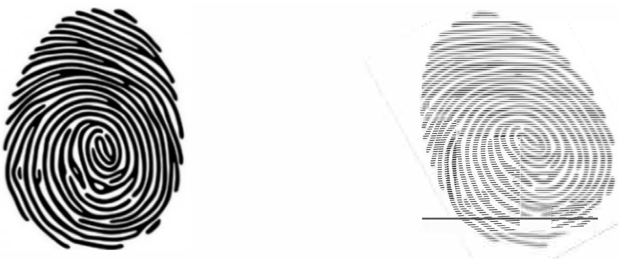
Figure 2. Image rotation

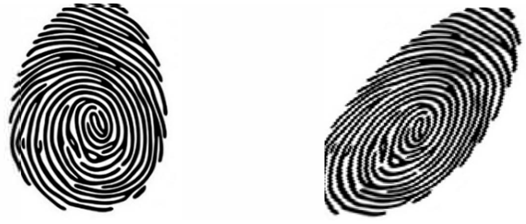

Figure 3. Skewing the image

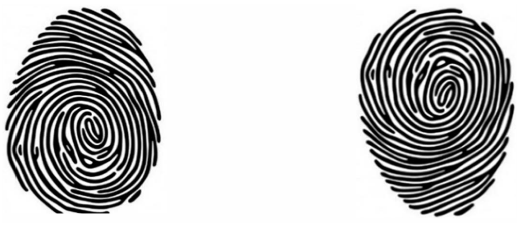

Figure 4. Mirroring the image relative to the horizontal axis

– Scaling resizes the image by making it larger or smaller. When enlarging the image, its dimensions may exceed the original display area, which will require an expansion of the screen area. Similarly, when reducing an image, the size of the display area may be too large compared to the new, reduced image.

– Rotation can cause the corners of the image to go beyond the original display area, especially if the rotation angle is not a multiple of 90 degrees. In this case, to correctly display the image without cropping, you need to expand the screen area. For example, when rotating a rectangular image by 45 degrees, it may take up more space diagonally than in its original position.

– Skewing the image can lead to a change in shape, for example, from a rectangle to a parallelogram. As a result, the image may take up more or less space than the original display area. If the new image extends beyond the original area, it needs to be expanded.

– Shifting the image leads to the need to increase the size of the display area by the size of the shift.

As a result of performing the specified affine transformations, the need to expand the display area arises when the image size increases, which leads to the exit of parts of the image beyond the original screen area.

The need to reduce the display area may occur when the image is reduced, when the new area is optimized to display a smaller image.

As for the second of the specified tasks, the need to take into account the possible color change of individual pixels is explained as follows:

– Scaling consists of enlarging or reducing the image, which leads to the need to determine the color of the new pixels that appear as a result of the increase or decrease in size.

– Rotating and skewing the image can cause some pixels to move to non-integer positions. In such cases, interpolation is required to determine the color of pixels that do not match the original image pixels. This is important from the point of view of avoiding the appearance of "torn" edges or sharp transitions in color.

In the basic case, the calculation of the modified dimensions of the display area is implemented according to the following conditions [2; 18]:

The display area has the shape of a rectangle.

The size of the display area is equal to the size of the image:

$$D_x = L_x , \tag{8}$$

$$D_y = L_y , \tag{9}$$

where D_x, D_y – are the horizontal and vertical dimensions of the display area; L_x, L_y – horizontal and vertical dimensions of the image.

In turn, the dimensions of the modified image can be calculated using the following expressions:

$$L_{x,m} = L_{x.b} \cdot k_x , \tag{10}$$

$$L_{y,m} = L_{y,b} \cdot k_y , \tag{11}$$

$$L_{x,r} = L_{x,b} \cdot \left|\cos(\varphi)\right| + L_{y,b} \cdot \left|\sin(\varphi)\right| , \tag{12}$$

$$L_{y,r} = L_{x,b} \cdot \left|\sin(\varphi)\right| + L_{y,b} \cdot \left|\cos(\varphi)\right| , \tag{13}$$

$$L_{x,s1} = L_{x,b} + L_y \cdot \left|\tan(\varphi_x)\right| , \tag{14}$$

$$L_{y,s1} = L_{y,b} , \tag{15}$$

$$L_{x,s2} = L_{x,b} , \tag{16}$$

$$L_{y,s2} = L_{y,b} + L_{x,b} \cdot \left|\tan\left(\varphi_y\right)\right|, \tag{17}$$

where $L_{x,b}$, $L_{y,b}$ are the initial horizontal and vertical dimensions of the image; $L_{x,m}$, $L_{y,m}$ – image dimensions after scaling; k_x, k_y – scale factors along the X and Y axes; $L_{x,r}$, $L_{y,r}$ – image dimensions after rotation; φ – image rotation angle; $L_{x,s1}$, $L_{y,s1}$ – image dimensions after skewing relative to the X axis; $L_{y,s2}$, $L_{x,s2}$ – image dimensions after skewing relative to the Y axis; φ_x, φ_y – skew angles along the X and Y axes, respectively.

