UDC 624.012

2D CONVOLUTIONAL NEURAL NETWORK IN THE DESIGN OF MONOLITHIC SELF-STRESSED SLABS ON BASE

A. E. Zheltkovich[1], K. G. Parchotz[2], V. V. Molosh[3], Haotian Jin[4], Shang Xu[5]

[1] Ph.D in Engineering, Associate Professor, Associate Professor of the Department of Applied Mechanics, Brest State Technical University, Brest, Belarus, e-mail: gelpek@mail.ru
[2] Programmer engineer, Belarus, e-mail: konstantinparhoc@gmail.com
[3] Ph.D in Engineering, Associate Professor, Associate Professor of the Department of Applied Mechanics, Brest State Technical University, Brest, Belarus, e-mail: m.vic@rambler.ru
[4] Postgraduate, Belarusian National Technical University, Minsk, Belarus, e-mail: 275479404@qq.com
[5] Undergraduate, Brest State Technical University, Brest, Belarus, e-mail: p0035799@g.bstu.by

Abstract

The purpose of this paper is to demonstrate the capabilities of convolutional neural networks in mechanics-related problems, in particular, in the design of monolithic self-stressed slabs on the base. In order to simplify the procedure of designing and calculating the displacements of slabs on the base has been developed a method that combines the advantages of theoretical models, and neural network technologies. The paper shows the possibility of using "soft computing", and also points out the promising potential of convolutional neural networks in predicting forced displacements in slabs of different geometrical shape.

Keywords: convolutional neural network, neurons, slabs on base, self-stressed concrete, hybrid.

ДВУМЕРНАЯ СВЁРТОЧНАЯ НЕЙРОСЕТЬ ПРИ ПРОЕКТИРОВАНИИ МОНОЛИТНЫХ САМОНАПРЯЖЕННЫХ ПЛИТ НА ОСНОВАНИИ

А. Е. Желткович, К. Г. Пархоц, В. В. Молош, Хаотянь Цзинь, Шань Сюй

Реферат

Целью настоящей статьи является демонстрация возможностей свёрточных нейросетей в задачах, связанных с механикой, в частности при проектировании монолитных самонапряжённых плит на основании. С целью упрощения процедуры проектирования и расчета перемещений плит по основанию был разработан метод, сочетающий в себе преимущества теоретических моделей и нейросетевых технологий. В статье показана возможность использования "мягких вычислений", а также отмечен перспективный потенциал свёрточных нейронных сетей в прогнозировании вынужденных перемещений в плитах различной геометрической формы.

Ключевые слова: свёрточная нейронная сеть, нейроны, плиты на основании, самонапряженный бетон, гибрид.

The aim is to obtain an improved method for slab design

The ultimate goal of this research is to create a slab design method that combines the advantages of neural network technologies and theoretical models. This paper discusses the first step in realizing this goal – the creation of a so-called hybrid, which should combine the advantages of theoretical models, finite element methods, biosimilar models, and neural network technologies. The step-by-step realization of the goal requires:

1. To illustrate the possibility of convergence of mechanics and neurotechnology.

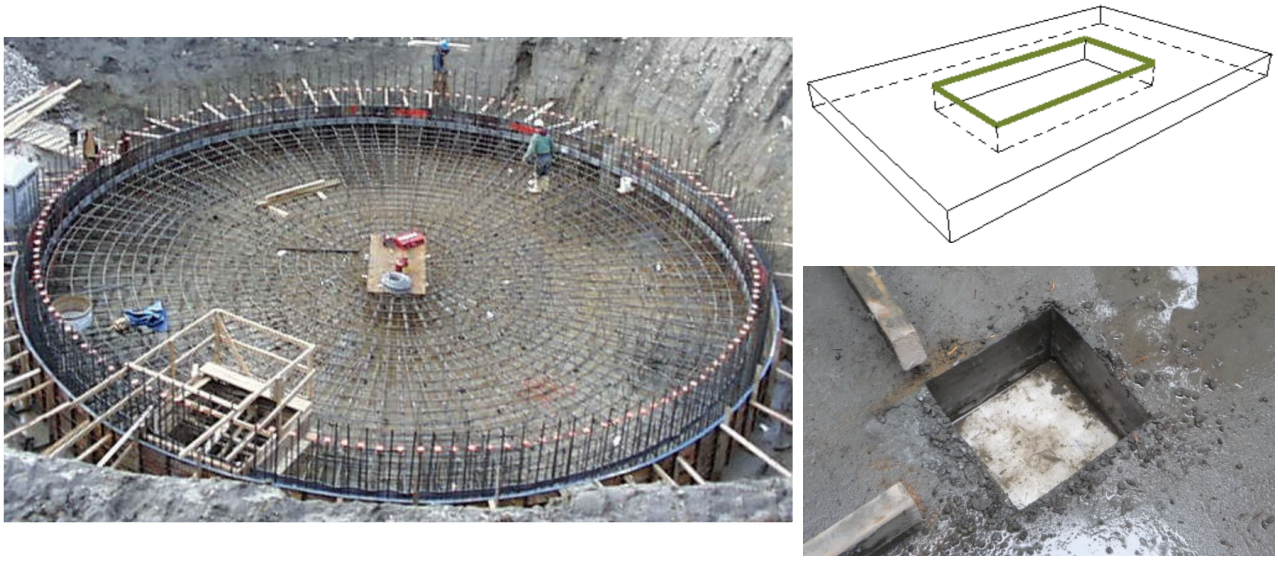

Figure 1 – Design of slabs with holes of different shapes [1]

https://doi.org/10.36773/1818-1112-2023-132-3-54-60

2. To show the possibility of using "soft computing" with the application of deep learning in tasks related to design.

3. To show the advantages of convolutional neural networks in predicting forced displacements in slabs on a base when there is a deficit (or absence) of initial data on displacements in the area of technological holes.

