J. Appl. Comput. Mech., xx(x) (2021) 1-9
DOI: 10.22055/JACM.2021.37805.3092

ISSN: 2383-4536
jacm.scu.ac.ir

Shahid Chamran
University of Ahvaz

Journal of
Applied and Computational Mechanics

Research Paper

A Deep Learning Approach to Predict the Flow Field and Thermal Patterns of Nonencapsulated Phase Change Materials Suspensions in an Enclosure

Mohammad Edalatifar[1], Mohammad Bagher Tavakoli[2], Farbod Setoudeh[2]

[1] Department of Electrical Engineering, Arak Branch, Islamic Azad University, Arak, Iran, Email: m.edalatifar@gmail.com
[2] Department of Electrical Engineering, Arak Branch, Islamic Azad University, Arak, Iran, Email: m-tavakoli@iau-arak.ac.ir
[3] Faculty of Electrical Engineering, Arak University of Technology, Arak, Iran, Email: f.setoudeh@arakut.ac.ir

Received June 22 2021; Revised July 02 2021; Accepted for publication July 03 2021.
Corresponding author: M. B. Tavakoli (m-tavakoli@iau-arak.ac.ir)

Abstract. The flow and heat transfer of a novel type of functional phase change nanofluids, nano-encapsulated phase change suspensions, is investigated in the present study using a deep neural networks framework. A deep neural network was used to learn the natural convection flow and heat transfer of the phase change nanofluid in an enclosure. A dataset of flow and heat transfer samples containing 3290 samples of the flow field and temperature distributions was used to train the deep neural network. The design variables were fusion temperature of nanoparticles, Stefan number, and Rayleigh number. The results showed that the proposed combination of a feed-forward neural network and a convolutional neural network as a deep neural network could robustly learn the complex physics of flow and heat transfer of phase change nanofluids. The trained neural network could estimate the flow and heat transfer without iterative and costly numerical computations. The present neural network framework is a promising tool for the design and prediction of complex physical systems.

Keywords: Nanoencapsulated phase-change suspension; deep convolutional neural networks; Natural convection heat transfer; deep learning.

1. Introduction

The natural convection heat transfer of nanofluids has been the subject of many recent studies due to its crucial applications in industrial cooling systems. Natural convection cooling does not need moving parts such as fans or pumps and external power. Hence, it benefits from low maintenance costs, safety, and low noise. Thus, natural convection cooling is an excellent cooling choice. However, the cooling power of natural convection is low compared to forced convection flow. The main reason for the low cooling power of natural convections is the poor thermal conductivity of the working fluid. Thus, any approach, which could improve the thermal conductivity of the heat transfer fluid is of great interest.

The nanofluids are a new type of engineered fluid, which is a stable suspension of nanoparticles. The nanofluids show enhanced thermophysical properties and could be are employed in various industrial thermal systems. Izadi et al. [1] and Sajid and Ali [2] have reviewed the recent application of nanofluids and their engineering aspects. Various aspects of heat transfer are explored in some of the recent publications [3-5].

The nanofluids are promising candidates for improving the heat transfer power of natural convection cooling systems. In this regard, many researchers tried to investigate and understand the natural convection behavior of nanofluids in enclosures. Various types of nanofluids, such as Al2O3-water [6-8], Cu-water [8, 9], CuO-water [10], Fe3O4-water [11, 12], and Ag-MgO-water [13] have been investigated in recent studies. The Nano-Encapsulated-Phase Change Material (NEPCM) suspensions are an advanced type of nanofluids, in which the core of nanoparticles is made of a Phase Change Material (PCM). The PCM core is enclosed in a nanoshell. Thus, NEPCM particles are composite particles that are stably suspended in a host fluid such as water. The NEPCM particles circulate in an enclosure with the host fluid and could reach hot or cold regions. When the temperature is higher than their fusion temperature, they melt and absorb the phase change's latent heat. Then, when they reach a cold area, they solidify and release the latent heat. Hence, the nanoparticles contribute to heat transfer by their latent heat [14]. The natural convection flows are inherently complex phenome as the fluid flow is coupled with the heat transfer. The presence of phase change heat transfer increases the complexity of the phoneme, and hence, advanced numerical techniques are demanded to handle the simulation of NEPCM flows.

Published online: July 05 2021

The governing equations, representing the flow, heat transfer, and phase change of the particles, are non-linear Partial Differential Equations (PDEs). These PDEs are mainly originated from the physical laws of conservation of mass, momentum, and energy. The numerical methods discrete the domain of the solution into small partitions (mesh) and employ numerical approximations to the original governing equations. This process leads to a system of algebraic equations, which should be solved iteratively. The iterative solution is a computationally costly and time-consuming procedure. Moreover, the computer codes generally start with an initial condition or an initial guess for the variable fields and commence the iterations toward the final solution.

Recently, Deep Neural Networks (DNNs) demonstrated promising capability to extract features. DNNs benefit from many hidden layers and show extraordinary potential in generating images. Until 2006, the neural networks could not adequately benefit from many hidden layers because of inadequate training techniques. However, Hastad and Goldmann [15] provided approaches to overcome this shortcoming and highlighted the enormous advantages of using many hidden layers. Since many hidden layers could be used in DNNs, they are excellent candidates for data-driven approaches [16]. For instance, neural networks have been used for modeling of human ureter [17], rotating machines faults classification [18], and prediction of entrance length for magnetohydrodynamics channels flow [19].

Some of NNs are used to estimate some characteristics parameters such as Nusselt number or skin friction. However, some NNs could also be used to estimate a variable field in the domain, such as temperature distribution. For example, Ermis [20] employed a NN to estimate the toral stored energy in latent heat thermal energy storage units. Their NN was made of one hidden layer. The NN receives the time, inlet heat transfer fluid temperature, Reynolds number, and heat transfer area and estimates the total thermal energy storage. The NN could estimate the total thermal energy storage with an absolute mean relative error of 5.58%.

