FEA-Net: A Deep Convolutional Neural Network With Physics Prior For Efficient Data Driven PDE Learning

Houpu Yao*, Yi Ren† and Yongming Liu‡
Arizona State University, Tempe, Arizona, 85281

We present FEA-Net as an efficient data driven approach to learn Partial Differential Equation (PDE). Specially designed based on physics prior knowledge, FEA-Net needs less trainable parameters and training data while has certifiable convergence. Moreover, FEA-Net is fully interpretable and we can even infer the physics parameters from it. In this paper, inspired by the local support of Finite Element Analysis (FEA), we will first construct a convolution kernel that is suitable to model PDE. Secondly, inspired by the numerical solvers, we constructed the FEA-Net based on the proposed convolution kernel. Experiment results in predicting elasticity problems show that, FEA-Net is able to outperform purely data driven approaches like Fully Convolutional Networks (FCN) by a large margin on multiple tasks.

I. Nomenclature

n	=	number of nodes alone one dimension
K^e, K	=	element and global stiffness matrix in FEA
B, C	=	shape matrix and constitutional matrix in FEA
x, X	=	vector and image form of the system response
f, F	=	vector and image form of external loading
D, R	=	diagonal and off-diagonal component of global stiffness matrix in FEA
W	=	equivalent convolution kernel derived from PDE
E, μ	=	elasticity modulus and Poisson ration
$\circledast$	=	convolution

II. Introduction

Physics processes are usually governed by its underlying Partial Differential Equations (PDE). There are many existing work using purely data driven approaches to predict the physics phenomena, such as in astronomy [1], topology optimization [2], fluid dynamics [3, 4], and material design [5, 6]. However, few work has been done to incorporate our prior knowledge in physics. If we can leverage our prior knowledge on the PDE we want to learn and predict, it can be very helpful in making the neural network more efficient and effective. Some of the related pioneering work includes integrating Ordinary Differential Equation (ODE) information into Recurrent Neural Networks (RNN) [7], adaptive sampling based on topology optimality[8], and learn from PDE via novel network inspired by Finite Difference Approximation (FDA) [9]. These networks with physics prior have shown many improvements over traditional purely data driven approaches in terms of data efficiency and generalizability.

Deep learning, especially deep neural network, has achieved great success in computer vision [10], speech recognition[11], natural language process [12], and control [13] over the recent years. The reason behind the success of it is universal approximation theory, which states that, given enough nodes, a neural network is able to approximate any function with arbitrary precision[14]. However, huge amount of data might be needed to train the network, sometimes can make the training very expensive. To this end, various attempts has been made to design efficient network architectures that work the best for specific problems. For example, in computer vision, Convolutional Neural Network (CNN) was proposed to mimic human visual system, which has shown to be very suitable for object recognition[15], detection[16] and generation[17]. Generally speaking, deeper network will will be more efficient; however, the training

*Ph.D. candidate, School for Engineering of Matter, 650 E Tyler Mall, GWC 435, Student member.
†Assistant Professor, School for Engineering of Matter, 650 E Tyler Mall, GWC 464
‡Professor, School for Engineering of Matter, 501 E Tyler Mall, ECG 301, Senior Member

of deeper networks can have issues like gradient vanishing[18]. To this end, researchers adopted ReLU activation [19], developed Residual Network (Res-Net) [20] and densely connected Res-Net [21] architecture, which largely increased the trainable network depth. Despite all these progresses, almost all current neural networks have following shortages: (1) the filter learned is not quite interpretable, (2) too many weights is needed which makes the training expensive, (3) generalibility of the network is limited, (4) there is no guaranteed convergence for traditional deep networks. In this paper we will show that, by incorporating our prior knowledge in physics, we will show that proposed FEA-Net is able to mitigate all these shortages for learning with PDEs.

Prior knowledge we have in physics can be valuable to design the network. It was observed that different numerical differential equations and different network structures and filters can be related in [22]. Based on this finding, [9] proposed PDE-Net based on finite difference scheme, and reported promising result in system identification. PDE-Net is very similar to our FEA-Net in this paper, as both networks aim at making use of prior knowledge in PDE and its solvers to build better network architecture. There are mainly two differences in our work. Firstly, while the single assumption made in PDE-Net is the order of differential equation, proposed FEA-Net is even more flexible and can lift this assumption. Secondly, FEA-Net is predicting boundary value problems, while PDE-Net is predicting initial value problems.

In this paper, inspired by Finite Element Analysis (FEA), we cast the matrix vector production with FEA matrix into a convolution operation, and design a convolutional neural network named FEA-Net based on weighted Jacobi solver. Proposed FEA-Net is a very flexible framework for learning PDE and we can easily include different prior knowledge of physics into it. We show that FEA-Net needs less trainable parameters, while has better performance compared with traditional purely data driven deep learning methods. What's more, unlike traditional neural networks, FEA-Net will have certifiable convergence with the number of layers of the network. Since FEA-Net is physically meaningful, it can be applied to system response prediction, or data-driven discovery of PDE. To the best of our knowledge, this is the first time people try to design deep learning models via prior knowledge in FEA. This also is also among the first approaches trying to bridge the gap between deep learning models and numerical PDE solvers.

