Generalizing Discretization Representations for the Physical Solutions via Flexible Spatial Image Data Structures

Zi Wang
Energy Science Engineering
Stanford University
ziwang3@stanford.edu

Abstract

This work highlights the relationship between computer vision and discretization representations of physical problems. For partial differential equations governing the physical problems, solutions can be represented in spatial fields, which is essentially the images of discretized physical quantities. From a computer vision perspective, this work explores the implementation of deep learning models with flexible spatial representations to predict physical solutions. Two architectures are implemented on two different data structures: an U-Net-based convolutional neural network (CNN) and a transformer-based model. These can be interpreted as learning from structured (grid-based) and unstructured (graph-like) discretization of space, respectively. Both the Unet and transformer achieves high accuracy on the task. Although Unet is more accurate, transformer shows better flexibility on spatial representation of physical solutions, highlighting a more promising path of integrating deep learning with general discretization data of physical problems.

1. Introduction

The deep learning method has been widely used in the computer vision domain, and shows its strong performance for spatial and temporal tasks. In this work, we aim to apply advanced computer vision technique in the discretized physical problems and reveal new insights on the relationship between spatial physical field and computer vision deep learning method.

For solving the physical problem, it's actually the process of finding solutions of partial differential equations in a given spatial and temporal domain. The solutions can be seen as dynamic images consisted of spatial distributed values. From the mathematical perspective, the PDEs can be solved by analytical methods, such as Green function, Fourier transform and Laplace transform. However, when the PDE system becoming more complex, these analytical methods fails on the mathematical solving tasks. For these complex but very common PDE systems in various engineering problems, the numerical methods are the most reliable and powerful tools.

To start with, solving PDEs numerically, we need to discretized the spatial domain into several finite elements or volumes, i.e. using mesh grid to represent the spatial space. The target is to assigning the physical quantities' value on these finite elements and decreases the residual loss of the original PDEs. It's extremely interesting to find that the discretized spatial domain can be highly aligned with image data in computer vision field. The structured mesh grid is representing the space by pixels, which is the same as the traditional pixel-based image data in computer vision, enabling us using convolutional neural networks to capture the local and global characteristics to predict the target value. The unstructured mesh is usually generated by triangular grid structure. The spatial domain is firstly described by setting several discretized points, which is called nodes. These node data can be seen as sparse sampling of a limited spatial domain, which is aligned with the point cloud structure. But in computer vision view, the point cloud data is usually sampled to maximize the representative capability of the point cloud for the image information, and thus it is usually more scattered and uniform in the entire domain. However, when nodes of unstructured mesh are generated, it tends to be more clustered in some regions with complex geometry or shape, where the PDEs solutions are usually sharp and unstable. It can enhance the solving accuracy and the solution stability. Based in the generated nodes, the triangular mesh is formed by connecting the neighboring nodes based on given scheme. As can be seen, the final tridiagonal mesh not only represents the spatial domain with a set of discretized points, but also induced some relations among these points. The structure is highly aligned with the graph data in computer vision view. It enables us to derive what the physical value is on the given points from their underlying relationship that is the partial differential equa-

tions. After discretizing the spatial domain into elements, we need to solve a huge linear algebra system to get the accurate solution. It is usually very computational expensive and time consuming.

Therefore, from the point of view in the connections between discretized domain and computer vision, we aimed to leveraging deep learning method bridging these two research domains, to accelerate the solving process of PDEs. My motivation is to explore how different image data structures can influence the deep learning prediction results on the fluid flow problem. Inspired by the structured and unstructured mesh concept in numerical method, the physical solutions can be expressed in different forms which enable us to explore the best discretized structure for deep learning method that can reach the highest accuracy and prediction efficiency.

2. Related Work

Deep learning for PDEs is a part of research in the field of AI4Science, a lot of previous work has been extended to integrate the deep learning method in the numerical solving process. Due to the potential to overcome the lack of high computational costs of the traditional numerical method, this interdisciplinary research direction has gained huge focus, including data-driven methods, physics-driven methods and neural operators.

