

**FORM 2**

**THE PATENTS ACT, 1970  
(39 of 1970)**

**COMPLETE SPECIFICATION**  
**(See section 10 and rule 13)**

**TITLE**

**MACHINE LEARNING, DEEP LEARNING AND ARTIFICIAL  
INTELLIGENCE FOR PHYSICAL TRANSPORT PHENOMENON IN  
THERMAL MANAGEMENT**

**INVENTORS:**

**NANDIGANA, Vishal V.R.** Indian Citizen  
**DASARI, Ananyananda,** Indian Citizen  
Department of Mechanical Engineering

Indian Institute of Technology Madras,  
Chennai-600036, India

**APPLICANTS**

**Indian Institute of Technology Madras (IIT Madras)**  
Office of the Dean ICSR  
Chennai – 600036, India

**THE FOLLOWING SPECIFICATION PARTICULARLY DESCRIBES THE  
INVENTION AND THE MANNER IN WHICH IT IS TO BE PERFORMED**

**MACHINE LEARNING, DEEP LEARNING AND ARTIFICIAL  
INTELLIGENCE FOR PHYSICAL TRANSPORT PHENOMENON IN  
THERMAL MANAGEMENT**

CROSS-REFERENCES TO RELATED APPLICATIONS

[0001] This patent is a complete specification filing of patent application no. 201941036792 entitled Machine Learning, Deep Learning and Artificial Intelligence for physical transport phenomenon in thermal management filed on October 12, 2019.

FIELD OF THE INVENTION

[0002] The present invention relates in general to methods of solving transport problems and in particular to AI or neural learning based methods of solving these problems.

DESCRIPTION OF RELATED ART

[0003] Artificial Intelligence (AI) has created a great value and buzz across the technological landscape in recent years. Artificial Intelligence has found use in diverse fields such as language processing, text completion, image reconstruction, autonomous cars, voice assistance, and in the fields of robotics, drone movement control, virtual reality applications, video games, graphics simulations, etc. However, the application of such techniques have not been explored to solve engineering problems in thermal management, electronic cooling industries, automobile industries like fluid dynamics predictions over a bonnet or inside the engine, aerospace industries like aerodynamics and fluid dynamics problems across an aero-foil or wing.

[0004] In thermal management, physical quantities such as temperature, pressure, velocity, etc. of a body depend on many factors, including but not limited to spatio-temporal variables. The three-dimensional pose, time and material properties such as Heat Capacity, Young's Modulus, etc. are some of the factors that determine the values of

the physical quantities in processes occurring around us. In many cases, these quantities are also interdependent among themselves, creating coupled situations. Differential equations formulate the basic physical laws governing such phenomena, and can model these dependencies. However, the multi-variable dependency of most quantities makes it infeasible to use Ordinary differential equations, except in extremely simplified cases. Partial differential equations (PDE) can both model the large scale, multi-variable dependency, as well as inter-variable dependencies. Depending on the process, a single PDE or a system of simultaneous PDEs is solved to determine the value of each physical quantity at a particular location and time.

**[0005]** Generally, PDEs are solved by analytical methods such as separation of variables or by assuming a suitable form for the solution. PDEs are also solved numerically, by dividing the area of interest into smaller parts, and satisfying the conservation laws between neighboring parts using a variety of specialized algorithms. Numerical solutions iterate the values of the quantities a large number of times to satisfy the conservation laws as closely as possible.

**[0006]** However, the discretization of the geometry into smaller parts or grid cells introduces a different set of problems. Firstly, the generation of the solution is dependent on the convergence of the algorithm, i.e., the reduction of the error term to within acceptable range. Convergence is dependent upon many factors, such as initial values, boundary conditions, geometry shape and size, mesh size, etc. Secondly, the increase in the number of grid cells causes a corresponding increase in the time required for convergence. In case of convergence of the algorithm, a finer mesh usually produces a better solution, but with the penalty of higher computational time and resources. This causes huge delays in running advanced simulations, which involve very fine or in some cases, adaptive meshes, which have different mesh sizes at different locations in the

geometry. Complex geometries further aggravate the problem. Thus, there is a need to find methods to reduce the time and effort in solving large computational problems.

[0007] The application of neural network models for solving and predicting partial differential equations is an emerging domain, initiated in 1990s. Various publications such as G.E. Karniadakis et al and MaziarRaissi et al work on solving and predicting non-linear partial differential equations using neural networks as approximated solutions to the partial differential equations has spearheaded the development in this field. Recently developed auto-differentiation techniques over the neural networks to approximate the other non-linear terms of the equations have further contributed to the gradual acceptance of neural network solvers for PDEs. However, there are no publications that solve physical transport phenomena problems without incorporating the underlying physics based PDEs, which makes the solutions complex and computation intensive.

[0008] The invention proposes devices and methods to address some of the drawbacks discussed here.

### **SUMMARY OF THE INVENTION**

[0009] According to various embodiments, a method of solving a heat transport problem over an object characterized by a geometry, using a hardware multi-threading process is disclosed. The hardware comprising: a processor configured to run a training model, a first number of storage process units configured to store input data, a second number of memory operation units configured to store output data, and a hardware switch configured to minimize idle time of the processor. The disclosed method comprising: providing a geometry and associated boundary conditions and discretizing the geometry into a grid, wherein the grid comprises a number of grid points. The method includes specifying temperature or heat flow conditions at the boundary surrounding the geometry

and an initial condition at each grid point. The method next includes solving a heat flow equation selected from one of conduction, convection or radiation for the geometry and the associated boundary conditions to obtain a temperature, or a heat flow rate, or both at each grid point at steady state. Next, the method includes storing the solution for each grid point in a training database; training a model selected from a PPRNN, a DRNN or a DANN model using the training database; inputting a modified boundary condition or initial condition or both associated with the geometry; and, generating a temperature, a heat flow rate at both at each grid point corresponding to the modified boundary condition or initial condition.

**[0010]** In various embodiments, the training includes calling a storage process unit for each input point  $i$  in an individual unit; storing all data points  $i=1, 2, 3 \dots n$  in  $j=1, 2, 3 \dots n$  storage units in a single logic call using AND logic enabled function; calling multi-threading/distributed RNN and executing the same; calling AND logic enabled machine code upon each point  $i=1, 2, 3 \dots n$  in memory unit  $k=1, 2, 3 \dots n$  with zero time lapse and zero  $k$  to  $k$  communication; storing output RNN data of each output point  $p$ , corresponding to each input point  $i=1, 2, 3 \dots n$  in  $j=1, 2, 3 \dots n$  storage units; and repeating the steps (i) to (v) for  $m$  samples and perform computations and storage for each  $m=1, 2, 3 \dots$  sample in the geometry.

**[0011]** In various embodiments, the heat flow equation solved is a conduction equation wherein the temperature  $T$  is given by:  $\frac{1}{a} \cdot \frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} + \frac{q}{k}$  where  $x$ ,  $y$ , and  $z$  represent Cartesian coordinates,  $q$  is rate of heat generation,  $k$  is thermal conductivity and  $a$  is a product of density and specific heat capacity of the material. In some embodiments, the heat flow equation solved is a convection equation given by:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + h \cdot \frac{\partial T}{\partial y} = 0.$$

In some embodiments, the heat flow equation solved is a radiation

$$\text{equation given by: } d_z \rho c_p u \cdot \nabla T + \nabla \cdot q = d_z Q + q_0 + d_z Q_{ted} + d_z Q_R.$$

**[0012]** In some embodiments, solving a heat flow equation comprises using an analytical solution, a finite element method solution (FEM) or finite difference method (FDM) solution to generate training data set for each input data  $i$  for each sample  $m$  in a geometry  $t$ . In various embodiments, the model is a PPRNN model wherein:  $PPRNN = \oint_{\Omega} \int_{j=1}^M \tanh(h_{j_i} + b_{2_i}) dj d\Omega_i$  where  $h_{j_i} = W_{1_i} \cdot h_{j-1_i} + W_{2_i} \cdot x_{j-1_i} + b_{1_i}x$  is the input,  $h$  is the hidden cell state and  $W_1, b_1$  and  $W_2, b_2$  are the weight and bias matrices for hidden-hidden and input-hidden connections,  $\Omega$  is the domain of interest,  $M$  is the number of examples for training,  $\tanh$  is an activation function.

**[0013]** In various embodiments, the method includes obtaining the weight and bias matrices by minimizing mean square error between the predicted and input temperatures for each mesh independent point  $i$  for an example  $m$  given by:  $MSE_i = \frac{1}{m} \sum_{n=1}^m |T_{Pred_i} - T_{Act_i}|^2$ . In various embodiments, the model is a DRNN model wherein:  $DRNN = \forall \oint_{\Omega} \int_{j=1}^M \tanh(h_{j_i} + b_{2_i}) dj d\Omega_i$  where,  $h_j = W_1 \cdot h_{j-1} + W_2 \cdot x_{j-1} + b_1x$  is the input,  $h$  is the hidden cell state and  $W_1, b_1$  and  $W_2, b_2$  are the weight and bias matrices for hidden-hidden and input-hidden connections,  $\Omega$  is the domain of interest,  $M$  is the number of examples for training,  $\tanh$  is an activation function.

**[0014]** In some embodiments, the method includes obtaining the weight and bias matrices by minimizing mean square error between the predicted and input temperatures for each mesh independent point  $i$  for an example  $m$  given by:  $MSE = \sum_{n=1}^m \|T_{Pred} - T_{Act}\|^2$ . In various embodiments, the model is a DANN model wherein:  $DANN =$

$$\forall \begin{cases} 0 & \text{if } x \leq 0 \\ \int_{j=1}^M (h_j + b_2) dj, & \text{if } x > 0 \end{cases} \text{ where, } h_j = W_1 \cdot h_{j-1} + W_2 \cdot x_{j-1} + b_1x \text{ is the input, } h \text{ is the}$$

hidden cell state and  $W_1, b_1$  and  $W_2, b_2$  are the weight and bias matrices for hidden-hidden

and input-hidden connections, and  $M$  is the number of examples for training. In various embodiments, the method includes obtaining the weight and bias matrices by minimizing mean square error between the predicted and input temperatures for each mesh independent point  $i$  for an example  $m$  given by:  $MSE_i = \frac{1}{m} \sum_{n=1}^m |T_{Pred_i} - T_{Act_i}|^2$ .

**[0015]** According to another embodiment, a system for solving a heat transport problem over an object characterized by a geometry is disclosed. The system includes a hardware switch configured to execute a multi-threading process, and a processor coupled to the hardware switch to run a neural network engine. The processor is configured to receive a geometry and associated boundary conditions and discretize the geometry into a grid, wherein the grid comprises a number of grid points. Temperature or heat flow conditions at the boundary surrounding the geometry and an initial condition at each grid point are received. The processor solves a heat flow equation selected from one of conduction, convection or radiation for the geometry and the associated boundary conditions to obtain a temperature, or a heat flow rate, or both at each grid point at steady state and stores the solution for each grid point in a training database. Next, the processor trains a model selected from a PPRNN, a DRNN or a DANN model using the training database. The trained model is configured to receive a modified boundary condition or initial condition or both associated with the geometry to generate a temperature, a heat flow rate at both at each grid point corresponding to the modified boundary condition or initial condition.

**[0016]** In various embodiments, the system includes a storage process unit comprising a plurality of sub-storage process unit configured to store one or more data subsets of geometry and associated boundary conditions, and temperature and heat-flow conditions. The processor is configured to divide the transport problem solution into a plurality of threads for concurrent execution. In various embodiments, the hardware

switch is configured to execute the plurality of threads in parallel and allocate the generated temperature and heat flow rate in memory operation units. The memory operation units include a plurality of memory operation units configured to store generated temperature and heat flow rate.

[0017] This and other aspects are disclosed herein.

### **BRIEF DESCRIPTION OF THE DRAWINGS**

[0018] The invention has other advantages and features which will be more readily apparent from the following detailed description of the invention and the appended claims, when taken in conjunction with the accompanying drawings, in which:

[0019] FIG. 1 illustrates a simple block diagram of a system for solving a transport problem using Neural Network (NN), according to one embodiment of the present subject matter.

[0020] FIG. 2 illustrates a method of solving a transport problem, machine learning or deep learning methods using Neural Network (NN), according to one embodiment of the present subject matter.

[0021] FIG. 3 illustrates a block diagram of a system for solving a transport problem using Neural Network (NN) engine, according to another embodiment of the present subject matter.

[0022] FIG. 4 illustrates a flow diagram of a method of multi-threading process for DRNN, according to another embodiment of the present subject matter.

[0023] FIG. 5A and FIG. 5B illustrate a simple architecture of recurrent neural network and a point-to-point recurrent neural network, according to one embodiment of the present subject matter.



[0024] FIG. 6A and FIG. 6B illustrate a simple architecture of artificial neural network and a deep artificial neural network, according to one embodiment of the present subject matter.

[0025] FIG. 7A and FIG. 7B illustrate a simple architecture of recurrent neural network and a deep recurrent neural network, according to one embodiment of the present subject matter.

[0026] FIG. 8A – FIG. 8C illustrate dirichlet condition for square geometry, circular geometry and Neumann condition for square and circular geometries.

