Deep Learning Neural Networks with Controlled Switching of Neural Planes*

Alexander Yu. Dorogov [0000-0002-7596-6761]

Saint Petersburg Electrotechnical University "LETI"
Saint Petersburg, Russia

vaksa2006@yandex.ru

Abstract. The algorithm of network topology construction and training of two-dimensional fast neural networks with additional switched planes is considered. It is noted that the structure of fast neural networks has a fractal nature. The constructed topology is ideologically close to the topology of convolutional neural networks of deep learning, but it has regular topology with the number of layers established by the factor representation of the dimensions of the image and the output plane of the classes. The learning algorithm has an analytical representation, and it is stable and converges in a finite number of steps. Additional planes extend the information capacity of the tunable transformation to the maximum possible. Control of the planes in the training and processing mode is realized by numerical coordinate codes of the output plane. The architecture of a regular neural network with additional planes is presented. Variants for image ordering in the output plane are considered. Examples are given.

Keywords: fast tunable transformation; neural network; learning; convolutional neural network; the bitmap; the planes of the neural layers First Section

1 Introduction

Deep learning technology involves a process of configuration complexity of informative features in the sequence of neural layers. Starting with the neocognitron by K. Fukushima [1] it has been proposed some variants of realization of this idea. One of the successful solutions is the architecture of convolutional neural networks [2], which has shown high efficiency in solving various problems. A distinctive feature of this architecture is the presence in convolutional layers of several data processing channels (called maps or planes). In each plane, the output image of the previous layer is being convoluted with a fixed kernel of small dimensions. Convolutional layers alternate with pooling layers that multiply reduce the dimension of the feature space. The pooling layers are optional and there are variants of completely eliminating them from the network architecture [3]. The second distinctive feature of the architecture is the use of

semi-linear activation functions that act as switching keys controlled by the values of hidden layers variables [4, 5, 6, 7].

The disadvantage of the convolution network architecture is the lack of theoretically justified methods for choosing the network structure and convolution kernel parameters. There are several well-functioning network configurations for specific tasks, but it is not clear how to build a network for a new task. Until now, the choice of the structure of the convolution network is an art object. The second significant drawback is related to the training time of convolutional networks. On a typical processor, the time can vary from several hours to several days, therefore high-performance GPUs are often used to train networks. In [8] the authors proposed the idea of using fast transformation algorithms to construct the structure and topology of multilayer perceptron neural networks. We show that this approach with some modification can also be used to construct the structure and topology of convolutional neural networks.

At present, fast algorithms for linear Fourier, Walsh, Haar, and similar transformations are widely known. With the use of fast algorithms, the gain on computational operations increases exponentially with the increase in the dimension of the transformation. Since the end of the 20-th century, there has arisen a direction of fast tunable transformations [9], which are essentially neural networks with limited connections and linear activation functions. There have been developed methods for training such neural networks that converge in a finite number of steps. The number of layers in fast transformations and their configuration are determined by the dimension of the processed images. To construct fast algorithms, the dimension of the transformation must be a composite number, and the more multipliers in the composition of the dimension, the higher the computational efficiency of the fast algorithm. Despite the wide variety of fast algorithms, the configurations of their structures satisfy the system invariant of self-similarity [10]. Fractals are known to have the same property. Therefore, fast algorithms can be interpreted as quasi-fractals. The property of structural fractality allows solving two tasks simultaneously: to realize fast data processing and fast transformation training.

In this paper, we will show that a small modification of the system invariant of fast algorithms leads to convolutional neural network structures. At the same time, it is possible to preserve the algorithm of fast learning and increase the information capacity of the network recognition up to the maximum possible, determined by the number of neurons of the output layer of the network. The proposed architecture cannot be called convolution networks, because in the planes of neural layers more general transformations than convolution are used, and there are no pooling layers in the transformation, but the principle of pooling is used in the training of the network. All neurons have linear activation functions, however non-linear processing exists, but it is implemented not at the expense of activation functions, but by switching the planes of neural networks. To some extent, this is similar to the switching semi-linear activation functions of convolutional neural networks. Control of switching planes is carried out by the coordinates of neurons in the output plane of the network. Call this class of networks neural networks with controlled switching of planes (CSPNN).

