A transfer learning metamodel using artificial neural networks
applied to natural convection flows in enclosures

Majid Ashouri¹, Alireza Hashemi²

¹College of Engineering, Boston University, Boston, MA 02215, USA. ashouri@bu.edu
²Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, MA 02134, USA. ahashemi@mit.edu

Abstract

In this paper, we employed a transfer learning technique to predict the Nusselt number for natural convection flows in enclosures. Specifically, we numerically simulated a benchmark problem in square enclosures described by the Rayleigh and Prandtl numbers using the finite volume method. Given that the ideal grid size depends on the value of these parameters, we performed our simulations using a combination of different grid systems. This allowed us to train an artificial neural network in a cost-effective manner. We adopted two approaches to this problem. First, we generated a multi-grid training dataset that included both the Rayleigh and Prandtl numbers as input variables. By monitoring the training losses for this dataset, we were able to detect any significant anomalies that stemmed from an insufficient grid size. We then revised the grid size or added more data points to denoise the dataset and transferred the learning from our original dataset to build a computational metamodel that predicts the Nusselt number. Furthermore, we sought to endow our neural network model with the ability to account for additional input features. Therefore, in our second approach, we applied a deep neural network architecture for transfer learning to this problem. Initially, we trained a neural network with a single input feature (Rayleigh), and then, extended the network to incorporate the effects of a second feature (Prandtl). This learning framework can be applied to other systems of natural convection in enclosures that presumably have higher physical complexity, while bringing the computational and training costs down.

Keywords: Artificial neural networks; deep learning; machine learning; metamodel; natural convection; transfer learning.

1. Introduction

An exhaustive metamodel using machine learning and particularly artificial neural networks (ANNs) can be utilized to characterize the flow and heat transfer behaviors in an engineering system [1-10]. Data-driven ANN metamodels are especially useful for predicting the heating or cooling processes in natural convection. Natural
convection starts when temperature gradients, and subsequently, density differences in a fluid induce buoyancy effects. This phenomenon has applications in various engineering systems such as nuclear reactors, heat exchangers, solar energy collectors, and electronic devices. The strength of natural convection can be quantified by the Nusselt number (Nu). Several ANN metamodels for predicting Nu in natural convection systems have been investigated using numerical simulations [11-14] or by conducting experiments [14-18].

As the complexity of a physical system increases, the training of an ANN becomes more challenging. Given that creating a suitable training dataset may become laborious and expensive, a compromise between cost and accuracy should be made. Transfer learning (TL) can be employed to mitigate this issue by using knowledge from a source domain to improve the learning of a target domain with limited data. One approach to TL is to use multi-fidelity training in which inexpensive low-fidelity data are used to map the relationship between features and targets. Accuracy can be improved by applying higher-fidelity training data to the low-fidelity model. This method was shown to effectively reduce training cost [19-27].

One can conceive of a few case examples for the application of TL in heat transfer systems: The need for TL may arise when the performance of a heat exchanger changes due to fouling over time. A TL methodology can also be applied to build a semi-empirical metamodel using available simulation data and limited experimental data. As another example, the information from a low-cost two-dimensional metamodel can be transferred to account for three-dimensional effects using limited three-dimensional data. TL especially is beneficial for a complex model through domain adaptation [28]. Where the differences between two heat transfer systems are identified, information from one system can be transferred to the other. For example, heat transfer from vertical and horizontal walls can be related to heat transfer from an enclosure.

The key parameter to characterize a natural convection problem is the Rayleigh number (Ra). This parameter is defined as the product of the ratio of buoyancy to viscous forces (which is equivalent to the Grashof number), and the ratio of momentum and thermal diffusivities (the Prandtl number, Pr). However, various expected and unexpected factors can affect a natural convection system. Some features may become active under different conditions, or after a system is redesigned. For example, the system orientation with respect to gravity or the system boundary conditions may change. Or a magnetic force applied to a natural convection system will lead to thermomagnetic [29,30] or magnetohydrodynamic [31,32] effects. Moreover, generating a new set of simulations or experiments is laborious. Domain adaptation or TL with deep neural networks (DNN) can be employed to remedy this problem, provided that the differences between the source and target can be addressed.

