Data-Driven Electrostatics Analysis based on Physics-Constrained Deep learning

Wentian Jin, Shaoyi Peng, and Sheldon X.-D. Tan
Department of Electrical and Computer Engineering, University of California, Riverside, CA 92521
{wjin018, speng004, stan}@ece.ucr.edu

***Abstract*—Computing the electric potential and electric field is important for modeling and analysis of VLSI chip and high speed circuits. For instance, it is an important step for DC analysis for high speed circuits as well as dielectric reliability and capacitance extraction for VLSI interconnects. In this paper, we propose a new data-driven meshless 2D analysis method, called *PCEsolve*, of electric potential and electric fields based on the physics-constrained deep learning scheme. We show how to formulate the differential loss functions to consider the Laplace differential equations with voltage boundary conditions for typical electrostatic analysis problem so that the supervised learning process can be carried out. We apply the resulting *PCEsolve* solver to calculate electric potential and electric field for VLSI interconnects with complicated boundaries. We show the potential and limitations of physics-constrained deep learning for practical electrostatics analysis. Our study for purely label-free training (in which no information from FEM solver is provided) shows that *PCEsolve* can get accurate results around the boundaries, but the accuracy degenerates in regions far away from the boundaries. To mitigate this problem, we explore to add some simulation data or labels at collocation points derived from FEM analysis and resulting *PCEsolve* can be much more accurate across all the solution domain. Numerical results demonstrate that the *PCEsolve* achieves an average error rate of 3.6% on 64 cases with random boundary conditions and it is 27.5× faster than COMSOL on test cases. The speedup can be further boosted to $\sim$ 38000× in single-point estimations. We also study the impacts of weights on different components of loss functions to improve the model accuracy for both voltage and electric field.**

I. Introduction

Electrostatics is an important subject of study as it is pivotal in many VLSI modeling applications. The goal is to compute voltage potential and electric fields with some voltage and current boundary conditions for dielectrics and metal interconnects or planes. In the back-end of VLSI manufacturing, strong electric field can induce failure of the dielectrics, which is known as Time-dependent dielectric breakdown (TDDB) [1]. Simulation of this aging effect requires electrostatics. Also, several methods of parasitic extraction involves electrostatics simulations in the chip layouts [2]. Recently global placement is also modeled as electrostatic problem [3].

Traditionally, this problem is primarily solved by numerical methods with spatial discretization of the governing equation using polynomials into a finite-dimensional algebraic system (as is done in finite element method (FEM) or finite difference method (FDM)). Such numerical methods typically require meshing of a complicated layout or geometry, which can be very computationally prohibitive for large problems.

On the other hand, deep neural networks (DNN) have propelled an evolution in machine learning fields and redefined many existing applications with new human-level AI capabilities. DNNs such as convolution neural networks (CNN) have recently been applied to many cognitive applications such as visual object recognition, object detection, speech recognition, natural language understanding, etc. due to dramatic accuracy improvements in those tasks [4]. However, how to apply the deep learning techniques to learn and encode laws of physics and help to solve nonlinear partial differential equations still remains in its infancy.

Recently, the so called physics-informed neural networks (PINN) or physics-constrained neural networks (PCNN) have been proposed to learn and encode physics laws expressed by nonlinear partial differential equations (PDE) for complex physical, biological or engineering systems [5], [6]. The idea is to use DNN, which is differentiable, to regularize the loss functions via back-propagation based training to obtain so-called physics-informed/constrained surrogate models, which can quickly infer the solutions of the PDE to all input coordinates and parameters. The promise of PINN/PCNN is that we only need small number of training examples and the resulting DNN models will quickly respect the underlying principled physics laws for all the input coordinates and parameters so that fast and accurate numerical solutions can be obtained. Another significant benefits of PINN/PCNN idea is that they are mesh-free compared to traditional FEM or FDM based methods. However, only very simple PDE problems were demonstrated in [6]–[10] although some progresses were made for more complicated aerodynamics simulation recently [11].