2 The topology of Two-Dimensional Fast Tunable Transformations

Let us designate through $F\left(U_y,U_x\right)$ an image matrix by dimensionality $N_y \times N_x$. In case of impact on the image through linear transformation, $H\left(U_y,U_x;V_y,V_x\right)$ the array from $M_y \times M_x$ coefficients turns out. Two-dimensional transformation is executed by the rule:

$$S\left(V_y,V_x\right)=\sum_{Uy=0}^{Ny-1}\sum_{Ux=0}^{Nx-1}F\left(U_y,U_x\right)H\left(U_y,U_x;V_y,V_x\right). \tag{1}$$

A necessary condition of the existence of a fast algorithm is the possibility of multiplicative decomposition of values of input and output dimensionalities of the transformation to an equal number of multiplicands:

$$\begin{aligned} N_y &= p_0^y p_1^y \dots p_{n-1}^y, \qquad M_y = g_0^y g_1^y \dots g_{n-1}^y, \\ N_x &= p_0^x p_1^x \dots p_{n-1}^x, \qquad M_x = g_0^x g_1^x \dots g_{n-1}^x. \end{aligned} \tag{2}$$

Here indexes x, y mean the belonging to coordinate axes of the source image, and value n defines the number of layers in the graph of a fast algorithm. Using multiplicands of decomposition, coordinates of points of the image representation in a positional system notation with the mixed radices:

$$\begin{aligned} U_y &= \left\langle u_{n-1}^y u_{n-2}^y \cdots u_1^y u_0^y \right\rangle, \\ U_x &= \left\langle u_{n-1}^x u_{n-2}^x \cdots u_1^x u_0^x \right\rangle, \end{aligned} \tag{3}$$

where the weight of m's position digit is defined by an expression $p_{m-1}^* p_{m-2}^* \dots p_1^* p_0^*$ and u_m^* is the digit variable accepting values $\left[0, p_m^* - 1\right]$ (the asterisk replaces indexes x, y here). It is similarly possible to represent coordinates of spectral coefficients for the plane $\left[V_y,V_x\right]$:

$$\begin{aligned} V_y &= \left\langle v_{n-1}^y v_{n-2}^y \cdots v_1^y v_0^y \right\rangle, \\ V_x &= \left\langle v_{n-1}^x v_{n-2}^x \cdots v_1^x v_0^x \right\rangle, \end{aligned} \tag{4}$$

where the weight of m's position digit is defined by an expression $g_{m-1}^* g_{m-2}^* \dots g_1^* g_0^*$ and v_m^* is the digit variable accepting values $\left[0, g_m^* - 1\right]$.

The algorithm of fast transformation is usually presented in the form of a graph with a different topology. It is convenient to use the digit-by-digit form for the analytical description of a graph of the topology of a fast algorithm. For example, topology a

graph can be described by "Cooley–Tukey topology with decimation on to time" in the form of a linguistic sentence [10] (topological model):

$$\left[\begin{array}{l}\left\langle u^*_{n-1}u^*_{n-2}\ldots u^*_1u^*_0\right\rangle\left\langle u^*_{n-1}u^*_{n-2}\ldots u^*_1v^*_0\right\rangle\ldots \\ \ldots\left\langle u^*_{n-1}u^*_{n-2}\ldots u^*_{m+1}u^*_mv^*_{m-1}v^*_{m-2}\ldots v^*_0\right\rangle\ldots\left\langle v^*_{n-1}v^*_{n-2}\ldots v^*_1v^*_0\right\rangle\end{array}\right],$$

where words are digit-by-digit representations of coordinate numbers, and letters – names of digit variables. The number of words in the sentence is equal $n+1$. The first and last words in the sentence correspond to coordinates of points of the terminal planes provided by expressions. The intermediate words define input U^m_y, U^m_x and output coordinate V^m_y, V^m_x in the planes of inner layers of the fast algorithm. For an algorithm with substitution of values, the condition is follow-up satisfied:

$$U^{m+1}_y = V^m_y, \qquad U^{m+1}_x = V^m_x \tag{5}$$

Graph of topology contains basic operations in a layer $W^m_{i^m_x, i^m_y}\left(u^y_m u^x_m; v^y_m v^x_m\right)$, representing four-dimensional matrixes of dimensionality $\left[p^y_m, p^x_m; g^y_m, g^x_m\right]$. Where digit-by-digit expressions of indexes of kernels of a layer m for the selected topology have viewed:

$$\begin{aligned} i^m_x &= \left\langle u^x_{n-1}u^x_{n-2}\ldots u^x_{m+1}v^x_{m-1}v^x_{m-2}\ldots v^x_0\right\rangle, \\ i^m_y &= \left\langle u^y_{n-1}u^y_{n-2}\ldots u^y_{m+1}v^y_{m-1}v^y_{m-2}\ldots v^y_0\right\rangle. \end{aligned} \tag{6}$$