Azwadi et al. [21] utilized an adaptive network-based fuzzy inference system (ANFIS) to estimate the temperature and flow field in a lid-driven cavity. They solved the governing equation for heat and fluid flow by using Lattice Boltzman method and produced a dataset of simulations to be used for taring of their ANFIS neural network. The NN input parameters were x and y coordinates, Rayleigh, and Reynolds numbers.

Moreover, very recently, Selimefendigil et al. [22] applied NNs to estimate the magnetohydrodynamic free convection behavior of a fluid in a porous cavity. Some of the input parameters were the coordinate system, Rayleigh number (Ra), Hartmann number, Darcy Number, the porosity of the medium, and the target parameters were velocity and temperature. These authors created a comprehensive dataset of samples to be used to train a Long Short-Term Memory (LSTM) neural network. The results showed that the utilized LSTM could estimate temperature better than velocity. Zhou et al. [23] employed a NN with a hidden layer to predict the temperature of a monitor point through time in natural convection flow in a cavity. Authors numerically solved the flow and temperature distributions in the cavity and built a dataset. Later, the dataset was used to train the NN The NN receives the initial temperature and coordinate of a point in the cavity and estimates the temperature of a point after 9000s. Akbari et al. [24] used an ANFIS to model the natural convection in a cavity. They utilized an optimization algorithm, Jaya optimization algorithm, to train their neural network. The ANFIS neural network receives the Rayleigh number and a geometrical parameter and estimates the Nusselt number.

There are various engineering applications in which the input parameters are a group of controlling parameters, while the output is a 2D or 3D field distribution [21-23]. Generally, in such problems, the coordinates of the field variables are considered as the input parameters, and the NN computes the outputs for each set of the group controlling parameters and the individual input coordinates independently. Thus, each output pixel will be predicted individually based on the provided input pixel (provided input coordinate). Predicting the pixels of an image one by one is a tedious and time-consuming procedure. More importantly, in a physical image, the pixels of an image is related to the neighboring pixels, and predicting the pixels of the output image individually ignores such physical relationships. Thus, we believe that predicting all pixels at once could improve the accuracy of the solution and maintain the physical integrity of the predictions. The purpose of this study is to teach the complex physics of natural convection heat transfer of a NEPCM-suspension to a DNN. The trained DNN will be used to predict the velocity and temperature fields with no iterations directly.

2. Mathematical model & systematic analysis

The purpose of this study is to teach the physics of phase change natural convection heat transfer in an enclosure to a DNN. The present approach consists of two main steps. The first one is creating a dataset of pictures describing the flow field and heat distribution in an enclosure model. The mathematical model and its validation for natural convection heat transfer of Nano-Encapsulated Phase Change Material (NEPCM) suspensions in a rectangular enclosure have been addressed in the previous publication [14]. The same model and data have been used here to produce a dataset. The dataset contains images of temperature distribution, and velocity components in x and y directions, and pressure distribution. The images are produced by numerical computation of governing equations for various combinations of design parameters, which are the volume fraction of nanoparticles φ, non-dimensional phase change temperature θ_f, Stefan number (Ste), and Rayleigh number (Ra). The data set contains 3,290 sets of images for a combination of design parameters.

The second step is developing a DNN to get the design parameters as input and produce the output images. The DNN should be trained to learn the physics of phase change natural convection. During the training process, the coefficient of the DNN will be adjusted in a way that the trained DNN will be able to estimate the flow and temperature fields adequately. Preparing a dataset, definition of DNNs, and designation of estimator error are the primary actions for design a DNN that are considered in the following sections.

2.1 Dataset

A dataset containing 3,290 samples was produced in a uniform mesh of 128 × 128. As mentioned, the inputs of samples are the volume fraction of nanoparticles φ, non-dimensional phase change temperature θ_f, Stefan number (Ste), and Rayleigh number (Ra), and the outputs are velocity in the x-direction (U), velocity in the y-direction (V), pressure filed (P), and temperature distribution (θ) which are all in a general non-dimensional form. The input parameters and their range of variations have been summarized in Table 1. Here, Fig. 1 depicts a set of output images, which are the velocity components (U and V), pressure distribution (P), and temperature distribution (θ), corresponding to a set of input parameters (design parameters). Indeed, Fig. 1 shows a sample of dataset images. Each color map is made of 15 levels, and the color legend is depicted below each image. It should be noted that input data have been normalized between -1 and 1 as well as output data between 0 to 1. The normalization between -1 and 1 was done with the following formula:

Table 1. The range of variables in the dataset.

Type	Data Name	Minimum	Maximum
Input	Ra	1,000	100,000
	θ_f	0.05	0.95
	ϕ	0	0.05
	Ste	1	5
Output	U	-40.35	40.35
	V	-74.39	74.39
	P	-366527.83	336.08
	θ	0	1

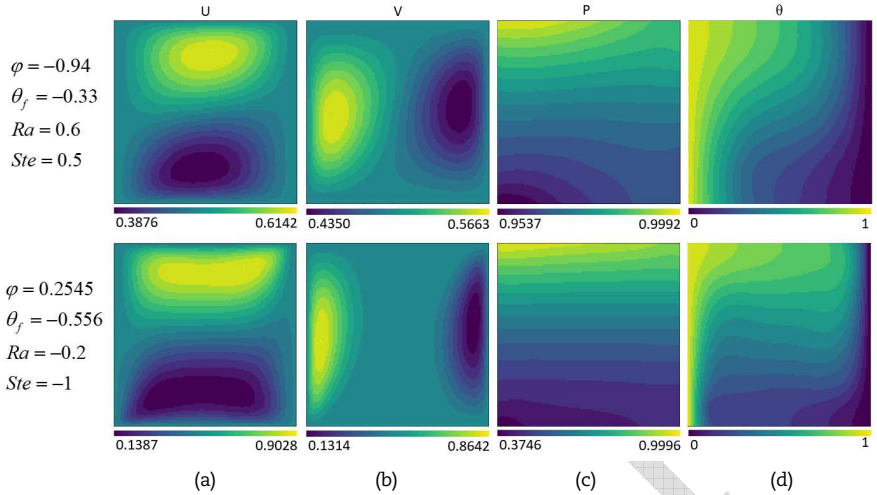

Fig. 1. Some samples of each output image. (a): U image, (b): V image, (c): P image, (d): θ image.

$$\bar{x}_0 = 2 \times \frac{x_0 - \min(X)}{\max(X) - \min(X)} - 1 \tag{1}$$

To normalization output data between 0 to 1 bellow formula was used:

$$\bar{x}_0 = \frac{x_0 - \min(X)}{\max(X) - \min(X)} \tag{2}$$

in which $\bar{x}_0$ is the normal value of a feature, x_0 and X is the array value of a feature that must be normalized. The values of minimum and maximum of all input and output features are also reported in Table 1. From now on, the normalized values of parameters are depicted for convenience. These normalized values could be transformed back to actual values using the above formula and data in Table 1.