As a summary, we claim three folds of contributions in this paper:

- We proposed a form of filter which is fully interpretable and very suitable for modeling PDE.
- We designed a novel convolutional neural network architecture that has far less trainable variables and certifiable convergence to predict the PDE response.
- Demonstrated that learning with physics prior knowledge is very effective and efficient via 2D elasticity problems.

III. Proposed methods

In the first part of this section, we will focus on constructing the most suitable convolutional kernel for a specific physics problem. The second part introduces FEA-Net, which is built upon the convolution operator derived in the first two parts.

A. convolution operator for physics process

As a quick review, solving partial differential equation using Finite Element Analysis will lead to solving a system of linear equations with the form:

$$K \cdot x = f \tag{1}$$

where f is the external loading on the system, x is the system response, and K is the global stiffness matrix which is obtained by assembling all individual element stiffness matrices K^e together:

$$K^e = \int_A B^T C B dx \tag{2}$$

where C is constitutional matrix depends on the material property, and B is geometry matrix which is decided by the element shape and order. It is worth noting that C will be the same for all elements if the material is homogeneous, and B will be the same if the mesh is uniform. Under these hypothesis, the element stiffness matrices K^e will be the same everywhere. We will make use of this property to reduce K in to a minimalist convolutional filter.

While numerous numerical solvers exist for solving the system of linear equations of Eq. 1, most of them involves iteratively computing the residual[23]:

$$r = f - K \cdot x \tag{3}$$

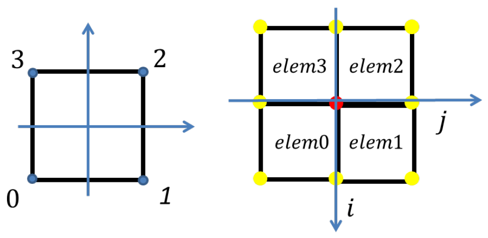


Fig. 1 Illustration of the coordinate system. Left: node index manner inside element. Right: we rotated the coordinate $90°$ to be coherent with image coordinate. Note that node(i,j) is affected by is surrounding nodes only (plotted out in red and yellow respectively).

If the solution domain is rectangular shaped, and the nodes are uniformly distributed in the solution domain, we can view the load and response vector f and x as images. In the rest of this paper, we use F and X to denote the load and response field as images. Further assume that the solution domain is split into $n-1$ first order square elements in each side, we will have $X \in R^{(n,n)}$, $F \in R^{(n,n)}$.

Now we will deal with the global stiffness matrix K. Consider the local support property of finite elements, the response at the red $node(i,j)$ in Fig.1 is affected only by the load on its four surrounding elements. In other words, the pixel value of F is only influenced by the neighboring pixel value of X. Thus, these four adjacent element stiffness matrix can be merged to form a convolutional kernel. Follow the numbering convention in Fig.1, relationship between response and load can be written as :

$$\begin{aligned} F_{ij} &= K^{e0}_{02}X_{i+1,j-1} + K^{e0}_{12}X_{i+1,j} + K^{e0}_{22}X_{i,j} + K^{e0}_{32}X_{i,j-1} \\ &+ K^{e1}_{03}X_{i+1,j} + K^{e1}_{13}X_{i+1,j+1} + K^{e1}_{23}X_{i,j+1} + K^{e1}_{33}X_{i,j} \\ &+ K^{e2}_{00}X_{i,j} + K^{e2}_{10}X_{i,j+1} + K^{e2}_{20}X_{i-1,j+1} + K^{e2}_{30}X_{i-1,j} \\ &+ K^{e3}_{01}X_{i,j-1} + K^{e3}_{11}X_{i,j} + K^{e3}_{21}X_{i-1,j} + K^{e3}_{31}X_{i-1,j-1} \end{aligned} \tag{4}$$

Simply by re-arranging the coefficients in Eq.4, we will be able to obtain its equivalent convolutional form:

$$F = W \circledast X \tag{5}$$

where $W \in R^{(3,3)}$ is the equivalent convolutional filter derived from the element stiffness matrix:

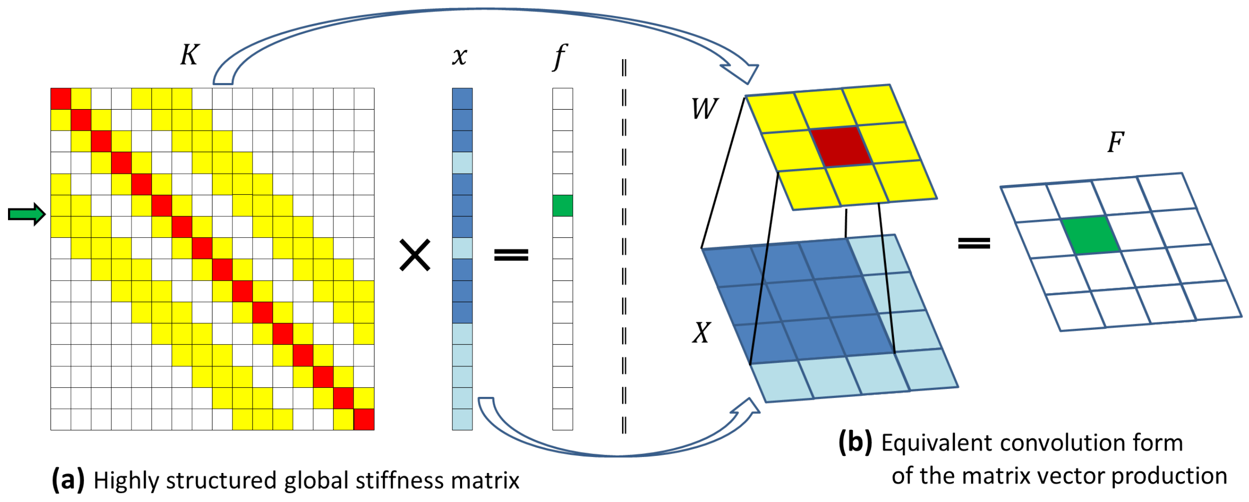
$$W = \begin{bmatrix} K^e_{31} & K^e_{21} + K^e_{30} & K^e_{20} \\ K^e_{32} + K^e_{01} & K^e_{00} + K^e_{11} + K^e_{22} + K^e_{33} & K^e_{23} + K^e_{10} \\ K^e_{02} & K^e_{12} + K^e_{03} & K^e_{13} \end{bmatrix} \tag{6}$$

This equivalence is further illustrated in Fig.2. We will reshape the loading and response vectors x and f into loading and response images X and F. The big stiffness matrix K is reduced to a minimalist convolutional filter W, which will lead to huge savings in storage for large problems. And the matrix vector production in Fig.2a is equivalent to the convolution operation in Fig.2b. As an example, it can be easily seen that the row pointed by a green arrow in Fig.2a is equal to the value of the green grid in Fig.2b.

Since K^e is symmetric, without loss of generality, we can write W as:

$$W = \begin{bmatrix} w_{crnr1} & w_{side1} & w_{crnr2} \\ w_{side2} & w_{self} & w_{side2} \\ w_{crnr2} & w_{side1} & w_{crnr1} \end{bmatrix} \tag{7}$$

There are only five distinct entries in Eq.7, and these terms all have physical meanings. $w_{side1} and w_{side2}$ represents the loading needed when there is only unit response on its side node. w_{self} represents the loading magnitude on the node when there is only a unit response on itself. Note that this FEA-Net form is very general and can be used to model almost any PDE. As an example, we included the derivation of the exact expression of W for steady-state heat transfer and linear elasticity is given in the Appendix.



K
x
f
=
W
F
=
X
(a) Highly structured global stiffness matrix
(b) Equivalent convolution form of the matrix vector production

Fig. 2 Illustration of the equivalent convolution. Plot (a): System of linear equations obtained from FEA discritization. The global stiffness matrix is highly structured. There are only 9 non-zero entries each row, and they are similar across different rows. Plot (b): Corresponding convolutional operation of the matrix vector production. The filter is plotted out as the red and yellow stencil, and the potential distribution is plotted out as the blue stencil below.

B. FEA-Net architecture

In this section, we will focus on transforming the iterative solver Eq.1 into a convolutional neural network based on the convolutional kernel derived in Sec.III.A. Although our method is generally applicable, we will demonstrate on the very basic Jacobi solver here for its simplicity.

The general Jacobi solver of Eq.1 has the following form:

$$x_{t+1} = \omega D^{-1} \cdot \left(f - R \cdot x_t \right) + (1-\omega) x_t \tag{8}$$

where ω is a hyper-parameter which is set to 2/3 in this paper, D and R are the diagonal matrix and off-diagonal matrix of K respectively. Since D is the effect that a unit potential has on itself, it can be reformulated into a element-wise production:

$$D^{-1} \cdot x = PX \tag{9}$$

where P is a scalar, which is also known as pre-conditioner. Since R is the effect all the neighbors has on a point except for the point itself, and its equivalent convolutional filter is zero in the center. Furthermore, as discussed in Sec.III.A, the matrix $-R$ can be transformed into a equivalent convolutional kernel:

$$W_R = -\begin{bmatrix} w_{crnr1} & w_{side1} & w_{crnr2} \\ w_{side2} & 0 & w_{side2} \\ w_{crnr2} & w_{side1} & w_{crnr1} \end{bmatrix} \tag{10}$$

By substituting Eq.9 and Eq.10 into Eq.8 we will have the equivalent convolutional form of the Jacobi solver:

$$X_{t+1} = \omega P \left(F + W_R \circledast X_t \right) + (1-\omega) X_t \tag{11}$$