In the case of implementation of several affine transformations, the dimensions of the modified image are calculated by successive application of the corresponding calculation expressions.

We will provide some explanations regarding the scaling procedure, which is traditionally considered key in the tasks of modifying the dimensions of digital images. As already mentioned, the purpose of the scaling procedure can be to increase or decrease the size of the image. A distinction is made between scaling with preservation of proportions and without preservation of proportions. Scaling with preservation of proportions involves increasing or decreasing the size of the image proportionally along both axes (width and height). The aspect ratio remains unchanged, that is, the image is not distorted. Scaling without preserving proportions involves modifying the width and height of the image independently of each other. This can cause the image to become distorted as its proportions change.

The scaling factor is a numerical value that determines the amount of change in image size during the scaling process. This coefficient shows how many times the size of the image should be increased or decreased along one or more axes (width and/or height). If the scaling factor is the same for width and height, the image is scaled proportionally. A scale factor less than one means a smaller image size. Scale factors can be different for width and height used for non-aspect scaling.

3. Pixel color calculation when scaling an image

Most known methods of image geometry modification use a digital image scaling procedure that takes into account the need to solve the problem of determining the color of individual pixels of the modified image. This procedure is implemented using various interpolation methods [5; 6; 12].

In general, the concept of interpolation describes the procedure for finding the values of a function at points between known values. Simply put, interpolation allows you to fill in the gaps between known data to construct a continuous function or sequence. In the problem of determining the color of pixels when scaling an image, the image interpolation procedure is to calculate new pixel values to create a modified version of the image. Well-known image interpolation methods are divided into adaptive and non-adaptive.

Adaptive methods of image interpolation involve determining the color of a pixel depending on the local characteristics of the image area to be interpolated. For example, when using adaptive methods, determining the color of a pixel located on the border of an object will differ from determining the color of a pixel related to the background of the image. Non-adaptive methods of image interpolation do not involve adapting the pixel color determination procedure to the subject of interpolation. In the case of their application, all pixels are processed in the same way without taking into account the local characteristics of the image.

Adaptive image interpolation methods are useful in cases where high quality image processing is required, especially if the images contain complex details, textures, or clearly defined boundaries. As a rule, these methods are used in the processing of medical images and in the restoration of images. At the same time, their use can lead to certain difficulties, which are primarily referred to:

– High computational complexity. Adaptive methods require more computing resources and are characterized by a longer processing interval. This is explained by the fact that their application involves the analysis of local characteristics of each image area, which complicates the processing process.

– Difficulty in implementing and defining settings. In adaptive interpolation methods. typically use complex algorithms that are more difficult to implement and optimize compared to simple algorithms used by non-adaptive methods. In addition, incorrect settings, for example, inaccurate definition of texture boundaries, can lead to distortion of content or loss of important image details.

Taking into account the identified difficulties, the use of adaptive methods in cases of processing relatively simple images, under the conditions of predictability and stability of the obtained results, limited processing time and limited computing resources, it is advisable to use non-adaptive methods of image interpolation. In tasks related to the scaling of digital images, the tested non-adaptive interpolation methods include: nearest neighbor, bilinear interpolation, bicubic interpolation, Lanzosh interpolation, box filter, triangular filter. Let's consider the main aspects of these methods.

Nearest Neighbor Method – involves determining the color of a new pixel based on the color of the nearest neighboring pixel. Calculating the pixel colors of a scaled image is implemented using following expressions:

$$x_{nr} = Round\left(\frac{x_m}{k_x}\right), \tag{18}$$

$$y_{nr} = Round\left(\frac{y_m}{k_y}\right), \tag{19}$$

$$I\left(x_m, y_m\right) = I\left(x_{nr}, y_{nr}\right), \tag{20}$$

$$I_R\left(x_m, y_m\right) = I_R\left(x_{nr}, y_{nr}\right), \tag{21}$$

$$I_G\left(x_m, y_m\right) = I_G\left(x_{nr}, y_{nr}\right), \tag{22}$$

$$I_B\left(x_m, y_m\right) = I_B\left(x_{nr}, y_{nr}\right), \tag{23}$$

where x_m, y_m are pixel coordinates of the scaled image; x_{nr}, y_{nr} – coordinates of the pixel of the initial image, which is considered the closest to the pixel of the scaled image with coordinates x_m, y_m; k_x, k_y – scale factors along the X and Y axes; $Round$ – the operation of finding the nearest larger whole number; I – color intensity for a single-channel image; I_R, I_G, I_G – color intensity for the RGB image in the R, G, B color channel, respectively.