Inspired by recent progress with PINN/PCNN for solving partial differential equations, in this work, we propose a new data-driven 2D analysis of electric potential and electric fields based on physics-constrained deep learning scheme, called *PCEsolve*. Our contribution are as follows:

- We first show how to formulate the differential loss functions to consider the Laplace differential equations with voltage boundary conditions for typical electrostatic analysis problem. The resulting solving process becomes a nonlinear optimization process, which are solved by the back-propagation method in existing DNN networks. We apply the resulting *PCEsolve* solver to calculate electric potential and electric field for VLSI interconnects with complicated boundaries.
- Our study for purely label-free training (in which no information from FEM solver is provided) shows that *PCEsolve* can get accurate results around the boundaries, but the accuracy degenerates in regions far away from the boundaries. To mitigate this problem, we explore to add some simulation data or labels at collocation points derived from FEM analysis. We explore both voltage and electric field (1st order derivative) label information and resulting *PCEsolve* can be much more accurate across all the solution domain.
- We also studied the impacts of weights on different components of loss functions to improve the model accuracy for both voltage and electric field.
- Numerical results demonstrate the *PCEsolve* achieves an average error rate of 3.6% on 64 cases with random boundary conditions and it is 27.5× faster than COMSOL on test cases. The speedup can be further boosted to $\sim$ 38000× in single-point estimations. Our study shows that the current PINN/PCNN frameworks have some

This work is supported in part by NSF grants under No. CCF-1816361, in part by NSF grant under No. CCF-2007135 and No. OISE-1854276.

potentials for solving practical electrostatics analysis but with limited accuracy.

This paper is organized as follows: Section II reviews some related work for using machine learning and recent deep learning to solve the nonlinear partial differential equations. Section III gives the basic background of electrostatic problem and related differential equations and boundary conditions. Section IV presents the proposed physics-constrained neural network solver for the electrostatic solution. Section V presents the numerical results and discussions of those results. Finally section VI concludes this paper.

II. Related work

In this section, we review some related work. Using the neural networks with physics laws to solve classic differential equations as constraints was originally proposed in late 90's [12], [13]. However, those works were limited by computational power at time. Recently, this idea have received much attention due to recent advances in deep learning for many cognitive tasks with ever-increasing computational resources [4]. Recently, Raissi *et al* proposed to solve 1-D PDEs [6], [7] using so-called PINN as shown in Fig. 1. In this neural network, the physics law in terms of partial differential equations, boundary conditions and initial conditions are explicitly checked for each input (coordinates) so that the resulting loss function and its gradient can be computed for back-propagation based training.

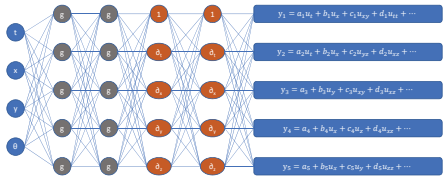


Fig. 1: Concept of physics-informed neural networks

The PINN or PCNN approaches have been recently extended to solve high dimensional PDEs by approximating Galerkin method using neural networks [9], assimilate multi-fidelity training data [14], and its variational analyses have been explored based on arbitrary polynomial chaos [15] and adversarial inference [16]. However, those models can only solve small-sized PDEs with simple boundary conditions. PDE problems with complex boundary and geometries actually still remain a challenging problem as demonstrated by this work. Berg *et al* proposed a new loss function so that the boundary conditions can be hard-coded to be automatically fulfilled by using two well-defined neural network-enabled auxiliary functions [10]. Recently Sun *et al* applied the PCNN concept to solve the uncertainty quantification problems of fluid flows described by Naive-Stokes PDEs [17]. This method introduced more parameters into the neural networks to model the parameter variations show that PCNN based models can yield significant speedup over the first-principle based numerical approaches. However, all those published work were demonstrated on many small problems with simple boundary conditions and the claimed accuracies were not verified on large engineering problems. As we show in this paper, PINN/PCNN based analysis framework still remain challenging for practical large analysis problems in terms of both speedup and accuracy.

III. Preliminaries

A. Electrostatics problem

As mentioned above, many VLSI related problems can be concluded to electrostatics problem, where electric current does not exist and there are only static electric fields due to the voltages applied or charges. They are governed by the first equation of the Maxwell's equations, also known as Gauss's law:

$$\nabla^2 u(x) = \frac{-\rho}{\epsilon}, x \in \Phi \tag{1}$$

with following Dirichlet and Neumann boundary conditions:

$$\begin{aligned} u &= f(x),\ x \in \Gamma_D, \\ \nabla u \cdot \vec{n} &= g(x),\ x \in \Gamma_N, \end{aligned} \tag{2}$$

where Φ is the solution domain, Γ_D is the part of the boundary where Dirichlet (voltage) boundary conditions are given, Γ_N is the part of the boundary where Neumann boundary conditions are given, $u(x)$ is unknown potential to be found, ρ is the charge density, ϵ is permittivity, $f(x)$, and $g(x)$ are given voltage sources and current sources at the boundaries.