Key prerequisites

In some cases, even at the design stage, the slab design provides for the presence of holes of various shapes, both along the contour and in the body of the slab (Fig. 1). Thus, for example, when installing slabs in the shops of production buildings, machine shops of nuclear power plants, at other facilities, in already existing premises, it is necessary to provide in advance how the slab will behave in the area of the enveloped holes.

The solution of such problems on determination of displacements, stress-strain state (SSS), in closed form is either very labor-intensive or not achievable at all.

The existing approaches related to the use of finite element models are known to have certain and sometimes very significant disadvantages. Among them we can mention: 1 – application of temperature effects to describe the free deformations of concrete during curing; 2 – nonlinear kinetics of concrete curing (in the time period up to 28 days) is not taken into account; 3 – nonlinear change of concrete temperature during the course of chemical reactions in the initial period of curing is not taken into account.

Hard & Soft Computing - from competition to synthesis

In order to simplify the procedure of designing such slabs, we propose a joint application of a spectrum of methods. For this purpose the main role is given to the development and training of a 2D convolutional neural network (CNN) capable of working with multidimensional data matrices [2].

Problem statement

At the first stage, the task of intelligent solution search was set for a displacement in test slabs (slabs on which the neural network's prediction ability was tested). At the same time, when training the neural network, there was no information about displacement in the area of holes in the central part of the slabs. In other words – in all training slabs technological holes were made along the perimeter. In the test ones, the holes were located in the center.

For this purpose, using the solution obtained in the closed form [3] for strip-slabs, displacements in characteristic points along the length of a number of strip-slabs were determined, from which, at the next stage, slabs of different geometric shapes were made up.

At the same time, the samples differed in the shape and location of holes. Next, two-dimensional matrices describing the displacements of the nodal points of the slabs were compiled.

For this purpose, using the solution obtained in the closed form [3] for strip-slabs, displacements in characteristic points along the length of a number of strip-slabs were determined, from which, at the next stage, slabs of different geometric shapes were made up. At the same time, the samples differed in the shape and location of holes. Next, two-dimensional matrices describing the displacements of the nodal points of the slabs were compiled.

The location of the nodal points was determined using radius vectors drawn from the geometric center of the slab to a specific grid intersection with a step of 0.4x0.4 m.

It was required to determine the displacements in OXY axes for the test slabs with dimensions 4x4x0.1 m., having technological holes in the center – 1.2x1.2 m., as well as 0.4x0.4 m.

Preparing samples for training

To train the convolutional neural network, a sample of 21x7 different slabs was created (Fig. 2, the slab without holes is not shown).

Two types of data were used: topology of the nodes of the coordinate grid "breaking" the slab and shape parameters of the slab with holes; the second type included geometric characteristics of the slab as a whole, physical and mechanical characteristics of the self-stressing concrete and characteristics of the contact layer in the slab-base system.

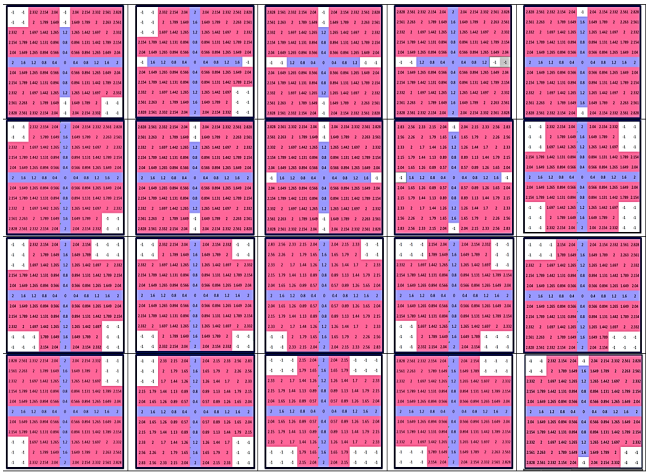

Figure 2 – A sample of 20 different slabs with peripheral holes for training the convolutional NN

To obtain the slab parameters, the slabs were marked into 11x11 points (nodes) with values of distances from the slab center and displacements defined for each grid node. The displacements of the grid nodes were used as target output values of the CNN. The resulting data were recorded as matrices that were fed to the input of the CNN. Data were stored in separate directories with csv files (each of the 7 directories contained 21 subdirectories with files for coordinates and displacements). This study tested several options for coding data describing slab holes (through "0" and through "–1").

CNN framework for predicting displacement

A convolutional neural network combines three approaches in image processing. These are the use of a local receptive field for each neuron of the convolutional layer, the formation of convolutional layers as a set of maps whose neural elements have identical synaptic connections, and the presence of subsampling layer maps that increase the network's resistance to distortions [4, 5, 6, 7].