The main advantages of the nearest neighbor method are: ease of implementation, high productivity, the ability to preserve clear boundaries of objects, the absence of parameters that require adaptation.

Disadvantages of the method: occurrence of torn edges of objects, noise of small details and transitions between colorful parts, distortion of thin lines and small elements. In Figure 5 shows the result of applying this method for scaling a halftone image ($k = 4$), which displays a text fragment on a background of a texture consisting of diagonal lines.

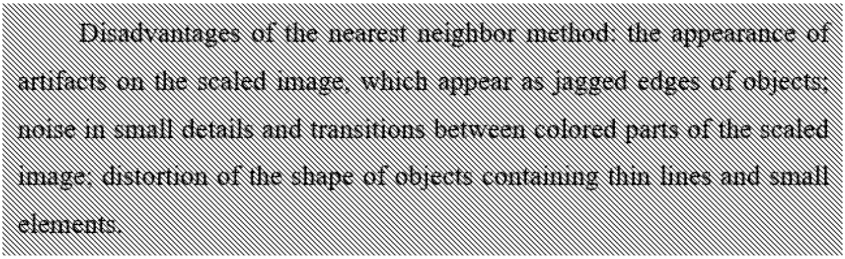
Disadvantages of the nearest neighbor method: the appearance of artifacts on the scaled image, which appear as jagged edges of objects; noise in small details and transitions between colored parts of the scaled image; distortion of the shape of objects containing thin lines and small elements.

Figure 5. Scaled image using the nearest neighbor method

As shown in Figure 5, distortions appeared on the scaled image, consisting in the appearance of torn edges of the letters and characterized by the loss of straightness of the diagonal lines of the texture.

Similar distortions also occur when scaling images containing small elements, which is illustrated by Figs. 6 and Figure 7, which show the original image and the image scaled using the nearest neighbor method with $k = 3$.

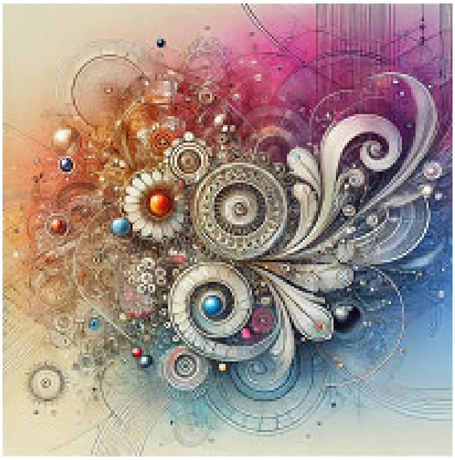

Figure 6. Initial image to be scaled

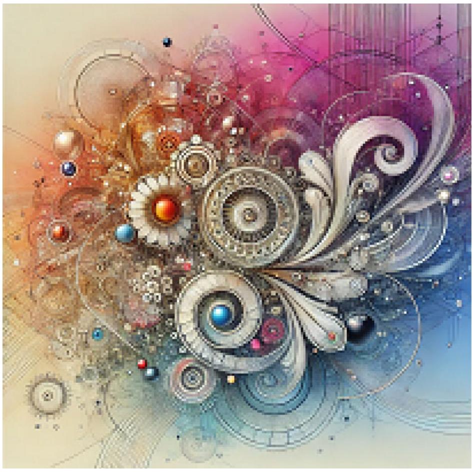

Figure 7. The result of using the nearest neighbor method with *k* = 3

A comparison of the indicated figures shows the loss of the quality of the scaled image even with a small value of the scaling factor. The nearest neighbor method is used when high scaling speed is required and the quality of the processed image is not a critical factor.

The bilinear interpolation method involves determining the color of a pixel of a scaled image based on the weighted average value of the four nearest pixels of the original image. The pixel colors of an image scaled using this method are calculated as follows:

$$x_p = \frac{x_m}{k_x}, \tag{24}$$

$$y_p = \frac{y_m}{y_x}, \tag{25}$$

$$x_1 = Trunc\left(x_p\right), \tag{26}$$

$$x_2 = x_1 + 1, \tag{27}$$

$$y_1 = Trunc\left(y_p\right), \tag{28}$$

$$y_2 = y_1 + 1, \tag{29}$$

$$w_x = x_p - x_1, \tag{30}$$

$$w_y = y_p - y_1, \tag{31}$$

$$K_{y_1} = I(x_1, y_1) \cdot (1 - w_x) + I(x_2, y_1) \cdot w_x, \tag{32}$$

$$K_{y_2} = I(x_1, y_1) \cdot (1 - w_x) + I(x_2, y_2) \cdot w_x, \tag{33}$$

$$I(x_p, y_p) = K_{y_1} \cdot (1 - w_y) + K_{y_2} \cdot w_y, \tag{34}$$

where x_m, y_m are the pixel coordinates of the scaled image (integer); x_p, y_p – coordinates of the pixel of the initial image (real number), which corresponds to the pixel with coordinates x_m, y_m in the scaled image; w_x, w_y – weighting factors; I is the color brightness for a single-channel image.