In cases where static charges are absent, which this work focuses on, (1) becomes Laplace equation:

$$\nabla^2 u = 0 \tag{3}$$

After solving (3), distribution of electric field is usually obtained by calculating the gradient as per its definition:

$$\vec{E} = -\nabla u \tag{4}$$

Solving for $\vec{E}$ under given voltage boundary conditions $f(u)$ is often of more interest for many practical problems. For example, for capacitance extraction, first the voltage is set to $1V$ for one interconnect wire (indexed i), $0V$ for other wires. Then the induced static charge in any other wire j can be computed using Gauss's flux theorem.

B. Finite element method

In convention, electrostatics problems are solved using discretization methods such as FEM or FDM. Results given by commercial tools based on FEM such as COMSOL are usually deemed golden.

In conclusive words, FEM first discretizes the domain to be solved by a mesh. The mesh and the shape function chosen define a function space. Elements of the function space are defined by expansion coefficients of the shape functions. A numeric solution to the original PDEs can be then found by searching in this function space for the one that best fits the original equations. This is done by setting up and solving a linear system from the original PDEs, with the expansion coefficients to be solved. Typically, the final linear system is composed of tens of thousands of unknowns, known as DOF (degree of freedom). Solving a problem of this scale is not very expensive, yet is still noticeable in a longer routine.

IV. The proposed physics-constrained neural network solver for electrostatics

In this section, we present the proposed physics-constrained neural network solver for electrostatics problem. We first present how the loss functions are built and then we show how to extend the PCNN concept for parameterized PCNN network so that the trained models can be applied to simulation works for different parameters and conditions.

A. PCNN models for electrostatics analysis

As we can see, PCNN essentially leverage the well-known capability of DNN as universal function approximation [5], [6]. PCNN learns to model the behaviors of any dynamic time-dependent, nonlinear system, expressed by the given PDE with boundary and initial conditions. For electrostatic problem as we see from (1), we are computing the steady-state solution. As a result, we only use multilayer perceptron (MLP) neural network architecture as we do not need to learn temporal information. Fig. 2 show the proposed MLP-based PCNN electrostatic solver, that takes the inputs (location x, y) and output an approximate solution. The specific hyperparameters such as number of hidden layers, and number of nodes in each layer are determined experimentally. For each hidden layer, ReLU is used as the activation function. For the output layer, sigmoid is used as the activation layer as we scale the output voltage range to $[0, 1]$.

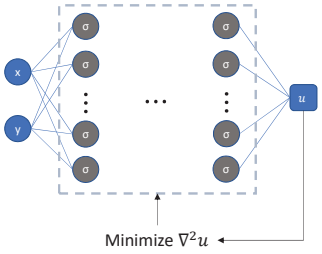


Fig. 2: The proposed PCNN based electrostatic model. σ stands for nonlinear activations. The training of PCNN is done by minimizing the physics-based loss.

The training process of PCNN is essentially an optimization process, it looks for a set of parameters $(\mathbf{W}, \mathbf{b})$ which minimizes the physics based loss defined by the original differential equation.

Specifically for our electrostatics problems, the physics based loss function can be defined by the original equations (3) and (2)

$$L_{phy}(\mathbf{W}, \mathbf{b}) = \underbrace{||\nabla^2 u||_\Phi}_{\text{Gauss's law}} + \underbrace{||u - f(x)||_{\Gamma_D}}_{\text{Boundary condition}} \tag{5}$$

where $||\cdot||$ is L2 norm over a specific domain. Then, training of the network is defined as an optimization problem, looking for the optimal weights and biases $(\mathbf{W}^*, \mathbf{b}^*)$ that minimizes the loss. The process is also shown in Fig. 2:

$$\mathbf{W}^*, \mathbf{b}^* = \underset{\mathbf{W},\mathbf{b}}{\operatorname{argmin}} L_{phy}(\mathbf{W}, \mathbf{b}) \tag{6}$$

In practice, the L2 norm is computed using the collocation method [13]. The domain Φ and the boundary Γ is discretized into sets of collocation points Φ_d and Γ_d, with the number of points $|\Phi_d| = N_f$ and $|\Gamma_d| = N_b$ respectively. Then the loss is computed through the root mean square error on these points. For the part that corresponds to gauss's law in the PDE form (equation residual),

$$L_{pde}(\mathbf{W}, \mathbf{b}) = \frac{1}{N_f} \sum_{i=0}^{N_f} |\nabla^2 U(x^i, y^i)|^2, (x^i, y^i) \in \Phi_d \tag{7}$$

where $U(x, y)$ stands for the PCNN solution.