The objective of the present study is to develop a TL framework for building metamodels for the prediction of Nu in natural convection flows in enclosures. First, we show that it is possible to exploit an imperfect dataset that was generated by cost-effective simulations using different grid systems. Second, we developed a DNN architecture for transfer learning. We consider the benchmark problem of an air-filled enclosure and build a Nu metamodel. We demonstrate how its learning can be transferred to an enclosure with arbitrary fluid (i.e.,
incorporating \( \text{Pr} \). While we considered two input parameters, namely, \( \text{Ra} \) and \( \text{Pr} \), the present TL framework can also be used for more complex natural convection problems.

2. Problem Description

We aimed to extract a metamode out of a physical model that numerically predicts the natural convection characteristics in a square enclosure, filled with a Newtonian fluid. This problem is governed by two parameters: \( \text{Ra} \) and \( \text{Pr} \) (see Appendix A for details about the mathematical analysis). We consider \( \text{Ra} \) of up to \( 10^8 \) and \( \text{Pr} \) of greater than 0.05 (\( 0 < \text{Ra} \leq 10^8 \) and \( 0.05 \leq \text{Pr} < \infty \)); however, lower \( \text{Pr} \) were also considered provided that the ratio of \( \text{Ra}/\text{Pr} \) is at most less than \( 10^6 \).

A 400×400 grid system was shown to provide precise results for the average \( \text{Nu} \) even for the most stringent cases. Appendix B includes details related to the numerical method and grid independence test. Using a single logical processor on a 2.6 GHz Intel Core i7-3720QM CPU, an average computational time of about 4,850 seconds (as high as about 13,000 seconds for low \( \text{Pr} \)) was spent for obtaining the numerical solutions using a 400×400 grid system. Nonetheless, as demonstrated in Appendix C, lower grid systems can provide accurate numerical solutions for limited ranges of \( \text{Ra} \). For example, a 200×200 grid system (with an average simulation time of 1,300 seconds) can reliably be used for \( \text{Ra} \) of up to \( 10^7 \) with errors of less than 0.5%. Therefore, we consider a multi-grid simulation that also uses lower grid systems, wherever possible, to decrease the simulation cost in training our AI model.

3. Results and Discussion

The multi-grid dataset that we used in our training is shown in the scatter graph of Fig. 1a. This dataset includes a limited number of simulation data using the 400×400 grid system (5% of data) for cases with high \( \text{Ra} \) or low \( \text{Pr} \). The rest of this dataset includes numerical results using 200×200 (9%), 50×50 (15%), and 25×25 grids (71%), for \( \text{Ra} \) within the range of \( 10^5 \leq \text{Ra} \leq 10^7 \), \( 10^3 \leq \text{Ra} < 10^5 \), and \( 0 < \text{Ra} \leq 10^3 \), respectively. Nonetheless, we included simulations of higher fidelity solutions that were carried out as part of the analysis in Appendix C. As can be seen in Fig. 1a, the low-cost simulations using 25×25 grid systems (with an average computational time of about 40 seconds) allowed us to generate more data in the low \( \text{Ra} \) region to capture the nonlinear variation of \( \text{Nu} \) for \( \text{Ra} \approx 10^3 \). In contrast, for higher \( \text{Ra} \), \( \text{Nu} \) varies in a logarithmically linear manner with \( \text{Ra} \) (Fig. C-Ia).

An optimized ANN as described in Appendix D was used to make a metamodel for predicting \( \text{Nu} \). We trained our ANN using 480 data points (in which 15% of the dataset was considered for validation during training) according to the details presented in Table 1 under “step 1". The contour graph for the relative errors associated with our training and validation datasets is presented in Fig. 1b.
Fig. 1. a) The scatter graph for the training dataset generated by numerical simulations using a combination of 25×25, 50×50, 200×200, and 400×400 grid systems. b) the error contour for the training dataset.

The error contour in Fig. 1b can be used to check if the grid system for each data point was properly selected (other than the analysis of Appendix C). We assumed that the source to any irregularity in the error values in Fig. 1b was due to a lack of sufficient data points or possibly an inconsistency between the results of different grid systems. Considering these two points, we revise our training dataset as illustrated in the scatter graph of Fig. 2a. As can be ascertained, we added more data at the regions with high Ra or low Pr. We also replaced some of the data with higher fidelity simulation results. This process could also be done using an over-complete autoencoder. We then fed the new dataset to our previously trained ANN to achieve a better validation loss (by 48%) as summarized in Table 1 under “step 2”. The contour of the relative error for the training and validation datasets is shown in Fig. 2b.