2.1. Data-driven Neural Network

The application of deep learning in solving physical problems has witnessed significant growth, primarily due to its potential to mitigate the high computational costs associated with traditional numerical methods. Initially, data-driven deep learning approaches were adopted to model physical systems. Yang et al.[8] implemented fully connected neural network on predicting the grid-based fluid simulations, showing that the computational cost is decreased by using deep learning network. Cheng et al.[1] adopted convolutional generative adversarial network to predict the spatial and temporal fluid flow images, and obtained high fidelity results that is consistent with numerical model. However, these methods often exhibit limited generalization capabilities, being highly sensitive to specific problem details and computational domains.

2.2. Physics-driven Neural Network

To overcome these limitations, researchers have focused on integrating physical information directly into neural network architectures. One well-known approach is the Physics-Informed Neural Network (PINN)[5], which embeds governing equations into the loss function, enabling the network to learn physical solutions from the underlying physical principles. The results indicate that the physics-informed method allows the neural network to learns more physical informations with less data, and could predict longer temporal behavior of physical systems. This method has been widely adopted in many physical problems, such as elastic mechanics, fluid mechanics and geosciences. It shows better generalization ability than the data-driven neural network, and gives more flexibility for us to design what physical information that we want to feed in the neural network. However, one main limitation of this method arises, it is found that the trained PINN only performs well on a specific problems. It means that once we trained the PINN with a fixed residual loss of PDEs, the network shows very strong performance on that PDE system but performs extremely weak on any other problems. This conclusion again indicates the weak generalization ability of neural networks to different PDE systems.

2.3. Neural Operators

Beyond PINNs, the development of neural operators has marked a significant advancement in modeling complex and different physical systems. Instead of focusing on the specific problems, the neural operator aims at learning the relationship between vectors on one space and another space, which means it focuses on learning the function of vector operations i.e. the differential operators. The Fourier Neural Operator (FNO)[3] introduces a novel framework by parameterizing integral kernels in Fourier space, facilitating efficient and accurate modeling of parametric partial differential equations. Similarly, Deep Operator Networks (DeepONets)[4] are designed to learn nonlinear operators, effectively capturing the mappings between function spaces. By combining the physics-informed method and the concept of neural operators, the Euler operators is proposed to predict the mis-specified fluid problem[2]. The neural operator seems to be the most promising method for physical problem. However, it still faces the same problem of weak generalization ability. The emergence of transformer architectures has further expanded the capabilities of deep learning in physical simulations. Transolver[7] leverages transformer models to solve partial differential equations on general geometries, demonstrating enhanced scalability and adaptability.

However, none of the studies has dived into the relationship between deep learning network and numerical solvers deeply. he usage of deep learning in solving physical problems from strict view of discretizing spatial domain is unrevealed. For more promising methods of deep learning and physical problems, the data structure should be further designed carefully in order to capture the physical characteristics. This work will explore how deep learning network can perform with different discretization forms from numerical solver.

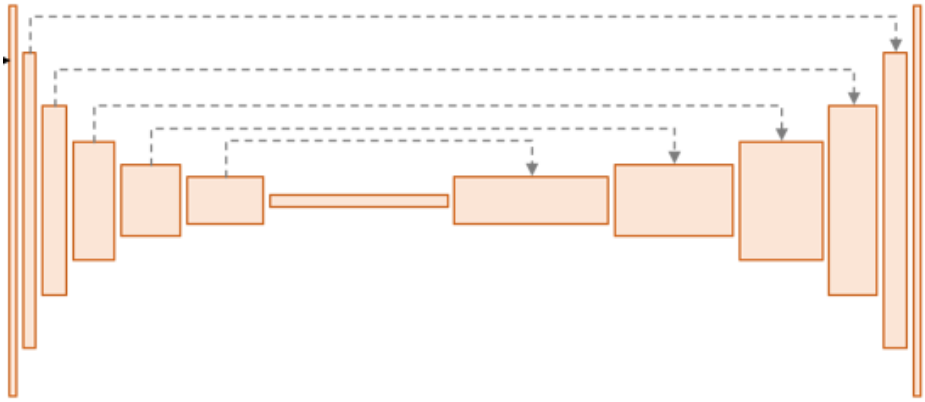

Figure 1. Unet Convolutional Net Structure

3. Methods

In this work, the convolutional neural network on the pixel image and the transformer on point cloud data are implemented to predict the fluid flow field. The convolutional neural network is designed to follow the architecture of Unet consisted of encoding and decoding part. The transformer is directly applied on the point cloud data.