For the part that covers the boundary condition,

$$L_{bou}(\mathbf{W}, \mathbf{b}) = \frac{1}{N_b} \sum_{i=0}^{N_b} |U(x^i, y^i) - u_b^i|^2, (x^i, y^i) \in \Gamma_d \tag{8}$$

where u_b^i is the value of the voltage boundary condition at collocation point (x^i, y^i).

Combining the two parts of the loss function, the complete loss function used in practice is defined as

$$L_{pcnn}(\mathbf{W}, \mathbf{b}) = L_{pde}(\mathbf{W}, \mathbf{b}) + L_{bou}(\mathbf{W}, \mathbf{b}) \tag{9}$$

By minimizing the loss function, the network $U(x, y)$ will converge to an accurate solution to the original problem.

B. Improved loss function with labels

Our study shows that for many practical problems with complicated boundary conditions, PCNN loss function defined in (9) may still lead to large errors especially for the region far away from the boundary. In this case, introducing some data from numerical solutions or measurement as labels can be instrumental for the training of PCNN networks. For example, one can get the solution with FEM on a coarse mesh, and use it to aid training the PCNN. Compared with the much longer runtime required to solve the PDE on a finer mesh or training the label-free PCNN, the cost of getting such assistant data is negligible.

Denote this set of data $((x, y), u)$ as Ψ_{label}, a new part of data-defined loss is then

$$L_{label}(\mathbf{W}, \mathbf{b}) = \frac{1}{N_l} \sum_{\Psi_{label}} |U(x, y) - u|^2 \tag{10}$$

where Ψ_{label} is the data points and $|\Psi_{label}| = N_l$.

Furthermore, we can add first derivatives into the label data. In this case the data set is in the form of $((x, y), u, \nabla u)$. The loss function becomes

$$L_{label}(\mathbf{W}, \mathbf{b}) = \frac{1}{N_l} \sum_{\Psi_{label}} |U(x, y) - u|^2 + \lambda |\nabla U(x, y) - \nabla u|^2 \tag{11}$$

where λ is a coefficient, which can be determined experimentally. The final loss function with label data $L_{pcnn,L}(\mathbf{W}, \mathbf{b})$ becomes

$$\begin{aligned} L_{pcnn,L}(\mathbf{W}, \mathbf{b}) =& L_{pde}(\mathbf{W}, \mathbf{b}) + L_{bou}(\mathbf{W}, \mathbf{b}) + \\ & L_{label}(\mathbf{W}, \mathbf{b}) \end{aligned} \tag{12}$$

C. Parameterized PCNN surrogate models

The trained PCNN surrogate model discussed in the previous section can only solve particular problem described by given PDEs and boundary and initial conditions. To make the model adaptive to various scenarios, PCNN models need to be parameterized [17]. For our problems, the supply voltages are parameterized as an demonstration of this idea, and are denoted θ. Then the final loss function with labels and parameters becomes

$$\begin{aligned} L_{pcnn,L,P}(\mathbf{W}, \mathbf{b}, \theta) =& L_{pde}(\mathbf{W}, \mathbf{b}, \theta) + L_{bou}(\mathbf{W}, \mathbf{b}, \theta) + \\ & L_{label}(\mathbf{W}, \mathbf{b}, \theta) \end{aligned} \tag{13}$$

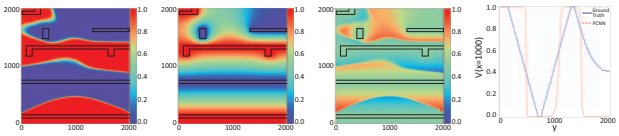
We note that once the parameterized PCNN model is trained, the inference time of the network is much faster than running a conventional FEM based solver for the differential equations for different parameter, which can be very useful for process variational and uncertainty quantification analysis.

V. Numerical Results and Discussions

In this section, we present the experimental results of our proposed PINN solver for electric potential and electric fields of VLSI interconnects. All the models are implemented in Python basing on TensorFlow(1.14.0) library [18] which is an open-source machine learning platform.

A. Label-free PCNN network

We first implement the original PCNN which is based in [6]. It is called label-free PCNN solver as the model is solely trained by loss functions defined by the physics-constraint of PDE in (6) with no simulation data (labels). We test this solver on a VLSI design with complicated contour of interconnects. The resulting electric field is shown in Fig. 3a and the result derived from COMSOL is shown as the ground truth in Fig. 3b.