The produced dataset is published as a part of the present manuscript and could be accessed using the following address: https://data.mendeley.com/datasets/j5f6r56jnb/draft?preview=1

For the training part, 70% of the dataset is randomly selected as training data, 15% as testing data, and 15% as validation data. Determining the best epoch of the training is the main usage of validation data. They never apply to train DNN and only are used to calculate loss function at the end of each epoch. Thus, they were used for validation loss function. Among all epochs, an epoch with less validation loss error is the best point to save as the ultimate DNN.

2.2 Deep Neural Network (DNN) structure

In the present study, the input parameters are design parameters which are numbers, and the output parameters are *U*, *V*, *P*, and θ distribution, which are images. In the literature works, for a similar situation where the input parameters are numbers and the output parameters are images, a set of *x* and *y* is given to the network as separate inputs. Then, the NN should predict each pixel separately. Then again, a new set of *x* and *y* coordinates will be feed as input to NN, and the predicted value will be generated in the output, and the process will be repeated. Finally, the generated data will be arranged, and the final image will be created. As a result, the prediction process should be repeated for each pixel (point) separately. In such an approach, the correlation between pixels could be ignored, and repeating the process, could be a tedious and time-consuming process.

The other alternative is to produce the image at once, which could improve the prediction capacity of a NN. However, estimating an image at once by using a typical NN requires a full-connection between layers of the NN. Thus, the parameters of NN will be increased drastically, which will not be practical. As a result, the training time, computational cost, and demanded memory could be increases hugely. In the present investigation, a special type of DNN is proposed to estimate an output image at once with a reasonable and practical number of parameters. This proposed DNN benefit an advanced design of DNN. Figure (2) shows the structure of the proposed DNN. This DNN is made of 387,415 parameters. For the sake of comparison, the same DNN with fully connected layers and a hidden layer with 128 neurons could be made of 8,454,784 parameters. Thus, the proposed DNN, depicted in Fig. (2), has 2182% fewer parameters than a typical fully-connected DNN. It should be noted that the fewer parameters, the less required memory, computational cons, training time, and estimation time.

As depicted in Fig. 2, the proposed DNN consists of three parts. The first part is a simple shallow NN with a hidden layer in which the layer is fully-connected (dense). In dense layers, each layer's neuron is connected to all of the previous neurons through weights. Thus, all of the input values could affect the input value of a neuron in the hidden layer. For instance, the output of k^{th} neuron in the layer *l* could be computed using the following relation:

$$y_{l,k} = f\left(\sum_{i=1}^{m} W_{l,i,k} \times y_{l-1,i} + b_{l,k}\right) \tag{3}$$

where m is the number of neurons in layer l-1, and $W_{l,i,k}$ is the weight of i^{th} output neuron in the layer l-1 to k^{th} neuron in layer l. Here, $y_{l-1,i}$ denotes the i^{th} output of l-1 layer, and $b_{l,k}$ indicates the bias for k^{th} neuron in layer l. The bias is typically a constant number that should be adjusted during the training process. Moreover, f represents the neuron's activation function, which is typically a non-linear function. The total output of neurons of a layer can be computed by using matrix computations for convenience. For example, the output of layer l can be computed by the following relation:

$$Y_l = f(W_l.Y_{l-1} + B_l) \tag{4}$$

where W_l represents the weight coefficients matrix between layers l and l-1. In the above relation, Y_{l-1} is the output vector of layer l-1, and B_l denotes the bias coefficients of layer l. The hidden layer of Fig. 2 is made of 128 neurons. The output layer of this part of DNN is consists of 1024 neurons. The hidden layer of this part of the proposed DNN is made of rectified linear unit (Relu) activation functions [25], but the output layer is made of a sigmoid activation function.

The second part of the proposed DNN acts as a transformer. As described, the output of the first part was a vector of size 1024. The second part takes this vector as input and transforms it into a 32×32 matrix. The third part is a convolutional neural network with convolutional layers. In contrast with a dense NN, a neuron in a convolutional layer does not essentially connect to all of the neurons in a previous layer. A convolutional layer has some regular matrixes, so-called filters whose elements must be adjusted by the training process and typically have the size of 3×3 or 5×5. The output of a convolutional layer is calculated by convolving the filters and the inputs of it, while the inputs are input data of the DNN or output of the previous layer.

As illustrated in Fig. 2, before each convolutional layer, two more layers were added, Relu and BN. Relu is an active function that can be replaced with sigmoid, tanh, etc. however, Relu has less computational complexity and more rate of convergence[26]. BN is a symbol for the batch normalization method [27], which enhances the speed, performance, and stability of a DNN. Two sigmoid active-functions are added at the end of part 1 and part 2 of the DNN for the non-linearity aspect. Moreover, range of output data are between zero and one and sigmoid is one of best active function that maps data to this range.

Attention to Fig. 2 shows that the dimension of the input section of part three in the proposed DNN is a 32×32 matrix. However, since the output images are of size 128×128, the output of some layers should be extended to meet the output resolution. Whenever needed, the upsampling was applied before the Batch normalization layer to increase inputs two-fold. The increase of input size was carried out by repeating the input columns and rows.