The result of using the bilinear interpolation method for scaling shown in Figure 8 image is illustrated with Figure 9.

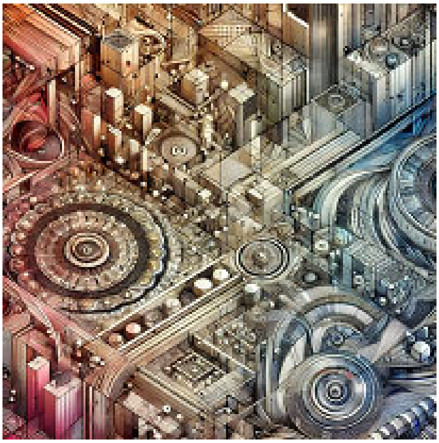

Figure 8. Initial image to be scaled

Analysis of Figure 9 confirms the results of [5; 6], which indicate that at small values of the scale coefficients, the use of the bilinear interpolation method ensures the preservation of smooth transitions and smoothed contours of multi-colored areas. At high values of the scaling factor (more than 3), using this method can lead to blurring of details and loss of clarity, since in this case a significant part of the image pixels needs interpolation, and the initial information is lost. The bilinear interpolation method is recommended for use under the conditions of ensuring a balance between the quality of the scaled image and computational costs. This method is quite often used in information systems of general application, when the requirements for the quality of a scaled image cannot be met when using the nearest neighbor method.

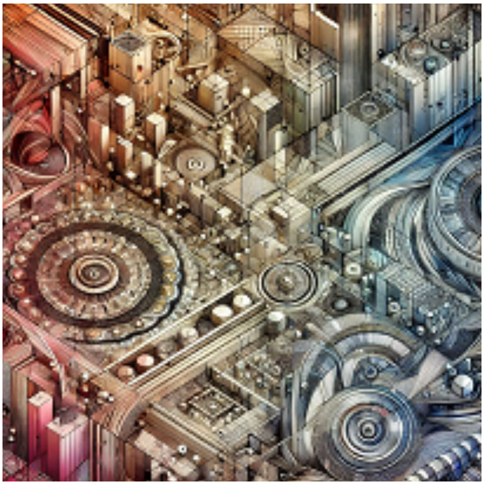

Figure 9. The result of using the bilinear interpolation method with *k* = 3

The bicubic interpolation method involves determining the color of a scaled image pixel based on the weighted average value of the 16 nearest pixels of the original image placed in a 4x4 grid that covers the target fractional coordinates and includes pixels located one coordinate unit to the left, right, up, and down of these coordinates . The pixel colors of an image scaled using this method are calculated using expressions of the form (24-26, 28, 35-39).

$$w(t)=\begin{cases}(a+2)\cdot|t|^3-(a+3)\cdot|t|^2+1, if\,|t|\le 1\\ a\cdot|t|^3-5\cdot a\cdot|t|^2+8\cdot a\cdot|t|-4\cdot a, if\,1<|t|\le 2,\\ 0, if\,|t|>2\end{cases} \quad (35)$$

$$C_i(x,y)=\sum_{k=-1}^{2}C(x_1+k,y_1+i)\cdot w(x_2-k), i=-1;0;1;2, \quad (36)$$

$$I(x_m,y_m)=\sum_{l=-1}^{2}I_l(x_p,y_p)\cdot w(y_2-l), \quad (37)$$

where $w(t)$ is a function for determining the weighting coefficients of interpolation; t is a parameter describing the distance between the integer coordinates of the nearest pixel and the current fractional coordinate for which the interpolation is calculated; a s a predetermined coefficient; $I(x_m,y_m)$ is the color brightness of a pixel in a scaled single-channel image determined using the bicubic interpolation method.

In this case, the value of the parameter t is calculated using expression (38) when calculating $w(t)$ along the X axis, and using expression (2.39) when calculating $w(t)$ along the Y axis. The value of the parameter a is, as a rule, equal to $-0,5$ or -1.

$$t=x_2-k, k=-1;0;1;2, \quad (38)$$

$$t=y_2-l, l=-1;0;1;2. \quad (39)$$

Also note that the parameter k used in expressions (36, 38) represents an integer shift along the X axis relative to the integer coordinate x_1.

Parameter k can take the following values:

-1 – corresponds to the pixel located one position to the left of the pixel with coordinate x_1;

0 – corresponds to the pixel itself with coordinate x_1;

1 – corresponds to the pixel located one position to the right of the pixel with coordinate x_1;

2 – corresponds to the pixel located two positions to the right of the pixel with coordinate x_1.