As a matter of fact, Eq.11 can be viewed as a block of convolutional layer as most of the computation lies in compute the convolution. And by stacking these convolutional blocks together, we can establish a convolutional neural network which is very suitable to handle physics problems. Furthermore, by set the initial guess $X_0 = F$ we can express Eq.11 as:

$$X = g\left(F | P, W_R\right) \tag{12}$$

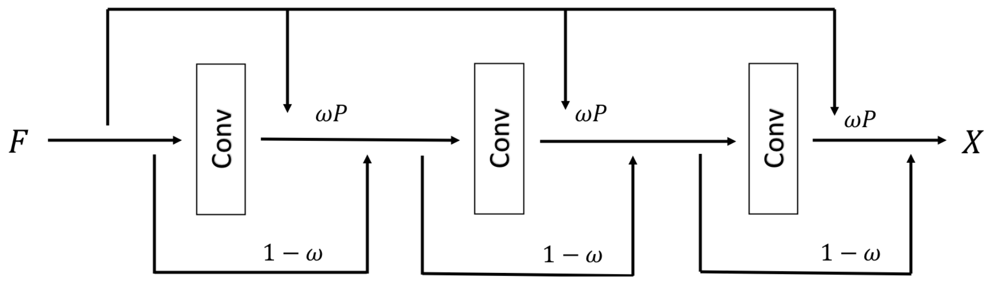


Fig. 3 Illustration of FEA-Net. Dashed lines will disappear for single phase material, and mask convolutions will be come conventional convolutions. It can be seen that there are some densely connected Res-Net structures [21] FEA-Net.

where g denotes the network structure. We name this network FEA-net since its filter is derived from the Finite Element Analysis and the network architecture is derived from FEA solvers. The structure of FEA-Net can be illustrated in Fig.3. FEA-Net bears some similarities to the cutting-edge densely connected Res-Net[21], as there are "short-cuts" across different layers and the input to the network is passed to all other layers. It is also similar to FCN, as the whole network only consists of convolutional layers.

Consider the problem setting that given observed loading and response pair F_i, X_i on an unknown material, and we wish to predict what the response would be if there is a new loading applied. To do this, we will first train the proposed FEA-Net, which is to optimize the filters to minimize the network output given loading with the reference response:

$$\begin{aligned} P^*, W_R^* &= \underset{P, W_R}{\operatorname{argmin}}\, L\left(X^{ref}, X^{pred}\right) + \beta \\ X^{pred} &= g\left(F^{ref} | P, W_R\right) \end{aligned} \tag{13}$$

where L is the error measurement, and the form we choose is:

$$L(x, y) = \sum_{ij}^{n} |x_{ij}|(x_{ij} - y_{ij})^2 \tag{14}$$

which is reported to have better performance in inverse problems[**?**]. And β term in Eq.13 is the penalty for other prior knowledge that we might have. In this paper, it is assumed that we roughly know the range of the physics parameter E and μ, and we can write the term as :

$$\beta = max(E_l - E, 0) + max(E - E_u, 0) + max(\mu - \mu_u, 0) + max(\mu - \mu_u, 0) \tag{15}$$

As with all other neural networks, standard back-propagation can be used to train FEA-Net based on Eq.13. Thus, standard deep learning packages can be directly used without any modifications. Once the network has been trained, we can use it to predict the system responses under new loading conditions.

IV. Results and Discussion

This section is organized as follows: the first part introduces the training and testing data that we will use, the second part introduces the details of the network setup, the third part we show how the networks predicts after training, and the last part we will discuss the convergence property of FEA-Net.

A. dataset setup

To demonstrate the data efficiency of FEA-Net, we will include only 6 training data in total in the training set. These training data are obtained by numerical simulation of a squared plate under different but similar loading condition. As shown in Fig.4, the white lines in the first row are the locations where a uniform x-directional force is applied. The second and third rows are the displacement response along x and y directions obtained from numerical simulation. These plates are sized $0.2m \times 0.2m$ and made of steel with elasticity modulus of 200 GPa and Poisson's ratio of 0.25.

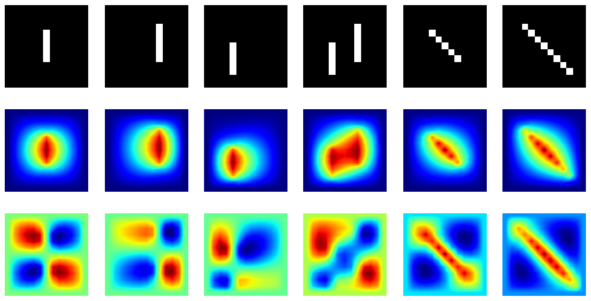


Fig. 4 Visualization of all of the training data used. The first row corresponds to the loading location. A unit x directional loading is applied along the white lines. All boundaries are fixed at both x and y directions. The second and third row corresponds to the response along x and y directions. The resolution of both input and output is 13x13.

And all of the boundary of these plates are fixed at all directions. We used 12 elements along each direction to discritize the region, which leads to a resolution 13x13 for both loading and displacement images.