One of the reasons for high CNN performance, according to the authors [4], is the use of identical neurons in each map, which makes it possible to reduce the number of customizable synaptic connections of the network.

From a technical point of view, this task is similar to image transformation. Therefore, the Pix-to-pix architecture was used [8]. The Pix-to-pix architecture consists of two blocks, an encoder and a decoder with connections between them (fig. 3).

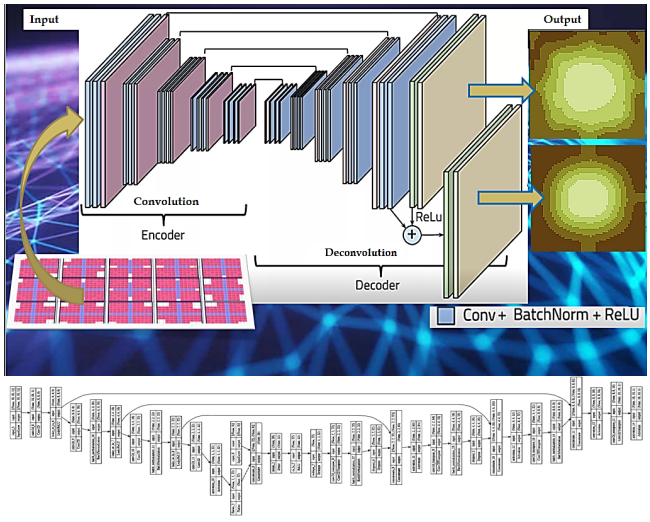

Figure 3 – Convolutional neural network with Pix-to-pix architecture [9], and diagrams of displacements in slabs with small and large center holes at the "output" of the CNN

All the steps of creating the convolutional NN, training and validation were implemented using the Python programming language and the Tensorflow framework [10].

Due to the pix-to-pix nature of the CNN, the two-dimensional data was enlarged to the number of grid nodes, 16x16 (with total 256 features), before being submitted for training. Two models were investigated, model #1 used the value "0" in the process of holes coding, model #2 used "–1". Simultaneously the task was set to identify the most optimal ratio between validation and training samples, as well as the number of training epochs. Thus, in both models from 50 to 100 training epochs were assigned. In models with 50 training epochs for optimal regularization, 75 % of the initial data for the training dataset were selected (randomly), and 25 % of the data were left for testing the quality of the model. In models with 100 training epochs, the ratio between the validation and training samples was 20 % versus 80 %.

The developed two-dimensional convolutional neural network works as follows: a list with three-dimensional matrices for training describing the features inherent in the slab is fed to the input. The first dimension is responsible for the "length coordinates of the grid points", the second for the "width coordinates", and the third for the "distance" of the grid nodes from the center of the slab.

At each layer, the encoder block wraps up the three-dimensional matrix, reducing the number of points of the slab grid by half and increasing the number of features responsible for the characteristic features of deformation in individual patterns – grid nodes. The convolution continues until a single point remains.

Then the inverse de-convolution to the previous size starts, where the output of the CNN "waits" for a matrix of displacements at characteristic points of the slab. The “sliding window” method is used for image scanning [4]. A sliding window is otherwise called a local receptive field or filter kernel for the corresponding (usually one) neuron of a feature map (each receptive field in the input image space is mapped to a separate neuron in each feature map). If the filter scans the image with stripe – ***s***, the number of neurons in each feature map is generally calculated by the formula:

$$D(C_1)=\left(\frac{n-p}{s}+1\right)\times\left(\frac{n-p}{s}+1\right). \quad (1)$$

Where: ***p*** x ***p*** is the size of the filter kernel, ***n*** x ***n*** – the dimensionality of the original image.

As follows from the last expression, the use of a convolutional network reduces the total number of configurable synaptic connections compared to a multilayer perceptron due to the use of identical neurons in each feature map [4].

The convolution layers of the encoder block have the following sequence of the number of feature maps and corresponding neurons in them: 8@8x8, 16@4x4, 32@2x2, 32@1x1. Similarly for the decoder block (fig. 3). The grid pitch of one filter kernel – 4x4 (fig. 4a). The filter kernels in this study work like independent neural networks. The filter kernel "passes" over the slab image (a grid with a certain step or stripe) shifting (at each step) by 2 values (thus, the "stripe" parameter was set equal – "2"). A "same padding" method (fig. 4c) was applied to the input data. Padding adds rows and columns of zeros around the data, which allow the kernel filter to start from the corner of the data and keep the size of the output data.

To determine the number of neurons in the first feature map (after the filter kernel passes through an image of size – 16x16), taking into account the padding method, one value is added to the existing 16x16 grid from the right and left, as well as from the top and bottom. Hence the final dimension of the incoming map is 18x18. Applying formula (1) [4] we obtain the number of neurons of the first feature map: $D(C_1)=\left(\frac{18-4}{2}+1\right)\times\left(\frac{18-4}{2}+1\right)$ = 8x8. Proceeding in a similar way for subsequent maps, we obtain – 4x4, 2x2, 1x1 neurons in convolutional layer maps.

The decoder layers are followed by a layer – "dropout" (Fig. 4b), which "disconnects" the filter neurons with a probability of 50% to minimize the overtraining of the network.

In the current study, the number of trained parameters exceeds the number of samples on which the model is trained, which is a drawback of the model. For the model to be built, the number of sample slabs should be at least 65000. At the moment the data of missing samples are being generated, which will be reflected in the nearest works.