According to the dataset, the input parameters of the DNN are ϕ, θ_f, Ste, and Ra, which pass through the first and second parts of DNN and create a 2D of 32×32 data. The created data flows through the convolutional layers of the third part and are produced U, V, P, and θ images.

3. Loss function and evaluation parameters

To design a DNN, in addition to its structure, its parameters must be tuned. These parameters are called trainable parameters. The tuning of trainable parameters occurs during a process training process, using an iterable mathematical function so-call optimizer. The optimizer frequently calculates the error of estimation by using a loss function and then changes the trainable parameters in a way to reduce estimation error. Hence, the loss function is an important element of the training process that must be defined correctly. Mean Square Error (MSE) is the most common loss function is used to train neural networks that generate data, as well as to evaluate a trained neural network. Besides MSE, there are some parameters, for instance, MAE, RMSE, etc., to evaluate the results of estimation, which in the continue they are introduced.

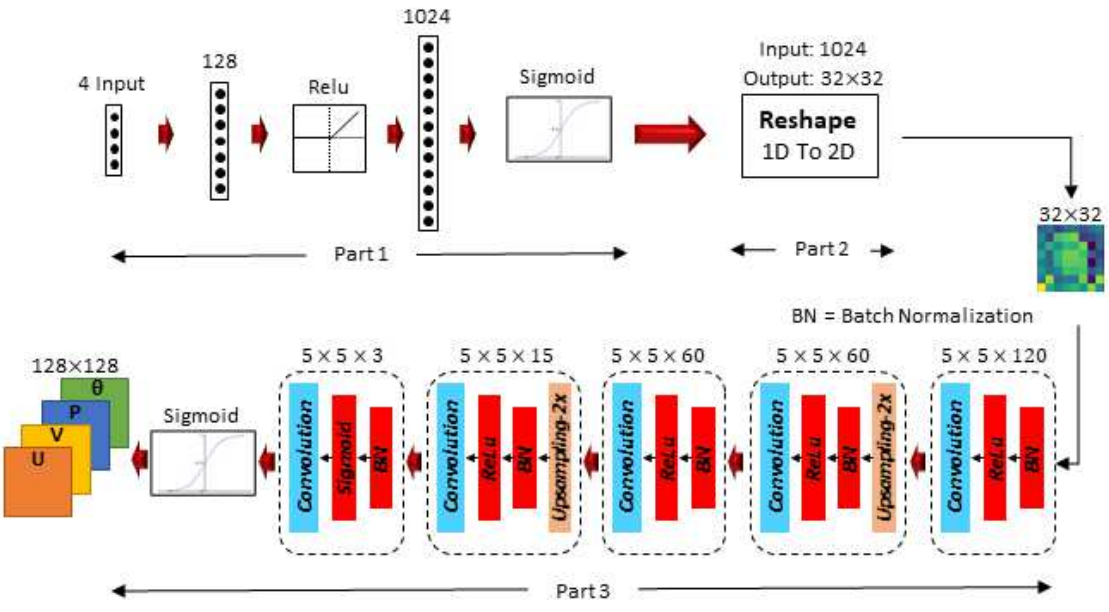

Fig. 2. Structure, details, and characteristics of the DNN used in this paper.

3.1 MSE

Suppose T is an actual target that the neural network must estimate it and P is the predicted value with the neural network. T and P could be only a number or a multi-dimension matrix, for instance, 2D matrix. However, Square Error (SE) calculates as:

$$SE = (T - P) \odot (T - P) \tag{5}$$

That $\odot$ demonstrates element-wise product, and SE is a matrix with dimensions equal to T and P. If neural network estimate only a number, T and P are a number, and eq. 5 could simplify as:

$$SE = (T - P)^2 \tag{6}$$

MSE is the mean of SE's elements. Hence it calculated as:

$$MSE = \text{mean}(SE) = \frac{1}{N}\sum_{i=1}^{N} SE \tag{7}$$

That N is the number of SE's elements. According to Fig. 2, our DNN predicts 4 images of 128×128 pixels. Hence, to predict a batch of the images with the size of B, N is equal to 4×128×128×B.

3.2 Maximum of Square Errors (MaSE)

MaSE is defined as the maximum of the SE elements. Hence, it determines the worst prediction square error. MaSE is calculated as below:

$$MaSE = \max(SE) \tag{8}$$

3.3 Mean Absolute Error (MAE)

Square is a non-linear operator, and it changes the scale of errors non-linearly. Therefore, judging and comparing errors is difficult by MSE. MAE is one of those evaluation parameters that does not change the scale of errors. To calculate MAE, first should be produced Absolute Error (AE) matrix as follow:

$$AE = |T - P| \tag{9}$$

Then, MAE is calculated with average over all AE elements:

$$MAE = \text{mean}(AE) = \frac{1}{N}\sum_{i=1}^{N} AE \tag{10}$$

Comparing eq. 7 and eq. 9 reveal that SE and AE are similar except for square and absolute operation.

3.4 Maximum Absolute Error (MaAE)

MaAE defined as maximum absolute error that exist in AE and calculated as:

$$MaAE = \max(AE) \tag{11}$$

Other evaluation parameters

The results of this paper, in addition to MSE and MAE are investigated with some more evaluation parameters include Root Mean Square Error (RMSE), Coefficient of determination (R2), and Pearson correlation coefficient (r). They are defined as below formulations:

$$RMSE = \sqrt{MSE} \tag{12}$$

$$R^2 = 1 - \frac{\sum_{n=1}^{N}\sum_{r=1}^{R}\sum_{c=1}^{C}(T_{n,r,c} - P_{n,r,c})^2}{\sum_{n=1}^{N}\sum_{r=1}^{R}\sum_{c=1}^{C}(T_{n,r,c} - \overline{T})^2} \tag{13}$$

$$\overline{T} = \text{mean}(T) = \frac{1}{N \times R \times C}\sum_{n=1}^{N}\sum_{r=1}^{R}\sum_{c=1}^{C} T_{n,r,c} \tag{14}$$