[0027] FIG. 9A – FIG. 9C illustrate a comparison between truth and predicted solution for square geometry domain conduction with Dirichlet boundary condition.

[0028] FIG. 10A – FIG. 10C illustrate a comparison between truth and predicted solution for circular geometry domain conduction with Neumann boundary condition.

[0029] FIG. 11A – FIG. 11C illustrate a comparison between truth and predicted solution for square geometry domain convection with Dirichlet boundary condition.

[0030] FIG. 12A – FIG. 12C illustrate a comparison between truth and predicted solution for circular geometry domain convection with Neumann boundary condition.

[0031] FIG. 13A – FIG. 13C illustrate a comparison between truth and predicted solution for square geometry domain radiation with Dirichlet boundary condition.

[0032] FIG. 14A – FIG. 14C illustrate a comparison between truth and predicted solution for circular geometry domain radiation with Neumann boundary condition.

[0033] Referring to the drawings, like numbers indicate like parts throughout the views.

### **DETAILED DESCRIPTION OF THE EMBODIMENTS**

[0034] While the invention has been disclosed with reference to certain embodiments, it will be understood by those skilled in the art that various changes may be made and equivalents may be substituted without departing from the scope of the invention. In addition, many modifications may be made to adapt to a particular situation or material to the teachings of the invention without departing from its scope.

[0035] Throughout the specification and claims, the following terms take the meanings explicitly associated herein unless the context clearly dictates otherwise. The meaning of "a", "an", and "the" include plural references. The meaning of "in" includes "in" and "on." Referring to the drawings, like numbers indicate like parts throughout the views. Additionally, a reference to the singular includes a reference to the plural unless otherwise stated or inconsistent with the disclosure herein.

[0036] The invention in its various embodiments discloses systems and methods of solving a transport problem across a geometrical object using machine learning. In some embodiments, the transport problem may be a heat flow problem in 2 or 3 dimensions. A system **100** for implementing the method of the invention is disclosed in FIG. 1. The system **100** may include a processor **101** configured to run a neural network engine, a first number of storage process units **110** configured to store input data, a second number of memory operation units **120** configured to store output data, and a hardware switch **102** configured to minimize idle time of the processor. The hardware switch **102** may execute a multi-threading process on receiving instructions from the processor to solve the transport problem while minimizing idle time of the processor.

[0037] In various embodiments, the processor **101** may be configured to divide the process of solving the transport problem into a plurality of threads or tasks for

concurrent execution. The storage process unit **110** may include a plurality of sub-storage process units **111-1, 111-2, 111-3, ...,111-X** configured to store the input data. In some embodiments, each one of the plurality of sub-storage process unit **111-1, 111-2,...,111-X** may be configured a subset of the input data. For instance, the input data may be classified into a plurality of subsets based on a metric, such as timestamp.

**[0038]** The processor **101** running the neural network engine may receive the input data from the storage process units **110** and generate the output data. In some embodiments, the neural network engine may include multiple layers for computing the output concurrently. The hardware switch **102** may be configured to execute the multiple threads or tasks in parallel and retrieve the output data. The output data may be allocated and stored in the memory operation units **120**, which may include a plurality of memory operation units **121-1, 121-2, 121-3....., 121-X**,

**[0039]** In various embodiments a method of solving or computing solution to a heat transport problem over an object characterized by a geometry and having a boundary is disclosed with reference to FIG. 2. The method may use a hardware system as disclosed in FIG. 1. The method may include a first step of providing **(201)** a discretized geometry. The object geometry may be discretized into multiple elements forming a grid, which may include a number of grid points. In various embodiments, the discretization may depend on the complexity of the geometry. The number of elements or element density or both may be varied depending on whether the object has discontinuity in section, or material characteristics or both.

**[0040]** In the next step **202**, temperature or heat flow conditions may be specified at the boundary surrounding the geometry. The boundary condition may be a Dirichlet boundary condition or a Neumann boundary condition. An initial condition may also be specified at each grid point. The method may then involve the step **203** of solving a heat flow equation for the geometry and corresponding boundary conditions to obtain

a temperature, or a heat flow rate, or both at each grid point at steady state. In various embodiments, the heat flow equation may be selected from one of a conduction equation, a convection equation or a radiation equation as illustrated further.

[0041] In the next step **204**, the method involves storing the solution for each grid point in a training database. The training database may be configured to be trained using a variety of different models such as a Point by Point Recurrent Neural Network (PPRNN), a Distributed DANN (DANN) or a Distributed Recurrent Neural Network (DRNN). In step **205**, the method may involve training a model selected from a PPRNN, a DRNN or a DANN model using the training database. In the next step **206**, the method involves inputting a modified boundary condition or initial condition at the geometry, and running the trained model. In a final step **207**, a temperature or a heat flow rate is generated, at each grid point corresponding to the modified boundary or heat flow condition across the geometry.

[0042] Another block diagram of the system **300** is illustrated in FIG. 3, according to another embodiment of the present subject matter. The system **300** may include one or more processors **101**, one or more memory units **302**, a neural network (NN) engine **304**, a communication unit **306**, and a display unit **308**. The one or more processors **101** may be configured to receive a geometry having a boundary and discretize the geometry into a grid comprising a number of points. The processor **101** may also receive specific boundary conditions surrounding the geometry and an initial condition at each grid point. The processor **101** may then solve a heat flow equation for the geometry and boundary conditions to obtain a temperature, or a heat flow rate or both at each grid point. The solution for each grid point may be stored in a training database. The processor **101** may train a model selected from a neural network engine **304**, such as a PPRNN engine using the training database. In some embodiments, the neural network engine **304** may be a point to point neural network

PPRNN engine, deep recurrent neural network DRNN engine or a deep artificial neural network DANN engine.

[0043] In various embodiments, the NN engine **304** may include a training component **310**, a validating component **312**, and a testing component **314**. Each component of the NN engine **304** may receive different datasets. For example, the geometry and associated boundary condition data along with truth values comprising heat transfer solutions may be segregated into one or more of a training dataset and a validation dataset. In some embodiments, the training dataset may encompass 80% and the validation dataset may encompass 20% of the received data. In various embodiments, the training data may be accessed from a remote database. In other embodiments, the training dataset may encompass 70% of the received data, the validation dataset may include 15% of the received data, and the test dataset may encompass 15% of the received data. In various embodiments, the communication unit **306** may be configured to transmit the predicted solutions to one or more external components. In some embodiments, the display unit **308** may be configured to display the predicted solutions to a user.

[0044] In various embodiments of the method, a method **400** of training a database using a neural network model is disclosed. The model trained may in some embodiments be a PPRNN, a DRNN or a DANN model. The method comprises the steps of **401** calling a storage process unit for each input point  $i$  in an individual unit, storing **402** all data points  $i=1, 2, 3 \dots n$  in  $j=1, 2, 3 \dots n$  storage units in a single logic call using AND logic enabled function, calling **403** and a multi-threading/distributed method and executing the same. The method in the next steps **404** involves calling AND logic enabled machine code upon each point  $i=1, 2, 3 \dots n$  in memory unit  $k=1, 2, 3 \dots n$  with zero time lapse and zero  $k$  to  $k$  communication and **405** storing output data of each output point  $p$ , corresponding to each input point  $i=1, 2, 3 \dots n$  in  $j=1, 2,$

3... n storage units. The method may further involve in step **406**, repeating the steps **401** to **405** for m samples and perform computations and storage of output data for each m=1, 2, 3... sample in the geometry.

[0045] In some embodiments of the method **200**, the heat flow equation solved may be a conduction equation wherein the temperature T is given by:

$$\frac{1}{a} \cdot \frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} + \frac{q}{k} \dots \dots \dots (1)$$

where x, y, and z represent Cartesian coordinates, q is rate of heat generation, k is thermal conductivity and a is a product of density and specific heat capacity of the material.

[0046] In some embodiments of the method **200**, the heat flow equation solved is a convection equation given by:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + h \cdot \frac{\partial T}{\partial y} = 0 \dots \dots \dots (2)$$

[0047] In some embodiments of the method **200**, the heat flow equation solved is a radiation equation given by:

$$d_z \rho c_p u \cdot \nabla T + \nabla \cdot q = d_z Q + q_0 + d_z Q_{ted} + d_z Q_R \dots \dots \dots (3)$$

[0048] In step **203** in FIG. 2 may involve solving a heat flow equation using a suitable known method to generate the training data. The method of solving may include using an analytical solution, a finite element method solution (FEM) or a finite difference method (FDM) solution to generate training data set for each input data i for each sample m in a geometry t.

[0049] In some embodiments of the method **400**, the training model is a PPRNN model wherein:

$$PPRNN = \oint_{\Omega} \int_{j=1}^M \tanh(h_{ji} + b_{2i}) dj d\Omega_i \dots \dots \dots (4)$$

where

$$h_{j_i} = W_{1_i} \cdot h_{j-1_i} + W_{2_i} \cdot x_{j-1_i} + b_{1_i} \dots \dots \dots (5)$$

$x$  is the input,  $h$  is the hidden cell state and  $W_1, b_1$  and  $W_2, b_2$  are the weight and bias matrices for hidden-hidden and input-hidden connections,  $\Omega$  is the domain of interest,  $M$  is the number of examples for training, and  $\tanh$  is an activation function.

[0050] The method 400 may further involve obtaining the weight and bias matrices for the PPRNN model by minimizing mean square error between the predicted and input temperatures for each mesh independent point  $i$  for an example  $m$  given by:

$$MSE_i = \frac{1}{m} \sum_{n=1}^m |T_{Pred_i} - T_{Act_i}|^2 \dots \dots \dots (6)$$

[0051] In some embodiments of the method 400, the training model is a DRNN model wherein:

$$DRNN = \forall \oint_{\Omega} \int_{j=1}^M \tanh(h_{j_i} + b_{2_i}) dj d\Omega_i \dots \dots \dots (7)$$

where,  $h_j$  is as given in (5),  $x$  is the input,  $h$  is the hidden cell state and  $W_1, b_1$  and  $W_2, b_2$  are the weight and bias matrices for hidden-hidden and input-hidden connections,  $\Omega$  is the domain of interest,  $M$  is the number of examples for training,  $\tanh$  is an activation function.

[0052] The method 400 may further involve obtaining the weight and bias matrices of the DRNN model by minimizing mean square error between the predicted and input temperatures for each mesh independent point  $i$  for an example  $m$  given by:

$$MSE = \sum_{n=1}^m \|T_{Pred} - T_{Act}\|^2 \dots \dots \dots (8)$$

[0053] In some embodiments of the method 400, the training model is a DANN model wherein:

$$DANN = \forall \begin{cases} 0 & \text{if } x \leq 0 \\ \int_{j=1}^M (h_j + b_2) dj & \text{if } x > 0 \end{cases} \dots \dots \dots (9)$$

where,  $h_j$  is as given in (5),  $x$  is the input,  $h$  is the hidden cell state and  $W_1$ ,  $b_1$  and  $W_2$ , are the weight and bias matrices for hidden-hidden and input-hidden connections, and  $M$  is the number of examples for training.

[0054] The method **400** may further involve obtaining the weight and bias matrices by minimizing mean square error between the predicted and input temperatures for each mesh independent point  $i$  for an example  $m$  given by equation (6).

[0055] A simple architecture of recurrent neural network and a point-to-point recurrent neural network are illustrated in FIG. 5A and FIG. 5B, according to one embodiment of the present subject matter. As shown, the recurrent neural network architecture **500** may include an input layer **503**, a plurality of LSTMs **504-N**, and an output layer **505**. The input layer may receive  $5 \times 1$  dimension vector comprising temperatures. The hidden layer **504-N** may receive the input temperature and feedback values as inputs to predict the temperature. The output layer **505** may include a node representing an output  $1 \times 1$  vector representing the predicted temperature. The PPRNN architecture **502** may include a plurality of point-to-point recurrent neural networks PPRNN **506**, each receiving training examples **501** and a bias vector input **507**. The outputs of the PPRNN engines are provided to the output layer **508** as a forward pass to predict the temperatures.

[0056] A simple architecture of artificial neural network and a deep artificial neural network are illustrated in FIG. 6A and FIG. 6B, according to one embodiment of the present subject matter. As shown, the ANN **600** may include an input layer **603**, one or more hidden layers **604-1**, **604-2** and an output layer **604**. The deep artificial neural network DANN architecture **602** may include a DANN **606**, which receives training examples **603** and a bias vector input **607**. The outputs of the DANN engines **606** may be provided to the output layer **608**.



[0057] Further, a simple architecture of recurrent neural network and a deep recurrent neural network are illustrated in FIG. 7A and FIG. 7B, according to one embodiment of the present subject matter. As shown, the recurrent neural network architecture **700** may include an input layer **703**, a plurality of LSTMs **704-N**, and an output layer **705**. The input layer may receive 900x4 dimension matrix comprising input temperature data. The hidden layer **704-N** may receive the input temperature and feedback values as inputs to predict the temperature. The output layer **705** may include a node representing an output 900x1 vector representing the predicted temperature. The deep artificial neural network DANN architecture **706** may include a DANN **706**, each receiving training examples **701** and a bias vector input **707**. The outputs of the DANN engines are provided to the output layer **708** as a forward pass to predict the temperatures.