Expression is an analytical representation of a system invariant of fast algorithms [10]. In general topologies, for the directions, x and y may be different. Connections between basic operations are defined by the structural model of fast transformation, where to each node there corresponds to basic operation (differently called hereinafter as neural kernels). For the selected topology of graphs of the structural model is described by the following linguistic sentence:

$$\left[\begin{array}{l}\left\langle u^*_{n-1}u^*_{n-2}\ldots u^*_1\right\rangle\left\langle u^*_{n-1}u^*_{n-2}\ldots u^*_2v^*_0\right\rangle\ldots \\ \ldots\left\langle u^*_{n-1}u^*_{n-2}\ldots u^*_{m+1}v^*_{m-1}v^*_{m-2}\ldots v^*_0\right\rangle\ldots\left\langle v^*_{n-1}v^*_{n-2}\ldots v^*_1v^*_0\right\rangle\end{array}\right].$$

Each word in this sentence defines the number of basic operations i^m_* in the layer m. The number of words is equal n in the sentence.

In Fig. 1 structural model of fast two-dimensional transformation for dimensionality 8×8 is shown. Input image enters on the low layer and spectral coefficients turn out in the high layer. Nodes of the model there correspond to basic operations (neural kernels) with dimensionality [2,2;2,2]. Kernels in layer m execute two-dimensional transformation over the spatial unit with size $p^y_m \times p^x_m$:

$$S^m\left(V_y^m,V_x^m\right)=\sum_{u_m^y}\sum_{u_m^x}F^m\left(U_y^m,U_x^m\right)W^m_{i_x^m,i_y^m}\left(u_m^y u_m^x;v_m^y v_m^x\right). \tag{7}$$

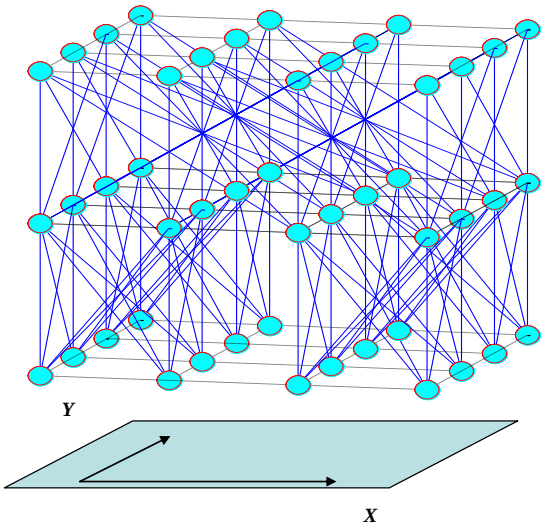

Fig. 1. Structure model two-dimensional transformation

3 Multiplicative Decomposition of Fast Transformation

Setting specific values for all digit variables u_m^*, v_m^* (where m runs through the values $0,1,\ldots n-1$) defines some path in a topological graph between pair of nodes of an initial and finite layer. From the uniqueness of digit-by-digit representation of coordinate numbers, it follows that such path is single for each pair combination of spatial points of the input and output plane. This circumstance allows obtaining the convenient analytical expression connecting array elements of fast transformation with elements of kernels. From expression it follows:

$$H\left(U_y,U_x;V_y,V_x\right)=\frac{\partial S\left(V_y,V_x\right)}{\partial F\left(U_y,U_x\right)}. \tag{8}$$

Differentiating by the rule of differentiation of the composite function we will obtain:

$$H\left(U_y,U_x;V_y,V_x\right)=\frac{\partial S^{n-1}}{\partial F^{n-1}}\frac{\partial F^{n-1}}{\partial S^{n-2}}\frac{\partial S^{n-2}}{\partial F^{n-2}}\cdots\frac{\partial F^1}{\partial S^0}\frac{\partial S^0}{\partial F^0}.$$