3.1. Unet CNN

Convolutional neural network is the most fundamental and a very efficient model in computer vision area. Especially, the U-net structured CNN shows its strong ability in different research area. The Unet structure in this work is shown in Fig. 1.

Both encoding and decoding blocks contain five convolutional layers. In detail, the encoder part uses the convolutional net and the decoder part uses the transposed convolutional net, which achieves the down sampling and up sampling, respectively. Except for the last transposed convolutional net of the decoder, after every convolutional net, the batch normalization layer and LeakyReLU activation are applied. The detail setting of convolutional layers are shown in Table. 1. It should be mentioned that the skip connection is introduced between corresponding layers between encoders and decoders, which is marked by grey dashed line in the figure. This skip connection allows us to enhance data information fusion and gradient calculation.

The structured mesh describe the spatial domain pixelwisely. By using convolutional neural networks, the local and neighbor information can be captured by the convolutional kernel. In detail, calculating the gradient of velocity is actually using neighboring pixel values. For example, when calculating the first-order gradient of x velocity, we only need to use a 3×3 convolutional kernel with a fixed kernel value of ((0,0,0), (1,0,1), (0,0,0)). Therefore, convolutional neural networks are expected to fully capture the local gradient information.

Table 1. Convolutional Layers Configuration

Layer	Channel	Kernel	Stride	Pad	Dilute
Conv_1	(2, 8)	(4, 4)	2	1	1
Conv_2	(8, 32)	(4, 4)	2	1	1
Conv_3	(32, 128)	(3, 3)	1	0	1
Conv_4	(128, 256)	(3, 3)	3	0	1
Conv_5	(256, 512)	(3, 3)	1	0	1
ConvT_1	(512, 256)	(3, 3)	1	0	1
ConvT_2	(256, 128)	(3, 3)	3	0	1
ConvT_3	(128, 32)	(3, 3)	1	0	1
ConvT_4	(32, 8)	(4, 4)	2	1	1
ConvT_5	(8, 2)	(4, 4)	2	1	1

3.2. Transformer

The Transformer[6] is a neural network architecture originally introduced for natural language processing tasks, which has since become a foundational model across diverse domains, including computer vision, point cloud analysis, and scientific computing. Its key innovation lies in the use of self-attention mechanisms, which allow the model to capture long-range dependencies and contextual relationships between elements of the input sequence without relying on recurrent or convolutional structures, as shown in Fig. 2. In scientific applications, the Transformer can be adapted to attend over coordinates, features, or latent representations, providing a powerful framework for learning from structured, irregular, or high-dimensional data.

In this project, we adapt the Transformer architecture to a point cloud regression task, where each data sample consists of a set of spatial points characterized by their 2D coordinates and associated physical features given from the finite element method. Unlike pixel-based data, point clouds are more irregular. But it is more efficient for representing physical problem in the spatial domain. To address this, we leverage the Transformer's self-attention mechanism to model interactions between all node points within point cloud, allowing the network to learn global geomet-

Table 2. Transformer Net Architecture for Point Cloud Regression

Stage	Layer	Input Shape	Output Shape
Input Projection	Linear ($2 \rightarrow 96$)	(N, 2)	(N, 96)
Transformer Encoder 1	LayerNorm	(N, 96)	(N, 96)
	Multihead Attention (4 heads)	(N, 96)	(N, 96)
	Residual Connection	(N, 96)	(N, 96)
	LayerNorm	(N, 96)	(N, 96)
	Feedforward: Linear ($96 \rightarrow 256 \rightarrow 96$), ReLU	(N, 96)	(N, 96)
Transformer Encoder 2–8	Repeat same structure as Encoder 1	(N, 96)	(N, 96)
Output Head	MLP: Linear ($96 \rightarrow 96 \rightarrow 2$), ReLU	(N, 96)	(N, 2)