The l parameter used in expressions (37, 39) is analogous to the k parameter, but refers to an integer shift along the Y axis. The l parameter can take the following values:

-1 – corresponds to a pixel located one position higher than the pixel with coordinate y_1;

0 – corresponds to the pixel itself with the coordinate y_1;

1 – corresponds to a pixel located one position below the pixel with coordinate y_1;

2 – corresponds to the pixel located two positions below the pixel with the y_1 coordinate.

The bicubic interpolation method is recommended to be used if there are no strict limits on the computing resource intensity of the processing process, when it is necessary to achieve high image quality when scaling. As a rule, this method is used in information systems that are used for professional image processing with high detail of small details, clear boundaries, complex textures and color transitions. The result of applying the bicubic interpolation method for scaling shown in Figure 8, the image with the scaling factor $k = 3$ is shown in Figure 10.

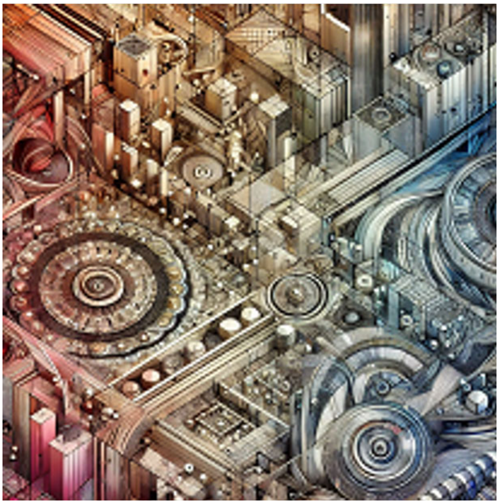

Figure 10. The result of using the bicubic interpolation method with $k = 3$

Comparison of Figs. 9 from Figure 10, which shows the result of using the bilinear interpolation method at the same value of the scale factor, confirms the conclusions formulated in [5] that the use of the bicubic interpolation method makes it possible to obtain a better image. The quality improvement is achieved by providing smoother color transitions and better preservation of details, especially in areas with sharp boundaries and high contrasts.

The Lanzosh interpolation method involves determining the color of a pixel of a scaled image as a weighted sum of the colors of surrounding pixels, taking into account that the weighting coefficients of the colors of the surrounding pixels is determined by the Lanzosh function, which is described by expression (40).

$$sinc(\tau) = \begin{cases} \dfrac{sin(\pi \cdot \tau)}{\pi \cdot \tau} if\, \tau \neq 0 \\ 1 if\, \tau = 0 \end{cases}, \tag{40}$$

where $sinc$ is a normalized sine-cardinal function; τ is a parameter.

Using the Lanzosh function provides smoother transitions between pixels compared to other methods, such as nearest neighbor or bilinear interpolation. The pixel colors of an image scaled using the Lanzosh method are calculated using expressions (24-26, 28, 40-44).

$$I(x_m, y_m) = \sum_{k=-a+1}^{a} \sum_{n=-a+1}^{a} I(x_1 + k, y_1 + n) \cdot Lanc(\delta_x) \cdot Lanc(\delta_y), \tag{41}$$

$$\delta_x = x_m - (x_1 + m), \tag{42}$$

$$\delta_y = y_m - (y_1 + n), \tag{43}$$

$$Lanc(\tau) = \begin{cases} sinc(\tau) \cdot sinc\left(\dfrac{\tau}{a}\right), if -a < \tau < a \\ 0, if\, \tau \leq -a \vee \tau \geq a \end{cases}, \tag{44}$$

where $Lanc$ is the normalized Lanzosh function; τ is a parameter; a – parameter of the Lanzosh window, which determines the size of the interpolation window (as a rule, $a = 2$ or $a = 3$); x_1, y_1 - are calculated using expressions (24-26, 28).

Note that the parameters k and n take integer values in the range from $-a+1$ to a, which defines the interpolation area, which covers

$2a \times 2a$ pixels around the pixel with coordinates (x_1, y_1). For each pixel from this area, its contribution to the color of the pixel with coordinates (x_1, y_1) is calculated using the Lanzosh function, which weights this contribution depending on the distance from this pixel to the pixel with coordinates (x_1, y_1).

The result of applying the Lanzosh interpolation method for scaling shown in Figure 8 image with scaling factor $k = 3$ at $a = 2$ and $a = 3$, shown in Figure 11 ($a = 2$) and Figure 12 ($a = 3$).

Comparison of Figs. 11 and Figure 12 from Figure 10, which shows the result of using the bicubic interpolation method at the same value of the scale factor k, confirms the results of [5; 9], which indicate that:

– The Lanzosh method ensures better preservation of small details and textures, while the use of the bicubic interpolation method can lead to their smoothing.