From condition it follows that for all m the following equals take place $\frac{\partial F^m}{\partial S^{m-1}} = 1$, and from, – that $\frac{\partial S^m}{\partial F^m} = W^m_{i_x^m, i_y^m}\left(u_m^y u_m^x; v_m^y v_m^x\right)$. Thus, we will obtain that each element of a four-dimensional transformation matrix H expresses through elements of kernels in the form of the following product:

$$H\left(U_y, U_x; V_y, V_x\right) = W^{n-1}_{i_x^{n-1}, i_y^{n-1}}\left(u_{n-1}^y u_{n-1}^x; v_{n-1}^y v_{n-1}^x\right)\cdots$$
$$\ldots W^{n-2}_{i_x^{n-2}, i_y^{n-2}}\left(u_{n-2}^y u_{n-2}^x; v_{n-2}^y v_{n-2}^x\right)\ldots W^{0}_{i_x^{0}, i_y^{0}}\left(u_0^y u_0^x; v_0^y v_0^x\right), \quad (9)$$

where digit-by-digit expressions of kernel indexes for a layer m for the selected topology are defined by expression Multiplicative Decomposition of Two-Dimensional Images.

The algorithm of multiplicative decomposition is based on the ideas of fractal filtering [10] (in the notation of convolution neural networks this operation corresponds to pooling). For a two-dimensional case, fractal filtering represents the multiple scale image processing sequentially squeezing its sizes up to a single point. The diagram of fractal filtering can be presented in the form of the pyramid shown in Fig 2.

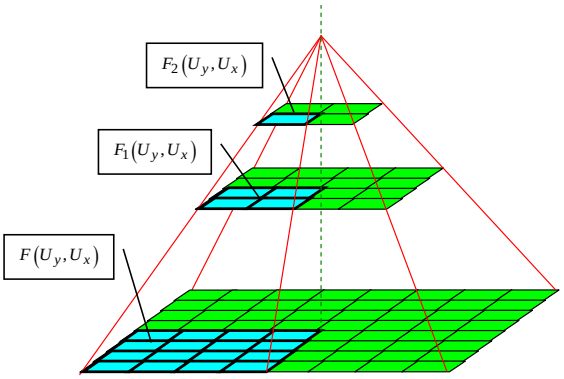

Fig. 2. The scheme of the fractal image filtering

The base of the pyramid is the source image, $F\left(U_y, U_x\right)$ for which arguments U_y and U_x are presented in a radix notation (see expression. In this positional representation, we will fix all digits except two the lowest u_0^y and u_0^x. If to vary these digits on all possible values, then we will obtain a two-dimensional selection with the size $p_0^y \times p_0^x$. The fractal filter is understood as any functional Φ, acting on this selection. Formally, it can be written in the form of the following expression:

$$F_1\left(\left\langle u_{n-1}^y u_{n-2}^y \cdots u_1^y \right\rangle, \left\langle u_{n-1}^x u_{n-2}^x \cdots u_1^x \right\rangle\right) =$$
$$= \underset{\left(u_0^y, u_0^x\right)}{\Phi}\left[F\left(\left\langle u_{n-1}^y u_{n-2}^y \cdots u_1^y u_0^y \right\rangle, \left\langle u_{n-1}^x u_{n-2}^x \cdots u_1^x u_0^x \right\rangle\right)\right].$$

The image F_1 will be multiply reduced by the sizes with the source image. For example, the rule of average calculation of selection or its median line can be such functional. The source image can be now formally presented in the form of a product:

$$F\left(\left\langle u_{n-1}^y u_{n-2}^y \cdots u_1^y u_0^y \right\rangle, \left\langle u_{n-1}^x u_{n-2}^x \cdots u_1^x u_0^x \right\rangle\right) =$$
$$= F_1\left(\left\langle u_{n-1}^y u_{n-2}^y \cdots u_1^y \right\rangle, \left\langle u_{n-1}^x u_{n-2}^x \cdots u_1^x \right\rangle\right) f_{j_y^0 j_x^0}\left(u_0^y, u_0^x\right),$$

where $f_{j_0^y j_0^x}\left(u_0^y, u_0^x\right)$ - is a set of the two-dimensional function factors depending on digit variables u_0^y and u_0^x, and indexes j_y^0, j_x^0 selecting a two-dimensional function from this set. The value of these indexes is set to equal values of arguments of the image F_1, so that $j_y^0 = \left\langle u_{n-1}^y u_{n-2}^y \cdots u_1^y \right\rangle$ and $j_x^0 = \left\langle u_{n-1}^x u_{n-2}^x \cdots u_1^x \right\rangle$. For obtaining the factor functions, it is enough to execute scalar division of the image F to the image F_1 in case of variation of all digit variables. In turn, the image F_1 can also be represented as the product of the image F_2 on factors from the set $f_{j_y^1 j_x^1}\left(u_1^y, u_1^x\right)$. Repeating multiply the operation of fractal filtering and decomposition, we will reach the peak of the pyramid of images and we will obtain multiplicative decomposition:

$$F\left(\left\langle u_{n-1}^y u_{n-2}^y \cdots u_1^y u_0^y \right\rangle, \left\langle u_{n-1}^x u_{n-2}^x \cdots u_1^x u_0^x \right\rangle\right) =$$
$$f_{j_y^{n-1} j_x^{n-1}}\left(u_{n-1}^y, u_{n-1}^x\right) f_{j_y^{n-2} j_x^{n-2}}\left(u_{n-2}^y, u_{n-2}^x\right) \ldots f_{j_y^1 j_x^1}\left(u_1^y, u_1^x\right) f_{j_y^0 j_x^0}\left(u_0^y, u_0^x\right), \tag{10}$$

where indexes of multiplicands are defined by expressions:

$$j_x^m = \left\langle u_{n-1}^x u_{n-2}^x \cdots u_{m+1}^x \right\rangle, \qquad j_y^m = \left\langle u_{n-1}^y u_{n-2}^y \cdots u_{m+1}^y \right\rangle. \tag{11}$$

4 Attuning of Adapted Transformations

We will call transformation adapted to the image if one of the transformation base functions with coordinates $\left(V_y, V_x\right)$ coincides with this image. The value of a scalar product of the image with this function will be maximal among other coefficients of the spectral area of the transformation. The purpose of transformation attuning consists in it. Value of coordinates $\left(V_y, V_x\right)$ we will call as adaptation point the function in the spectral plane.

Attuning can be realized also in several images. If to compare the obtained multiplicative decomposition of the image with the decomposition of fast transformation, it is

easy to note that they are similar, and set kernel indexes in each layer cover a set of indexes of function multiplicands. From this constructive result, follows that fast transformation will be attuned to the image when transformation kernels are attuned to function multiplicands. Attuning of transformation kernels is defined by the rule:

$$W^m_{i^m_x,i^m_y}\left(u^y_m u^x_m; v^y_m v^x_m\right) = f_{j^m_x j^m_y}\left(u^y_m, u^x_m\right). \tag{12}$$

Comparing expression for i^m_x, i^m_y and for j^m_x, j^m_y, it is possible to obtain the result conclusion that quantity of components in the multiplicative expansion of the image and quantity of kernels of transformation coincide for a layer $m = 0$ (thus it takes place following equals $i^0_x = j^0_x$ and $i^0_y = j^0_y$), and less number of kernels for all remaining layers. Therefore, in the case of attuning a part of degrees of freedom of a transformation is not used. Digit variables v^y_0, v^x_0 are freely variational variables; therefore, the kernel may be attuned to $D = g^y_0 g^x_0$ images. The remaining layers have a bigger number of degrees of freedom and cannot worsen this value. Thus, it is possible to conclude that a fast transformation cannot adapt more than to D different images. On this, the opportunities of this algorithm of attuning are exhausted. Value D let us call as the level of transformation attuning.

5 Regular Neural Networks with Additional Planes

Remaining within the considered topology, not used in case of attuning degrees of freedom it is possible to determine in several ways (in more detail see. [10]). In this case, the level of attuning does not change, but at the same time remaining transformation functions change.

Let us consider the alternative decision, which consists of an extension of the topology through additional planes to use the remained degrees of freedom for an increase in the level of attuning. In this case, the number of computing operations in the new topology increases, but the structural regularity of the network remains.

In the beginning, let us specify a choice rule of the adapted kernels for the former topology. By adaptation let's express point coordinates in a radix notation, having designated digit variables through y and x:

$$V_y = \left\langle y_{n-1}, y_{n-2} \ldots y_0 \right\rangle, \quad V_x = \left\langle x_{n-1}, x_{n-2} \ldots x_0 \right\rangle. \tag{13}$$

The fixed values of digit variables y_m, x_m correspond to variables v^y_m, v^x_m, therefore (as it follows from) in case of a choice of this point of adaptation, according to the rule need adapt only kernels with numbers:

$$\begin{aligned} i^m_x &= \left\langle u^x_{n-1} u^x_{n-2} \ldots u^x_{m+1} x_{m-1} x_{m-2} \ldots x_0 \right\rangle, \\ i^m_y &= \left\langle u^y_{n-1} u^y_{n-2} \ldots u^y_{m+1} y_{m-1} y_{m-2} \ldots y_0 \right\rangle. \end{aligned} \tag{14}$$

In particular for $m = 0$ we have:

$$i_x^0 = \left\langle u_{n-1}^x u_{n-2}^x \cdots u_1^x \right\rangle, \qquad i_y^0 = \left\langle u_{n-1}^y u_{n-2}^y \cdots u_1^y \right\rangle.$$

Thus, irrespective of a choice of points of adaptation all kernels of a zero layer always shall be adapted. At the same time, the level of transformation attuning is restricted to value D.