The testing set is two problems with different scale and loading condition. Consider we have a plate made of the same material with size $1m \times 1m$, with all boundaries fixed at both x and y directions as well. The first testing data is when a vertical crack-like region in the center with x-directional loading applied. This is more similar to the loading in the training set. The second testing data is when a rectangle-shaped region in the upper left corner has an x-directional loading applied. We use 60 elements to discritize the plate so that the size of the element will be the same as the training set. This gives us loading and displacement images of size 61x61. The testing set is different from the training set as: (1) The scale of the problem is different, which poses difficulty on traditional learning approaches. (2) The loading condition in testing set varies a lot from the training set. Besides, the amount of training data is very limited, which makes this prediction a very challenging task for traditional neural networks.

B. Network setup

We train a FEA-Net of 500 layers in this experiment to learn from the training dataset. We further assume that somehow we have a prior knowledge on the material elasticity modulus: $E \in (0.1GPa, 3GPa)$. Note that this is a very loss constrain, as almost all metal material will fail into this range. Furthermore, we know that conventional material will have $\mu \in (0, 0.5)$. We will add these prior knowledge to Eq.15 as training objective for FEA-Net.

As a comparison, we choose fully convolutional networks [24] as a benchmark. The reason is that the training and testing image is of different resolution, and only FCN is able to handle this situation. We build a FCN with 7 layers but with way more parameters. The first layer of FCN has 2 input channels and 64 output channels, the middle 5 layers have 64 input and 64 output channels, and the output layer has 64 input channels and 2 output channels. The filter size is kept as 3x3 which is the same as FEA-Net. ReLU activation is applied after every convolutional layer aside from the last one. Under this network setting, it will lead to over 4k filters and 186k trainable variables, which is way more complicated than FEA-Net.

Aside from the physics prior knowledge term Eq.15, the training objective as stated in Eq.14is the same for both FEA-Net and FCN. We use Adam optimizer [25] with learning rate of 1e-2 and 1e-3 for FEA-Net and FCN respectively. Both FEA-Net and FCN are built and trained with Tensorflow* package.

C. prediction performance

Now we will apply the trained network in Sec.IV.B to predict the response of system under the loading from the testing data in Sec.IV.A. A trick used in this paper to further improve the testing accuracy of FEA-Net is to stack more layers to the original network during testing. We create FEA-Net6x by increasing the number of layers in the trained FEA-Net 6 times. The reason for this is that: (1) the filter learned is physically meaningful and can be shared across

*https://www.tensorflow.org/

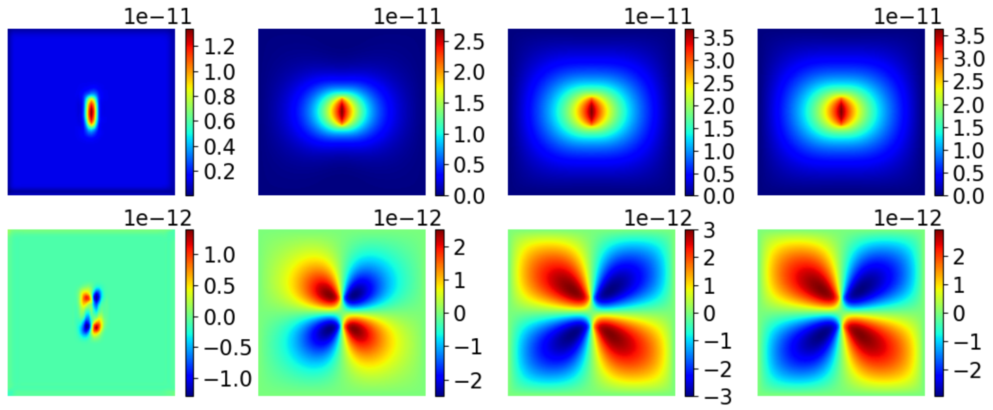


Fig. 5 Comparison of the network prediction from FCN and FEA-Net on the first testing data. From top to bottm: predicted displacement in x and y direction. From left to right: FCN, FEA-Net, FEA-Net6x, and ground truth.

Table 1 Learned parameter with different network depth.

Number of layers	100	200	300	400	500	ground truth
Estimated E (100GPa)	1.94	1.92	1.92	1.93	1.94	2.00
Estimated μ	0.216	0.225	0.227	0.227	0.251	0.250

different layers, (2) FEA-Net is inspired by numerical solvers and has nice convergence property with increasing depth, which will be further analyzed in Sec.IV.D.

A comparison of the prediction between FCN and FEA-Net on different testing data is shown in Fig.5 and Fig.6 respectively. It is interesting to see that FCN is able to predict the trend of the displacement for the first testing data, although the distribution and magnitude of the prediction is way off. This may be that the loading of this testing case is "crack-like", which is very similar to the training data in terms of both size and shape. When it comes to FEA-Net, vanilla FEA-Net is able to give plausible prediction on both testing cases. And with FEA-Net6x, we are able to achive even better predictions. From the first testing case we can see that FEA-Net needs less data and parameters.