Table 1. The details of the ANN training for the prediction of Nu.

| Step No. | 25² | 50² | 200² | 400² | Training data | Training error | Validation error | Model cost |
|----------|-----|-----|------|------|---------------|----------------|-----------------|------------|
|          |     |     |      |      | MSE (10⁻⁶) | MAE (10⁻⁶) | MSE (10⁻⁶) | MAE (10⁻⁶) | ST (hr) | NNT (hr) |
| 1        | 340 | 73  | 43   | 24   | 1.12 | 0.64 | 1.77 | 0.86 | 53.7 | 2.0 |
| 2        | 339 | 68  | 65   | 36   | 1.03 | 0.60 | 0.93 | 0.57 | 79.6 | 0.8 |

MSE: mean squared error loss; MAE: mean absolute error; ST: simulation time; NNT: ANN training time.

* Also includes the simulation time for the data that was removed from the original dataset.
Fig. 2. a) The scatter graph for the training dataset that was revised based on the training error by either replacing some data points with data generated using higher fidelity models or adding more data points. b) the relative error contour for the training dataset. The line contour points to the areas with errors greater than 0.4%.

We tested our ANN metamodel using a test dataset of 100 simulations using a 400×400 grid system. The test points were selected randomly and transformed to normal distributions of log(Ra) = 4.8 ± 2.1 and log(Pr) = 0 ± 1.7. A scatter graph for the test dataset is presented in Fig. 3a. The test result for our ANN model is summarized in Table 2. Comparing the test loss between the two steps shows an improvement of 49% at an added simulation cost of 46%. The test error contour for the final metamodel is presented in Fig. 3b. This metamodel predicts Nu with an error of 0.22 ± 0.21 % (the first and second terms are the mean and standard deviation of the relative errors, respectively). In Fig. 3b, the highest errors are for low Pr. Then again, based on the error contour of Fig. 2b one can revise the training dataset to achieve a higher accuracy. For example, we could enhance the training dataset by replacing data in regions above the 0.4% error lines in Fig. 2b.

| Step No. | MSE (/10^{-6}) | MAE (/10^{-3}) | MRE (%) | SD (%) |
|----------|----------------|----------------|---------|--------|
| 1        | 3.48           | 1.19           | 0.27    | 0.33   |
| 2        | 1.77           | 0.96           | 0.22    | 0.21   |

MSE: mean squared error loss; MAE: mean absolute error; MRE: mean relative error; SD: standard deviation of relative error.
Fig. 3. a) The scatter graph for the test dataset generated by numerical simulations using 400×400 grids. b) The relative error contour for the test dataset. The highest relative error is 0.97%.

As an alternative to the above approach, we developed another TL framework that enables us to incorporate any potential features. We started with constructing a metamodel based on one input (in this case, Ra) and later added other input parameters (namely, Pr).

For simplicity, we only used 200×200 and 400×400 grid systems for our simulations. We initially generated a metamodel that predicts the variation of Nu with Ra for an air-filled cavity (Pr = 0.71). We trained an ANN as illustrated in Fig. 4 (as block I) using 30 data points (ranging from Ra = 1 to 2 × 10^8), without using a validation dataset during the training (with the purpose of lowering the simulation cost). The results of testing our Ra-Nu metamodel using 10 test data points is presented in Table 3 under Step 1.

We then extended our metamodel to also consider the effects of Pr. We applied the same structure as Ra (the right branch in Fig. 4). We then merged the outputs of the Pr branch and block I using a Multiply layer (Keras API). We added a one-node layer after the Multiply layer to adjust for the multiplication coefficient. We froze the layers on block I, and subsequently, trained the new hidden layers using the data points from Step 1 as well as a new dataset of constant Ra simulation points (24 training data at a fixed Ra = 10^5 and variable Pr ranging from Pr = 10^{-3} to 10^5). We trained the ANN using 54 data points and validated it using 20 data points (10 data points of Pr-constant and 10 data points of Ra-constant). The test results (using the dataset of Fig. 3a) are presented in Table 3 under Step 2.
Fig. 4. The structure of the DNN for transfer learning. Block I builds a metamodel that predicts the variation of $\text{Nu}$ with $\text{Ra}$ for cases where $\text{Pr}$ is unchanged. In Block II, there is a second branch that takes $\text{Pr}$ into account as a new input. The outputs of the two branches are Multiplied to produce the output of Block II. In Block III, the outputs are concatenated and two hidden layers are added to improve the training.
Table 3. The details of the ANN training for the prediction of Nu.

| Step No. | Training data | ANN parameters | Test error | Model cost |
|----------|---------------|----------------|------------|------------|
|          | 2002          | 4002           |            |            |
| 1: Block I | 23 7          | 377 Non-trainable | 0.01 0.30 0.07±0.04 | 19.2 0.2 |
| 2: Block II | 42 32        | 379 377        | 22.2 9.35 2.12±2.55 | 42.7 0.8 |
| 3: Block III | 64 53       | 381 754        | 2.39 3.08 0.71±0.89 | 75.7 3.6 |

*Also includes the simulation time for the test data that was generated for this step.