To increase of transformation attuning level we will enter the additional plane structure copying the main plane in each layer. We will determine the order numbers of the additional planes within a layer by the rule:

$$\pi_m = \left\langle x_{n-1} x_{n-2} \ldots x_{m+1}, y_{n-1} y_{n-2} \ldots y_{m+1} \right\rangle.$$

The maximum quantity of the additional planes will be in the zero layers, and in process of increase in a number of a layer the quantity of the additional planes will decrease, and in the last layer we will obtain $\pi_{n-1} = \langle\ \rangle$, i.e. the additional planes will not be absolute. Thus, in the new topology, the plane of the last layer will remain the same, and in younger layers, the additional planes will appear.

The architecture of the neural network with the additional planes is shown in Fig. 3. The input image is fed at the same time to all planes of the input layer. Layers are divided by switchboards which are controlled by position digits of coordinate numbers of output class.

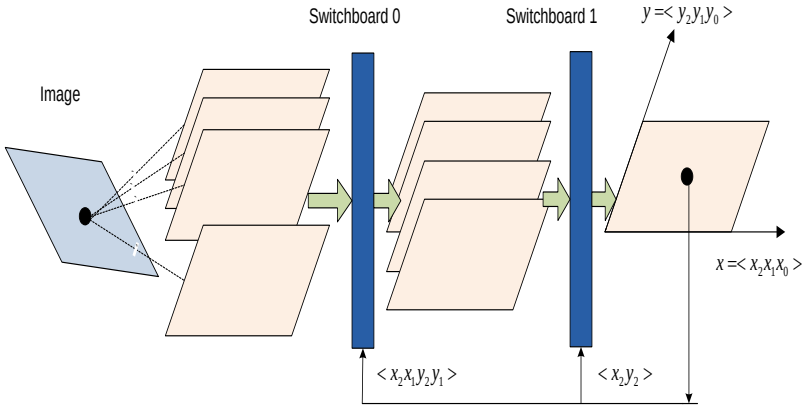

Fig. 3. Regular neural network architecture with additional planes

Since the rule of a generation of the new planes does not contradict to condition, so for attuning of kernels of the transformation it is possible to use the former rule, with an additional argument for selecting the planes:

$$W^m_{i_x^m, i_y^m} \left\langle \pi_m \right\rangle \left(u_m^y u_m^x ; v_m^y v_m^x \right) = f^k_{j_x^m j_y^m} \left(u_m^y, u_m^x \right).$$

The index k in the right part enumerates an adaptation point. For $m=0$ we have $i_x^0 = j_x^0$ and $i_y^0 = j_y^0$, here variational variables are the number of plane $\pi_0 = \langle x_{n-1}x_{n-2}\ldots x_1, y_{n-1}y_{n-2}\ldots y_1 \rangle$ and digit variables v_0^y, v_0^x. Together, they cover the full coordinate range of the output plane. Possible index values k correspond to this range. The remaining layers do not impair the level of transformation attuning. Thus, the transformation with additional planes can be adapted to $D_\pi = M_y \times M_x$ images, i.e. each point of the output plane will exactly correspond to one image of the learning set.