$$r = \sqrt{R^2} \tag{15}$$

3.5 Numerical method

NumPy [28], scikit-image[29], and Matplotlib [30] are three main libraries of Python which had been utilized to generate the dataset. NumPy is a powerful python library for mathematical operation on big and multi-dimensional array and matrices. Scikit-image contain main image processing algorithms for python, and Matplotlib is a plotting library. The DNN of this study (Fig. 2) was built in Keras [31] with Google TensorFlow[32] backend. The Adam optimizer [33] was utilized with a fixed learning rate of 0.001, β_1 of 0.9, and β_2 of 0.999. The β_1 and β_2 are the two main parameters of Adam, which for more information, refer to [33]. Before starting the optimization process, all DNN parameters had been initialized with the Glorot uniform technique [51]. Training

a DNN with one sample in a step is time-consuming and it some case cause overfitting, so in this study, the DNN trained with a batch of data consists of 32 samples instead of one sample. To investigate the training process, a history of variation of validation parameters during the training process is needed. Thus, they were calculated for training and validation data at the end of each epoch and saved as the history of training.

The variation of loss error throughout the training is one of the main points that must be considered. The loss error may increase or decrease during the training process. Hence, after each epoch loss error was calculated for validation data, loss validation error, and at the end of the training, DNN was saved as the ultimate trained DNN at an epoch that has minimal loss validation error.

4. Results and discussion

In this paper, the DNN illustrated in Fig. 2 is used to estimate U, V, P, and θ, which for simplifying refer to it as Net_DNN. A feed-forward neural network with one hidden layer, shown in Fig. 3, and called Net_NN, is designed as well to compare their result with Net_DNN. All layers of Net_NN are full-connection, and it has 8,454,784 trainable parameters. These numbers of parameters are a huge value in comparison to Net_DNN that has only 387,415 trainable parameters. Both of them in a similar condition were trained with MSE loss function for 2,000 epochs. To preventing overfitting, each network was saved in an epoch with the lowest MSE of validation data.

The evaluation parameters for estimated data with Net_NN and Net_DNN are presented in Table 2. This table shows that Net_DNN had better performance and predicted data with lower error than Net_NN, it is while Net_NN has a huge number of training parameters about 2182% more than Net_DNN. Hence, Net_DNN, with their filters, could extract features of neighbor pixels and estimated data with acceptable accuracy better than Net_NN.

In Table 2, the evaluation parameters were calculated for U, V, P, and ϑ images together. However, in order to investigate more details of estimations, in the continue of the paper, the evaluation parameters will calculate for U, V, ϑ, and P separately. Tables 3 to 5 show evaluation parameters of U, V, P, and ϑ when estimated with Net_DNN and Net_NN. To produce these tables first, the networks estimated U, V, P, and ϑ together, then for each of them, the values of tables were calculated. The values of these tables revealed that Net_DNN was more successful than Net_NN to estimate U, V, and specially ϑ. Only, P has been estimated with Net_NN a bit small better than Net_DNN. This small difference can be ignored because of the small number of Net_DNN's trainable parameters and its accuracy to estimate U, V, and ϑ.

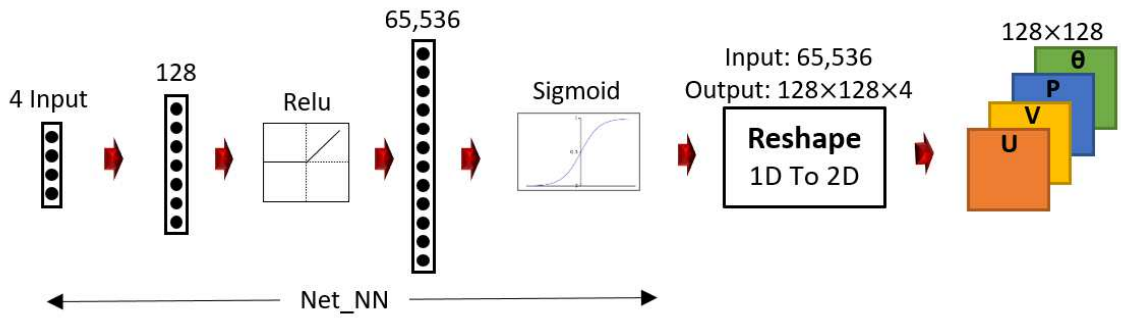

Fig. 3. Structure and characteristics of the simple feed-forward neural network with one hidden layer.

Table 2. The evaluation parameters for estimated data with Net_NN and Net_DNN. These parameters are calculated for U, V, P, and θ together.

Data type	DDN type	MSE	RMSE	MAE	MaAE	R^2	r
Train	Net_DNN	2.8538e-6	0.00169	0.00132	0.0768	0.9999332	0.999967
	Net_NN	1.1621e-5	0.00341	0.00174	0.0824	0.999798	0.999899
Validation	Net_DNN	4.1962e-6	0.00205	0.00136	0.0713	0.9999280	0.999964
	Net_NN	1.3444e-5	0.003667	0.00185	0.0827	0.999769	0.999885
Test	Net_DNN	4.492e-6	0.00212	0.00142	0.0802	0.9999218	0.999961
	Net_NN	1.2780e-5	0.00357	0.00186	0.0856	0.999777	0.99989

Table 3. The evaluation parameters when are calculated only for U images.

Data type	DDN type	MSE	RMSE	MAE	MaAE	R^2	r
Train	Net_CNN	3.0988e-6	0.00176	0.00127	0.0293	0.999826	0.9999132
	Net_NN	9.726e-6	0.00312	0.00174	0.0803	0.999455	0.999728
Validation	Net_DNN	3.4696e-6	0.00186	0.00131	0.0346	0.999809	0.9999042
	Net_NN	1.1856e-5	0.00344	0.001889	0.0902	0.999346	0.999673
Test	Net_DNN	3.666e-6	0.00191	0.001364	0.025	0.999826	0.9999130
	Net_NN	1.077e-5	0.00328	0.00191	0.0407	0.999489	0.99974

Table 4. The evaluation parameters when are calculated only for V images.