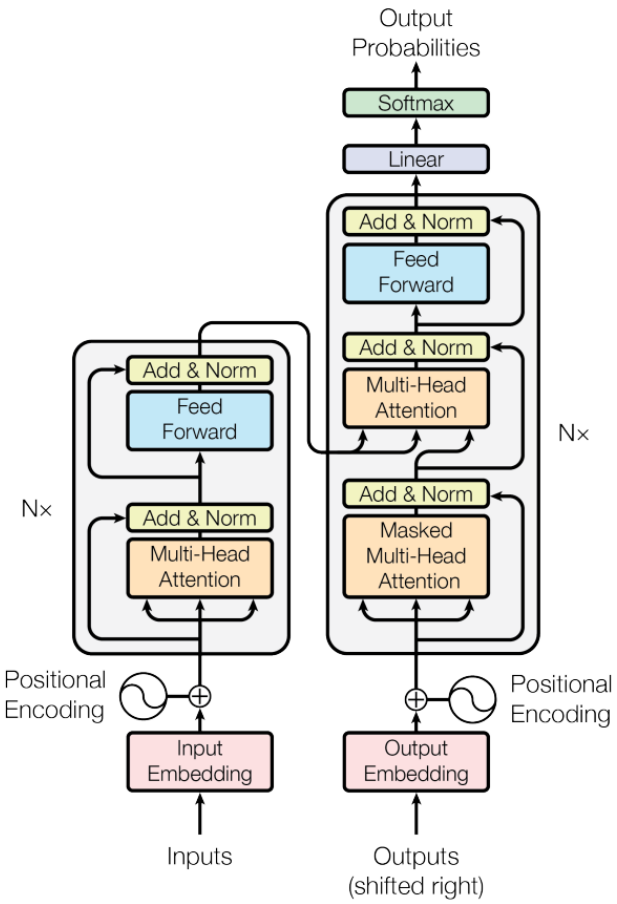

Figure 2. Schematic of Original Transformer[6]

ric relationships and local feature dependencies simultaneously. Each point attends to others based on both feature similarity and spatial proximity, enabling the model to infer underlying patterns in the data. The goal is to predict two target values at each point—representing discretization nodes, including the flow velocities along x direction and y direction.

The transformer net is consisted of three parts, including the linear projection, the transformer encoder and final output layer. The input projection takes the point cloud data (B, N, 4) as input, and projects the data to the Q/K/V domain by a fully connected layer, leading to the output Q/K/V data of shape (B, N, 96). Then these Q/K/V data is fed into the transformer encoder layer consisted of 8 transformer encoder block. For each transformer encoder block, it is consisted of 4 self-attention heads with standard dot product attention, feedforward layer with 256 hidden dimensions and the ReLU activation functions. The pre-layer normalization is adopted before all encoder blocks. For the output layer, it's consisted of two fully connected layer with ReLU activation functions, which transfer the data from 96 dimensions to 2 dimensions. The two dimensional output data is our target fluid velocity (u, v) at all data positions. The details of the transformer net is given in Table. 2. It should be mentioned that the positional encoding has not been implemented here, and the experiments indicates that the transformer still shows good performance without positional encoding on our tasks. It is due to that our input is the point coordinate, the positional information is implicitly embedded into the input data and is enough for transformer to learn the spatial relationship.

4. Datasets

Pore-scale fluid flow is difficult to be numerically solved due to the complexity of porous structures. Herein, we aimed at the pore-scale fluid flow inside circular package structures, as shown in Fig. 3(a), where region outside the circles represents the fluid flow space and the circle region represents the solid structures.