[0058] As disclosed herein, the systems and methods have great computational advantage compared to existing systems and methods of solving these problems. The present methods could save computational time to the extent of  $10^5$  to  $10^9$  times, compared to existing methods. Further, the computational error could be minimized by optimally training the networks. In some instances, the methods may be used to minimize computational error to 0.2% or less of the actual value as predicted by standard numerical or analytical solutions. In some instances, with optimal training using appropriately sized data sets, the error could be minimized to 0.01% or less. A representation of max error of various thermal management problems considered using three deep learning based methods is provided in Table 3 in the examples section.

[0059] The systems and methods illustrated in the foregoing embodiments and the examples are generally applicable to any heat flow problem, whether in 2D or 3D geometry that involve mechanical complexity. Some 3D problems may be approximated using a 2D solution. Although the method is illustrated using 2D heat

flow examples, the systems and methods are equally applicable to the generalized 3D heat flow solution situation. The computational advantages of the method may become more valuable for 3D problems because of the highly computation-intensive nature of 3D heat flow solutions using current numerical methods of solution.

**[0060]** Although the detailed description contains many specifics, these should not be construed as limiting the scope of the invention but merely as illustrating different examples and aspects of the invention. It should be appreciated that the scope of the invention includes other embodiments not discussed herein. Various other modifications, changes and variations which will be apparent to those skilled in the art may be made in the arrangement, operation and details of the system and method of the present invention disclosed herein without departing from the scope of the invention as described here.

## EXAMPLES

### **[0061] Example 1: Generation of Training Data and Training the Networks**

**[0062]** The following cases as illustrated in Table 1 were determined using PPRNN, DRNN and DANN. Training data for each module was generated using either ANSYS or in house developed finite difference or finite element codes, as summarized in Table 2, which also includes runtime.

**TABLE 1: Heat Flow Problems Considered with Boundary Conditions**

	Dirichlet Boundary Condition	Neumann Boundary Condition	Dirichlet Boundary Condition	Neumann Boundary Condition
Conduction	Square	Square	Circular	Circular
Convection	Square	Square	Circular	Circular
Radiation	Square	Square	Circular	Circular

**[0063]** The above twelve problems considered were solved using each of the 3 methods. In the case of a 2D heat conduction over a square geometry a 25x25 mesh was chosen to solve using a conventional finite difference method (FDM) as illustrative example to compare with the analytical solution for the problems considered.

**[0064]** For Deep Learning methods, training data is essential to make the neural network get trained and result in accurate solution for a new test boundary condition or test example. For the implementation of the PPRNN into the 2D heat conduction problem, the in-house developed FDM solution was used as a training data. In this example problem, 1670 examples were used as training data and 800 examples are used as testing cases. Also, the boundary conditions for each example  $m$ , are given as the biases  $b_1$ ,  $b_2$ ,  $b_3$  and  $b_4$ , for all the four falls in the case where Dirichlet BCs are considered. The input  $x_i$  is the mesh point (excluding the boundary condition mesh points) for each of total sample  $m$ , which for this said 25 by 25 mesh points makes 529. 529 Recurrent neural networks are coded in a loop to take in a single mesh point input,  $x_i$  so that there is no dependency between mesh points for a said  $m$ , in developing the new PPRNN for 2D heat conduction application that is also mesh independent and physics independent as the information of the physics like the gradients and their derivatives need not be incorporated. The PPRNN is trained using the tanh activation function, biasing the input and biases with the trained weights. In order to obtain the trained weights, the input data  $x_i$  is compared with PPRNN solution (PPRNN) by minimizing the mean square error (MSE) say between  $T_{\text{Predi}}$  (same as Temperature PPRNN) and  $T_{\text{Input}}$  (same as  $T_{\text{Acti}}$ ) (obtained from Eq. (6)) for each mesh independent point  $i$  for an example  $m$ .

**[0065]** The said error minimization is performed for  $n = 1$  to  $m$  examples and also averaged to calculate the mean. The calculation of the mean squared error is repeated

or looped for each input grid point  $i$ , independent of the other grid point, to make the PPRNN truly mesh independent. L-BFGS optimiser is used to minimize the error for each input point  $i$ . The final weight for each point  $i$  is obtained, is saved and is used to predict the temperature for a new test boundary condition or test example using again PPRNN solution Eq 1. A representation of max error of various thermal management application problems considered using the deep learning neural network based methods is provided in Table 3.

[0066] The disclosed PPRNN method facilitates predictions for sparse data points as the method is not mesh dependent, making it free from calculating physics driven gradients and its derivatives and other higher order PDEs. Our developed PPRNN method is PDE or Physics less model and is also mesh independent for the first time in literature for solving engineering application problems.

[0067] **B. Multi-threading/Distributed RNN (DRNN):** In this method, a new method Multi-threading/Distributed RNN is proposed. Unlike PPRNN, where input data for each point  $i$  was trained using a separate Recurrent neural network. A single Recurrent neural network was used with multi-threading like or distributed like architecture where the same neural network was used for all  $i$  points, for a given example  $m$ , but the points were not communicating between each other in the proposed Multi-threading/Distributed Recurrent Neural Network (DRNN) because the weights while getting trained see these set of points still as independent point  $i$ , while performing the said Mean square error calculation using L-BFGS operation. The mathematical form for DRNN was obtained as shown in Eq. (7). The major advantage of DRNN unlike PPRNN is we need only 15 samples to do the training and test on various geometry and boundary conditions unlike 1000s of training samples needed for PPRNN as discussed before.

[0068] The equations are similar to PPRNN with the only difference the entire input information (in the said 2D heat conduction example, all 529 points) were inputted once for all to find the DRNN solution which however considers individual input  $i$ , independent of its neighboring input point for each sample, of total samples  $m$ , while calculating the weights minimizing Mean squared error (MSE) with L-BFGS operator. The advantage of DRNN method reduces the training time as compared to PPRNN, due to the model consisting of a single DRNN equation rather than multiple RNN equations used in PPRNN solution.

[0069] **B1. Multi-threading/Distributed RNN hardware implementation:**

Hardware multi-threading allows multiple threads to share the RNN operation to each processor in an overlapping fashion to try to utilize the hardware resources efficiently. FIG. 1 shows a storage processor unit in a computer architecture which consists of memory storage units to store the python based method of a input data point  $i$  set to RNN's method using a python based architecture and this data point from RNN is fed to memory processing units in a time lapse controlled fashion using an AND logic enabled with zero time lapse machine symbolic code to enable multi-threading and to perform the RNN calculation for each data point  $i$  independently in individual memory processing units. A schematic flow chart of the discussed multi-threading process is shown in FIG. 4 for better understanding of the method schematic shown in FIG. 2. This way the utilization of a processor was increased with no idle wait time for calculating individual data point  $i$ .

[0070] **C. Multi-threading/Distributed DANN (DANN):** The Multi-threading/Distributed Artificial Neural Network (DANN) is similar to DRNN but the DANN solution uses a RELU/Linear activation function instead of a tanh function. The major advantage of DANN like DRNN is we need only 15 samples to do the training and test on various geometry and boundary conditions unlike 1000 or more of

training samples needed for PPRNN. The mathematical equation for DANN is given in Eq. (9) and (10) as described earlier.

**[0071] Example 2: Prediction of Heat Flow using the Trained Models**

**[0072]** Two different geometries, 2D square/rectangle and 2D circular disc type geometries are considered to test the methods on given rectilinear and curvilinear geometries (see FIG. (8A-8D)).

**[0073] Conduction:** The multidimensional heat conduction equation and its variants were used in diverse fields, such as explaining Brownian motion in probability theory, wave function characteristics in quantum mechanics and also in solving the Black Scholes PDE in financial mathematics. In Cartesian coordinates it is written as Eq. (1).

**[0074]** The equation (1) determines the temperature distribution for a 3D, unsteady heat conduction problem with heat generation. The equation requires six boundary conditions, (two for each axis, x, y, and z) and one initial condition for the time dependence. The rate of heat generation is given by q and the thermal conductivity of the material is given by k. The two-dimensional steady state heat conduction equation, without heat generation is given by

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \dots \dots \dots (10)$$

$$T(0, y) = T1, T(x, 0) = T2, T(L, y) = T3, T(x, L) = T4 \dots \dots \dots (11)$$

**[0075]** This equation (10) requires the four boundary conditions (11) shown, two each for the x and y axes and no initial condition. The conditions shown are for the Dirichlet boundary condition on all four sides of the square domain. The Neumann case was obtained by changing one or more of these boundary conditions to  $\frac{dT}{dx} = a$  or  $\frac{dT}{dy} = b$ . a and b may be constants or even functions. Two Neumann conditions were used on the top and bottom sides of the square and sinusoidal function forms for the

slope values. For the circular domain, only the circumference and centre boundary conditions are required. The Dirichlet case specifies fixed temperatures for both circumference and centre, while for the Neumann case, the circumference temperature slope was defined, while the centre temperature remains fixed. The boundary temperatures for square and circular geometries were between  $T = 100^\circ$  to  $T = 1100^\circ$ . A comparison between truth (finite difference numerical solution) and predicted (PPRNN solution) for square geometry domain conduction with Dirichlet boundary condition (B) is illustrated in FIG. 9A – FIG. 9C. A comparison between truth (finite difference numerical solution) and predicted (PPRNN solution) for circular geometry domain conduction with Neumann boundary condition is illustrated in FIG. 10A – FIG. 10C.

**[0076] B. Heat Convection:** The advection-diffusion equation describes the transfer of physical properties, particles or energy inside a system, due to convection and diffusion. This equation is vital in the solution of other important equations such as Navier-Stokes in fluid dynamics and Black-Scholes in finance. The two-dimensional steady state heat convection equation, without heat generation and involving a non-linear convection term is given by Eq. (2)

**[0077]** The boundary conditions are given in Eqs (12). A comparison between truth (finite difference numerical solution) and predicted (PPRNN solution) for square geometry domain convection with Dirichlet boundary condition (B) is illustrated in FIG. 11A – FIG. 11C. A comparison between truth (finite difference numerical solution) and predicted (PPRNN solution) for circular geometry domain convection with Neumann boundary condition is illustrated in FIG. 12A – FIG. 12C.

**[0078] C. Heat Radiation:** Generation of electromagnetic waves by the thermal motion of all particles above 0 K is termed as radiation. In practice, radiative heat transfer between two surfaces takes place through a medium, whose properties

influence the nature of the waves. Such a medium is called a participating medium. The governing differential equations for radiative heat transfer through a participating medium was the Radiative Transfer Equation given by Eq (3) as described earlier. The equation includes the absorption, emission and scattering of incident radiation due to the participating medium.

$$d_z \rho c_p u \cdot \nabla T + \nabla \cdot q = d_z Q + q_0 + d_z Q_{ted} + d_z Q_R \dots \dots \dots (3)$$

where,  $d_z$  is the domain thickness (unit thickness assumed here),  $\rho$  and  $c_p$  are material properties and  $q$  is the thermal flux, given by

$$q = -d_z k \nabla T \dots \dots \dots (12)$$

$Q_R$  is the sum of the thermal and radiative heat fluxes, given by

$$Q_R = \kappa(G - 4\pi I_b) \dots \dots \dots (13)$$

$$G = \sum_{i=1}^n \omega_i I_i \dots \dots \dots (14)$$

where  $\omega$  is a quadrature weight (a weight used for numerical integration, in the discrete ordinates method). The scattering term is given by

$$S_i \cdot \nabla I_i = \kappa I_b(T) - \beta I_i + \frac{\sigma_s}{4\pi} \sum_{j=1}^N \omega_j I_j \phi(S_j, S_i) \dots \dots \dots (15)$$

where  $\sigma_s$  is the scattering coefficient and  $\beta = \kappa + \sigma_s$ .

[0079] To negate the influence of the medium,  $Q_R$  was set to zero, to obtain the pure surface-surface radiation case. COMSOL multi-physics Finite Element based method was used to generate the data. The material used was air at room temperature and pressure. The heat generation and transient terms were set to zero. The boundary conditions were of the form  $T = T_0$  for Dirichlet and  $-n \cdot q = d_z q_0 - n$  for Neumann. A comparison between truth (finite element numerical solution) and predicted (PPRNN solution) for square geometry domain radiation with Dirichlet boundary condition (B)



is illustrated in FIG. 13A – FIG. 13C. A comparison between truth (finite element numerical solution) and predicted (PPRNN solution) for circular geometry domain radiation with Neumann boundary condition (B) is illustrated in FIG. 14A – FIG. 14C.