Data type	DDN type	MSE	RMSE	MAE	MaAE	R^2	r
Train	Net_DNN	2.351e-6	0.00176	0.00102	0.0424	0.999609	0.999805
	Net_NN	3.1757e-6	0.00311	0.000934	0.0599	0.999472	0.999736
Validation	Net_DNN	2.439e-6	0.00156	0.00103	0.0424	0.999605	0.999802
	Net_NN	3.838e-6	0.00196	0.000998	0.0661	0.999378	0.999689
Test	Net_DNN	2.88e-6	0.0017	0.00109	0.0424	0.999602	0.99980
	Net_NN	3.448e-6	0.00186	0.00102	0.0401	0.999523	0.999761

Table 5. The evaluation parameters when are calculated only for P images.

Data type	DDN type	MSE	RMSE	MAE	MaAE	R^2	r
Train	Net_DNN	4.0933e-6	0.00202	0.00134	0.08643	0.999883	0.9999419
	Net_NN	1.5908e-6	0.00126	0.000979	0.0177	0.9999549	0.9999774
Validation	Net_DNN	4.189e-6	0.00205	0.00136	0.08564	0.999894	0.999947
	Net_NN	1.802e-6	0.00134	0.00105	0.0152	0.9999544	0.9999772
Test	Net_DNN	4.725e-6	0.00217	0.00144	0.0866	0.9999893	0.9999467
	Net_NN	1.746e-6	0.00132	0.00101	0.0268	0.9999606	0.9999803

Table 6. The evaluation parameters when are calculated only for θ images.

Data type	DDN type	MSE	RMSE	MAE	MaAE	R^2	r
Train	Net_DNN	5.8715e-6	0.00242	0.001675	0.0941	0.9999183	0.9999592
	Net_NN	3.199e-5	0.00566	0.0033	0.1650	0.999555	0.999777
Validation	Net_DNN	6.687e-6	0.00259	0.00175	0.0521	0.9999074	0.9999537
	Net_NN	3.628e-5	0.00602	0.00348	0.1154	0.999498	0.999749
Test	Net_DNN	6.6976e-6	0.00259	0.00177	0.0491	0.9999056	0.9999528
	Net_NN	3.5156e-5	0.00593	0.00351	0.1095	0.999505	0.999752

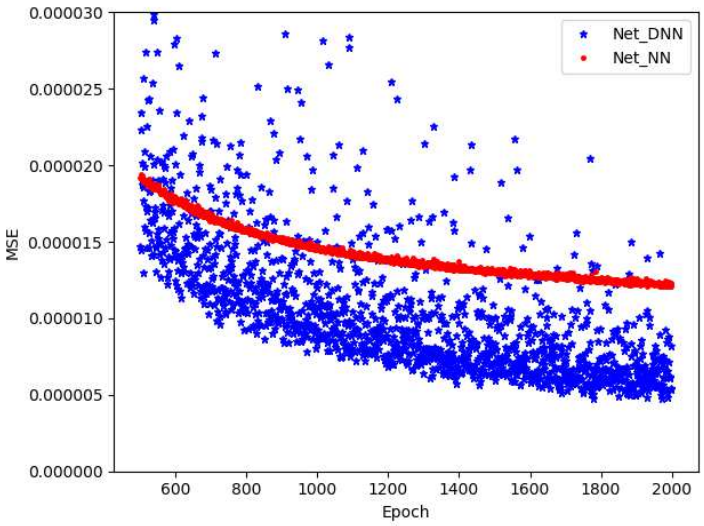

Fig. 4. The variation of MSE during epoch 500 to 2000.

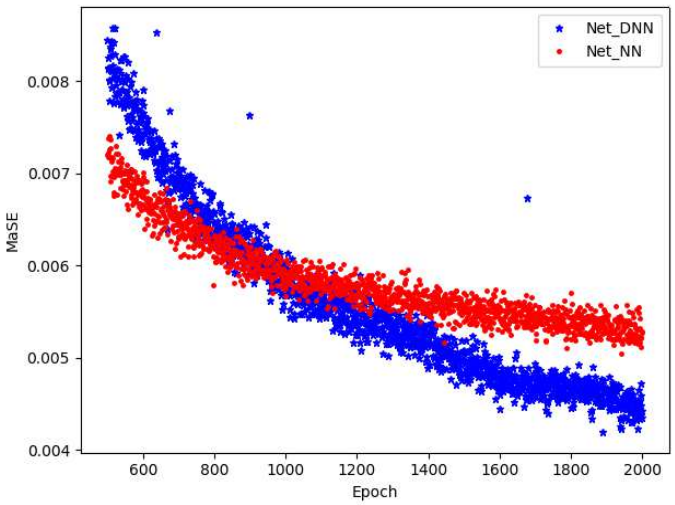

Fig. 5. The variation of MaSE during epoch 500 to 2000

The speed of reducing errors during the training process is one of the valuable parameters to compare multi neural networks. Figure 4 and 5 illustrate the variation of MSE and MaSE during the training process from epoch 500 to 2000, in which the first 500 epochs were removed to allow a better plot and representation of results. Figure 4 shows although MSE of Net_DNN has more variation during the process, in most epochs, it achieves a value lower value than Net_NN. The distance between charts that show higher accuracy of Net_DNN is considerable, too. The slop of charts on Fig. 4 is another attractive point that reveals a probability of achieving lower error and increasing the difference between charts with Net_DNN if continuing the training process.