**Mean ± standard deviation of relative error.

To increase the accuracy, one can remove the Multiply layer from block II and replace it with a concatenated layer followed by two additional hidden layers (block III in Fig. 4). We froze the training on the two branches and transferred the learning from step 2. We trained the new hidden layers using new simulation points at fixed $Ra = 10^8$ or at fixed $Pr = 0.05$ on top of the previous dataset. As presented in Table 3 (Step 3), our results on the test dataset remarkably improved after this step. The scatter graph for different training datasets used in this approach is presented in Fig. 5a. The test error contour is shown in Fig. 5b (for Step 3). Further improvement in the metamodel accuracy is possible after retraining the DNN from step 3 using more data. For example, using additional data points (shown by grey symbols in Fig. 5a) decreased the test loss by 51% (RE = 0.45±0.66) with an increased simulation cost of 28%.

Fig. 5. a) The scatter graph for the training datasets for the different steps described in Table 3. b) the relative error contour for the test dataset applied to the final metamodel.
4. Conclusion

While buoyancy is what drives natural convection, the performance of a natural convection system can be affected by other physical phenomena and different factors such as geometry, boundary conditions, and the behavior of a fluid. Data-driven metamodels governing real-world natural convection systems require datasets that cover the entire feature space. Moreover, some unforeseen features may become active under a different process, or after a system is redesigned. Among other limiting factors, generating a new full set of simulations or experiments is time-consuming. Our methodology using TL with DNN can flexibly adapt to the expansion of the feature space when a natural convection system becomes more complicated and needs to be described more precisely.

We considered the benchmark problem of a two-dimensional enclosure with the horizontal walls being isolated, and the vertical walls being at constant temperatures. We carried out a mesh refinement study on the input space to find the appropriate grid size to perform accurate simulations. Although simulations using coarse grid systems may provide precise results, they only do so for a limited range of input parameters. For example, the higher the Ra the finer the grid size required. As such, we utilized a multi-grid dataset to train our ANN in order to reduce simulation time. Any irregularity in the training loss could be an indication of inconsistency in the dataset due to grid error. We effectively denoised the dataset, and retrained the ANN, based on abnormalities observed in the training losses.

Secondly, we adopted a TL approach using DNN. We demonstrated the capability of this approach to incorporate any additional input features. We built a metamodel to predict Nu as a function of Ra (the only input variable), for an air-filled enclosure. We then successfully applied a DNN and transferred the learning of an air-filled enclosure to an enclosure with arbitrary fluid (i.e., Pr was added as a new input). Therefore, we have shown that this metamodel can be retrained to predict Nu in different natural convection problems. Furthermore, this TL strategy is versatile and can handle straightforward metamodeling for different engineering systems.

Appendix A: Mathematical Analysis for the Physical Model

We modeled the natural convection in a square enclosure of width $D$ filled with a Newtonian fluid. The top and the bottom walls of the enclosure are adiabatic, and the left and right vertical walls are kept at a constant hot ($T_h$) and cold ($T_c$) temperature, respectively. The flow is considered laminar, incompressible and two-dimensional. Constant thermal properties are assumed except for density which is modeled by Boussinesq approximation. The governing conservation equations in the non-dimensional form can be expressed as Eqs. (A-1)-(A-4). In these equations, $u$ and $v$ are the velocity components in $x$ and $y$ directions, $P$ is the pressure, and $T$ is the temperature of the fluid. The superscript * is used to point out that the variables are in their dimensionless forms.
\( \frac{\partial u^*}{\partial x^*} + \frac{\partial v^*}{\partial y^*} = 0 \)  
(A-1)