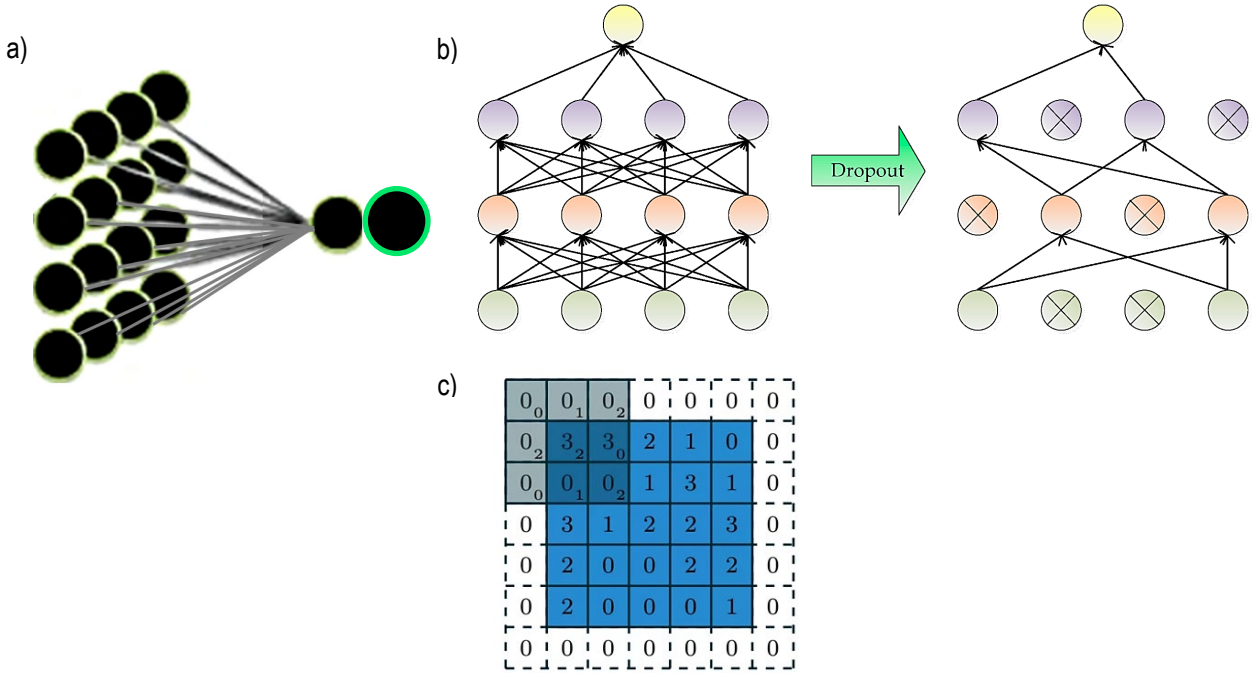

Figure 4 – Schemes: a) 4x4 filter kernel structure, b) "dropout" operator [11], c) “padding method” [12]

https://doi.org/10.36773/1818-1112-2023-132-3-54-60

Mechanism of operation of ANN and a filter

Each filter neuron is considered as an operator that changes the input data [4, 13, 14]. The ANN receives the values of grid node coordinates as input, while the signal at the output of the neuron is defined as follows:

$$y_i = f_{act}\left(\sum x_i \cdot w_i + b\right). \quad (2)$$

Where: ***x*** and ***y*** – input and output signals of NN, ***w*** – weight parameter of synapse, ***b*** is a bias, f_{act} – the neuron activation functions – *LeakyReLu* for encoder and *ReLu* for decoder. Since all data fed to the input and outputs of the convolutional neural network were normalized using Z-normalization, the activation of the last layer was performed using the function – *Tanh*. After training the neural network, the data was returned to the normal dimensionality using the "invers transformation".

The value of the transformed function (1) at the output of the resulting neuron of the filter (in given case – 1 neuron, Fig. 4a).

Normalization

A number of methods are known in mathematical statistics: decimal scaling, minimum normalisation, normalisation by mean (Z-normalisation) and others [15]. In this paper has been applied normalisation by mean (to compare values of different dimensions, as well as to bring the data to a more convenient form for training a neural network). Z-normalisation sets the mean (mathematical expectation) and variance of the data and is represented by the formula:

$$z = (x - \mu)/\sigma, \quad (3)$$

where: μ and σ – mathematical expectation (mean) and standard deviation, respectively.

Quality criteria for ANN performance

When testing the neural network, the mean absolute error with L1 norm [16, 17] was used, as this metric reflects the accuracy of the prediction result quite well. The loss function was defined as:

$$E = \frac{1}{n}\sum\left|Y_{target} - Y_{predicted}\right|, \quad (4)$$

where: n – number of examples, Y_{target} – actual initial data, $Y_{predicted}$ – predicted values of the predicted parameter.

Parametric optimization, gradient descent algorithm

As optimizer has been used the method for stochastic optimization "Adam". Adam [18] is a first-order-gradient-based algorithm of stochastic objective functions, based on adaptive estimates of lower-order moments [19].

