Neural networks for multicontinuum models

A. Grigorev, S. Stepanov, Dj. Nikiforov
Ammosov North-Eastern Federal University, Yakutsk, Russia
E-mail: re5itsme@gmail.com, cepe2a@inbox.ru, dju92@mail.ru

Abstract.
Neural networks are developed for multicontinuum models. Multicontinuum models [1] assume that processes divided into several continuums and they are connected by some exchange member. In problems connected with PDE not only solution, but also coefficients can depend on time. First we construct LSTM neural network for exchange coefficient that depend on time. We compare the results obtained on the basis of the reference coefficient and on the basis of the neural network. Next we apply same neural network on the solutions. The proposed approach involves the use of a neural network for the approximate generation of a solutions and the further adaptation of the solution of a neural network for generate coarse grid solution based on DG finite elements.

1. Introduction
Nowadays neural networks [2] are very popular technology for wide range of problems. Neural networks are subclass of machine learning methods. About last 5 years neural networks demonstrate better results than other methods in many applications — language analysis, image processing, artificial intellect, etc. There are several reasons for this — big amount of data (internet, data generation methods, kaggle, etc), high-performance computations (parallel calculations, supercomputers, GPU computing), modern algorithms (convolutional neural networks, recurrent neural networks, open-source libraries like tensorflow). For every kind of problem we need to apply special neural network. For example for image classification we need to use CNN. In problems connected with PDE not only solution, but also coefficients can depend on time. We will use LSTM [3] (one effective recurrent neural network) for coefficient predictions on time and expand this approach on solution prediction.

2. Neural network approach
Most of neural networks methods can be expressed as optimization algorithm to setup model parameters as weights and bias values for minimize error functional. A recurrent neural network (RNN) is a class of artificial neural network where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks that connected with sequences. Let’s discuss idea of LSTM. An LSTM network is a special type of RNN capable of learning long-term dependencies. LSTM networks were introduced [in Hochreiter and Schmidhuber, 1997]. Such networks do an excellent job with many tasks and are widely used at this time. LSTM networks are designed specifically to solve the problem of long-term dependencies. Storing information for long periods...
of time is their default behavior. All RNN have the form of a chain of repeating modules. The repeating module of a standard RNN has a very simple structure, for example, a single tanh layer (the activation function is the hyperbolic tangent). An LSTM network is a similar chain but a repeating module has a different structure. Instead of one layer it contains four layers which interact in a special way. A key feature of the LSTM network is the cell state. The state of the cell can be compared with a conveyor belt. It runs through the entire chain with only minor linear interactions. The LSTM network has the ability to remove and add information to the cell state. This process is governed by special structures called gateways. A gate is a mechanism for allowing information to be passed selectively. It consists of a sigmoid layer (activation function sigmoid) and a pointwise multiplication operation. The output of the sigmoid layer is a number from 0 to 1 which determines the level of transmission. The cell has three gates controlling its state. At the first stage it is necessary to decide what information should be removed from the state of the cell. This decision is made by a sigmoid layer called the forget gate. The next step is to decide what new information should be written to the cell state. This stage is divided into two parts. First the sigmoid layer called the input gate (input gate) decides which values need to be updated. The tanh layer then creates a vector of candidate values that can be added to the cell state. It’s time to update the previous state of the cell to the current state. At the previous stages it was already decided what needs to be done now it remains only to accomplish this. Finally you need to decide what to send out. The output will be the filtered state of the cell. First the sigmoid layer decides which elements of the cell state should be transferred to the output. The state of the cell is then converted using a tanh layer to the interval from \(-1\) to \(1\) and multiplied by the output of the sigmoid layer in order to output only what was decided to output.

3. Preprocessing problem of exchange coefficient
In many work researchers interested in exchange mechanism between continua (e.g. [4]). In our case we suppose that exchange coefficient depends on time. We generated CSV file that consist values of exchange coefficient that depends on time. This data builded with step 1 month. For simulate real data we add noise See Figure 1.

- Let’s suppose exchange coefficient depends on time
- Let’s suppose one-phase filtration flow not so deep underground
- Let’s suppose the soil degrades over time so the exchange coefficient grows
- Period: 1999 Jan – 2028 Dec

![Figure 1. Exchange coefficient on time](image-url)
We using open source libraries on Laptop (i7-8750H + Nvidia GeForce GTX 1050 Ti). Moreover for realise neural network part on GPU version of Tensorflow. TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks.