**Table 2: Comparison of Simulation Times of Various Thermal Management Application Problems Using Existing FDM/FEM Based Commercial Software vs Three Novel Deep Learning Based Methods**

Method	Problem Solved	Simulation Time
Finite Difference method (FDM)	2D heat <b>conduction</b> on <b>square</b> plate with Dirichlet and Neumann boundary conditions	10 minutes
Point by point Recurrent Neural network (PPRNN)	-ditto-	1 ms
Distributed Recurrent Neural network (DRNN)	-ditto-	0.1 ms
Distributed Artificial Neural network (DANN)	-ditto-	0.1 ms
Finite Difference method (FDM)	2D heat conduction on <b>circular</b> plate with Dirichlet and Neumann boundary conditions	15 minutes
Point by point Recurrent Neural network (PPRNN)	-ditto-	1 ms
Distributed Recurrent Neural network (DRNN)	-ditto-	0.1 ms
Distributed Artificial Neural network (DANN)	-ditto-	0.1 ms
Finite Element method (FEM)	2D heat <b>convection</b> on <b>square</b> plate with Dirichlet and Neumann boundary conditions	20 minutes
Point by point Recurrent Neural network (PPRNN)	-ditto-	1 ms
Distributed Recurrent Neural network (DRNN)	-ditto-	0.1 ms
Distributed Artificial	-ditto-	0.1 ms

Neural network (DANN)		
Finite Element method (FEM)	2D heat <b>convection</b> on <b>circular</b> plate with Dirichlet and Neumann boundary conditions	25 minutes
Point by point Recurrent Neural network (PPRNN)	-ditto-	1 ms
Distributed Recurrent Neural network (DRNN)	-ditto-	0.1 ms
Distributed Artificial Neural network (DANN)	-ditto-	0.1 ms
Finite Element method (FEM)	2D heat <b>radiation</b> on <b>square</b> plate with Dirichlet and Neumann boundary conditions	40 minutes
Point by point Recurrent Neural network (PPRNN)	-ditto-	1 ms
Distributed Recurrent Neural network (DRNN)	-ditto-	0.1 ms
Distributed Artificial Neural network (DANN)	-ditto-	0.1 ms
Finite Element method (FEM)	2D heat <b>radiation</b> on <b>circular</b> plate with Dirichlet and Neumann boundary conditions	45 minutes
Point by point Recurrent Neural network (PPRNN)	-ditto-	1 ms
Distributed Recurrent Neural network (DRNN)	-ditto-	0.1 ms
Distributed Artificial Neural network (DANN)	-ditto-	0.1 ms

**Table 3: Representation of max error of Various Thermal Management Application Problems considered using three Novel Deep Learning Based Methods**

Method	Problem Solved	Max error (%)
Point by point Recurrent Neural	2D heat <b>conduction</b> on	2.88

network (PPRNN)	<b>square</b> plate with Dirichlet and Neumann boundary conditions	
Distributed Recurrent Neural network (DRNN)	-ditto-	1.5
Distributed Artificial Neural network (DANN)	-ditto-	0.8
Point by point Recurrent Neural network (PPRNN)2D heat conduction on <b>circular</b> plate with Dirichlet and Neumann boundary conditions	2D heat conduction on <b>circular</b> plate with Dirichlet and Neumann boundary conditions	0.05
Distributed Recurrent Neural network (DRNN)	-ditto-	0.03
Distributed Artificial Neural network (DANN)	-ditto-	0.01
Point by point Recurrent Neural network (PPRNN)2D heat <b>convection</b> on <b>square</b> plate with Dirichlet and Neumann boundary conditions	2D heat <b>convection</b> on <b>square</b> plate with Dirichlet and Neumann boundary conditions	2.88
Distributed Recurrent Neural network (DRNN)	-ditto-	1.4
Distributed Artificial Neural network (DANN)	-ditto-	0.85
Point by point Recurrent Neural network (PPRNN)	2D heat <b>convection</b> on <b>circular</b> plate with Dirichlet and Neumann boundary conditions	1.2
Distributed Recurrent Neural network (DRNN)	-ditto-	0.7
Distributed Artificial Neural network (DANN)	-ditto-	0.3
Point by point Recurrent Neural network (PPRNN)	2D heat <b>radiation</b> on <b>square</b> plate with Dirichlet and Neumann boundary conditions	2.25
Distributed Recurrent Neural network (DRNN)	-ditto-	1.8
Distributed Artificial Neural network (DANN)	-ditto-	1.1
Point by point Recurrent Neural network (PPRNN)2D heat	2D heat <b>radiation</b> on <b>circular</b> plate with Dirichlet and	0.5

<b>radiation</b> on <b>circular</b> plate with Dirichlet and Neumann boundary conditions	Neumann boundary conditions	
Distributed Recurrent Neural network (DRNN)	-ditto-	0.3
Distributed Artificial Neural network (DANN)	-ditto-	0.08

**WE CLAIM:**

1. A method **200** of solving a heat transport problem over an object characterized by a geometry, using a hardware multi-threading process, the hardware comprising: a processor configured to run a training model, a first number of storage process units configured to store input data, a second number of memory operation units configured to store output data, and a hardware switch configured to minimize idle time of the processor, the method comprising:

providing **(201)** a geometry and associated boundary conditions and discretizing the geometry into a grid, wherein the grid comprises a number of grid points;

specifying **(202)** temperature or heat flow conditions at the boundary surrounding the geometry and an initial condition at each grid point;

solving **(203)** a heat flow equation selected from one of conduction, convection or radiation for the geometry and the associated boundary conditions to obtain a temperature, or a heat flow rate, or both at each grid point at steady state;

storing **(204)** the solution for each grid point in a training database;

training **(205)** a model selected from a PPRNN, a DRNN or a DANN model using the training database;

inputting **(206)** a modified boundary condition or initial condition or both associated with the geometry; and,

generating **(207)** a temperature, a heat flow rate at both at each grid point corresponding to the modified boundary condition or initial condition.

2. The method as claimed in claim 1, wherein the training **400** comprises:

calling **(402)** a storage process unit for each input point  $i$  in an individual unit;

storing **(403)** all data points  $i=1, 2, 3 \dots n$  in  $j=1, 2, 3 \dots n$  storage units in a single logic call using AND logic enabled function;

calling(404) multi-threading/distributed RNN and executing the same;  
 calling(405) AND logic enabled machine code upon each point  $i=1, 2, 3 \dots n$  in memory unit  $k=1, 2, 3 \dots n$  with zero time lapse and zero  $k$  to  $k$  communication;  
 storing (406) output RNN data of each output point  $p$ , corresponding to each input point  $i=1, 2, 3 \dots n$  in  $j=1, 2, 3 \dots n$  storage units; and  
 repeating(407) the steps (i) to (v) for  $m$  samples and perform computations and storage for each  $m=1, 2, 3 \dots$  sample in the geometry.

3. The method as claimed in claim 1, wherein the heat flow equation solved is a conduction equation wherein the temperature  $T$  is given by:

$$\frac{1}{a} \cdot \frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} + \frac{q}{k} \dots \dots \dots (1)$$

where  $x, y,$  and  $z$  represent Cartesian coordinates,  $q$  is rate of heat generation,  $k$  is thermal conductivity and  $a$  is a product of density and specific heat capacity of the material.

4. The method as claimed in claim 1, wherein the heat flow equation solved is a convection equation given by:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + h \cdot \frac{\partial T}{\partial y} = 0 \dots \dots \dots (2)$$

5. The method as claimed in claim 1, wherein the heat flow equation solved is a radiation equation given by:

$$d_z \rho c_p u \cdot \nabla T + \nabla \cdot q = d_z Q + q_0 + d_z Q_{ted} + d_z Q_R \dots \dots \dots (3)$$

6. The method as claimed in claim 1, wherein the solving a heat flow equation comprises using an analytical solution, a finite element method solution (FEM) or finite difference method (FDM) solution to generate training data set for each input data  $i$  for each sample  $m$  in a geometry  $t$ .

7. The method as claimed in claim 1, wherein the model is a PPRNN model wherein:

$$PPRNN = \oint_{\Omega} \int_{j=1}^M \tanh(h_{j_i} + b_{2_i}) dj d\Omega_i \dots \dots \dots (4)$$

where

$$h_{j_i} = W_{1_i} \cdot h_{j-1_i} + W_{2_i} \cdot x_{j-1_i} + b_{1_i} \dots \dots \dots (5)$$

$x$  is the input,  $h$  is the hidden cell state and  $W_1$ ,  $b_1$  and  $W_2$ ,  $b_2$  are the weight and bias matrices for hidden-hidden and input-hidden connections,  $\Omega$  is the domain of interest,  $M$  is the number of examples for training,  $\tanh$  is an activation function.

8. The method as claimed in claim 7, comprising obtaining the weight and bias matrices by minimizing mean square error between the predicted and input temperatures for each mesh independent point  $i$  for an example  $m$  given by:

$$MSE_i = \frac{1}{m} \sum_{n=1}^m |T_{Pred_i} - T_{Act_i}|^2 \dots \dots \dots (6)$$

9. The method as claimed in claim 1, wherein the model is a DRNN model wherein:

$$DRNN = \forall \oint_{\Omega} \int_{j=1}^M \tanh(h_{j_i} + b_{2_i}) dj d\Omega_i \dots \dots \dots (7)$$

where,

$$h_j = W_1 \cdot h_{j-1} + W_2 \cdot x_{j-1} + b_1 \dots \dots \dots (8)$$

$x$  is the input,  $h$  is the hidden cell state and  $W_1$ ,  $b_1$  and  $W_2$ , are the weight and bias matrices for hidden-hidden and input-hidden connections,  $\Omega$  is the domain of interest,  $M$  is the number of examples for training,  $\tanh$  is an activation function.

10. The method as claimed in claim 9, comprising obtaining the weight and bias matrices by minimizing mean square error between the predicted and input temperatures for each mesh independent point  $i$  for an example  $m$  given by:

$$MSE = \sum_{n=1}^m \|T_{Pred} - T_{Act}\|^2 \dots\dots\dots(9)$$

11. The method as claimed in claim 1, wherein the model is a DANN model wherein:

$$DANN = \forall \begin{cases} 0 & \text{if } x \leq 0 \\ \int_{j=1}^M (h_j + b_2) dx & \text{if } x > 0 \end{cases} \dots\dots\dots(10)$$

where,

$$h_j = W_1 \cdot h_{j-1} + W_2 \cdot x_{j-1} + b_1 \dots\dots\dots(8)$$

$x$  is the input,  $h$  is the hidden cell state and  $W_1$ ,  $b_1$  and  $W_2$ , are the weight and bias matrices for hidden-hidden and input-hidden connections, and  $M$  is the number of examples for training.

12. The method as claimed in claim 11, comprising obtaining the weight and bias matrices by minimizing mean square error between the predicted and input temperatures for each mesh independent point  $i$  for an example  $m$  given by:

$$MSE_i = \frac{1}{m} \sum_{n=1}^m |T_{Pred_i} - T_{Act_i}|^2 \dots\dots\dots(6)$$

13. A system for solving a heat transport problem over an object characterized by a geometry, the system comprising:

- a hardware switch **(102)**;
- a processor **(101)** coupled to the hardware switch **(102)** to run a neural network engine, wherein the processor **(101)** is configured to:



receive a geometry and associated boundary conditions and discretize the geometry into a grid, wherein the grid comprises a number of grid points;  
receive temperature or heat flow conditions at the boundary surrounding the geometry and an initial condition at each grid point;  
solve a heat flow equation selected from one of conduction, convection or radiation for the geometry and the associated boundary conditions to obtain a temperature, or a heat flow rate, or both at each grid point at steady state;  
store the solution for each grid point in a training database;  
train a model selected from a PPRNN, a DRNN or a DANN model using the training database;  
receive a modified boundary condition or initial condition or both associated with the geometry as input to the trained model; and,  
generate a temperature, a heat flow rate at both at each grid point corresponding to the modified boundary condition or initial condition.

14. The system as claimed in claim 13, comprising a storage process unit (**110**) comprising a plurality of sub-storage process unit (**111-1, 111-2, ..., 111-X**) configured to store one or more data subsets of geometry and associated boundary conditions, and temperature and heat-flow conditions.

15. The system as claimed in claim 13, wherein the processor (**101**) is configured to divide the transport problem solution into a plurality of threads for concurrent execution.

16. The system as claimed in claim 15, wherein the hardware switch (**102**) is configured to execute the one or more threads in parallel and allocate the generated temperature and heat flow rate in memory operation units **120**.

17. The system as claimed in claim 15, wherein the memory operation units **120** comprises a plurality of memory operation units (**121-1, 121-2, 121-3....., 121-X**) configured to store generated temperature and heat flow rate.

Dated this 12<sup>th</sup> Day of October, 2020.

Sd.- Dr V. SHANKAR IN/PA-1733  
For Indian Institute of Technology Madras (IIT Madras)

**MACHINE LEARNING, DEEP LEARNING AND ARTIFICIAL  
INTELLIGENCE FOR PHYSICAL TRANSPORT PHENOMENON IN  
THERMAL MANAGEMENT**

ABSTRACT

Disclosed is a method of solving a heat transport problem over an object characterized by a geometry using a hardware multi-threading process. The method includes geometry and associated boundary conditions and discretizing the geometry into a grid. The method includes specifying temperature or heat flow conditions at the boundary surrounding the geometry and an initial condition at each grid point. A heat flow equation selected from one of conduction, convection or radiation for the geometry is selected and associated boundary conditions, is solved using a known method such as analytical, FDM or FEM to obtain a temperature at each grid point. Next, the solution for each grid point is used for training a neural network model. The neural network model may be a point to point recurrent neural network model. The trained model may be used to predict temperatures for any modified boundary condition associated with the geometry.