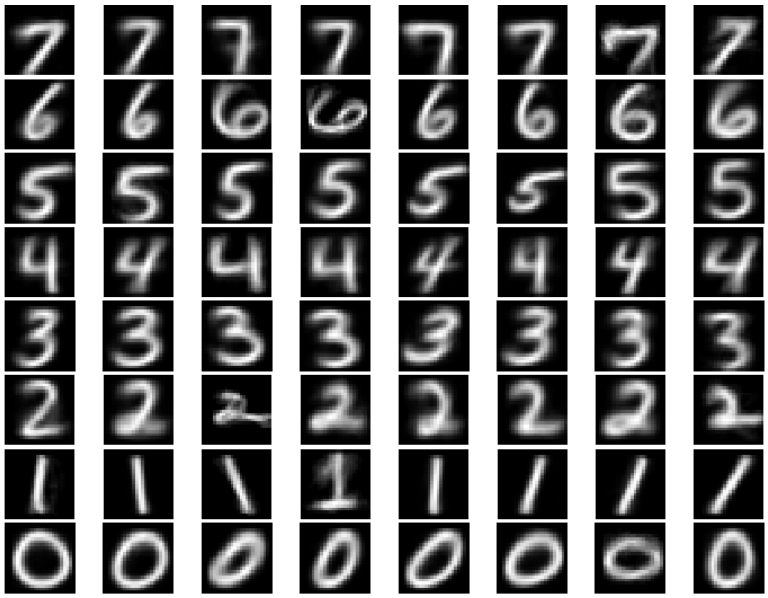

Fig. 4. Two-dimensional functions of the adapted transformation

Fig. 4 shows the two-dimensional transformation function adapted to the average images of handwritten numbers [2]. For this transformation:

$$N_y = N_x = p_0 p_1 p_2 = 4 \cdot 3 \cdot 2 = 24, \qquad M_y = M_x = g_0 g_0 g_2 = 2 \cdot 2 \cdot 2 = 8 .$$

If the image corresponds to one of the adaptation points, the value of this spectral plane coefficient will be maximal. The transformation result for above the image of the digit "0" is shown in Fig. 5. It is seen that the coefficients corresponding to the subclasses of the number “0” have maximum values.

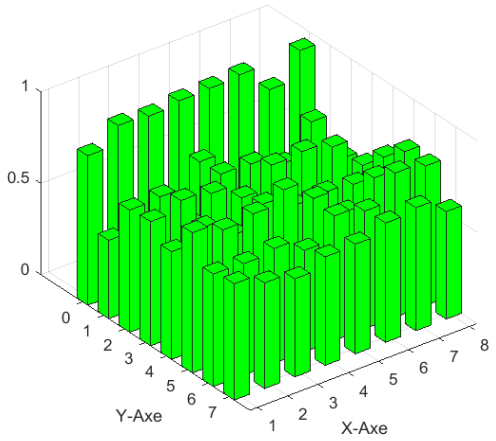

Fig. 5. The result of the two-dimensional transformation

6 Ordering of the Adaptation Points

For each k 's adaptation point in the range of $k=[0, D_{\pi}-1]$, you must set your values for bit variables y_m, x_m. A one-to-one correspondence $k \to \langle V_y, V_x \rangle$ defines rules for ordering adaptation points in the output plane. Let's look at some typical variants. Recall that the bitwise representation of coordinates is defined by expressions (13). The task of ordering is to establish a correspondence between the ordinal number k and the bit variables $\langle y_i \rangle, \langle x_i \rangle$. We assume that moving in the spectral plane along a column is determined by changing the coordinate V_y, and along a row by changing the coordinate V_x.

6.1 Ordering Along of Columns

The ordering algorithm can be specified by the following sequence number representations:

$$k = \langle x_{n-1} x_{n-2} \ldots x_0 y_{n-1} y_{n-2} \ldots y_0 \rangle.$$

In this expression, the digit y_0 is the lowest, so when you increase the number k, the digits $\langle y_i \rangle$ will change first, and as a result, the adaptation points will be placed along with the columns. Fig. 4 shows the variant of the ordering where classes are placed along columns and subclasses are placed along rows.

6.2 Ordering Along of Rows

The ordering algorithm can be specified by the following sequence number representations:

$$k = \langle y_{n-1} y_{n-2} \ldots y_0 x_{n-1} x_{n-2} \ldots x_0 \rangle .$$

In this expression, the digit x_0 is the lowest, so when you increase the number k, the digits $\langle x_i \rangle$ will change first, and as a result, the adaptation points will be placed along the rows. Fig. 6 shows a variant of implementing a transformation with the ordering of the adaption points along rows.

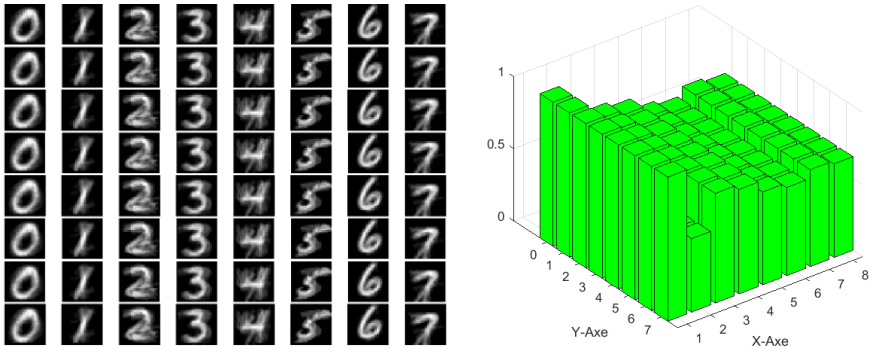

Fig. 6. The transformation with the ordering of basis functions along the rows

6.3 Ordering Along of Circular Segments

The ordering algorithm can be specified by the following sequence number representations:

$$k = \langle x_{n-1} y_{n-1} x_{n-2} y_{n-2} \ldots x_1 y_1 x_0 y_0 \rangle .$$

In this case, the spectral plane will be filled with increasing values k when moving clockwise along the "circular" segments. Fig. 7 shows a variant of implementing the transformation with the ordering of basic functions along with circular segments.

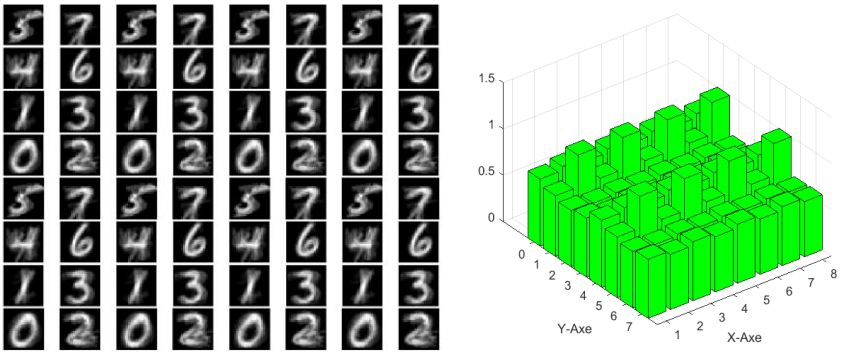

Fig. 7. Transformation with the ordering of basic functions by circular segments

7 Summary

Regular tunable transformations have a unique possibility of analytical representation of the topology of the implementing network, which allows developing learning algorithms that converge in a finite number of steps. It is shown that the implementing topology is easily expanded by additional planes, and the number of recognized images increases dramatically and covers all elements of the output plane. Moreover, the topology extension does not violate the principle of building a training algorithm. The constructed topology is ideologically close to the topology of convolutional deep learning networks [2] but is regular. The presented solution provides a constructive answer to the fundamental questions of deep learning neural networks: how to choose a topology and how to reduce the learning time of the network.

References

1. Fukushima K., Miyake S., Takayuki I. Neocognitron: A neural network model for a mechanism of visual pattern recognition. IEEE Transaction on Systems, Man and Cybernetics SMC–13(5):826–34.-1983.
2. Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel: Backpropagation Applied to Handwritten Zip Code Recognition, Neural Computation, 1(4):541-551, Winter 1989.
3. Springenberg, Jost Tobias; Dosovitskiy, Alexey; Brox, Thomas & Riedmiller, Martin (2014-12-21), "Striving for Simplicity: The All Convolutional Net", arXiv:1412.6806 https://arxiv.org/pdf/1412.6806.
4. Romanuke, Vadim. Appropriate number and allocation of ReLUs in convolutional neural networks (англ.) // Research Bulletin of NTUU “Kyiv Polytechnic Institute”: journal. — 2017. — Vol. 1. — P. 69—78.
5. Xavier Glorot, Antoine Bordes, BengioY. Deep Sparse Rectifier Neural Networks January 2010Journal of Machine Learning Research 15 Conference: Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS).
6. V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In Proc. 27th. International Conference on Machine Learning, 2010.

7. Maas, Andrew L.; Hannun, Awni Y.; Ng, Andrew Y. (June 2013). "Rectifier nonlinearities improve neural network acoustic models" (PDF). Proc. ICML. 30 (1). Retrieved 2 January 2017.
8. Dorogov A. Yu., Alekseev A. A. // Mathematical models of fast neural networks. In: collection of scientific. Tr. SPbGETU “Information management and processing systems”. Issue.490, 1996, p. 79-84. In Russian.
9. Solodovnikov A. I., Spivakovsky, A. M. Fundamentals of the theory and methods of spectral information processing: Proc. benefit. L.: publishing House of LSU, 1986. 272c. In Russian.
10. Dorogov A. Yu. Theory and design of fast tunable transformations and weakly connected neural networks. SPb.: "Polytechnic", 2014. 328 pp. In Russian. http://dorogov.su/