\[ u^* \frac{\partial u^*}{\partial x^*} + v^* \frac{\partial u^*}{\partial y^*} = - \frac{\partial p^*}{\partial x^*} + \text{Pr} \left( \frac{\partial^2 u^*}{\partial x^*^2} + \frac{\partial^2 u^*}{\partial y^*^2} \right) \]
(A-2)

\[ u^* \frac{\partial v^*}{\partial x^*} + v^* \frac{\partial v^*}{\partial y^*} = - \frac{\partial p^*}{\partial y^*} + \text{Pr} \left( \frac{\partial^2 v^*}{\partial x^*^2} + \frac{\partial^2 v^*}{\partial y^*^2} \right) + \text{Pr} \text{Ra} T^* \]
(A-3)

\[ u^* \frac{\partial T^*}{\partial x^*} + v^* \frac{\partial T^*}{\partial y^*} = \left( \frac{\partial^2 T^*}{\partial x^*^2} + \frac{\partial^2 T^*}{\partial y^*^2} \right) \]
(A-4)

The non-dimensional variables in the above equations are defined as \( x^* = x/D, y^* = y/D, u^* = u/(\alpha/D), v^* = v/(\alpha/D), p^* = P/(\rho \alpha^2/D^2), T^* = (T - T_c)/(T_h - T_c) \), where \( \alpha = k/(\rho c) \) is the thermal diffusivity of fluid. We define Pr and Ra as

\[ \text{Pr} = \frac{\nu}{\alpha}, \quad \text{Ra} = \frac{g \beta (T_h - T_c) D^3}{\nu \alpha} \]
(A-5)

where \( g \) is the gravitational acceleration; \( \nu, \rho, c, k, \) and \( \beta \) are kinematic viscosity, density, specific heat, thermal conductivity, and the volumetric coefficient of thermal expansion, respectively.

The hydrodynamic and thermal boundary conditions are specified in Eqs. (A-6) and (A-7), respectively.

\[ u^* = 0 \text{ and } v^* = 0 \text{ (for all walls)} \]
(A-6)

\[ T^* = 1 \text{ (on left wall)}, \quad T^* = 0 \text{ (on right wall)}, \quad \frac{\partial T^*}{\partial y^*} = 0 \text{ (on top and bottom walls)}, \]
(A-7)

The local and average Nu on the left \((X = 0)\) and right \((X = 1)\) walls are defined as

\[ \text{Nu}_y = \left[ \frac{\partial T^*}{\partial x^*} \right]_{x^* = X}, \quad \overline{\text{Nu}} = \int_0^1 \left[ \frac{\partial T^*}{\partial x^*} \right]_{x^* = X} dy^* \]
(A-8)

where \( \overline{\text{Nu}} \) has the same value at both the vertical walls due to the steady-state condition.

Appendix B: Numerical Method and Validation

The coupled governing Eqs. (A-1)–(A-4) were transformed into sets of algebraic equations using the finite volume method, and the SIMPLE algorithm was used for pressure-velocity coupling in the momentum equations. The numerical code was validated for the case of an air-filled enclosure, which showed agreement with published results. More details on the numerical method and its verification can be found in our previous papers [29-31].

The results of the average Nu and the maximum dimensionless stream function \( \psi^* = \psi/\alpha \) were evaluated for different grid systems at different Pr for the highest Ra, as summarized in Table B-1. The tested grids are
all uniform and structured and a boundary mesh is applied in which the cells adjacent to the walls were split in half to account for higher gradients near the walls. In comparison to the results obtained by an 800×800 grid system, the highest errors of the results obtained using the 200×200 and 400×400 grid systems were about 1 and 0.5 percent, respectively, for the most stringent cases.

Table B-1. Comparison of the average Nu and the maximum stream function for Ra = 10^8 for different grid sizes; the values in parentheses are the relative errors compared with the most accurate results.