**FIG. 2**

**F O R M 2**

**THE PATENTS ACT, 1970  
(39 of 1970)**

**COMPLETE SPECIFICATION**  
**(See section 10 and rule 13)**

**TITLE**

**MACHINE LEARNING, DEEP LEARNING AND ARTIFICIAL  
INTELLIGENCE FOR PHYSICAL TRANSPORT PHENOMENON IN  
THERMAL MANAGEMENT**

**INVENTORS:**

**NANDIGANA, Vishal V.R.** Indian Citizen  
**DASARI, Ananyananda,** Indian Citizen  
Department of Mechanical Engineering

Indian Institute of Technology Madras,  
Chennai-600036, India

**APPLICANTS**

**Indian Institute of Technology Madras (IIT Madras)**  
Office of the Dean ICSR  
Chennai – 600036, India

**THE FOLLOWING SPECIFICATION PARTICULARLY DESCRIBES THE  
INVENTION AND THE MANNER IN WHICH IT IS TO BE PERFORMED**

**MACHINE LEARNING, DEEP LEARNING AND ARTIFICIAL  
INTELLIGENCE FOR PHYSICAL TRANSPORT PHENOMENON IN  
THERMAL MANAGEMENT**

CROSS-REFERENCES TO RELATED APPLICATIONS

[0001] This patent is a complete specification filing of patent application no. 201941036792 entitled Machine Learning, Deep Learning and Artificial Intelligence for physical transport phenomenon in thermal management filed on October 12, 2019.

FIELD OF THE INVENTION

[0002] The present invention relates in general to methods of solving transport problems and in particular to AI or neural learning based methods of solving these problems.

DESCRIPTION OF RELATED ART

[0003] Artificial Intelligence (AI) has created a great value and buzz across the technological landscape in recent years. Artificial Intelligence has found use in diverse fields such as language processing, text completion, image reconstruction, autonomous cars, voice assistance, and in the fields of robotics, drone movement control, virtual reality applications, video games, graphics simulations, etc. However, the application of such techniques have not been explored to solve engineering problems in thermal management, electronic cooling industries, automobile industries like fluid dynamics predictions over a bonnet or inside the engine, aerospace industries like aerodynamics and fluid dynamics problems across an aero-foil or wing.

[0004] In thermal management, physical quantities such as temperature, pressure, velocity, etc. of a body depend on many factors, including but not limited to spatio-temporal variables. The three-dimensional pose, time and material properties such as Heat Capacity, Young's Modulus, etc. are some of the factors that determine the values of

the physical quantities in processes occurring around us. In many cases, these quantities are also interdependent among themselves, creating coupled situations. Differential equations formulate the basic physical laws governing such phenomena, and can model these dependencies. However, the multi-variable dependency of most quantities makes it infeasible to use Ordinary differential equations, except in extremely simplified cases. Partial differential equations (PDE) can both model the large scale, multi-variable dependency, as well as inter-variable dependencies. Depending on the process, a single PDE or a system of simultaneous PDEs is solved to determine the value of each physical quantity at a particular location and time.

**[0005]** Generally, PDEs are solved by analytical methods such as separation of variables or by assuming a suitable form for the solution. PDEs are also solved numerically, by dividing the area of interest into smaller parts, and satisfying the conservation laws between neighboring parts using a variety of specialized algorithms. Numerical solutions iterate the values of the quantities a large number of times to satisfy the conservation laws as closely as possible.

**[0006]** However, the discretization of the geometry into smaller parts or grid cells introduces a different set of problems. Firstly, the generation of the solution is dependent on the convergence of the algorithm, i.e., the reduction of the error term to within acceptable range. Convergence is dependent upon many factors, such as initial values, boundary conditions, geometry shape and size, mesh size, etc. Secondly, the increase in the number of grid cells causes a corresponding increase in the time required for convergence. In case of convergence of the algorithm, a finer mesh usually produces a better solution, but with the penalty of higher computational time and resources. This causes huge delays in running advanced simulations, which involve very fine or in some cases, adaptive meshes, which have different mesh sizes at different locations in the

geometry. Complex geometries further aggravate the problem. Thus, there is a need to find methods to reduce the time and effort in solving large computational problems.

**[0007]** The application of neural network models for solving and predicting partial differential equations is an emerging domain, initiated in 1990s. Various publications such as G.E. Karniadakis et al and MaziarRaissi et al work on solving and predicting non-linear partial differential equations using neural networks as approximated solutions to the partial differential equations has spearheaded the development in this field. Recently developed auto-differentiation techniques over the neural networks to approximate the other non-linear terms of the equations have further contributed to the gradual acceptance of neural network solvers for PDEs. However, there are no publications that solve physical transport phenomena problems without incorporating the underlying physics based PDEs, which makes the solutions complex and computation intensive.

**[0008]** The invention proposes devices and methods to address some of the drawbacks discussed here.

### **SUMMARY OF THE INVENTION**

**[0009]** According to various embodiments, a method of solving a heat transport problem over an object characterized by a geometry, using a hardware multi-threading process is disclosed. The hardware comprising: a processor configured to run a training model, a first number of storage process units configured to store input data, a second number of memory operation units configured to store output data, and a hardware switch configured to minimize idle time of the processor. The disclosed method comprising: providing a geometry and associated boundary conditions and discretizing the geometry into a grid, wherein the grid comprises a number of grid points. The method includes specifying temperature or heat flow conditions at the boundary surrounding the geometry

and an initial condition at each grid point. The method next includes solving a heat flow equation selected from one of conduction, convection or radiation for the geometry and the associated boundary conditions to obtain a temperature, or a heat flow rate, or both at each grid point at steady state. Next, the method includes storing the solution for each grid point in a training database; training a model selected from a PPRNN, a DRNN or a DANN model using the training database; inputting a modified boundary condition or initial condition or both associated with the geometry; and, generating a temperature, a heat flow rate at both at each grid point corresponding to the modified boundary condition or initial condition.

**[0010]** In various embodiments, the training includes calling a storage process unit for each input point  $i$  in an individual unit; storing all data points  $i=1, 2, 3 \dots n$  in  $j=1, 2, 3 \dots n$  storage units in a single logic call using AND logic enabled function; calling multi-threading/distributed RNN and executing the same; calling AND logic enabled machine code upon each point  $i=1, 2, 3 \dots n$  in memory unit  $k=1, 2, 3 \dots n$  with zero time lapse and zero  $k$  to  $k$  communication; storing output RNN data of each output point  $p$ , corresponding to each input point  $i=1, 2, 3 \dots n$  in  $j=1, 2, 3 \dots n$  storage units; and repeating the steps (i) to (v) for  $m$  samples and perform computations and storage for each  $m=1, 2, 3 \dots$  sample in the geometry.

**[0011]** In various embodiments, the heat flow equation solved is a conduction equation wherein the temperature  $T$  is given by:  $\frac{1}{a} \cdot \frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} + \frac{q}{k}$  where  $x$ ,  $y$ , and  $z$  represent Cartesian coordinates,  $q$  is rate of heat generation,  $k$  is thermal conductivity and  $a$  is a product of density and specific heat capacity of the material. In some embodiments, the heat flow equation solved is a convection equation given by:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + h \cdot \frac{\partial T}{\partial y} = 0.$$

In some embodiments, the heat flow equation solved is a radiation

$$\text{equation given by: } d_z \rho c_p u \cdot \nabla T + \nabla \cdot q = d_z Q + q_0 + d_z Q_{ted} + d_z Q_R.$$



**[0012]** In some embodiments, solving a heat flow equation comprises using an analytical solution, a finite element method solution (FEM) or finite difference method (FDM) solution to generate training data set for each input data  $i$  for each sample  $m$  in a geometry  $t$ . In various embodiments, the model is a PPRNN model wherein:  $PPRNN = \oint_{\Omega} \int_{j=1}^M \tanh(h_{j_i} + b_{2_i}) dj d\Omega_i$  where  $h_{j_i} = W_{1_i} \cdot h_{j-1_i} + W_{2_i} \cdot x_{j-1_i} + b_{1_i}x$  is the input,  $h$  is the hidden cell state and  $W_1, b_1$  and  $W_2, b_2$  are the weight and bias matrices for hidden-hidden and input-hidden connections,  $\Omega$  is the domain of interest,  $M$  is the number of examples for training,  $\tanh$  is an activation function.

**[0013]** In various embodiments, the method includes obtaining the weight and bias matrices by minimizing mean square error between the predicted and input temperatures for each mesh independent point  $i$  for an example  $m$  given by:  $MSE_i = \frac{1}{m} \sum_{n=1}^m |T_{Pred_i} - T_{Act_i}|^2$ . In various embodiments, the model is a DRNN model wherein:  $DRNN = \forall \oint_{\Omega} \int_{j=1}^M \tanh(h_{j_i} + b_{2_i}) dj d\Omega_i$  where,  $h_j = W_1 \cdot h_{j-1} + W_2 \cdot x_{j-1} + b_1x$  is the input,  $h$  is the hidden cell state and  $W_1, b_1$  and  $W_2, b_2$  are the weight and bias matrices for hidden-hidden and input-hidden connections,  $\Omega$  is the domain of interest,  $M$  is the number of examples for training,  $\tanh$  is an activation function.

**[0014]** In some embodiments, the method includes obtaining the weight and bias matrices by minimizing mean square error between the predicted and input temperatures for each mesh independent point  $i$  for an example  $m$  given by:  $MSE = \sum_{n=1}^m \|T_{Pred} - T_{Act}\|^2$ . In various embodiments, the model is a DANN model wherein:  $DANN =$

$$\forall \begin{cases} 0 & \text{if } x \leq 0 \\ \int_{j=1}^M (h_j + b_2) dj, & \text{if } x > 0 \end{cases} \text{ where, } h_j = W_1 \cdot h_{j-1} + W_2 \cdot x_{j-1} + b_1x \text{ is the input, } h \text{ is the}$$

hidden cell state and  $W_1, b_1$  and  $W_2, b_2$  are the weight and bias matrices for hidden-hidden

and input-hidden connections, and  $M$  is the number of examples for training. In various embodiments, the method includes obtaining the weight and bias matrices by minimizing mean square error between the predicted and input temperatures for each mesh independent point  $i$  for an example  $m$  given by:  $MSE_i = \frac{1}{m} \sum_{n=1}^m |T_{Pred_i} - T_{Act_i}|^2$ .

**[0015]** According to another embodiment, a system for solving a heat transport problem over an object characterized by a geometry is disclosed. The system includes a hardware switch configured to execute a multi-threading process, and a processor coupled to the hardware switch to run a neural network engine. The processor is configured to receive a geometry and associated boundary conditions and discretize the geometry into a grid, wherein the grid comprises a number of grid points. Temperature or heat flow conditions at the boundary surrounding the geometry and an initial condition at each grid point are received. The processor solves a heat flow equation selected from one of conduction, convection or radiation for the geometry and the associated boundary conditions to obtain a temperature, or a heat flow rate, or both at each grid point at steady state and stores the solution for each grid point in a training database. Next, the processor trains a model selected from a PPRNN, a DRNN or a DANN model using the training database. The trained model is configured to receive a modified boundary condition or initial condition or both associated with the geometry to generate a temperature, a heat flow rate at both at each grid point corresponding to the modified boundary condition or initial condition.

**[0016]** In various embodiments, the system includes a storage process unit comprising a plurality of sub-storage process unit configured to store one or more data subsets of geometry and associated boundary conditions, and temperature and heat-flow conditions. The processor is configured to divide the transport problem solution into a plurality of threads for concurrent execution. In various embodiments, the hardware

switch is configured to execute the plurality of threads in parallel and allocate the generated temperature and heat flow rate in memory operation units. The memory operation units include a plurality of memory operation units configured to store generated temperature and heat flow rate.

[0017] This and other aspects are disclosed herein.

### **BRIEF DESCRIPTION OF THE DRAWINGS**

[0018] The invention has other advantages and features which will be more readily apparent from the following detailed description of the invention and the appended claims, when taken in conjunction with the accompanying drawings, in which:

[0019] FIG. 1 illustrates a simple block diagram of a system for solving a transport problem using Neural Network (NN), according to one embodiment of the present subject matter.

[0020] FIG. 2 illustrates a method of solving a transport problem, machine learning or deep learning methods using Neural Network (NN), according to one embodiment of the present subject matter.

[0021] FIG. 3 illustrates a block diagram of a system for solving a transport problem using Neural Network (NN) engine, according to another embodiment of the present subject matter.

[0022] FIG. 4 illustrates a flow diagram of a method of multi-threading process for DRNN, according to another embodiment of the present subject matter.

[0023] FIG. 5A and FIG. 5B illustrate a simple architecture of recurrent neural network and a point-to-point recurrent neural network, according to one embodiment of the present subject matter.

[0024] FIG. 6A and FIG. 6B illustrate a simple architecture of artificial neural network and a deep artificial neural network, according to one embodiment of the present subject matter.

[0025] FIG. 7A and FIG. 7B illustrate a simple architecture of recurrent neural network and a deep recurrent neural network, according to one embodiment of the present subject matter.

[0026] FIG. 8A – FIG. 8C illustrate dirichlet condition for square geometry, circular geometry and Neumann condition for square and circular geometries.