As for the second testing loading, where the loading region and scale is all different, the prediction from FCN is even not at the same magnitude as ground truth. This is because FCN isn't learning the physics behind the data. Because second testing load has much larger overall load than the training case, and the system will have larger response. However, since FCN has never seen this kind of loading during training, it will not be able to make a correct prediction. Instead, FEA-Net is still able to make good predictions and FEA-Net6x is able to make predictions that is almost not visually distinguishable from ground truth. The reason is that FEA-Net is aware of the physics behind it, and will learn the PDE behind the training data first to predict the response of a new loading condition. That is to say, with physics prior, proposed FEA-Net is able to generalize much better to novel data.

Since FEA-Net is specially designed from FEA and numerical solver, its filter is fully interpretable and is physical meaningful as analyzed in Sec.III.A. If we further have the prior knowledge that the physics is elasticity, we can further incorporate Eq.19-Eq.20 to infer the material properties from the trained network filter. With the above trained FEA-Net, we achieved a prediction error of 2.95% and 0.48% for E and μ respectively. Since traditional neural networks like FCN is not able to estimate the material properties, we compared FEA-Net with itself trained with different depth. We trained multiple FEA-Net with depth from 100 to 500 and listed the parameter estimation result in Tab.1. It can be seen that FEA-Net is able to produce stable and accurate estimation of the physical parameters.

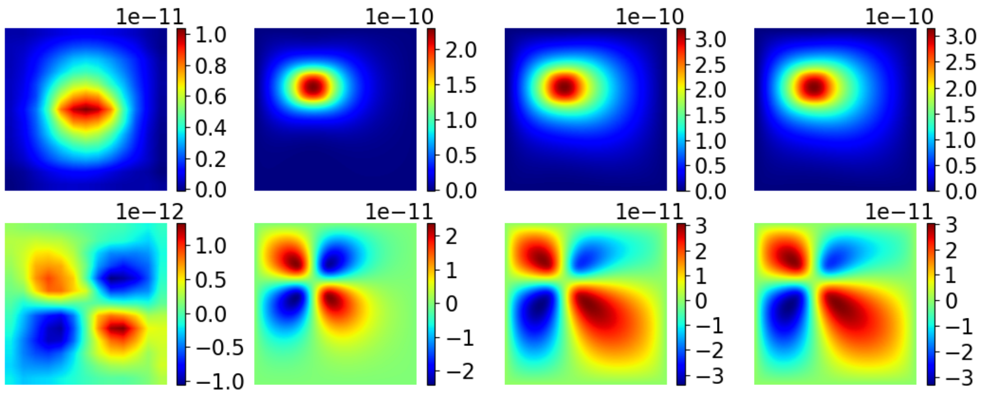


Fig. 6 Comparison of the network prediction from FCN and FEA-Net on the second testing data. From top to bottm: predicted displacement in x and y direction. From left to right: FCN, FEA-Net, FEA-Net6x, and ground truth. Note that there is no visible difference between FEA-Net6x and ground truth.

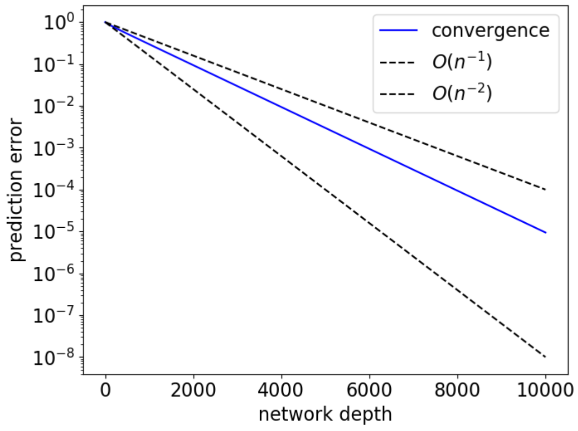


Fig. 7 Convergence of the feed-forward accuracy of FEA-Net with increasing network depth.

D. network analysis

As discussed in Sec.III.B, FEA-Net can be viewed as a surrogate model to the original PDE if the corresponding filter is known. In this experiment, we assume physics parameter is known and test how FEA-Net performs with increasing depth. We will substitute the elasticity modules and Poisson's ratio into Eq.19-Eq.20 and Eq.6 to obtain the optimum filter. With this optimum filter, we will track the prediction accuracy of FEA-Net with different depth.

We define the relative error rate of the network as:

$$\epsilon = \frac{|X^{pred} - X^{ref}|_2}{|X^{ref}|_2} \tag{16}$$

The error rate of FEA-Net with 1 to 10,000 layers is shown in Fig.7. we can observe that the network error rate is monotonically decreasing with increasing network depth. This is also the reason why FEA-Net6x will have a better performance compared with vanilla FEA-Net in Sec.IV.C. What's more, since the convolution filter derived in Sec.III.A is generally applicable, we expect much faster convergence can easily be achieved by designing network architecture based on more advanced solvers like multi-grid.

V. Conclusion

In this study, a novel FEA-Net is proposed for efficient learning of PDE. While FEA-Net is a general framework that can be used to predict almost any PDE, 2D linear elasticity examples are presented to demonstrate its performance. From the derivation and experiment, following conclusions can be drawn:

- FEA-Net is able to inject prior knowledge to successfully model partially unknown physics.
- Experiments on FEA-Net and FCN demonstrates by incorporating physics prior, proposed network is more data efficient and has much better generalizability.
- Proposed FEA-Net is fully interpretable, and can estimate the unknown physics parameters accurately.
- FEA-Net has nice convergence with increasing depth.