(a) (b) (c) (d)

Fig. 3: (a) Label-free PCNN solver result (b) Ground truth (c) Error map (d) Centerline voltage profile

As is shown in Fig. 3c, the label-free PCNN solver yields accurate results near the boundaries while the errors get large in other parts, especially at the points that are far away from any adjacent boundary. The overall root-mean-square-error (RMSE) of the estimated electric potential result is 0.29V, which can also be translated to 29% normalized-RMSE (NRMSE) considering the full 1V voltage range. The reason of this low accuracy is that the solver got stuck at a local minimum where it fails to generate smooth transitions from high to low electric potentials. As is mentioned at the end of Section II, our result here reveals the low capability of existing PCNN in modeling practical engineering problems with complicated boundaries.

The loss function (9) of the label-free PCNN solver consists of two parts: boundary loss ($\boldsymbol{L_{bou}}$) which penalizes errors on the boundaries and equation residuals ($\boldsymbol{L_{pde}}$) which enforces Laplace equation. Both parts contribute equally in the loss function. However, the loss of equation residuals is derived by gradient-based calculation which inherently has gradient vanishing problem. There are 8 fully-connected layers in the solver and the equation residuals loss was observed to be 5 to 7 orders of magnitude smaller than the boundary loss. This biased loss leads to a bad learning scheme in which the model is more focused on minimizing the boundary errors and ignores the equation residuals part. To make both parts more balanced, we introduce weight parameters ($\boldsymbol{W_{bou}}$ and $\boldsymbol{W_{pde}}$) to the loss function

$$L_{pcnn}(\mathbf{W},\mathbf{b}) = W_{pde}L_{pde}(\mathbf{W},\mathbf{b}) + W_{bou}L_{bou}(\mathbf{W},\mathbf{b}) \quad (14)$$

To figure out the best configuration of the weights, we fix $\boldsymbol{W_{bou}}$ to 1 and gradually increase $\boldsymbol{W_{pde}}$ from 10^3 to 10^9. Some of the results are shown in Fig. 4 and from which we can clearly see that increasing $\boldsymbol{W_{pde}}$ leads to better accuracy of the results in transition areas. However, as $\boldsymbol{W_{pde}}$ continues to increase, the accuracy near boundaries degrades. We finally set $\boldsymbol{W_{bou}}$ and $\boldsymbol{W_{pde}}$ to 1 and 10^8 respectively leads to an optimized solver with minimized RMSE of 0.059V or NRMSE of 5.9%. The accuracy is significantly improved compared to the original implementation which yields 0.29V RMSE.

The improvement we made in the loss function greatly increases the accuracy of existing PCNN model but still not very satisfactory. The problem is still due to the low accuracy in transition areas which is poorly handled by PCNN. Actually, the idea of completely excluding any simulation data from the training process in existing PCNN is arguable. Some prior data, especially the ones that can be acquired at limited cost, should be adopted into the model training. Thus we further explore the potential of our solver by proposing the label-assisted PCNN model.

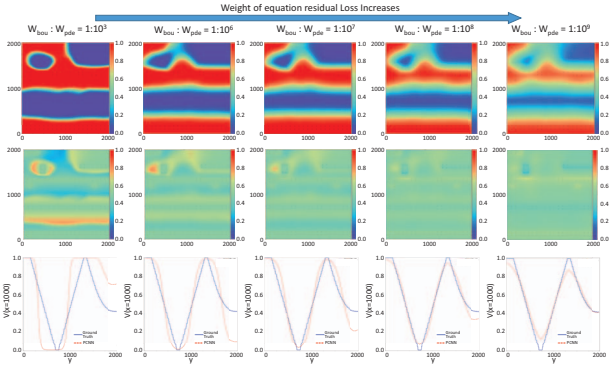
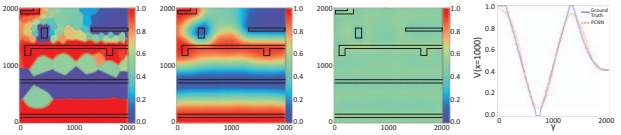


Fig. 4: Influence of weights on PCNN result

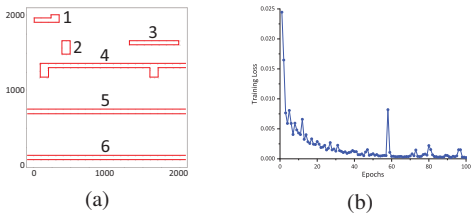
B. Simulation-label assisted PCNN

As described in Section IV-B, we introduce some simulation data as labels to assist the training, under the assumption that these simulation data can be obtained cheaply via existing numerical approaches. (as only small or coarse meshes are needed). As is shown in Fig. 5a, only a limited number of simulation results are sampled (black dots) from the coarse result derived by COMSOL and most sample points are located in the transition area. The label-assisted solver is trained in the same way as the label-free solver except that coarse data are added to facilitate the training process and the results are shown in Fig. 5. The overall accuracy is improved and the RMSE is reduced to 0.049V which is 4.9% in terms of NRMSE. The smoothness of the transition area is also improved as the label penalizes the model for abrupt changes and forces the solver to generate a continuously fading transition from high to low electric potentials.