[0027] FIG. 9A – FIG. 9C illustrate a comparison between truth and predicted solution for square geometry domain conduction with Dirichlet boundary condition.

[0028] FIG. 10A – FIG. 10C illustrate a comparison between truth and predicted solution for circular geometry domain conduction with Neumann boundary condition.

[0029] FIG. 11A – FIG. 11C illustrate a comparison between truth and predicted solution for square geometry domain convection with Dirichlet boundary condition.

[0030] FIG. 12A – FIG. 12C illustrate a comparison between truth and predicted solution for circular geometry domain convection with Neumann boundary condition.

[0031] FIG. 13A – FIG. 13C illustrate a comparison between truth and predicted solution for square geometry domain radiation with Dirichlet boundary condition.

[0032] FIG. 14A – FIG. 14C illustrate a comparison between truth and predicted solution for circular geometry domain radiation with Neumann boundary condition.

[0033] Referring to the drawings, like numbers indicate like parts throughout the views.

### **DETAILED DESCRIPTION OF THE EMBODIMENTS**

[0034] While the invention has been disclosed with reference to certain embodiments, it will be understood by those skilled in the art that various changes may be made and equivalents may be substituted without departing from the scope of the invention. In addition, many modifications may be made to adapt to a particular situation or material to the teachings of the invention without departing from its scope.

[0035] Throughout the specification and claims, the following terms take the meanings explicitly associated herein unless the context clearly dictates otherwise. The meaning of "a", "an", and "the" include plural references. The meaning of "in" includes "in" and "on." Referring to the drawings, like numbers indicate like parts throughout the views. Additionally, a reference to the singular includes a reference to the plural unless otherwise stated or inconsistent with the disclosure herein.

[0036] The invention in its various embodiments discloses systems and methods of solving a transport problem across a geometrical object using machine learning. In some embodiments, the transport problem may be a heat flow problem in 2 or 3 dimensions. A system **100** for implementing the method of the invention is disclosed in FIG. 1. The system **100** may include a processor **101** configured to run a neural network engine, a first number of storage process units **110** configured to store input data, a second number of memory operation units **120** configured to store output data, and a hardware switch **102** configured to minimize idle time of the processor. The hardware switch **102** may execute a multi-threading process on receiving instructions from the processor to solve the transport problem while minimizing idle time of the processor.

[0037] In various embodiments, the processor**101** may be configured to divide the process of solving the transport problem into a plurality of threads or tasks for

concurrent execution. The storage process unit **110** may include a plurality of sub-storage process units **111-1, 111-2, 111-3, ...,111-X** configured to store the input data. In some embodiments, each one of the plurality of sub-storage process unit **111-1, 111-2,...,111-X** may be configured a subset of the input data. For instance, the input data may be classified into a plurality of subsets based on a metric, such as timestamp.

**[0038]** The processor **101** running the neural network engine may receive the input data from the storage process units **110** and generate the output data. In some embodiments, the neural network engine may include multiple layers for computing the output concurrently. The hardware switch **102** may be configured to execute the multiple threads or tasks in parallel and retrieve the output data. The output data may be allocated and stored in the memory operation units **120**, which may include a plurality of memory operation units **121-1, 121-2, 121-3....., 121-X**,

**[0039]** In various embodiments a method of solving or computing solution to a heat transport problem over an object characterized by a geometry and having a boundary is disclosed with reference to FIG. 2. The method may use a hardware system as disclosed in FIG. 1. The method may include a first step of providing **(201)** a discretized geometry. The object geometry may be discretized into multiple elements forming a grid, which may include a number of grid points. In various embodiments, the discretization may depend on the complexity of the geometry. The number of elements or element density or both may be varied depending on whether the object has discontinuity in section, or material characteristics or both.

**[0040]** In the next step **202**, temperature or heat flow conditions may be specified at the boundary surrounding the geometry. The boundary condition may be a Dirichlet boundary condition or a Neumann boundary condition. An initial condition may also be specified at each grid point. The method may then involve the step **203** of solving a heat flow equation for the geometry and corresponding boundary conditions to obtain

a temperature, or a heat flow rate, or both at each grid point at steady state. In various embodiments, the heat flow equation may be selected from one of a conduction equation, a convection equation or a radiation equation as illustrated further.

[0041] In the next step **204**, the method involves storing the solution for each grid point in a training database. The training database may be configured to be trained using a variety of different models such as a Point by Point Recurrent Neural Network (PPRNN), a Distributed DANN (DANN) or a Distributed Recurrent Neural Network (DRNN). In step **205**, the method may involve training a model selected from a PPRNN, a DRNN or a DANN model using the training database. In the next step **206**, the method involves inputting a modified boundary condition or initial condition at the geometry, and running the trained model. In a final step **207**, a temperature or a heat flow rate is generated, at each grid point corresponding to the modified boundary or heat flow condition across the geometry.

[0042] Another block diagram of the system **300** is illustrated in FIG. 3, according to another embodiment of the present subject matter. The system **300** may include one or more processors **101**, one or more memory units **302**, a neural network (NN) engine **304**, a communication unit **306**, and a display unit **308**. The one or more processors **101** may be configured to receive a geometry having a boundary and discretize the geometry into a grid comprising a number of points. The processor **101** may also receive specific boundary conditions surrounding the geometry and an initial condition at each grid point. The processor **101** may then solve a heat flow equation for the geometry and boundary conditions to obtain a temperature, or a heat flow rate or both at each grid point. The solution for each grid point may be stored in a training database. The processor **101** may train a model selected from a neural network engine **304**, such as a PPRNN engine using the training database. In some embodiments, the neural network engine **304** may be a point to point neural network

PPRNN engine, deep recurrent neural network DRNN engine or a deep artificial neural network DANN engine.

[0043] In various embodiments, the NN engine **304** may include a training component **310**, a validating component **312**, and a testing component **314**. Each component of the NN engine **304** may receive different datasets. For example, the geometry and associated boundary condition data along with truth values comprising heat transfer solutions may be segregated into one or more of a training dataset and a validation dataset. In some embodiments, the training dataset may encompass 80% and the validation dataset may encompass 20% of the received data. In various embodiments, the training data may be accessed from a remote database. In other embodiments, the training dataset may encompass 70% of the received data, the validation dataset may include 15% of the received data, and the test dataset may encompass 15% of the received data. In various embodiments, the communication unit **306** may be configured to transmit the predicted solutions to one or more external components. In some embodiments, the display unit **308** may be configured to display the predicted solutions to a user.

[0044] In various embodiments of the method, a method **400** of training a database using a neural network model is disclosed. The model trained may in some embodiments be a PPRNN, a DRNN or a DANN model. The method comprises the steps of **401** calling a storage process unit for each input point  $i$  in an individual unit, storing **402** all data points  $i=1, 2, 3 \dots n$  in  $j=1, 2, 3 \dots n$  storage units in a single logic call using AND logic enabled function, calling **403** and a multi-threading/distributed method and executing the same. The method in the next steps **404** involves calling AND logic enabled machine code upon each point  $i=1, 2, 3 \dots n$  in memory unit  $k=1, 2, 3 \dots n$  with zero time lapse and zero  $k$  to  $k$  communication and **405** storing output data of each output point  $p$ , corresponding to each input point  $i=1, 2, 3 \dots n$  in  $j=1, 2,$



3... n storage units. The method may further involve in step **406**, repeating the steps **401** to **405** for m samples and perform computations and storage of output data for each m=1, 2, 3... sample in the geometry.

[0045] In some embodiments of the method **200**, the heat flow equation solved may be a conduction equation wherein the temperature T is given by:

$$\frac{1}{a} \cdot \frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} + \frac{q}{k} \dots \dots \dots (1)$$

where x, y, and z represent Cartesian coordinates, q is rate of heat generation, k is thermal conductivity and a is a product of density and specific heat capacity of the material.

[0046] In some embodiments of the method **200**, the heat flow equation solved is a convection equation given by:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + h \cdot \frac{\partial T}{\partial y} = 0 \dots \dots \dots (2)$$

[0047] In some embodiments of the method **200**, the heat flow equation solved is a radiation equation given by:

$$d_z \rho c_p u \cdot \nabla T + \nabla \cdot q = d_z Q + q_0 + d_z Q_{ted} + d_z Q_R \dots \dots \dots (3)$$

[0048] In step **203** in FIG. 2 may involve solving a heat flow equation using a suitable known method to generate the training data. The method of solving may include using an analytical solution, a finite element method solution (FEM) or a finite difference method (FDM) solution to generate training data set for each input data i for each sample m in a geometry t.

[0049] In some embodiments of the method **400**, the training model is a PPRNN model wherein:

$$PPRNN = \oint_{\Omega} \int_{j=1}^M \tanh(h_{ji} + b_{2i}) dj d\Omega_i \dots \dots \dots (4)$$

where

$$h_{j_i} = W_{1_i} \cdot h_{j-1_i} + W_{2_i} \cdot x_{j-1_i} + b_{1_i} \dots \dots \dots (5)$$

$x$  is the input,  $h$  is the hidden cell state and  $W_1, b_1$  and  $W_2, b_2$  are the weight and bias matrices for hidden-hidden and input-hidden connections,  $\Omega$  is the domain of interest,  $M$  is the number of examples for training, and  $\tanh$  is an activation function.

[0050] The method 400 may further involve obtaining the weight and bias matrices for the PPRNN model by minimizing mean square error between the predicted and input temperatures for each mesh independent point  $i$  for an example  $m$  given by:

$$MSE_i = \frac{1}{m} \sum_{n=1}^m |T_{Pred_i} - T_{Act_i}|^2 \dots \dots \dots (6)$$

[0051] In some embodiments of the method 400, the training model is a DRNN model wherein:

$$DRNN = \forall \phi_{\Omega} \int_{j=1}^M \tanh(h_{j_i} + b_{2_i}) dj d\Omega_i \dots \dots \dots (7)$$

where,  $h_j$  is as given in (5),  $x$  is the input,  $h$  is the hidden cell state and  $W_1, b_1$  and  $W_2, b_2$  are the weight and bias matrices for hidden-hidden and input-hidden connections,  $\Omega$  is the domain of interest,  $M$  is the number of examples for training,  $\tanh$  is an activation function.

[0052] The method 400 may further involve obtaining the weight and bias matrices of the DRNN model by minimizing mean square error between the predicted and input temperatures for each mesh independent point  $i$  for an example  $m$  given by:

$$MSE = \sum_{n=1}^m \|T_{Pred} - T_{Act}\|^2 \dots \dots \dots (8)$$

[0053] In some embodiments of the method 400, the training model is a DANN model wherein:

$$DANN = \forall \begin{cases} 0 & \text{if } x \leq 0 \\ \int_{j=1}^M (h_j + b_2) dj & \text{if } x > 0 \end{cases} \dots \dots \dots (9)$$

where,  $h_j$  is as given in (5),  $x$  is the input,  $h$  is the hidden cell state and  $W_1$ ,  $b_1$  and  $W_2$ , are the weight and bias matrices for hidden-hidden and input-hidden connections, and  $M$  is the number of examples for training.

[0054] The method **400** may further involve obtaining the weight and bias matrices by minimizing mean square error between the predicted and input temperatures for each mesh independent point  $i$  for an example  $m$  given by equation (6).

[0055] A simple architecture of recurrent neural network and a point-to-point recurrent neural network are illustrated in FIG. 5A and FIG. 5B, according to one embodiment of the present subject matter. As shown, the recurrent neural network architecture **500** may include an input layer **503**, a plurality of LSTMs **504-N**, and an output layer **505**. The input layer may receive  $5 \times 1$  dimension vector comprising temperatures. The hidden layer **504-N** may receive the input temperature and feedback values as inputs to predict the temperature. The output layer **505** may include a node representing an output  $1 \times 1$  vector representing the predicted temperature. The PPRNN architecture **502** may include a plurality of point-to-point recurrent neural networks PPRNN **506**, each receiving training examples **501** and a bias vector input **507**. The outputs of the PPRNN engines are provided to the output layer **508** as a forward pass to predict the temperatures.

[0056] A simple architecture of artificial neural network and a deep artificial neural network are illustrated in FIG. 6A and FIG. 6B, according to one embodiment of the present subject matter. As shown, the ANN **600** may include an input layer **603**, one or more hidden layers **604-1**, **604-2** and an output layer **604**. The deep artificial neural network DANN architecture **602** may include a DANN **606**, which receives training examples **603** and a bias vector input **607**. The outputs of the DANN engines **606** may be provided to the output layer **608**.

[0057] Further, a simple architecture of recurrent neural network and a deep recurrent neural network are illustrated in FIG. 7A and FIG. 7B, according to one embodiment of the present subject matter. As shown, the recurrent neural network architecture **700** may include an input layer **703**, a plurality of LSTMs **704-N**, and an output layer **705**. The input layer may receive 900x4 dimension matrix comprising input temperature data. The hidden layer **704-N** may receive the input temperature and feedback values as inputs to predict the temperature. The output layer **705** may include a node representing an output 900x1 vector representing the predicted temperature. The deep artificial neural network DANN architecture **706** may include a DANN **706**, each receiving training examples **701** and a bias vector input **707**. The outputs of the DANN engines are provided to the output layer **708** as a forward pass to predict the temperatures.