4.1. Porous Structure Image Generation

Circular packed porous structures are generated by Random Monte Carlo movement of circular. Initially, several circulars are fixed at given locations in a square domain with pixel size of (240, 240). Then all criculars are randomly moved inside this region, and they will collide with each other but follows two principles: 1. Their collision is elastic, which means there will be no overlap, 2. The boundary of the region is periodic, the part of circle outside the boundary will appears on the other side. When the random movement is evolving with time, several structures

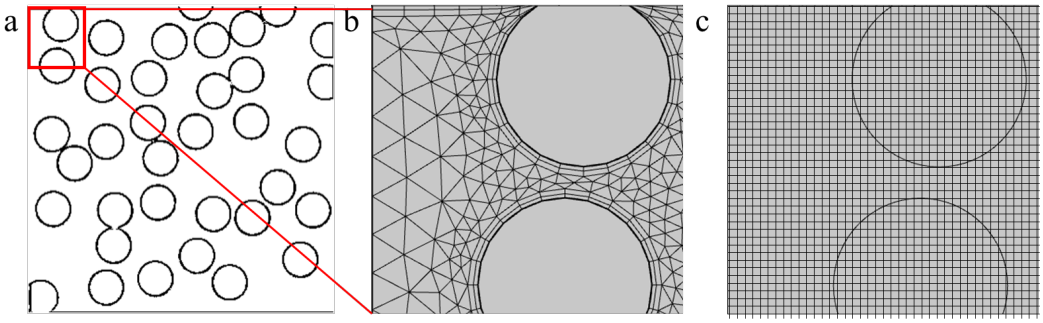

Figure 3. Schematic of Image Data Structure

with different distributions of circles are generated. In total, 8600 structures are generated, and they are used to simulate the fluid flow and used as training dataset.

4.2. Fluid Flow Data Generation

The fluid flow data is generated by numerical simulation of fluid flow inside porous media. The graph-like unstructured data is generated by COMSOL simulation, a commercial software based on finite element methods. The fluid is injected from the left boundary with a given inlet velocity, and then flows out the domain through the right boundary. The solutions can be represented as about thousands of nodes from the tri-diagonal mesh, as shown in Fig. 3(b). The data is consisted of node points and the triangular elements, where the triangular elements are formed by connecting the nodes. Currently, we only focus on the node points and implement transformer on the point cloud consisted of these node points. By saving the simulation results from COMSOL, the node points data can be saved in a point cloud of size (N, 4), where N is the node point number, the four dimension columns represent the coordinates (x, y) and the fluid velocity in horizontal and vertical direction (u, v), respectively.

4.3. Data Pre-process

In total, 8600 point clouds with shape of (N, 4) are generated by above procedure, as shown in Fig. 4(b). For every point cloud, the number of nodes is different, because it is automatically generated by COMSOL, which is the best representation for corresponding geometry. In general, the point number varies from 1793 to 3284. Since the nodes number (the point density) is not in this work's scope, we simply sampled 1793 points from every point cloud, and use it as our dataset. It should be mentioned that such random sampling may be not the best operation here, there could be better choice like padding or farthest point sampling. However, these random sampled points don't influence our goal to test transformer on this unstructured discretization data. In future work, the data should be taken more carefully, such as using a mask-mounted padding or embedding

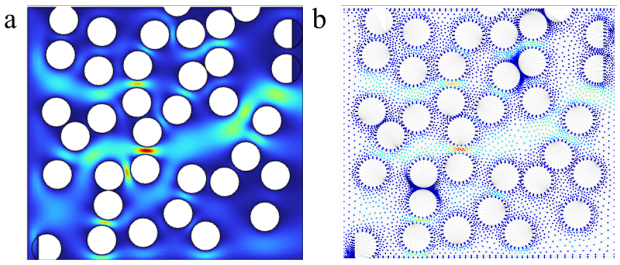

Figure 4. Dataset of Physical Images on Different Mesh

in the latent space. The dataset is also preprocessed before training. Since the coordinates should be used for positional encoding and we want to maintain its spatial relative correlation, the coordinates and the velocities are preprocessed separately. For the coordinates, the min-max normalization is adopted, as given in Eq. (1). For the velocity, the mean-deviation normalization is adopted, as given in Eq. (2). The 8600 pairs of point clouds are divided into training, validation and test dataset as 6300:1800:500 pairs.

$$x = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (1)$$

$$u = \frac{u - x_{mean}}{u_{std}} \quad (2)$$

5. Experiments and Results

The fluid flow field is predicted by both Unet and transformer based on pixel-based image and point cloud, respectively. In this work, the variations of loss function, the distributions of fluid velocity and the relative error of velocity field are compared between these two networks. The main target is to evaluate how the different image data structure (discretization generalization form) and corresponding neural networks could perform on predicting the flow field.