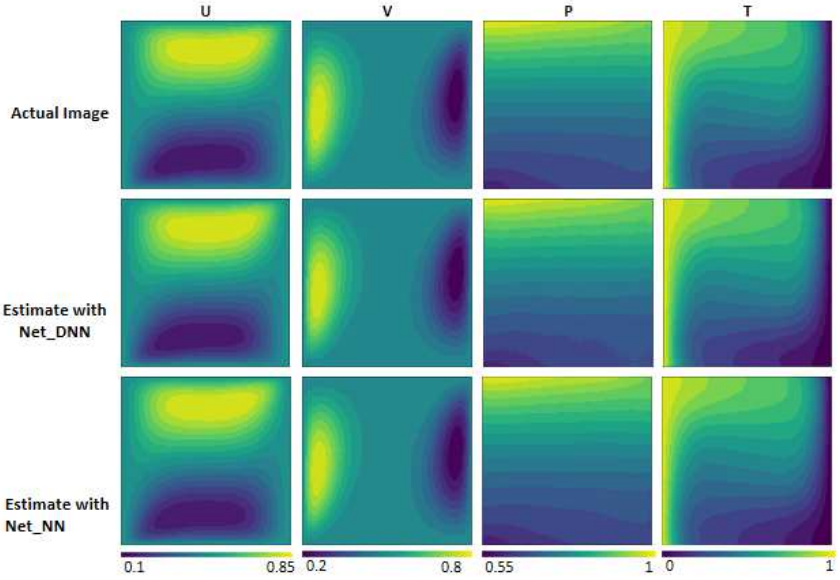

Fig. 6. One sample of test images estimated with Net_DNN and Net_NN.

The trends of charts in Fig. (5) are somewhat different than Fig. 4. Figure (5) show that Net_NN had lower MaSE before epoch 1000. However, after this epoch, the chart of Net_DNN has been reduced rapidly, and this reduction has continued until the last epoch. Similar to Fig. 4, the slope of Net_DNN's chart on Fig. 5 is more than Net_NN, so there is a probability of achieving lower MaSE with more difference between charts the training continues; however, the computational cost could be not feasible.

Figure 6 illustrates some samples of estimated images with Net_DNN and Net_NN. The first row belongs to actual images that exist in the dataset. The second and third rows correspond to images of row one that estimated with Net_DNN and Net_NN, respectively.

5. Conclusion

In this paper, a deep neural network was utilized to learn the complex physics of natural convection heat transfer of a NEPCM-suspension. The trained DNN was used to estimate the velocity and temperature fields with no iterations directly. In the first step, a dataset was produced with 3,290 samples includes the volume fraction of nanoparticles ϕ, non-dimensional phase change temperature θ_f, Stefan number (Ste), and Rayleigh number (Ra) as input images and velocity in the x-direction (U), velocity in the y-direction (V), pressure filed (P), and temperature distribution (θ) as output images. Next, a new DNN model (Net_DNN) was designed and trained with the dataset. A feed-forward neural network with one hidden layer (Net_NN) was also trained with the dataset was used to learn the convective heat transfer behavior of nano-encapsulated phase change material. The results showed that while Net_DNN has far fewer parameters than Net_NN, it estimates the images more accurately. The speed of decrease error during the training process with Net_DNN was also significant in comparison to Net_NN. The outcome of this research could be summarized as follow:

1. A hybrid DNN consisting of a feed-forward neural network and a convolutional neural network could reduce training parameters in comparison to a feed-forward neural network tremendously without losing the accuracy.
2. In most literature, pixels of the physical images are estimated one by one. It is a tedious process and causes the relationship between neighbor pixels was gone. The results showed Net_DNN could estimate images at once with more accuracy than Net_NN.
3. The Net_DNN, compared to Net_NN, could reduce errors faster. Moreover, Net_DNN could fairly and robustly learn the complex physics of natural convection heat transfer of a NEPCM-suspension.

This study demonstrated the capability of DNNs to learn the physics of phenomena and increase accuracy compared to traditional feed-forward neural networks. Hence, utilizing new DNNs to simulate new physical phenomena can be the subject of future studies.

Author Contributions

M. Edalatifar: Conceptualization, Methodology, Validation, Formal analysis, Data Curation; M. B. Tavakoli: Conceptualization, Methodology, Writing- Original draft preparation, Writing - Review & Editing, Supervision, F. Setoudeh: Conceptualization, Methodology, Writing- Original draft preparation, Writing - Review & Editing.

Acknowledgments

Authors are grateful to Arak Branch, Islamic Azad University, Arak, Iran for its support of the project.

Conflict of Interest

There is no conflict of interest to report.

Funding

The authors received no financial support for the research, authorship, and publication of this article.