(a) (b) (c) (d)

Fig. 5: (a) Coarse label data (b) Label-assisted PCNN solver result (c) Error map (d) Centerline voltage profile



(a) (b)

Fig. 6: (a) Boundary voltage inputs of parameterized PCNN (b) Learning curve of parameterized PCNN

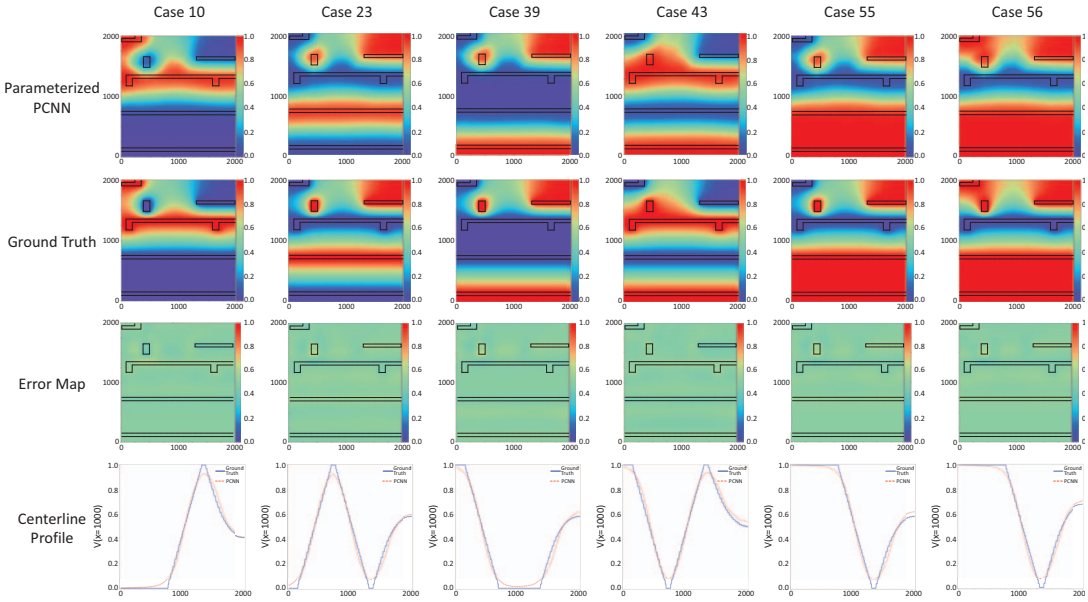


Fig. 7: Selected training results of Parameterized PCNN

C. Parameterized PCNN surrogate models

The aforementioned PCNN models are all dedicated for a specific case with fixed boundary condition, i.e. voltages on the interconnects. Once a new boundary condition is given, the training process has to start all over again even if the voltages might only be slightly altered. This is also the case for COMSOL which has to be rerun for every new boundary condition. To mitigate this problem, we add the voltages on boundaries as extra parameter inputs to PCNN in addition to the original x, y coordinate inputs, and we refer to this modified model as parameterized PCNN.

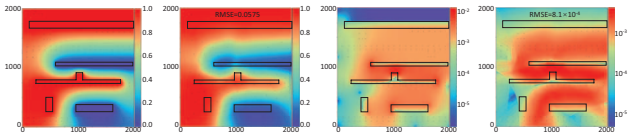
The interconnects topology used to train parameterized PCNN remains the same as we used in aforementioned experiments. The difference is that the boundary voltages can now be changed to random values. We generated 64 different boundary conditions by randomly assigning 0V or 1V voltage to each of the 6 wires as shown in Fig. 6a. In this work, the parameterized PCNN has 8 inputs with 2 for location and 6 for voltages. It is trained using all 64 cases simultaneously and the learning curve is shown in Fig. 6b. The electric field estimation results for some of the training cases are shown in Fig. 7. The average RMSE across all training cases is 0.036V and the maximum and minimum RMSEs are 0.052V and 0.007V respectively. The accuracy remains at the same level compared to previous solver, but the model capacity is significantly enhanced. The parameterized PCNN is able to model much more cases and achieves quite good accuracy for each of them.