5.1. Training Settings

For both the Unet and the transformer training, the Adam optimizer and the mean squared error loss function (MSE)

are adopted. The Adam optimizer shows good performance on most tasks without careful tuning of parameters, and thus it is adopted for this work. The prediction of velocity field is essentially the regression task of features on given spatial locations. Therefore, the MSE loss function is adopted which can be calculated by

$$L_{MSE} = \sum_{i} (u_{i,pred} - u_{i,truth})^2 (3)$$

For the learning rate, the scheduled changing learning rate strategy on the Unet is experimented, in which the learning rate is continuously decreased by 10 times if the relative error of validation set doesn't decrease in 20 batches. Currently, the fixed learning rate as is adopted for the transformer training. The initial learning rate for both network is set as 10^{-3}.

5.2. Unet Prediction on Pixel-based images

By using Adam optimizer with change learning rate based on scheduled alteration, the loss variations and changing learning rate can be seen in Fig. 5. It can be demonstrated that both the training loss and validation loss are decreased extremely fast during the training process. It indicates the U-net performs extremely well on the velocity field prediction. It should be mentioned that this extremely low loss on both training and validation set is attributed to the physics-informed settings. As mentioned in the physics-driven neural network, the physics-informed procedure is applied before the Unet, and thus the input data can be embedded with physical information in advance, leading to significant improvement on the training process and network performance. This fast decreasing trend of loss is the reason that we implement the decreasing learning rate based on validation error. We tried to further decrease the prediction error, but the profit is much less. Note that, the physics-informed part is not this work's focus, and thus we'd like not to discuss this in detail.

The trained U-net is further tested on the unknown test dataset. One velocity field result is shown in Fig. 6, in which the first row is the velocity along horizontal direction, the second row is the velocity along vertical direction. It can be seen that the velocity field is almost the same between the ground truth and prediction results. Especially for the high velocity region, the prediction results is well consistent with the ground truth.

Based on the distribution of velocity, the performance is further evaluated by using relative error of the whole velocity field as calculated in Eq. (4).

$$Error = \frac{|u_{i,pred} - u_{i,truth}|}{|u_{i,truth}|} (4)$$

As can be seen from the relative error map, most error is distributed near the solid surface of circular. Besides, the