References

1. Izadi S., Armaghani T., Ghasemiasl R., Chamkha A.J., Molana M. A comprehensive review on mixed convection of nanofluids in various shapes of enclosures, *Powder Technology*, 2019, 343, 880-907.
2. Sajid M.U., Ali H.M. Recent advances in application of nanofluids in heat transfer devices: a critical review, *Renewable and Sustainable Energy Reviews*, 2019, 103, 556-92.
3. Sadeghi H.M., Babayan M., Chamkha A. Investigation of using multi-layer PCMs in the tubular heat exchanger with periodic heat transfer boundary condition, *International Journal of Heat and Mass Transfer,* 2020, 147, 118970.
4. Tayebi T., Chamkha A.J. Magnetohydrodynamic natural convection heat transfer of hybrid nanofluid in a square enclosure in the presence of a wavy circular conductive cylinder, *Journal of Thermal Science and Engineering Applications*, 2020, 12(3), 031009.
5. Selimefendigil F., Öztop H.F., Chamkha A.J. Role of magnetic field on forced convection of nanofluid in a branching channel, *International Journal of Numerical Methods for Heat & Fluid Flow*, 2020, 30(4), 1755-1772.
6. Sheremet M.A., Öztop H.F., Abu-Hamdeh N. Thermogravitational convection of Al2O3-H2O nanoliquid in a square chamber with intermittent blocks, *International Journal of Numerical Methods for Heat & Fluid Flow*, 2020, 30(3), 1365-1378.
7. Bondarenko D.S., Sheremet M.A., Oztop H.F., Ali M.E. Natural convection of Al2O3/H2O nanofluid in a cavity with a heat-generating element, Heatline visualization, *International Journal of Heat and Mass Transfer,* 2019, 130, 564-74.
8. Alsabery A.I., Ismael M.A., Chamkha A.J., Hashim I. Impact of finite wavy wall thickness on entropy generation and natural convection of nanofluid in cavity partially filled with non-Darcy porous layer, *Neural Computing and Applications*, 2020, 1-21.
9. Dogonchi A., Sheremet M.A., Ganji D., Pop I. Free convection of copper–water nanofluid in a porous gap between hot rectangular cylinder and cold circular cylinder under the effect of inclined magnetic field, *Journal of Thermal Analysis and Calorimetry,* 2019, 135(2), 1171-84.
10. Selimefendigil F., Chamkha A.J. MHD mixed convection of nanofluid in a three-dimensional vented cavity with surface corrugation and inner rotating cylinder, *International Journal of Numerical Methods for Heat & Fluid Flow*, 2020, 30(4), 1637-1660.
11. Selimefendigil F., Oztop H.F., Sheremet M.A., Abu-Hamdeh N. Forced convection of Fe3O4-water nanofluid in a bifurcating channel under the effect of variable magnetic field, *Energies*, 2019, 12(4), 666.
12. Molana M., Dogonchi A., Armaghani T., Chamkha A.J., Ganji D., Tlili I. Investigation of hydrothermal behavior of Fe3O4-H2O nanofluid natural convection in a novel shape of porous cavity subjected to magnetic field dependent (MFD) viscosity, *Journal of Energy Storage,* 2020, 30, 101395.
13. Selimefendigil F., Chamkha A.J. MHD mixed convection of Ag–MgO/water nanofluid in a triangular shape partitioned lid-driven square cavity involving a porous compound, *Journal of Thermal Analysis and Calorimetry,* 2020, 1-18.
14. Ghalambaz M., Chamkha A.J., Wen D. Natural convective flow and heat transfer of nano-encapsulated phase change materials (NEPCMs) in a cavity, *International Journal of Heat and Mass Transfer,* 2019, 138, 738-49.
15. Håstad J., Goldmann M. On the power of small-depth threshold circuits, *Computational Complexity,* 1991, 1(2), 113-29.
16. Farimani A.B., Gomes J., Pande V.S. Deep learning the physics of transport phenomena, *arXiv preprint,* arXiv: 170902432, 2017.
17. Seyfi B., Rassoli A., Imeni M., Fatouraee N. Characterization of the Nonlinear Biaxial Mechanical Behavior of Human Ureter Using Constitutive Modeling and Artificial Neural Networks, *Journal of Applied and Computational Mechanics,* 2020, doi: 10.22055/JACM.2020.33703.2272.
18. Alzghoul A., Jarndal A., Alsyouf I., Bingamil A.A., Ali M.A., Albaiti S. On the Usefulness of Pre-processing Methods in Rotating Machines Faults Classification using Artificial Neural Network, *Journal of Applied and Computational Mechanics,* 2021, 7(1), 254-261.
19. Taheri M.H., Askari N., Mahdavi M.H. Prediction of entrance length for magnetohydrodynamics channels flow using numerical simulation and artificial neural network, *Journal of Applied and Computational Mechanics*, 2020, 6(3), 582-92.
20. Ermis K., Erek A., Dincer I. Heat transfer analysis of phase change process in a finned-tube thermal energy storage system using artificial neural network, *International Journal of Heat and Mass Transfer*, 2007, 50(15), 3163-75.
21. Azwadi C.S.N., Zeinali M., Safdari A., Kazemi A. Adaptive-Network-Based Fuzzy Inference System Analysis to Predict the Temperature and Flow Fields in a Lid-Driven Cavity, *Numerical Heat Transfer, Part A: Applications*, 2013, 63(12), 906-20.
22. Selimefendigil F., Akbulut Y., Sengur A., Oztop H.F. MHD conjugate natural convection in a porous cavity involving a curved conductive partition and estimations by using Long Short-Term Memory Networks, *Journal of Thermal Analysis and Calorimetry*, 2020, 140(3), 1457-68.
23. Zhou S., Liu X., Du G., Liu C., Zhou Y. Comparison study of CFD and artificial neural networks in predicting temperature fields induced by natural convention in a square enclosure, *Thermal Science*, 2019, 23, 3481-92.
24. Akbari E., Karami A., Nazari S., Ashjaee M. An intelligent integrated approach of Jaya optimization algorithm and neuro-fuzzy network to model the natural convection in an open round cavity, *International Journal of Modelling and Simulation*, 2020, 40(2), 87-103.
25. Agarap A.F. Deep learning using rectified linear units (relu), *arXiv preprint*, arXiv: 180308375, 2018.
26. Krizhevsky A., Sutskever I., Hinton G.E., editors. *Imagenet classification with deep convolutional neural networks*, Advances in neural information processing systems, 2012.
27. Ioffe S., Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift, *arXiv preprint*, arXiv: 150203167, 2015.
28. Oliphant T.E. *A guide to NumPy*, Trelgol Publishing, USA, 2006.
29. Van Der Walt S., Schönberger J., Nunez-Iglesias J., Boulogne F., Warner J., Yager N., et al. scikit-image: image processing in Python, *PeerJ*, 2014, 2, 453.
30. Hunter J.D. Matplotlib: A 2D graphics environment, *Computing in Science & Engineering*, 2007, 9(3), 90.
31. Chollet F., Others. Keras. https://keras.io; 2015.
32. Martín A., Ashish A., Barham P., Brevdo E, Chen Z., Citro C., et al. *TensorFlow : Large-Scale Machine Learning on Heterogeneous Systems*, 2015.
33. Kingma D.P., Ba J. *Adam: A method for stochastic optimization*, *arXiv preprint*, arXiv: 14126980, 2014.

ORCID iD

Mohammad Edalatifar https://orcid.org/0000-0003-4610-9522
Mohammad Bagher Tavakoli https://orcid.org/0000-0001-7829-7885
Farbod Setoudeh http://orcid.org/0000-0001-5866-756X

How to cite this article: Edalatifar M., Tavakoli M.B., Setoudeh F. A Deep Learning Approach to Predict the Flow Field and Thermal Patterns of Nonencapsulated Phase Change Materials Suspensions in an Enclosure, *J. Appl. Comput. Mech.*, xx(x), 2021, 1–9. https://doi.org/10.22055/JACM.2021.37805.3092