Another significant advantage of parameterized PCNN is that it can extrapolate to brand new cases. By altering 6 voltage inputs, the trained model can do the inference for unseen test cases with no need of retraining. We test the model on two new cases with random boundary voltages and the RMSEs are 0.21V and 0.23V respectively. More importantly, as there is no training required for inference, the time costs for both cases are 0.84s and 0.42s respectively, which is quite fast compared to the 23.14s COMSOL spent to solve each of them.

The speedup can be more significant when doing inferences for a large number of unseen new cases. Also, when the interconnects boundary gets more complicated, it would cost COMSOL more time to generate the mesh and solve for the result. However, the time cost of PCNN remains the same as the forward propagation structure is unchanged. Moreover, parameterized PCNN can solve for a small sub-area or even a single point. If the voltage potential at only one point is required, the time cost of parameterized PCNN is drastically reduced to 6×10^{-4} which makes the speedup against COMSOL boosts to $\sim 38000\times$ since COMSOL still has to solve the complete layout and then pick the required point out.

D. Electric field estimation

Once the PCNN model is trained, both x- and y-axis components of the electric field can be derived by computing the partial derivatives of voltage potential with respect to x and y inputs. In this work, we leverage the existing automatic differentiation, i.e. back-propagation, to compute the derivatives. We apply the label-assisted solver on a different layout with new boundary conditions. The estimated voltage potential and its corresponding electric field is shown in Fig. 8. The RMSE is 0.058V for voltage potential and is 8.1×10^{-4} V/nm for electric field. We use this result as our baseline error in this subsection.



(a) (b) (c) (d)

Fig. 8: (a) Ground truth of voltage potential (b) PCNN voltage potential estimation (c) Ground truth of electric field (d) PCNN electric field estimation

Since the accuracy of electric field is newly taken into consideration here, the model and the hyperparameters we optimized in previous section may not apply to this new case. Thus, we further fine tune the solver to get the best accuracy

in electric field estimation. The coarse simulation results of electric field are taken as label data in addition to existing electric potential labels. The weight for these new labels are also introduced into the loss as $\boldsymbol{W_{elec}}$, which converts the loss function into (15)

$$L_{pcnn,L}(\mathbf{W},\mathbf{b}) = W_{pde}L_{pde}(\mathbf{W},\mathbf{b}) + W_{bou}L_{bou}(\mathbf{W},\mathbf{b}) + \frac{1}{N_l}\sum_{\Psi_{label}} |U(x,y)-u|^2 + W_{elec}|\nabla U(x,y) - \nabla u|^2 \tag{15}$$

To search for the optimized weights configuration, we first set $\boldsymbol{W_{pde}}$ to zero so that the influence of second-order derivatives is eliminated. We then fix $\boldsymbol{W_{bou}}$ to 1 and gradually increase $\boldsymbol{W_{elec}}$. The results evolution is shown in Fig. 9. The lowest error of both voltage potential (0.047V) and electric field (9.3×10^{-4} V/nm) are achieved when the weights ($\boldsymbol{W_{bou}}$: $\boldsymbol{W_{elec}}$: $\boldsymbol{W_{pde}}$) are set to $1:10^3:0$. The result accuracy is worse than the baseline, which is within our expectation since the influence of $\boldsymbol{W_{pde}}$ is temporarily isolated.

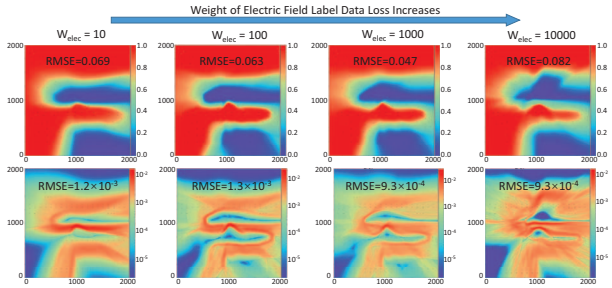


Fig. 9: Influence of $\boldsymbol{W_{elec}}$ on voltage potential and electric field results

We then add $\boldsymbol{W_{pde}}$ back to the loss function and gradually increase it to find the final optimized combination of three weights. The results evolution is shown in Fig. 10. The best accuracy is achieved when the weights are set to $1:10^3:10^7$. Compared to baseline, the RMSE of voltage potential is reduced from 0.058V to 0.036V and from 8.1×10^{-4} V/nm to 7.7×10^{-4} V/nm for electric field.