– The use of the Lanzosh method allows you to more accurately preserve the sharpness on the contrast boundaries, while the use of the bicubic interpolation method leads to smoothing of the edges on the contrast boundaries.

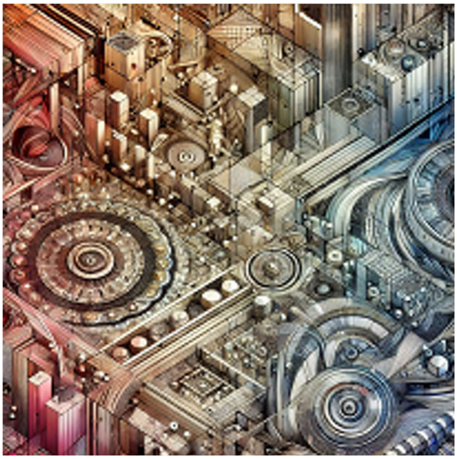

Figure 11. The result of applying the Lanzosh interpolation method at *k* = 3 and *a* = 2

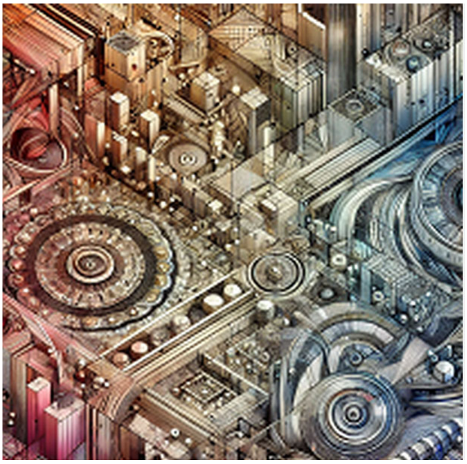

Figure 12. The result of applying the Lanzosh interpolation method at *k* = 3 and *a* = 2

– Using the Lanzosh method can lead to the creation of artifacts around sharp transitions, which does not happen with bicubic interpolation.

Thus, compared to the bicubic interpolation method, the Lanzosh method provides a sharper and more detailed image, while the bicubic interpolation method provides a softer and smoother scaling. In addition, a comparison of Figs. 11 from Figure 12, which displays images scaled by the Lanzosh method with the same scaling factor, but using different values of the Lanzosh window parameter, confirms the results of [3; 9], which indicate the following:

When the value of the Lanzosh window parameter $a = 2$ preservation of details is better ensured, but artifacts become more pronounced.

At the value of the Lanzosh window parameter $a = 3$, the probability of occurrence of artifacts decreases, but the probability of loss of sharpness increases.

The advantages of the Lanzosh interpolation method include the ability to ensure smooth transitions and preserve small details. Disadvantages of the method are high computational complexity and the occurrence of distortions around sharp color transitions in the image. The Lanzosh interpolation method is recommended for scaling images when maximum preservation of quality and detail is required, which determines the prospects for its use in photo processing tools, technical drawings, and image preparation for high-quality printing.

The box filter method involves averaging the pixel color values of the original image in a smoothing (or "boxy") region around the target pixel to calculate the pixel color value of the scaled image. In general, the pixel colors of an image scaled using the box filter method are calculated using expressions of the form:

$$I\left(x_m, y_m\right) = \frac{1}{N^2} \times \sum_{i=-a}^{a} \sum_{j=-a}^{a} I\left(x+i, y+j\right), \tag{45}$$

$$a = \frac{N-1}{2}, \tag{46}$$

where $I\left(x_m, y_m\right)$ is the brightness of the pixel with coordinates $\left(x_m, y_m\right)$ in the scaled image; N s the size of the smoothing area; $I\left(x, y\right)$ is the brightness of the pixel with coordinates $\left(x, y\right)$ in the original image.

As a rule, the smoothing area is a square with a size 3×3, 5×5 or 7×7 pixels. At the same time, the analysis of expressions (45, 46) shows that the minimum size of the smoothing area is 3×3 pixels. For example, in the case of a smoothing area with a size of 5×5 pixels (corresponds $N=5$), expressions (45, 46) are modified as follows:

$$I_{5\times5}\left(x_m, y_m\right) = \frac{1}{9} \times \sum_{i=-2}^{2} \sum_{j=-2}^{2} \left(x+i, y+j\right). \tag{47}$$

Applying a minimum anti-aliasing area of 3x3 pixels when enlarging an image helps prevent image blurring, but may result in jagged edge artifacts. When reducing the image, the use of such an area allows you to preserve sharpness, but it leads to the appearance of noise and distortion of textures. This area of smoothing is recommended to be used if it is necessary to preserve small details and high sharpness of the resulting image. The result of applying the box method for scaling shown in Figure 8 image with $k=3$ when using a smoothing area of 3×3 pixels is shown in Figure 13.