In the future study, we will investigate the application of FEA-Net to multi-physics problems and build architectures based on more advanced solvers like multi-gride.

Appendix

A. elasticity convolutional kernel

We derive the equivalent convolutional kernel for 2D in-plane elasticity problem in this section. Because both the loading and response for 2D elasticity has both x and y component, the equivalent filter $W_{elast} \in R^{(2,3,3,2)}$ which has 2 input channels and two output channels.

By substituting the constitutional matrix D of plane elasticity and the shape matrix B of first order square element into Eq.2, we can obtain the corresponding element stiffness matrix:

$$K^e_{elast} = \frac{E}{4(1-\mu^2)} \begin{bmatrix} 8-\frac{8}{3}\mu & 2\mu+2 & -\frac{4}{3}\mu-4 & 6\mu-2 & \frac{4}{3}\mu-4 & -2\mu-2 & \frac{8}{3}\mu & 2-6\mu \\ 2\mu+2 & 8-\frac{8}{3}\mu & 2-6\mu & \frac{8}{3}\mu & -2\mu-2 & -\frac{4}{3}\mu-4 & 6\mu-2 & \frac{4}{3}\mu-4 \\ -\frac{4}{3}\mu-4 & 2-6\mu & 8-\frac{8}{3}\mu & -2\mu-2 & \frac{8}{3}\mu & 6\mu-2 & \frac{4}{3}\mu-4 & 2\mu+2 \\ 6\mu-2 & \frac{8}{3}\mu & -2\mu-2 & 8-\frac{8}{3}\mu & 2-6\mu & -\frac{4}{3}\mu-4 & 2\mu+2 & \frac{4}{3}\mu-4 \\ \frac{4}{3}\mu-4 & -2\mu-2 & \frac{8}{3}\mu & 2-6\mu & 8-\frac{8}{3}\mu & 2\mu+2 & -\frac{4}{3}\mu-4 & 6\mu-2 \\ -2\mu-2 & -\frac{4}{3}\mu-4 & 6\mu-2 & \frac{4}{3}\mu-4 & 2\mu+2 & 8-\frac{8}{3}\mu & 2-6\mu & \frac{8}{3}\mu \\ \frac{8}{3}\mu & 6\mu-2 & \frac{4}{3}\mu-4 & 2\mu+2 & -\frac{4}{3}\mu-4 & 2-6\mu & 8-\frac{8}{3}\mu & -2\mu-2 \\ 2-6\mu & -\frac{4}{3}\mu-4 & 2\mu+2 & \frac{4}{3}\mu-4 & 6\mu-2 & \frac{8}{3}\mu & -2\mu-2 & 8-\frac{8}{3}\mu \end{bmatrix} \tag{17}$$

where odd rows and columns corresponds to x directional response and loading, and even rows and columns corresponds to y directional response and loading. We will start by considering only the relationship between x directional loading and x directional response by extracting the entries of the odd rows and columns in Eq.17:

$$K^e_{elast,xx} = \frac{E}{4(1-\mu^2)} \begin{bmatrix} 8-\frac{8}{3}\mu & -\frac{4}{3}\mu-4 & \frac{4}{3}\mu-4 & \frac{8}{3}\mu \\ -\frac{4}{3}\mu-4 & 8-\frac{8}{3}\mu & \frac{8}{3}\mu & \frac{4}{3}\mu-4 \\ \frac{4}{3}\mu-4 & \frac{8}{3}\mu & 8-\frac{8}{3}\mu & -\frac{4}{3}\mu-4 \\ \frac{8}{3}\mu & \frac{4}{3}\mu-4 & -\frac{4}{3}\mu-4 & 8-\frac{8}{3}\mu \end{bmatrix} \tag{18}$$

Based on Eq.6, we can find it equivalent convolutional kernel:

$$W^{xx}_{elast} = \frac{E}{4(1-\mu^2)} \begin{bmatrix} -(1-\mu/3) & 4\mu/3 & -(1-\mu/3) \\ -2(1+\mu/3) & 8(1-\mu/3) & -2(1+\mu/3) \\ -(1-\mu/3) & 4\mu/3 & -(1-\mu/3) \end{bmatrix} \tag{19}$$

W^{xx}_{elast} is the convolutional kernel represents the interaction between x directional force and x directional displacement.

Similarly, the relationship between x directional loading and y directional response W^{yx}_{elast} can be obtained from the odd row and even column of Eq.17, the relationship between y directional loading and y directional response W^{yy}_{elast} can be obtained from the even row and even column of Eq.17, the relationship between y directional loading and x directional response W_{elast}^{xy} can be obtained from the odd row and even column of Eq.17:

$$
\begin{aligned}
W_{elast}^{xy} = W_{elast}^{yx} &= \frac{E}{2(1-\mu)} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix} \\
W_{elast}^{yy} &= \frac{E}{4(1-\mu^2)} \begin{bmatrix} -(1-\mu/3) & -2(1+\mu/3) & -(1-\mu/3) \\ 4\mu/3 & 8(1-\mu/3) & 4\mu/3 \\ -(1-\mu/3) & -2(1+\mu/3) & -(1-\mu/3) \end{bmatrix}
\end{aligned} \tag{20}
$$

where W_{xy}, W_{yx}, W_{yy} represents the interaction between: y directional displacement and x directional force, x directional displacement and y directional force, y directional displacement and y directional force.