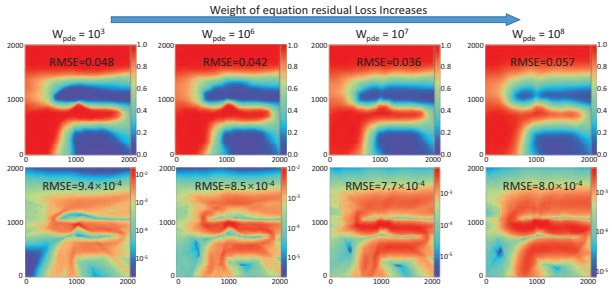


Fig. 10: Influence of $\boldsymbol{W_{pde}}$ on voltage potential and electric field results

VI. Conclusions

This work proposed a novel 2D electric field analysis method based on the physics-constrained deep learning concept. We show how to formulate the differential loss functions to consider the Laplace differential equations with voltage boundary conditions for typical electrostatic analysis problem so that the supervised learning process can be carried out. We apply the resulting *PCEsolve* solver to calculate electric potential and electric field for VLSI interconnects with complicated boundaries. Our study for purely label-free training (in which no information from FEM solver is provided), *PCEsolve* can give accurate results around the boundaries, but the accuracy degenerates at in regions far away from the boundaries. However, with the assistance of coarse simulation data at collocation points derived from FEM analysis, *PCEsolve* can be much more accurate across all the solution domain. Numerical results demonstrate the *PCEsolve* achieves an average error rate of 3.6% on 64 cases with random boundary conditions and it is 27.5× faster than COMSOL on test cases. The speedup can be further boosted to $\sim 38000\times$ in single-point estimations. We also studied the impacts of weights on different components of loss functions to improve the model accuracy for both voltage and electric field.

References

[1] J. McPherson and H. Mogul, "Underlying Physics of the Thermochemical E Model in Describing Low-Field Time-Dependent Dielectric Breakdown in SiO_2 Thin Films," *Journal of Applied Physics*, vol. 84, no. 3, pp. 1513–1523, 1998.

[2] T.-Y. Chou and Z. J. Cendes, "Capacitance calculation of ic packages using the finite element method and planes of symmetry," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 13, no. 9, pp. 1159–1166, 1994.

[3] J. Lu, P. Chen, C.-C. Chang, L. Sha, D. J.-H. Huang, C.-C. Teng, and C.-K. Cheng, "Eplace: Electrostatics-based placement using fast fourier transform and nesterov's method," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 20, Mar. 2015.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[5] M. Raissi, "Deep hidden physics models: Deep learning of nonlinear partial differential equations," *The Journal of Machine Learning Research*, vol. 19, no. 1, pp. 932–955, 2018.

[6] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[7] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations," *arXiv e-prints*, p. arXiv:1711.10561, Nov. 2017.

[8] Y. Yang and P. Perdikaris, "Physics-informed deep generative models," *arXiv e-prints*, p. arXiv:1812.03511, Dec. 2018.

[9] J. Sirignano and K. Spiliopoulos, "DGM: A deep learning algorithm for solving partial differential equations," *Journal of Computational Physics*, vol. 375, pp. 1339 – 1364, 2018.

[10] J. Berg and K. Nyström, "A unified deep artificial neural network approach to partial differential equations in complex geometries," *Neurocomputing*, vol. 317, pp. 28 – 41, 2018.

[11] S. Choudhary, "Physics informed neural networks." Electronic Design Process Symposium Lecture, 10 2019.

[12] H. Lee and I. S. Kang, "Neural algorithm for solving differential equations," *Journal of Computational Physics*, vol. 91, no. 1, pp. 110–131, 1990.

[13] I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE transactions on neural networks*, vol. 9, no. 5, pp. 987–1000, 1998.

[14] X. Meng and G. E. Karniadakis, "A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse pde problems," *Journal of Computational Physics*, vol. 401, p. 109020, 2020.

[15] D. Zhang, L. Lu, L. Guo, and G. E. Karniadakis, "Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems," *Journal of Computational Physics*, vol. 397, p. 108850, 2019.

[16] Y. Yang and P. Perdikaris, "Adversarial uncertainty quantification in physics-informed neural networks," *Journal of Computational Physics*, vol. 394, pp. 136–152, 2019.

[17] L. Sun, H. Gao, S. Pan, and J.-X. Wang, "Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data," *Computer Methods in Applied Mechanics and Engineering*, vol. 361, p. 112732, 2020.

[18] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, (Savannah, GA), pp. 265–283, USENIX Association, Nov. 2016.