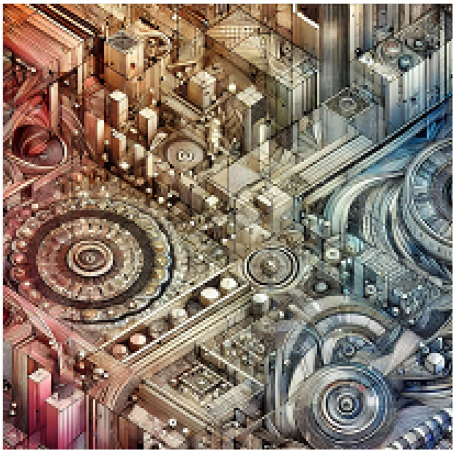

Figure 13. The result of using the box method with $k = 3$ and $N = 2$

Applying the average area of smoothing with the size of 5×5 pixels in the case of enlarging the image ensures sufficient sharpness and smoothness of transitions between pixels. In the case of image reduction, the use of the middle area eliminates noise and allows you to preserve acceptable detail of the scaled image. The average area of antialiasing is usually used when there is a need to maintain a balance between quality and scaling speed. Applying smoothing areas larger than 7×7 when enlarging the image leads to the creation of rather smooth transitions between individual objects, which can negatively affect the blurring of the boundaries of such objects. When reducing an image, applying large areas of smoothing allows you to get rid of noise, but can lead to the loss of fine details and distortion of textures. Large areas of smoothing are recommended to be used when it is necessary to remove noise and smooth sharp transitions when scaling low-quality images. However, using a larger smoothing window results in increased scaling execution time and increased memory consumption. The main advantages of the box method include ease of implementation and high speed of image processing. Disadvantages of the box method include loss of sharpness, blurring of details, distortion of object boundaries, and distortion of text. The box method is often used when the scaling speed is more important than the quality of the image obtained after scaling.

The triangular filter method assumes that a so-called triangular filter is used to calculate the color of a pixel of a scaled image, which can be represented as a function given by expression (48), which ensures a uniform distribution of interpolation weighting coefficients between the central and neighboring pixels. At the same time, the use of the modulus of the number x in the expression (48) is explained by the fact that, in the general case, the distance between the current and target pixels can be determined taking into account the given origin of coordinates and the positive direction of the X axis.

$$w(x) = \max\left(0;1-|x|\right), \tag{48}$$

where $w(x)$ is the weighting factor; x is the distance between the current and target pixels.

The pixel colors of the scaled image using the triangular filter method are calculated using expressions:

$$d_{x_i} = x_m - x_i, \tag{49}$$

$$d_{y_i} = y_m - y_i, \tag{50}$$

$$w_i = \max\left(0;1-\sqrt{d_{x_1}^2 + d_{x_1}^2}\right), \tag{51}$$

$$\overline{w_i} = w_i \times \left(\sum_{j=1}^{J} w_j\right)^{-1}, \tag{52}$$

$$I\left(x_m, y_m\right) = \sum_{i=1}^{I}\left(\overline{w_i} \cdot I\left(x_i, y_i\right)\right), \tag{53}$$

where x_m, y_m are the coordinates of the target pixel in the scaled image; x_i, y_i – coordinates of the i-th pixel in the initial image, which belongs to the set of pixels closest to the target pixel; d_{x_i}, d_{y_i} is the distance between the target pixel and the pixel with coordinates (x_i, y_i) horizontally and vertically; w_i is the weight factor of a pixel with coordinates x_i, y_i; $\overline{w_i}$ is the normalized weight coefficient of a pixel with coordinates x_i, y_i; $I\left(x_m, y_m\right)$ is the brightness of the target pixel of the scaled image; $I\left(x_i, y_i\right)$ is the brightness of the pixel with coordinates (x_i, y_i) in the initial image.

The result of applying the triangular filter method for scaling shown in Figure 8 image at $k = 3$ is shown in Figure 14.

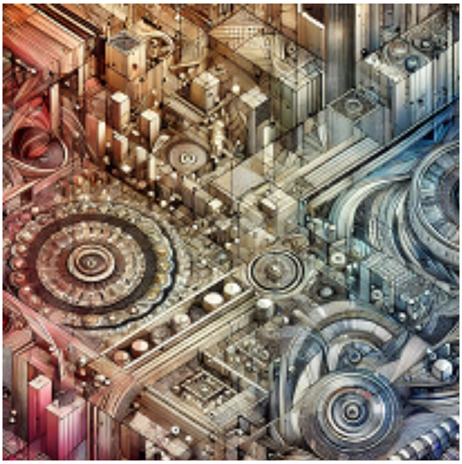

Figure 14. The result of using the triangular filter method with *k* = 3

The main advantages of the triangular filter method include ease of implementation compared to more complex interpolation methods (for example, bicubic), ensuring the smoothness and uniformity of smoothing of the scaled image and less blurring of the scaled image in relation to box filtering. Disadvantages of the triangular filter method include the distortion of the borders of the scaled image elements and the appearance of artifacts at sufficiently large values of the scaling factor.