The goal of parameter optimization is to find the minimum value of the loss function E. At each iteration, the algorithm updates the weight parameters w, as showed in function (5).

$$\omega_{n+1} = \omega_n - \frac{\alpha}{\sqrt{\hat{v}_n} + \varepsilon} \cdot \hat{m}_n, \quad (5)$$

where: α=0.001 is the learning rate parameter, g_t^2 – indicates the element-wise square $g \odot g$. $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 1 \cdot 10^{-7}$. All operations on vectors are element-wise. With β_1^t and β_2^t are denoted β_1 and β_2 to the power *t* (*t=0* at the first initializing). The 1st moment vector at the first initializing – $m_0 = 0$, 2st moment vector at the first initializing – $v_0 = 0$, ω_0 – initial parameter vector, initializing by random generator. The weights are updated until the current and previous values converge.

Was used the next algorithm [19]:

1) $t \leftarrow t + 1$;

2) $g_t \leftarrow \nabla_w f_t(w_{t-1})$ – Get gradient w.r.t. stochastic objective at time-step t, i.e. the vector of partial derivatives of f_t, w.r.t (w) evaluated at time-step t;

3) $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) g_t$ – Update biased first moment estimate;

4) $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) g_t^2$ – Update biased second row moment estimate;

5) $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ – Compute bias-corrector first moment estimate;

6) $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ – Compute bias-corrector second row moment estimate;

7) $\omega_t = \omega_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \varepsilon)$ – Update parameters.

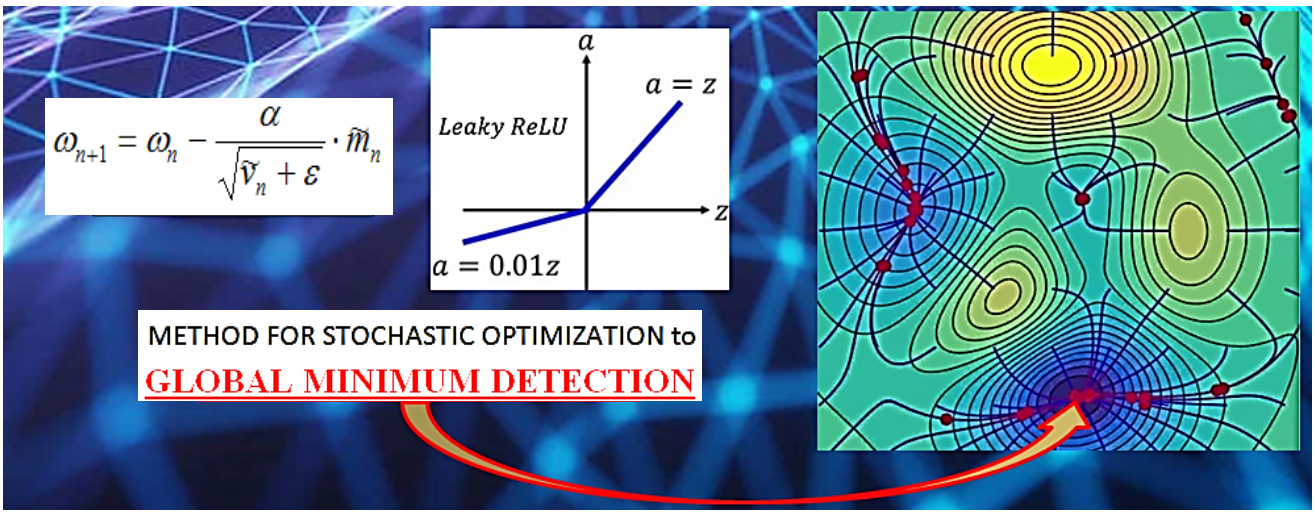

Figure 5 – Gradient descent on the loss hypersurface [20]

where: $f(w)$ is a noisy objective function: a stochastic scalar function that is differentiable w.r.t. parameters ω.

After acquiring loss function value, the antigradient is computed, updating the synapse weights. Thus, using the gradient information, the optimal path to achieve the global minimum on the loss hypersurface is determined.

Network Training

Since the developed neural network is a convolutional network with many layers, the launch of 100 epochs on an "Intel" processor with 4 GB RAM and 2 GB video card lasted from several minutes to half an hour (depending from model to model). In order to optimize the process, have been decided to train CNN in remote access mode inside the Google Drive environment. Has been used Python 3 server accelerator based on Google Compute Engine. The time of passing 100 epochs was reduced to 5–10 seconds.

Displacement results in slabs with center hole

In the next step, the data were fed into the de-encoder and unwrapped. The CNN thus matched the data on geometric parameters (node coordinates, distances from the center, slab shape features, hole locations), with the data on displacements of the slab grid nodes used for training. The displacement information was translated into colors of a particular spectrum and displayed as an output image. In the center of figures 6a-d, 7a-d the color spectrum of displacements is displayed (the color corresponds to the magnitude of the displacements).

In order to show that the prediction accuracy does not change significantly after 50 epochs of training (even when the training duration is doubled, i.e., up to 100 epochs), here presented plots of prediction of displacements in full-body slabs (fig. 6g-j, 7g-j). A comparative modeling analyze of the displacements (in full-body slabs on the base) has been carried out in the figures below (fig. 6e-g, 7e-g). The relative and absolute errors in determining the displacements of the slabs are showed (fig. 6h-i, 7h-i).