[0058] As disclosed herein, the systems and methods have great computational advantage compared to existing systems and methods of solving these problems. The present methods could save computational time to the extent of  $10^5$  to  $10^9$  times, compared to existing methods. Further, the computational error could be minimized by optimally training the networks. In some instances, the methods may be used to minimize computational error to 0.2% or less of the actual value as predicted by standard numerical or analytical solutions. In some instances, with optimal training using appropriately sized data sets, the error could be minimized to 0.01% or less. A representation of max error of various thermal management problems considered using three deep learning based methods is provided in Table 3 in the examples section.

[0059] The systems and methods illustrated in the foregoing embodiments and the examples are generally applicable to any heat flow problem, whether in 2D or 3D geometry that involve mechanical complexity. Some 3D problems may be approximated using a 2D solution. Although the method is illustrated using 2D heat

flow examples, the systems and methods are equally applicable to the generalized 3D heat flow solution situation. The computational advantages of the method may become more valuable for 3D problems because of the highly computation-intensive nature of 3D heat flow solutions using current numerical methods of solution.

**[0060]** Although the detailed description contains many specifics, these should not be construed as limiting the scope of the invention but merely as illustrating different examples and aspects of the invention. It should be appreciated that the scope of the invention includes other embodiments not discussed herein. Various other modifications, changes and variations which will be apparent to those skilled in the art may be made in the arrangement, operation and details of the system and method of the present invention disclosed herein without departing from the ~~spirit and~~ scope of the invention as described here.

## EXAMPLES

### **[0061] Example 1: Generation of Training Data and Training the Networks**

**[0062]** The following cases as illustrated in Table 1 were determined using PPRNN, DRNN and DANN. Training data for each module was generated using either ANSYS or in house developed finite difference or finite element codes, as summarized in Table 2, which also includes runtime.

**TABLE 1: Heat Flow Problems Considered with Boundary Conditions**

	Dirichlet Boundary Condition	Neumann Boundary Condition	Dirichlet Boundary Condition	Neumann Boundary Condition
Conduction	Square	Square	Circular	Circular
Convection	Square	Square	Circular	Circular
Radiation	Square	Square	Circular	Circular

**[0063]** The above twelve problems considered were solved using each of the 3 methods. In the case of a 2D heat conduction over a square geometry a 25x25 mesh was chosen to solve using a conventional finite difference method (FDM) as illustrative example to compare with the analytical solution for the problems considered.

**[0064]** For Deep Learning methods, training data is essential to make the neural network get trained and result in accurate solution for a new test boundary condition or test example. For the implementation of the PPRNN into the 2D heat conduction problem, the in-house developed FDM solution was used as a training data. In this example problem, 1670 examples were used as training data and 800 examples are used as testing cases. Also, the boundary conditions for each example  $m$ , are given as the biases  $b_1, b_2, b_3$  and  $b_4$ , for all the four falls in the case where Dirichlet BCs are considered. The input  $x_i$  is the mesh point (excluding the boundary condition mesh points) for each of total sample  $m$ , which for this said 25 by 25 mesh points makes 529. 529 Recurrent neural networks are coded in a loop to take in a single mesh point input,  $x_i$  so that there is no dependency between mesh points for a said  $m$ , in developing the new PPRNN for 2D heat conduction application that is also mesh independent and physics independent as the information of the physics like the gradients and their derivatives need not be incorporated. The PPRNN is trained using the tanh activation function, biasing the input and biases with the trained weights. In order to obtain the trained weights, the input data  $x_i$  is compared with PPRNN solution (PPRNN) by minimizing the mean square error (MSE) say between  $T_{\text{Predi}}$  (same as Temperature PPRNN) and  $T_{\text{Input}}$  (same as  $T_{\text{Acti}}$ ) (obtained from Eq. (6)) for each mesh independent point  $i$  for an example  $m$ .

**[0065]** The said error minimization is performed for  $n = 1$  to  $m$  examples and also averaged to calculate the mean. The calculation of the mean squared error is repeated

or looped for each input grid point  $i$ , independent of the other grid point, to make the PPRNN truly mesh independent. L-BFGS optimiser is used to minimize the error for each input point  $i$ . The final weight for each point  $i$  is obtained, is saved and is used to predict the temperature for a new test boundary condition or test example using again PPRNN solution Eq 1. A representation of max error of various thermal management application problems considered using the deep learning neural network based methods is provided in Table 3.

[0066] The disclosed PPRNN method facilitates predictions for sparse data points as the method is not mesh dependent, making it free from calculating physics driven gradients and its derivatives and other higher order PDEs. Our developed PPRNN method is PDE or Physics less model and is also mesh independent for the first time in literature for solving engineering application problems.

[0067] **B. Multi-threading/Distributed RNN (DRNN):** In this method, a new method Multi-threading/Distributed RNN is proposed. Unlike PPRNN, where input data for each point  $i$  was trained using a separate Recurrent neural network. A single Recurrent neural network was used with multi-threading like or distributed like architecture where the same neural network was used for all  $i$  points, for a given example  $m$ , but the points were not communicating between each other in the proposed Multi-threading/Distributed Recurrent Neural Network (DRNN) because the weights while getting trained see these set of points still as independent point  $i$ , while performing the said Mean square error calculation using L-BFGS operation. The mathematical form for DRNN was obtained as shown in Eq. (7). The major advantage of DRNN unlike PPRNN is we need only 15 samples to do the training and test on various geometry and boundary conditions unlike 1000s of training samples needed for PPRNN as discussed before.

[0068] The equations are similar to PPRNN with the only difference the entire input information (in the said 2D heat conduction example, all 529 points) were inputted once for all to find the DRNN solution which however considers individual input  $i$ , independent of its neighboring input point for each sample, of total samples  $m$ , while calculating the weights minimizing Mean squared error (MSE) with L-BFGS operator. The advantage of DRNN method reduces the training time as compared to PPRNN, due to the model consisting of a single DRNN equation rather than multiple RNN equations used in PPRNN solution.

[0069] **B1. Multi-threading/Distributed RNN hardware implementation:**

Hardware multi-threading allows multiple threads to share the RNN operation to each processor in an overlapping fashion to try to utilize the hardware resources efficiently. FIG. 1 shows a storage processor unit in a computer architecture which consists of memory storage units to store the python based method of a input data point  $i$  set to RNN's method using a python based architecture and this data point from RNN is fed to memory processing units in a time lapse controlled fashion using an AND logic enabled with zero time lapse machine symbolic code to enable multi-threading and to perform the RNN calculation for each data point  $i$  independently in individual memory processing units. A schematic flow chart of the discussed multi-threading process is shown in FIG. 4 for better understanding of the method schematic shown in FIG. 2. This way the utilization of a processor was increased with no idle wait time for calculating individual data point  $i$ .

[0070] **C. Multi-threading/Distributed DANN (DANN):** The Multi-threading/Distributed Artificial Neural Network (DANN) is similar to DRNN but the DANN solution uses a RELU/Linear activation function instead of a tanh function. The major advantage of DANN like DRNN is we need only 15 samples to do the training and test on various geometry and boundary conditions unlike 1000 or more of



training samples needed for PPRNN. The mathematical equation for DANN is given in Eq. (9) and (10) as described earlier.

**[0071] Example 2: Prediction of Heat Flow using the Trained Models**

**[0072]** Two different geometries, 2D square/rectangle and 2D circular disc type geometries are considered to test the methods on given rectilinear and curvilinear geometries (see FIG. (8A-8D)).

**[0073] Conduction:** The multidimensional heat conduction equation and its variants were used in diverse fields, such as explaining Brownian motion in probability theory, wave function characteristics in quantum mechanics and also in solving the Black Scholes PDE in financial mathematics. In Cartesian coordinates it is written as Eq. (1).

**[0074]** The equation (1) determines the temperature distribution for a 3D, unsteady heat conduction problem with heat generation. The equation requires six boundary conditions, (two for each axis, x, y, and z) and one initial condition for the time dependence. The rate of heat generation is given by q and the thermal conductivity of the material is given by k. The two-dimensional steady state heat conduction equation, without heat generation is given by

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \dots \dots \dots (10)$$

$$T(0, y) = T1, T(x, 0) = T2, T(L, y) = T3, T(x, L) = T4 \dots \dots \dots (11)$$

**[0075]** This equation (10) requires the four boundary conditions (11) shown, two each for the x and y axes and no initial condition. The conditions shown are for the Dirichlet boundary condition on all four sides of the square domain. The Neumann case was obtained by changing one or more of these boundary conditions to  $\frac{dT}{dx} = a$  or  $\frac{dT}{dy} = b$ . a and b may be constants or even functions. Two Neumann conditions were used on the top and bottom sides of the square and sinusoidal function forms for the

slope values. For the circular domain, only the circumference and centre boundary conditions are required. The Dirichlet case specifies fixed temperatures for both circumference and centre, while for the Neumann case, the circumference temperature slope was defined, while the centre temperature remains fixed. The boundary temperatures for square and circular geometries were between  $T = 100^\circ$  to  $T = 1100^\circ$ . A comparison between truth (finite difference numerical solution) and predicted (PPRNN solution) for square geometry domain conduction with Dirichlet boundary condition (B) is illustrated in FIG. 9A – FIG. 9C. A comparison between truth (finite difference numerical solution) and predicted (PPRNN solution) for circular geometry domain conduction with Neumann boundary condition is illustrated in FIG. 10A – FIG. 10C.

**[0076] B. Heat Convection:** The advection-diffusion equation describes the transfer of physical properties, particles or energy inside a system, due to convection and diffusion. This equation is vital in the solution of other important equations such as Navier-Stokes in fluid dynamics and Black-Scholes in finance. The two-dimensional steady state heat convection equation, without heat generation and involving a non-linear convection term is given by Eq. (2)

**[0077]** The boundary conditions are given in Eqs (12). A comparison between truth (finite difference numerical solution) and predicted (PPRNN solution) for square geometry domain convection with Dirichlet boundary condition (B) is illustrated in FIG. 11A – FIG. 11C. A comparison between truth (finite difference numerical solution) and predicted (PPRNN solution) for circular geometry domain convection with Neumann boundary condition is illustrated in FIG. 12A – FIG. 12C.

**[0078] C. Heat Radiation:** Generation of electromagnetic waves by the thermal motion of all particles above 0 K is termed as radiation. In practice, radiative heat transfer between two surfaces takes place through a medium, whose properties

influence the nature of the waves. Such a medium is called a participating medium. The governing differential equations for radiative heat transfer through a participating medium was the Radiative Transfer Equation given by Eq (3) as described earlier. The equation includes the absorption, emission and scattering of incident radiation due to the participating medium.

$$d_z \rho c_p u \cdot \nabla T + \nabla \cdot q = d_z Q + q_0 + d_z Q_{ted} + d_z Q_R \dots \dots \dots (3)$$

where,  $d_z$  is the domain thickness (unit thickness assumed here),  $\rho$  and  $c_p$  are material properties and  $q$  is the thermal flux, given by

$$q = -d_z k \nabla T \dots \dots \dots (12)$$

$Q_R$  is the sum of the thermal and radiative heat fluxes, given by

$$Q_R = \kappa(G - 4\pi I_b) \dots \dots \dots (13)$$

$$G = \sum_{i=1}^n \omega_i I_i \dots \dots \dots (14)$$

where  $\omega$  is a quadrature weight (a weight used for numerical integration, in the discrete ordinates method). The scattering term is given by

$$S_i \cdot \nabla I_i = \kappa I_b(T) - \beta I_i + \frac{\sigma_s}{4\pi} \sum_{j=1}^N \omega_j I_j \phi(S_j, S_i) \dots \dots \dots (15)$$

where  $\sigma_s$  is the scattering coefficient and  $\beta = \kappa + \sigma_s$ .

**[0079]** To negate the influence of the medium,  $Q_R$  was set to zero, to obtain the pure surface-surface radiation case. COMSOL multi-physics Finite Element based method was used to generate the data. The material used was air at room temperature and pressure. The heat generation and transient terms were set to zero. The boundary conditions were of the form  $T = T_0$  for Dirichlet and  $-n \cdot q = d_z q_0 - n$  for Neumann. A comparison between truth (finite element numerical solution) and predicted (PPRNN solution) for square geometry domain radiation with Dirichlet boundary condition (B)

is illustrated in FIG. 13A – FIG. 13C. A comparison between truth (finite element numerical solution) and predicted (PPRNN solution) for circular geometry domain radiation with Neumann boundary condition (B) is illustrated in FIG. 14A – FIG. 14C.