4. Conclusions

As a result of the conducted research, mathematical support was formed, which is used to implement geometric transformations of images during the augmentation of training data of neural network tools. The key components of the mathematical support of affine transformations related to the determination of the dimensions of the area of the screen on which the modified image should be displayed and the determination of the color of individual points of the modified image are determined. A mathematical apparatus has been defined that allows you to calculate the dimensions of the display area of the modified image, provided that a rectangular display area is provided.

It is shown that in the task of augmentation of training data of neural network tools, it is advisable to perform image scaling using proven non-adaptive interpolation methods: nearest neighbor, bilinear interpolation, bicubic interpolation, Lanzosh interpolation, box filter, triangular filter.

A mathematical apparatus for calculating the pixel color of a scaled image using each of the above interpolation methods is defined.

The conditions that determine the expediency of using the specified methods in solving practical problems are outlined. It is shown that the method of the nearest neighbor is advisable to use in the case of the need to ensure the simplicity of implementation and high productivity of the augmentation procedure, when the noise of small details and the distortion of the shape of objects containing thin lines can be neglected. The bilinear interpolation method is recommended for use under the conditions of ensuring a balance between the quality of the scaled image and computational costs. The bicubic interpolation method is recommended to be used if there are no strict limits on the computing resource intensity of the processing process, when it is necessary to achieve high image quality when scaling. The Lanzosh interpolation method is recommended for scaling images in cases where maximum preservation of quality and detail is required. The box method and the triangular filter method are appropriate to use when the speed of scaling is more important than the quality of the image obtained after scaling.

Prospects for further research are related to the development of a methodology for adapting the means of integrated modification of geometry and visual characteristics of images to the conditions of augmentation of training data of neural network systems for video stream analysis.

References:

1. Antoniou, A., Storkey, A., & Edwards, H. (2017). Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*.

2. Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M., & Kalinin, A. A. (2020). Albumentations: Fast and flexible image augmentations. *Information, 11*(2), 125.

3. Buslaev, A., Seferbekov, S., & Iglovikov, V. (2018). OpenCV tools for efficient geometric image augmentations. *Proceedings of the Computer Vision Tools Symposium*.

4. Chaurasia, A., & Culurciello, E. (2017). LinkNet: Exploiting encoder representations for efficient semantic segmentation. *2017 IEEE Visual Communications and Image Processing (VCIP)*.

5. Chernyshev, D.O., Tereikovska, L.O., & Tereikovskyi, I.A. (2024). *Mathematical support for information technology of digital image processing.* Kyiv: KNUCA.

6. Data augmentation for object detection: A practical survey. (2021). *arXiv preprint arXiv:2103.01992*.

7. Howard, J., & Gugger, S. (2020). Fastai: A layered API for deep learning. *Information and Software Technology, 106*. https://doi.org/10.1016/j.infsof.2019.106139

8. Karras, T., Laine, S., & Aila, T. (2019). A style-based generator architecture for generative adversarial networks. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4401–4410.

9. Liu, Y., Zhang, Z., Lian, Y., & Liang, W. (2020). Geometric transformations in image augmentation: A survey. *arXiv preprint arXiv:2004.14636*.

10. MAug: Multimodal geometric augmentation in latent spaces of image deformations. (2023). *arXiv preprint arXiv:2312.13440*.

11. Perez, L., & Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*.

12. Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data, 6*(1), 60.

13. Sato, K., & Nishimura, H. (2016). Effective data augmentation techniques for convolutional neural networks. *Neural Information Processing Systems Workshop*.

14. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2818–2826.

15. Taylor, L., & Nitschke, G. (2018). Improving deep learning using generic data augmentation. *arXiv preprint arXiv:1708.06020*.

16. Tereikovska L., Tereikovskyi I. (2020) Application of a convolutional neural network for the analysis of biometric parameters. *Academic notes of TNU named after V.I. Vernadskyi. Series: technical sciences*, vol. 31 (70), No 5, pp. 124–128.

17. Tereikovskyi I., Bushuev D., Tereikovska L. (2022). *Artificial neural networks: basic principles*. Kyiv: KPI named after Igor Sikorsky, 123 p.

18. Volpi, R., & Murino, V. (2019). Addressing model vulnerability to distributional shifts over image transformation sets. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 7980–7989.