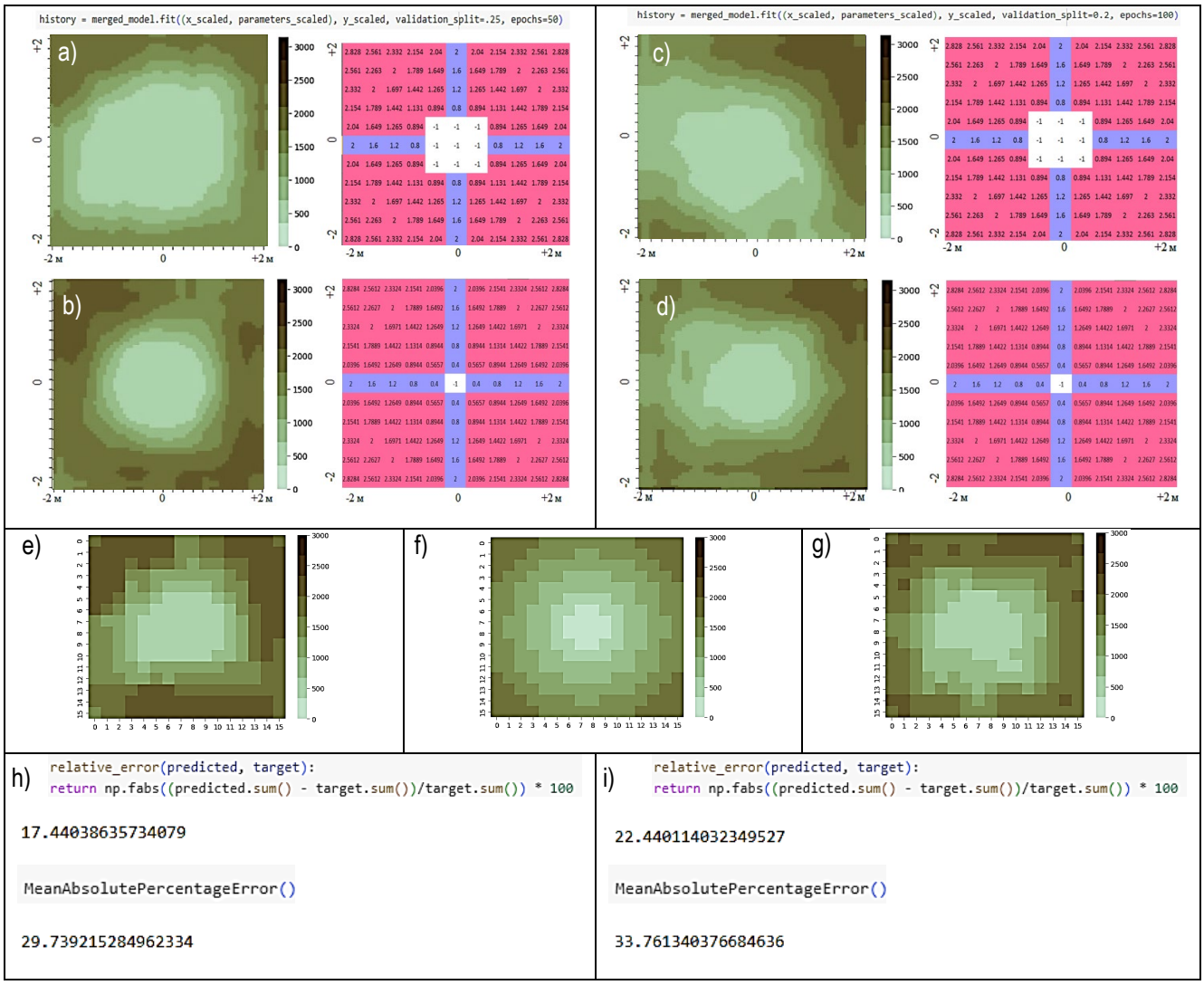

Figure 6 – Displacements in slabs with a central holes, holes coded through "0":
a) model #1–1, central hole 1.2x1.2 m, training-validation split 75/25 %, number of epoch – 50,
b) model #1–2, central hole 0.4x0.4 m, training-validation split 75/25 %, number of epoch – 50,
c) model #1–1, central hole 1.2x1.2 m, training-validation split 80/20 %, number of epoch – 100,
d) model #1–2, central hole 0.4x0.4 m, training-validation split 80/20 %, number of epoch – 100,
i) model #1–1, full-body slab, training-validation split 75/25 %, number of epoch – 50,
f) actual full-body slab 2x2x0.1 m,
g) model #1–2, full-body slab, training-validation split 80/20 %, number of epoch – 100,
h) model #1–1, relative and absolute errors, i) model #1–2, relative and absolute errors