**Table 2: Comparison of Simulation Times of Various Thermal Management Application Problems Using Existing FDM/FEM Based Commercial Software vs Three Novel Deep Learning Based Methods**

Method	Problem Solved	Simulation Time
Finite Difference method (FDM)	2D heat <b>conduction</b> on <b>square</b> plate with Dirichlet and Neumann boundary conditions	10 minutes
Point by point Recurrent Neural network (PPRNN)	-ditto-	1 ms
Distributed Recurrent Neural network (DRNN)	-ditto-	0.1 ms
Distributed Artificial Neural network (DANN)	-ditto-	0.1 ms
Finite Difference method (FDM)	2D heat conduction on <b>circular</b> plate with Dirichlet and Neumann boundary conditions	15 minutes
Point by point Recurrent Neural network (PPRNN)	-ditto-	1 ms
Distributed Recurrent Neural network (DRNN)	-ditto-	0.1 ms
Distributed Artificial Neural network (DANN)	-ditto-	0.1 ms
Finite Element method (FEM)	2D heat <b>convection</b> on <b>square</b> plate with Dirichlet and Neumann boundary conditions	20 minutes
Point by point Recurrent Neural network (PPRNN)	-ditto-	1 ms
Distributed Recurrent Neural network (DRNN)	-ditto-	0.1 ms
Distributed Artificial	-ditto-	0.1 ms

Neural network (DANN)		
Finite Element method (FEM)	2D heat <b>convection</b> on <b>circular</b> plate with Dirichlet and Neumann boundary conditions	25 minutes
Point by point Recurrent Neural network (PPRNN)	-ditto-	1 ms
Distributed Recurrent Neural network (DRNN)	-ditto-	0.1 ms
Distributed Artificial Neural network (DANN)	-ditto-	0.1 ms
Finite Element method (FEM)	2D heat <b>radiation</b> on <b>square</b> plate with Dirichlet and Neumann boundary conditions	40 minutes
Point by point Recurrent Neural network (PPRNN)	-ditto-	1 ms
Distributed Recurrent Neural network (DRNN)	-ditto-	0.1 ms
Distributed Artificial Neural network (DANN)	-ditto-	0.1 ms
Finite Element method (FEM)	2D heat <b>radiation</b> on <b>circular</b> plate with Dirichlet and Neumann boundary conditions	45 minutes
Point by point Recurrent Neural network (PPRNN)	-ditto-	1 ms
Distributed Recurrent Neural network (DRNN)	-ditto-	0.1 ms
Distributed Artificial Neural network (DANN)	-ditto-	0.1 ms

**Table 3: Representation of max error of Various Thermal Management Application Problems considered using three Novel Deep Learning Based Methods**

Method	Problem Solved	Max error (%)
Point by point Recurrent Neural	2D heat <b>conduction</b> on	2.88

network (PPRNN)	<b>square</b> plate with Dirichlet and Neumann boundary conditions	
Distributed Recurrent Neural network (DRNN)	-ditto-	1.5
Distributed Artificial Neural network (DANN)	-ditto-	0.8
Point by point Recurrent Neural network (PPRNN)2D heat conduction on <b>circular</b> plate with Dirichlet and Neumann boundary conditions	2D heat conduction on <b>circular</b> plate with Dirichlet and Neumann boundary conditions	0.05
Distributed Recurrent Neural network (DRNN)	-ditto-	0.03
Distributed Artificial Neural network (DANN)	-ditto-	0.01
Point by point Recurrent Neural network (PPRNN)2D heat <b>convection</b> on <b>square</b> plate with Dirichlet and Neumann boundary conditions	2D heat <b>convection</b> on <b>square</b> plate with Dirichlet and Neumann boundary conditions	2.88
Distributed Recurrent Neural network (DRNN)	-ditto-	1.4
Distributed Artificial Neural network (DANN)	-ditto-	0.85
Point by point Recurrent Neural network (PPRNN)	2D heat <b>convection</b> on <b>circular</b> plate with Dirichlet and Neumann boundary conditions	1.2
Distributed Recurrent Neural network (DRNN)	-ditto-	0.7
Distributed Artificial Neural network (DANN)	-ditto-	0.3
Point by point Recurrent Neural network (PPRNN)	2D heat <b>radiation</b> on <b>square</b> plate with Dirichlet and Neumann boundary conditions	2.25
Distributed Recurrent Neural network (DRNN)	-ditto-	1.8
Distributed Artificial Neural network (DANN)	-ditto-	1.1
Point by point Recurrent Neural network (PPRNN)2D heat	2D heat <b>radiation</b> on <b>circular</b> plate with Dirichlet and	0.5

<b>radiation</b> on <b>circular</b> plate with Dirichlet and Neumann boundary conditions	Neumann boundary conditions	
Distributed Recurrent Neural network (DRNN)	-ditto-	0.3
Distributed Artificial Neural network (DANN)	-ditto-	0.08

**WE CLAIM:**

1. A method **200** of solving a heat transport problem over an object characterized by a geometry, using a hardware multi-threading process, the hardware comprising: a processor configured to run a training model, a first number of storage process units configured to store input data, a second number of memory operation units configured to store output data, and a hardware switch configured to minimize idle time of the processor, the method comprising:

providing **(201)** a geometry and associated boundary conditions and discretizing the geometry into a grid, wherein the grid comprises a number of grid points;

specifying **(202)** temperature or heat flow conditions at the boundary surrounding the geometry and an initial condition at each grid point;

solving **(203)** a heat flow equation selected from one of conduction, convection or radiation for the geometry and the associated boundary conditions to obtain a temperature, or a heat flow rate, or both at each grid point at steady state;

storing **(204)** the solution for each grid point in a training database;

training **(205)** a model selected from a PPRNN, a DRNN or a DANN model using the training database;

inputting **(206)** a modified boundary condition or initial condition or both associated with the geometry; and,

generating **(207)** a temperature, a heat flow rate at both at each grid point corresponding to the modified boundary condition or initial condition.

2. The method as claimed in claim 1, wherein the training **400** comprises:

calling **(402)** a storage process unit for each input point  $i$  in an individual unit;

storing **(403)** all data points  $i=1, 2, 3 \dots n$  in  $j=1, 2, 3 \dots n$  storage units in a single logic call using AND logic enabled function;



calling(404) multi-threading/distributed RNN and executing the same;  
 calling(405) AND logic enabled machine code upon each point  $i=1, 2, 3 \dots n$  in memory unit  $k=1, 2, 3 \dots n$  with zero time lapse and zero  $k$  to  $k$  communication;  
 storing (406) output RNN data of each output point  $p$ , corresponding to each input point  $i=1, 2, 3 \dots n$  in  $j=1, 2, 3 \dots n$  storage units; and  
 repeating(407) the steps (i) to (v) for  $m$  samples and perform computations and storage for each  $m=1, 2, 3 \dots$  sample in the geometry.

3. The method as claimed in claim 1, wherein the heat flow equation solved is a conduction equation wherein the temperature  $T$  is given by:

$$\frac{1}{a} \cdot \frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} + \frac{q}{k} \dots \dots \dots (1)$$

where  $x, y,$  and  $z$  represent Cartesian coordinates,  $q$  is rate of heat generation,  $k$  is thermal conductivity and  $a$  is a product of density and specific heat capacity of the material.

4. The method as claimed in claim 1, wherein the heat flow equation solved is a convection equation given by:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + h \cdot \frac{\partial T}{\partial y} = 0 \dots \dots \dots (2)$$

5. The method as claimed in claim 1, wherein the heat flow equation solved is a radiation equation given by:

$$d_z \rho c_p u \cdot \nabla T + \nabla \cdot q = d_z Q + q_0 + d_z Q_{ted} + d_z Q_R \dots \dots \dots (3)$$

6. The method as claimed in claim 1, wherein the solving a heat flow equation comprises using an analytical solution, a finite element method solution (FEM) or finite difference method (FDM) solution to generate training data set for each input data  $i$  for each sample  $m$  in a geometry  $t$ .

7. The method as claimed in claim 1, wherein the model is a PPRNN model wherein:

$$PPRNN = \oint_{\Omega} \int_{j=1}^M \tanh(h_{j_i} + b_{2_i}) dj d\Omega_i \dots \dots \dots (4)$$

where

$$h_{j_i} = W_{1_i} \cdot h_{j-1_i} + W_{2_i} \cdot x_{j-1_i} + b_{1_i} \dots \dots \dots (5)$$

$x$  is the input,  $h$  is the hidden cell state and  $W_1$ ,  $b_1$  and  $W_2$ ,  $b_2$  are the weight and bias matrices for hidden-hidden and input-hidden connections,  $\Omega$  is the domain of interest,  $M$  is the number of examples for training,  $\tanh$  is an activation function.

8. The method as claimed in claim 7, comprising obtaining the weight and bias matrices by minimizing mean square error between the predicted and input temperatures for each mesh independent point  $i$  for an example  $m$  given by:

$$MSE_i = \frac{1}{m} \sum_{n=1}^m |T_{Pred_i} - T_{Act_i}|^2 \dots \dots \dots (6)$$

9. The method as claimed in claim 1, wherein the model is a DRNN model wherein:

$$DRNN = \forall \oint_{\Omega} \int_{j=1}^M \tanh(h_{j_i} + b_{2_i}) dj d\Omega_i \dots \dots \dots (7)$$

where,

$$h_j = W_1 \cdot h_{j-1} + W_2 \cdot x_{j-1} + b_1 \dots \dots \dots (8)$$

$x$  is the input,  $h$  is the hidden cell state and  $W_1$ ,  $b_1$  and  $W_2$ , are the weight and bias matrices for hidden-hidden and input-hidden connections,  $\Omega$  is the domain of interest,  $M$  is the number of examples for training,  $\tanh$  is an activation function.

10. The method as claimed in claim 9, comprising obtaining the weight and bias matrices by minimizing mean square error between the predicted and input temperatures for each mesh independent point  $i$  for an example  $m$  given by:

$$MSE = \sum_{n=1}^m \|T_{Pred} - T_{Act}\|^2 \dots\dots\dots(9)$$

11. The method as claimed in claim 1, wherein the model is a DANN model wherein:

$$DANN = \forall \begin{cases} 0 & \text{if } x \leq 0 \\ \int_{j=1}^M (h_j + b_2) dx & \text{if } x > 0 \end{cases} \dots\dots\dots(10)$$

where,

$$h_j = W_1 \cdot h_{j-1} + W_2 \cdot x_{j-1} + b_1 \dots\dots\dots(8)$$

$x$  is the input,  $h$  is the hidden cell state and  $W_1$ ,  $b_1$  and  $W_2$ , are the weight and bias matrices for hidden-hidden and input-hidden connections, and  $M$  is the number of examples for training.

12. The method as claimed in claim 11, comprising obtaining the weight and bias matrices by minimizing mean square error between the predicted and input temperatures for each mesh independent point  $i$  for an example  $m$  given by:

$$MSE_i = \frac{1}{m} \sum_{n=1}^m |T_{Pred_i} - T_{Act_i}|^2 \dots\dots\dots(6)$$

13. A system for solving a heat transport problem over an object characterized by a geometry, the system comprising:

- a hardware switch **(102)**;
- a processor **(101)** coupled to the hardware switch **(102)** to run a neural network engine, wherein the processor **(101)** is configured to:

receive a geometry and associated boundary conditions and discretize the geometry into a grid, wherein the grid comprises a number of grid points;  
receive temperature or heat flow conditions at the boundary surrounding the geometry and an initial condition at each grid point;  
solve a heat flow equation selected from one of conduction, convection or radiation for the geometry and the associated boundary conditions to obtain a temperature, or a heat flow rate, or both at each grid point at steady state;  
store the solution for each grid point in a training database;  
train a model selected from a PPRNN, a DRNN or a DANN model using the training database;  
receive a modified boundary condition or initial condition or both associated with the geometry as input to the trained model; and,  
generate a temperature, a heat flow rate at both at each grid point corresponding to the modified boundary condition or initial condition.

14. The system as claimed in claim 13, comprising a storage process unit (**110**) comprising a plurality of sub-storage process unit (**111-1, 111-2, ..., 111-X**) configured to store one or more data subsets of geometry and associated boundary conditions, and temperature and heat-flow conditions.

15. The system as claimed in claim 13, wherein the processor (**101**) is configured to divide the transport problem solution into a plurality of threads for concurrent execution.

16. The system as claimed in claim 15, wherein the hardware switch (**102**) is configured to execute the one or more threads in parallel and allocate the generated temperature and heat flow rate in memory operation units **120**.

17. The system as claimed in claim 15, wherein the memory operation units **120** comprises a plurality of memory operation units (**121-1, 121-2, 121-3....., 121-X**) configured to store generated temperature and heat flow rate.

Dated this 12<sup>th</sup> Day of October, 2020.

Sd.- Dr V. SHANKAR IN/PA-1733  
For Indian Institute of Technology Madras (IIT Madras)

**MACHINE LEARNING, DEEP LEARNING AND ARTIFICIAL  
INTELLIGENCE FOR PHYSICAL TRANSPORT PHENOMENON IN  
THERMAL MANAGEMENT**

ABSTRACT

Disclosed is a method of solving a heat transport problem over an object characterized by a geometry using a hardware multi-threading process. The method includes geometry and associated boundary conditions and discretizing the geometry into a grid. The method includes specifying temperature or heat flow conditions at the boundary surrounding the geometry and an initial condition at each grid point. A heat flow equation selected from one of conduction, convection or radiation for the geometry is selected and associated boundary conditions, is solved using a known method such as analytical, FDM or FEM to obtain a temperature at each grid point. Next, the solution for each grid point is used for training a neural network model. The neural network model may be a point to point recurrent neural network model. The trained model may be used to predict temperatures for any modified boundary condition associated with the geometry.

**FIG. 2**