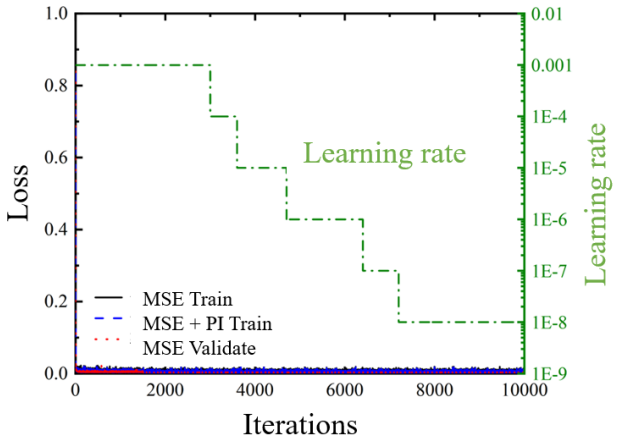

Figure 5. Unet training loss variations

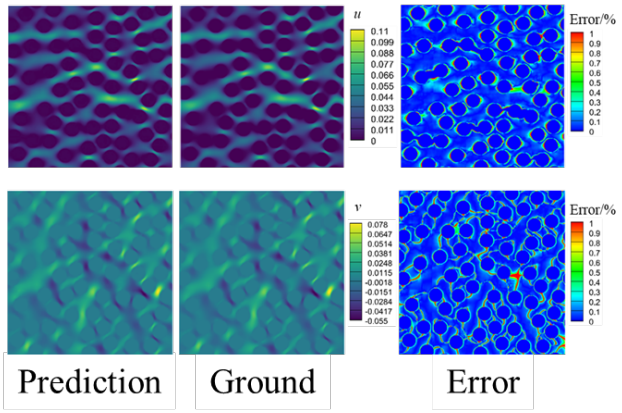

Figure 6. Prediction Results of Unet

large error is also concentrated in the region where flow velocity is relative low, as can be seen from Fig. 6. From the perspective of physical science, it is caused by lower value near solid surface, the low velocity has too much value difference from the velocity in the channel center, leading to the numerical error. From the perspective of computer vision, the pixels almost have the same value near circular surface, and thus very little difference can lead to huge relative error in velocity field. In general, the average relative error of test dataset is 0.0939.

5.3. Transformer Prediction on Point Clouds

During the training process of transformer, the variations of MSE loss on training dataset and validation dataset are shown in Fig. 7. It can be seen that the MSE loss is higher than the Unet training, which is caused by two reasons. Firstly, the transformer doesn't have the physics-informed block, which lacks physical information for the input data. Secondly, the data for transformer is consisted of the sparse points of the domain due to our simplification. Therefore, the MSE loss decreases slightly within the increasing

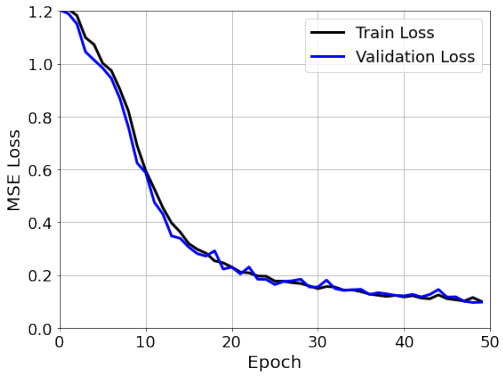

Figure 7. Loss Variations of Transformer

epochs, and finally stabilizes around 0.1. The MSE loss of training and validation set decreases at the same time and maintains in the similar range. Therefore, the transformer is well trained without overfitting or underfitting. More improvement procedure will be implemented to transformer in the future work, such as physics-informed block, padding sequence and learning rate tuning. However, the current results are enough to show the ability of transformer with point-wise data on prediction of flow field. The trained transformer is further tested on the unknown test dataset. For three test data, the distributions of horizontal and vertical velocities are shown in Fig. 8 and Fig. 9, respectively. The distributions are similar with the Unet results. The prediction velocity is almost the same with the ground truth. As can be seen from the results, under the point-wise data structure, the prediction results shows very good performance without physics-informed block and learning rate tuning, indicating the great potential of node point spatial data from unstructured triangular mesh. It should be noted that only 1793 points are included for a geometry, which is far less than the pixel-based image of (240, 240). It demonstrated that the triangular mesh data can provide larger potential for expressing the physical problem in the spatial domain. The saved computational resources enables us to build larger model with higher generalization ability.

The relative error is shown in Fig. 8 and 9, it can be seen that the relative error is still concentrated in the region where the flow velocity is relative low. It is consistent with our conclusion from Unet. However, the transformer shows one advantages on dealing with the boundary nodes. It can be seen that more node points are distributed in some narrow space near the boundaries. It increases the data density in the boundary surface, which helps improving the local accuracy. By calculating the global relative error, the transformer reaches 0.1625 on the test dataset, which is higher than the Unet.

5.4. Discussions

By comparing the prediction results from the Unet with pixel-wise data and the transformer with point-wise data, both of them performs well on our task. The Unet achieves higher accuracy than transformer due to physics-informed block and fine-tuned learning rate. However, more importantly, the undesigned transformer with randomly sampled point clouds shows strong capability on prediction. It should be foreseen that using transformer with unstructured mesh data can provide a better generalization of spatial discretization for physical problems. It not only allows more flexible and efficient spatial data input, but also saves the computational resources for data. allowing us to further extend the model.