https://doi.org/10.36773/1818-1112-2023-132-3-54-60

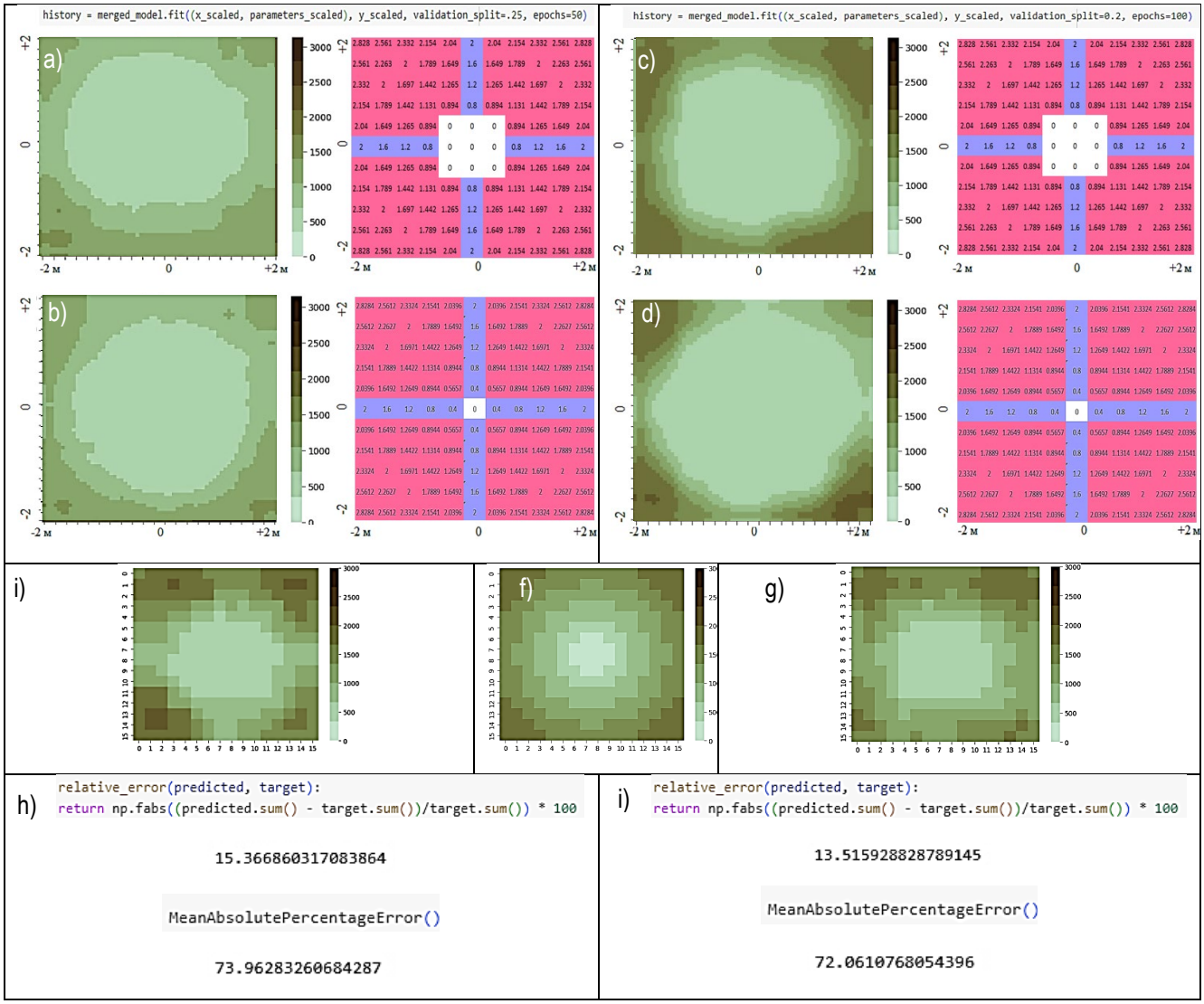

Figure 7 – Displacements in slabs with a central holes, holes coded through "–1":
a) model #2–1, central hole 1.2x1.2 m, training-validation split 75/25 %, number of epoch – 50,
b) model #2–2, central hole 0.4x0.4 m, training-validation split 75/25 %, number of epoch – 50,
c) model #2–1, central hole 1.2x1.2 m, training-validation split 80/20 %, number of epoch – 100,
d) model #2–2, central hole 0.4x0.4 m, training-validation split 80/20 %, number of epoch – 100,
i) model #2–1, full-body slab, training-validation split 75/25 %, number of epoch – 50,
f) actual full-body slab 2x2x0.1 m,
g) model #2–2, full-body slab, training-validation split 80/20 %, number of epoch – 100,
h) model #2–1, relative and absolute errors,
i) model #2–2, relative and absolute errors

Conclusions:

1. Several options for coding data describing holes in slabs (via "0" and via "–1") are investigated. Two models were created to test the quality of prediction of slab displacements. The relative error for a full-body slab (taken over 256 points, i.e., the entire 16x16 grid), when used model #1–1 was 17.44 %; for model #1–2 – 22.44 %; for model #2–1 – 15.37%; and to model #2–2 – 13.52 %.

2. We assume that model #1 has smaller absolute errors due to the coding of holes with the number "0". It was easier for the model to establish the absolute difference between the displacements of full-body slabs (which have no hole in the center, where the value is strictly "0") and the displacements of the central region of the test slabs (the central region of the slab – the light pixels in the plots have zero displacements). In model #2, where coding was performed through "–1" relative errors in determining the displacements are somewhat smaller, at the same time we observe large absolute errors. This is primarily due to the peculiarity of "holes" coding. At the same time, such coding, apparently, complicates the training of the convolutional network, which is evident from the comparison of loss diagrams in models #1 and #2.

3. The developed neural network, trained on the basis of 147 samples, is quite confident in predicting displacements in slabs with a central hole, using data only from peripheral cutouts. If the training sample is increased, the relative errors as well as losses in training of the SNN can be significantly reduced.