References

[1] Graff, P., Feroz, F., Hobson, M. P., and Lasenby, A., "SKYNET: an efficient and robust neural network training tool for machine learning in astronomy," *Monthly Notices of the Royal Astronomical Society*, Vol. 441, No. 2, 2014, pp. 1741–1759.

[2] Sosnovik, I., and Oseledets, I., "Neural networks for topology optimization," *arXiv preprint arXiv:1709.09578*, 2017.

[3] Tompson, J., Schlachter, K., Sprechmann, P., and Perlin, K., "Accelerating eulerian fluid simulation with convolutional networks," *arXiv preprint arXiv:1607.03597*, 2016.

[4] Chu, M., and Thuerey, N., "Data-driven synthesis of smoke flows with CNN-based feature descriptors," *ACM Transactions on Graphics (TOG)*, Vol. 36, No. 4, 2017, p. 69.

[5] Cang, R., and Ren, M. Y., "Deep network-based feature extraction and reconstruction of complex material microstructures," *ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers, 2016, pp. V02BT03A008–V02BT03A008.

[6] Cang, R., Xu, Y., Chen, S., Liu, Y., Jiao, Y., and Ren, M. Y., "Microstructure representation and reconstruction of heterogeneous materials via deep belief network for computational material design," *Journal of Mechanical Design*, Vol. 139, No. 7, 2017, p. 071404.

[7] Yu, Y., Yao, H., and Liu, Y., "Physics-based Learning for Aircraft Dynamics Simulation," *PHM Society Conference*, Vol. 10, 2018.

[8] Cang, R., Yao, H., and Ren, Y., "One-Shot Optimal Topology Generation through Theory-Driven Machine Learning," *arXiv preprint arXiv:1807.10787*, 2018.

[9] Long, Z., Lu, Y., Ma, X., and Dong, B., "PDE-Net: Learning PDEs from Data," *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, 2018.

[10] LeCun, Y., Bengio, Y., and Hinton, G., "Deep learning," *nature*, Vol. 521, No. 7553, 2015, p. 436.

[11] Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., et al., "Deep speech 2: End-to-end speech recognition in english and mandarin," *International Conference on Machine Learning*, 2016, pp. 173–182.

[12] Kumar, A., Irsoy, O., Ondruska, P., Iyyer, M., Bradbury, J., Gulrajani, I., Zhong, V., Paulus, R., and Socher, R., "Ask me anything: Dynamic memory networks for natural language processing," *International Conference on Machine Learning*, 2016, pp. 1378–1387.

[13] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al., "Human-level control through deep reinforcement learning," *Nature*, Vol. 518, No. 7540, 2015, p. 529.

[14] Csáji, B. C., "Approximation with artificial neural networks," *Faculty of Sciences, Etvs Lornd University, Hungary*, Vol. 24, 2001, p. 48.

[15] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D., "Backpropagation applied to handwritten zip code recognition," *Neural computation*, Vol. 1, No. 4, 1989, pp. 541–551.

[16] Ren, S., He, K., Girshick, R., and Sun, J., “Faster r-cnn: Towards real-time object detection with region proposal networks,” *Advances in neural information processing systems*, 2015, pp. 91–99.

[17] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A., “Image-to-image translation with conditional adversarial networks,” *arXiv preprint*, 2017.

[18] Glorot, X., Bordes, A., and Bengio, Y., “Deep sparse rectifier neural networks,” *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.

[19] Krizhevsky, A., Sutskever, I., and Hinton, G. E., “Imagenet classification with deep convolutional neural networks,” *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[20] He, K., Zhang, X., Ren, S., and Sun, J., “Deep residual learning for image recognition,” *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[21] Huang, G., Liu, Z., Weinberger, K. Q., and van der Maaten, L., “Densely connected convolutional networks,” *Proceedings of the IEEE conference on computer vision and pattern recognition*, Vol. 1, 2017, p. 3.

[22] Lu, Y., Zhong, A., Li, Q., and Dong, B., “Beyond Finite Layer Neural Networks: Bridging Deep Architectures and Numerical Differential Equations,” *arXiv preprint arXiv:1710.10121*, 2017.

[23] Yang, X. I., and Mittal, R., “Acceleration of the Jacobi iterative method by factors exceeding 100 using scheduled relaxation,” *Journal of Computational Physics*, Vol. 274, 2014, pp. 695–708.

[24] Long, J., Shelhamer, E., and Darrell, T., “Fully convolutional networks for semantic segmentation,” *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[25] Kingma, D. P., and Ba, J., “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.