| Grid system | Pr = 0.1 | Pr = 1 | Pr = 10 |
|-------------|----------|--------|---------|
| Nu | ψ_{max} | Nu | ψ_{max} | Nu | ψ_{max} |
| 25×25 | 28.94 | 56.06 | 34.65 | 63.46 | 35.64 | 68.29 |
| (14.8%) | (31.8%) | (12.4%) | (11.7%) | (12.1%) | (9.44%) |
| 50×50 | 26.78 | 46.86 | 33.63 | 59.15 | 35.07 | 66.19 |
| (6.27%) | (10.2%) | (9.12%) | (4.12%) | (10.3%) | (6.07%) |
| 100×100 | 24.79 | 44.94 | 31.05 | 57.77 | 32.19 | 63.96 |
| (1.65%) | (5.65%) | (0.74%) | (1.69%) | (1.27%) | (2.50%) |
| 200×200 | 25.09 | 42.84 | 31.07 | 57.04 | 32.13 | 62.72 |
| (0.43%) | (0.73%) | (0.82%) | (0.40%) | (1.06%) | (0.51%) |
| 400×400 | 25.14 | 42.47 | 30.93 | 56.90 | 31.95 | 62.51 |
| (0.26%) | (0.15%) | (0.34%) | (0.15%) | (0.51%) | (0.18%) |
| 800×800 | 25.20 | 42.53 | 30.82 | 56.81 | 31.79 | 62.40 |

Appendix C: The multi-grid simulation

We show that the grid dependency of the solution depends on the value of Ra and Pr. Figure C-1 presents the variations of Nu with Ra and Pr. As can be seen in the constant-Pr curve of Fig. C-1a, a 25×25 grid system reliably simulates the problem for Ra up to 10^3 (having a maximum relative difference of 0.2% with the result of the 200×200 grid system). Likewise, a 50×50 grid system predicts the Nu accurately enough provided that Ra < 10^5 (with a maximum relative difference of below 0.6% with the result of the 200×200 grid system). The validity of this statement is also checked for different Pr, as presented in Fig. C-1b for Ra = 10^5. We conclude that we can rely on the above statement except for low Pr that coarse grid systems result in high error values (due to experiencing convergence difficulties). Therefore, we employ models having 25×25 and 50×50 simulation grids for Ra ≤ 10^3 and Ra < 10^5, respectively, and a 200×200 grid system for other cases. Nonetheless, to maintain the accuracy of the Nu result above 99%, we employ a 400×400 grid system for low Pr or high Ra (as illustrated in Fig. 1a). Our analysis gives approximate criteria for the selection of the grid size for different input ranges. Nonetheless, based on the errors between the original data and the ANN model, we further revised the data points by using finer grid sizes.
Fig. C-1. The variation of Nu versus a) Ra at a fixed Pr = 1, and b) Pr at a fixed Ra = 10^5.

Appendix D: ANN Optimization

We optimized the ANN architecture using the original training dataset. The dataset (480 simulations) was first split into train and validation datasets with a ratio of 15% (408 training data and 72 validation data) using the model selection function in Scikit-learn library (the test dataset was not used in the ANN optimization process). In Keras 2.3.1 library with TensorFlow 2.2.0 as the backend, the Hyperband algorithm [33] that is a Bandit-based approach was employed to optimize the hyperparameters. We considered a hidden layer with 4 nodes next to the input layer, and a combination of 1 to 5 intermediate hidden layers with either 4, 8, 16, or 32 nodes within every hidden layer. As Fig. C-1 suggests, ‘tanh’ is a good approximator for the prediction of Nu; thus, we applied the ‘tanh’ activation function for all the layers of our ANN. We tested the Adam optimizer [34] with different learning rates from 5×10^-4 to 10^-3 with “log” sampling (which assigns equal probabilities to each order of magnitude range) in Keras HyperParameters container. After testing different combinations through Hyperband algorithm, a structure as described in Table D-1 is selected.

Next, we ran a limited number of brute-force tunings on the optimized model using more epochs. At this stage, we applied a multi-stage training procedure using different batch sizes of 2^n+1 applied sequentially (n = 1, 2, …, 7) and picked the best model among the models produced after each of these stages. Sequentially changing the batch size enabled us to find a well-trained ANN without worrying about having a proper value for the batch size as well as the learning rate, which changes adaptively in Adam optimizer. Nonetheless, we tested a limited number of batch sizes from 0.0001 to 0.001 and performed the multi-stage training separately. No dropout layer was required, and a learning rate of 0.0005 provided the least validation loss.
Table D-1. The structure of the optimized ANN for the Nu prediction.

|                         |                          |
|-------------------------|--------------------------|
| **ANN class:**          | multilayer perceptron (MLP) feedforward |
| **Layers:**             | input: 4 nodes, 2 inputs |
|                         | 2 hidden layers: 16 nodes each |
|                         | output layer: 1 node, 1 output |
| **Total parameters:**   | 381                      |
| **Activation function:**| ‘tanh’ (= 2/(1 + e⁻²ˣ) – 1) |
| **Optimizer:**          | Adam                     |
| **Learning rate:**      | 0.0005                   |
| **Batch size:**         | 2ⁿ⁺¹; n = 1, 2, ..., 7   |

References

[1] B. Chidambaram, M. Ravichandran, A. Seshadri, and V. Muniyandi, “Computational heat transfer analysis and genetic algorithm-artificial neural network-genetic algorithm-based multiobjective optimization of rectangular perforated plate fins,” IEEE Transactions on Components, Packaging and Manufacturing Technology, pp. 1–9, 2017.