We applying LSTM network based on next algorithm:

- We using SKlearn preprocessor to normalize data (0, 1)
- We divide generated data to train part and test part
- We build up Sequential model using Keras library
- We add LSTM model to Sequential with 4 extra layers
- Parameters of NN: epochs = 100, batch = 1

Comparing results

![Graphs](image)

**Figure 2.** Blue – reference, orange – training, green - prediction

4. Model problem
Let’s consider model problem with constant coefficient case excluding exchange coefficient:

\[
\begin{align*}
    c \frac{\partial u_1}{\partial t} - \nabla \cdot \nabla u_1 + r(t)(u_1 - u_2) &= 0, \\
    \frac{\partial u_2}{\partial t} - d \nabla \cdot \nabla u_2 - r(t)(u_1 - u_2) &= 0.
\end{align*}
\]

We want to compare not only coefficient values but also results of modeling that based on reference coefficient and coefficient that predicted by neural network (last 2 year). We using
Figure 3. Unit rectangle

next computation domain $\Omega$ ($\Gamma_D = \Gamma_1 \cup \Gamma_3$, $\Gamma_N = \Gamma_2 \cup \Gamma_4$):

We applying next boundary conditions:

\[
\begin{align*}
  u_\alpha(x, t) &= 1 - \exp(-10t), \quad x \in \Gamma_1, \\
  u_\alpha(x, t) &= 0, \quad x \in \Gamma_3, \\
  \frac{\partial u_\alpha}{\partial n}(x, t) &= 0, \quad x \in \Gamma_2 \cup \Gamma_4, \quad t \in (0, T],
\end{align*}
\]

and initial conditions:

\[
u_\alpha(x, 0) = 0, \quad x \in \Omega, \alpha = 1, 2.
\]

**Solving method**

We implement finite element method using Fenics library [5]: CG 1st order ($\tau = 0.015$, Structured triangular mesh $32 \times 32 \times 2$). And on time we use fully implicit scheme:

\[
\begin{align*}
  c \int_{\Omega} \frac{y_1^{n+1} - y_1^n}{\tau} v_1 dx + \int_{\Omega} \nabla y_1^{n+1} \nabla v_1 dx + r \int_{\Omega} (y_1^{n+1} - y_2^{n+1}) v_1 dx &= 0, \\
  \int_{\Omega} \frac{y_2^{n+1} - y_2^n}{\tau} v_2 dx + d \int_{\Omega} \nabla y_2^{n+1} \nabla v_2 dx + r \int_{\Omega} (y_2^{n+1} - y_1^{n+1}) v_2 dx &= 0.
\end{align*}
\]

On the model two-dimensional problem the dependence of the accuracy of an NN based solution on reference solution is investigated (See Figures 4, 5). Before we can see (Figure 2) LSTM neural network even can predict noise behavior of coefficient but also $L_2$ errornorm of solution is not exceed coefficient discrepancy maximum value (For example Figure 4).

Also we solution at the last moment of time which demonstrate adequate solution based on NN predicted coefficient (See Figure 6).

Next idea is we constructing continuation on the solution. We can apply the same LSTM neural network already on solutions that we obtain by simulating based coefficient predicted by LSTM. In this easier case LSTM neural network demonstrate much better results. And by predicting in set of points of domain we can construct coarse grid solution. On Figure 7 we demonstrate constructing coarse grid solution by pulling pointwise LSTM solution on piecewise constant DG finite elements [6].

**Conclusions**
Figure 4. Comparing reference solution and NN coefficient based

Figure 5. Comparing reference solution and NN coefficient based ($\times0.1$)

- We apply LSTM network for coefficient preprocessing problem. Results demonstrate that neural network can recreate noise behaviour of time dependent sequences.
- We extended predictions on solutions of model problem. And this approach let us construct coarse grid solution. This method provide high accuracy of NN predictions for pointwise solution.

4.1. Acknowledgments
The work was supported by RSCF grant 17-71-10106 for A. Grigorev, Stepanov’s work is supported by grant of RFBR N 18-31-00318

References
[1] Barenblatt, G. I., Zheltov, I. P., Kochina, I. N. (1960). Basic concepts in the theory of seepage of homogeneous liquids in fissured rocks [strata]. Journal of applied mathematics and mechanics, 24(5), 1286-1303.
[2] Haykin, S. (2009). Neural networks and learning machines. vol. 3 Pearson. Upper Saddle River, NJ, USA.
[3] Hochreiter, S., Schmidhuber, J. (1997). LSTM can solve hard long time lag problems. In Advances in neural information processing systems (pp. 473-479).
[4] Vabishchevich, P. N., Grigoriev, A. V. (2016). Numerical modeling of fluid flow in anisotropic fractured porous media. Numerical Analysis and Applications, 9(1), 45-56.
[5] Logg, A., Mardal, K. A., Wells, G. (Eds.). (2012). Automated solution of differential equations by the finite element method: The FEniCS book (Vol. 84). Springer Science & Business Media.
[6] Arnold, D. N., Brezzi, F., Cockburn, B., Marini, L. D. (2002). Unified analysis of discontinuous Galerkin methods for elliptic problems. SIAM journal on numerical analysis, 39(5), 1749-1779.
Ref. fractures

Ref. matrix

NN based fractures

NN based matrix

**Figure 6.** Solutions at last moment of time

Point solution prediction

coarse grid solution (DG 0)

**Figure 7.** Constructing coarse grid solution