References

1. Design of slabs with holes of different shapes [Electronic resource]. – Access mode: https://www.summitengineeringinc.com/concrete-liquid-storage-tanks-for-water-and-wastewater-treatment-plants/. – Date of access: 12.10.2023.

2. Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning [J] / H. C. Shin [et al.] // IEEE transactions on medical imaging. – 2016. – No. 35 (5). – P. 1285–1298.
3. ZHeltkovich, A. E. Raschyot vynuzhdennyh peremeshchenij i napryazhenij ot usadki v monolitnyh betonnyh plitah, vzaimodejstvuyushchih s osnovaniem / A. E. ZHeltkovich, V. V. Tur // Stroitel'naya nauka i tekhnika. – 2011. – № 2 (35). – S. 120–125.
4. Nejrosetevye tekhnologii obrabotki dannyh : ucheb. posobie / V. A. Golovko, V. V. Krasnoproshin. – Minsk : BGU, 2017. – 263 s.
5. Backpropagation applied to handwritten zip code recognition / Y. Le Cun [et al.] // Neural computation. – 1989. – № 1 (4). – P. 541–551.
6. Object recognition with gradient-based learning / Y. Le Cun [et al.] // In shape, contour and grouping in computer vision. – B. ; Heidelberg, 1999. – P. 319–345.
7. Gradient-based learning applied to document recognition / Y. Le Cun [et al.] // Proc. of the IEEE. – 1998. – № 86 (11). – P. 2278–2324.
8. Image-to-Image Translation with Conditional Adversarial Networks [Electronic resource] / P. Isola [et al.] // arXiv:1611.07004 [cs.CV]. – 2016. – 17 p.
9. Convolutional neural network with Pix-to-pix architecture [Electronic resource]. – Access mode: https://ai2-s2-public.s3.amazonaws.com/figures/2017-08-08/93c675e333380eead0adb9acbfcefdefca9270d4/4-Figure3-1.png. – Date of access: 30.10.2023.
10. Python programming language and the Tensorflow framework [Electronic resource]. – Access mode: https://www.tensorflow.org/about/bib. – Date of access: 22.10.2023.
11. Convolution operator "padding" [Electronic resource]. – Access mode: https://img-blog.csdn.net/20171229134500560. – Date of access: 10.11.2023.
12. Dropout operator [Electronic resource]. – Access mode: https://www.mdpi.com/2076-3417/9/10/2028/html – Date of access: 10.11.2023.
13. Full connected neural-network for simulation of extantion in self-stressed monolitic slabs on ground / A. E. Zheltkovich [et al.] // Perspektivnye napravleniya innovacionnogo razvitiya stroitel'stva i podgotovki inzhenernyh kadrov : sbornik nauchnyh statej XXII Mezhdunarodnogo nauchno-metodicheskogo seminara, Brest, 29–30 sentyabrya 2022 g. / Ministerstvo obrazovaniya Respubliki Belarus', Brestskij gosudarstvennyj tekhnicheskij universitet; redkol.: S. M. Semenyuk [i dr.]. – Brest, 2022. – P. 18–28.
14. Primenenie polnosvyaznoj nejronnoj seti v raschyotah soprotivleniya srezu pri prodavlivanii ploskih zhelezobetonnyh plit perekrytij bez poperechnoj armatury / V. V. Molosh [i dr.] // Perspektivnye napravleniya innovacionnogo razvitiya stroitel'stva i podgotovki inzhenernyh kadrov : sbornik nauchnyh statej XXII Mezhdunarodnogo nauchno-metodicheskogo seminara, Brest, 29–30 sentyabrya 2022 g. / Ministerstvo obrazovaniya Respubliki Belarus', Brestskij gosudarstvennyj tekhnicheskij universitet; redkol.: S. M. Semenyuk [i dr.]. – Brest, 2022. – S. 121–133.
15. Data normalisation [Electronic resource]. – Access mode: https://wiki.loginom.ru/articles/data-normalization.html. – Date of access: 15.11.2022.
16. Barrodale, I. L_1 Approximation and the Analysis of Data / I. Barrodale // Journal of the Royal Statistical Society. Series C (Applied Statistics) : JSTOR 2985267. – 1968. – Vol. 17, No. 1. – P. 51–57.
17. Mean absolute error with L1 norm [Electronic resource]. – Access mode: https://montjoile.medium.com/l0-norm-l1-norm-l2-norm-l-infinity-norm-7a7d18a4f40c. – Date of access: 20.12.2023.
18. Adam: A Method for Stochastic Optimization [Electronic resource] / D. P. Kingma, J. Ba // arXiv:1412.6980 [cs.LG]. – 2014. – 15 p.
19. First-order-gradient-based algorithm of stochastic objective functions, based on adaptive estimates of lower-order moments [Electronic resource]. – Access mode: https://medium.com/analytics-vidhya/a-complete-guide-to-adam-and-rmsprop-optimizer-75f4502d83be. – Date of access: 14.12.2023.
20. Gradient descent [Electronic resource]. – Access mode: https://commons.wikimedia.org/wiki/File:Gradient_Descent_in_2D.webm. – Date of access: 15.11.2023.

Material received 24/11/2023, approved 25/12/2023, accepted for publication 26/12/2023