[2] T. V. V. Sudhakar, C. Balaji, and S. P. Venkateshan, “Optimal configuration of discrete heat sources in a vertical duct under conjugate mixed convection using artificial neural networks,” International Journal of Thermal Sciences, vol. 48, no. 5, pp. 881–890, May 2009.

[3] S. I. Ahamad and C. Balaji, “Inverse conjugate mixed convection in a vertical substrate with protruding heat sources: a combined experimental and numerical study,” Heat and Mass Transfer, vol. 52, no. 6, pp. 1243–1254, Jul. 2015.

[4] H. Najafi and K. A. Woodbury, “Online heat flux estimation using artificial neural network as a digital filter approach,” International Journal of Heat and Mass Transfer, vol. 91, pp. 808–817, Dec. 2015.

[5] A. Berber, M. Gürdal, and K. Bağırsakçı, “Prediction of heat transfer in a circular tube with aluminum and Cr-Ni alloy pins using artificial neural network,” Experimental Heat Transfer, pp. 1–17, Jul. 2020.

[6] H. Taghavifar and E. Shabahangnia, “Prediction of thermal gradient in an air channel with presence of electrostatic field using artificial neural network,” Heat and Mass Transfer, vol. 50, no. 11, pp. 1515–1524, Apr. 2014.

[7] A. M. Bahman and S. A. Ebrahim, “Prediction of the minimum film boiling temperature using artificial neural network,” International Journal of Heat and Mass Transfer, vol. 155, p. 119834, Jul. 2020.

[8] Y. Islamoglu, “Modeling of thermal performance of a cooling tower using an artificial neural network,” Heat Transfer Engineering, vol. 26, no. 4, pp. 073–076, May 2005.
[9] A. Ben-Nakhi, M. A. Mahmoud, and A. M. Mahmoud, “Inter-model comparison of CFD and neural network analysis of natural convection heat transfer in a partitioned enclosure,” Applied Mathematical Modelling, vol. 32, no. 9, pp. 1834–1847, Sep. 2008.
[10] F. Selimefendigil and H. F. Öztop, “Estimation of the mixed convection heat transfer of a rotating cylinder in a vented cavity subjected to nanofluid by using generalized neural networks,” Numerical Heat Transfer, Part A: Applications, vol. 65, no. 2, pp. 165–185, Oct. 2013.
[11] S. Shahane, N. R. Aluru, and S. P. Vanka, “Uncertainty quantification in three dimensional natural convection using polynomial chaos expansion and deep neural networks,” International Journal of Heat and Mass Transfer, vol. 139, pp. 613–631, Aug. 2019.
[12] M. Sepehrnia, G. Sheikhzadeh, G. Abaei, and M. Motamedian, “Study of flow field, heat transfer, and entropy generation of nanofluid turbulent natural convection in an enclosure utilizing the computational fluid dynamics-artificial neural network hybrid method,” Heat Transfer-Asian Research, vol. 48, no. 4, pp. 1151–1179, Jan. 2019.
[13] A. R. Tahavvor and M. Yaghoubi, “Analysis of natural convection from a column of cold horizontal cylinders using Artificial Neural Network,” Applied Mathematical Modelling, vol. 36, no. 7, pp. 3176–3188, Jul. 2012.
[14] M. K. H. Kumar, P. S. Vishweshwara, and N. Gnanasekaran, “Evaluation of artificial neural network in data reduction for a natural convection conjugate heat transfer problem in an inverse approach: experiments combined with CFD solutions,” Sadhanā, vol. 45, no. 1, Mar. 2020.
[15] Ş. Ö. Atayılmaz, H. Demir, and Ö. Ağra, “Application of artificial neural networks for prediction of natural convection from a heated horizontal cylinder,” International Communications in Heat and Mass Transfer, vol. 37, no. 1, pp. 68–73, Jan. 2010.
[16] A. Amiri, A. Karami, T. Yousefi, and M. Zanjani, “Artificial neural network to predict the natural convection from vertical and inclined arrays of horizontal cylinders,” Polish Journal of Chemical Technology, vol. 14, no. 4, pp. 46–52, Dec. 2012.
[17] D. Colorado, M. E. Ali, O. García-Valladares, and J. A. Hernández, “Heat transfer using a correlation by neural network for natural convection from vertical helical coil in oil and glycerol/water solution,” Energy, vol. 36, no. 2, pp. 854–863, Feb. 2011.
[18] M. E. Poulad, D. Naylor, and A. S. Fung, “Prediction of local heat transfer in a vertical cavity using artificial neural networks,” Journal of Heat Transfer, vol. 132, no. 12, Sep. 2010.
[19] R. C. Aydin, F. A. Braeu, and C. J. Cyron, “General multi-fidelity framework for training artificial neural networks with computational models,” Frontiers in Materials, vol. 6, Apr. 2019.
[20] S. Chakraborty, “Transfer learning based multi-fidelity physics informed deep neural network,” arXiv preprint arXiv:2005.10614v2, Jun. 2020.
[21] S. De, J. Britton, M. Reynolds, R. Skinner, K. Jansen, and A. Doostan, “On transfer learning of neural networks using bi-fidelity data for uncertainty propagation,” arXiv preprint arXiv:2002.04495, Feb. 2020.
[22] D. Liu and Y. Wang, “Multi-fidelity physics-constrained neural network and its application in materials modeling,” Journal of Mechanical Design, vol. 141, no. 12, Sep. 2019.
[23] X. Meng and G. E. Karniadakis, “A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems,” Journal of Computational Physics, vol. 401, p. 109020, Jan. 2020.

[24] Q. Zhou, Y. Wang, S.-K. Choi, P. Jiang, X. Shao, J. Hu, and L. Shu, “A robust optimization approach based on multi-fidelity metamodel,” Structural and Multidisciplinary Optimization, vol. 57, no. 2, pp. 775–797, Aug. 2017.

[25] L. Shu, P. Jiang, X. Song, and Q. Zhou, “Novel approach for selecting low-fidelity scale factor in multifidelity metamodeling,” AIAA Journal, vol. 57, no. 12, pp. 5320–5330, Dec. 2019.

[26] Q. Zhou, Y. Wang, S.-K. Choi, P. Jiang, X. Shao, and J. Hu, “A sequential multi-fidelity metamodeling approach for data regression,” Knowledge-Based Systems, vol. 134, pp. 199–212, Oct. 2017.

[27] H. Babaei, P. Perdikaris, C. Chryssostomidis, and G. E. Karniadakis, “Multi-fidelity modelling of mixed convection based on experimental correlations and numerical simulations,” Journal of Fluid Mechanics, vol. 809, pp. 895–917, Nov. 2016.

[28] H. Venkateswara and S. Panchanathan, “Introduction to domain adaptation,” Domain Adaptation in Computer Vision with Deep Learning, pp. 3–21, 2020.

[29] M. Ashouri, B. Ebrahim, M. B. Shafii, M. H. Saidi, and M. S. Saidi, “Correlation for Nusselt number in pure magnetic convection ferrofluid flow in a square cavity by a numerical investigation,” Journal of Magnetism and Magnetic Materials, vol. 322, no. 22, pp. 3607–3613, Nov. 2010.

[30] M. Ashouri and M. Behshad Shafii, “Numerical simulation of magnetic convection ferrofluid flow in a permanent magnet–inserted cavity,” Journal of Magnetism and Magnetic Materials, vol. 442, pp. 270–278, Nov. 2017.

[31] M. Ashouri, M. Behshad Shafii, and H. Rajabi Kokande, “MHD natural convection flow in cavities filled with square solid blocks,” International Journal of Numerical Methods for Heat & Fluid Flow, vol. 24, no. 8, pp. 1813–1830, Oct. 2014.

[32] P. Mostaghimi, M. Ashouri, and B. Ebrahim, “Hydrodynamics of fingering instability in the presence of a magnetic field,” Fluid Dynamics Research, vol. 48, no. 5, p. 055504, Sep. 2016.

[33] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, Hyperband: A novel bandit-based approach to hyperparameter optimization,” The Journal of Machine Learning Research, vol. 18, pp. 1–52, Apr. 2018.

[34] D. P. Kingma, and J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980v9, Jan. 2017.