6. Conclusions and Future Work

6.1. Conclusions

The discretization form is the most important part to numerically solving physical problems. It describes the physical system (PDEs) from continuous equation view in the spatial domain. It not only determines the accuracy of solutions, but also influences the efficiency of solving process. From the computer vision view, leveraging deep learning method in the physical images represented by different data structures (discretization forms) is a promising routine to lead the innovation of numerical solutions of PDEs.

In this work, with focus on fluid flow inside porous media, the spatial solution is represented in pixel-based images and point clouds, and the Unet and transformer are implemented on each data form. The results show that the Unet exhibits a stronger performance on our task, while the transformer performs weaker. It should be mainly caused by the implemention of physics-informed block in Unet. However, the pixel-based images lacks flexibility on the discretization forms, with computational waste on the solid region where fluid flow doesn't happen. The transformer can handle more flexibile data structure, which should be a more promising solutions. In conclusion, generalizing flexible unstructured mesh into point form or graph form data and leveraging deep learning methods shows good performance on flow prediction, and it should be further investigated.

6.2. Future Work

1. Implement physics-informed block in transformer to improve its performance.

2. Apply padding and mask technology or the latent space prediction. Embed more spatial information of physical problem into the image structure. From the computer vision view, reveal the best data structure for discretization form with the richest and strongest representative ability.

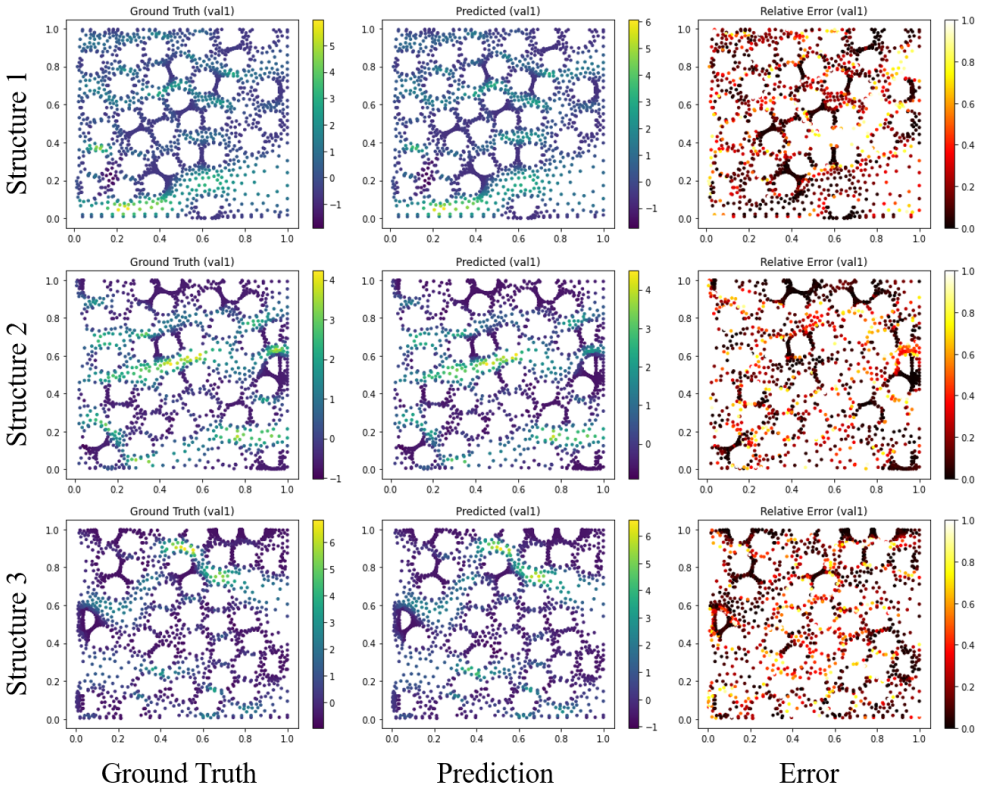

Figure 8. Prediction Results of Transformer on Horizontal Velocity

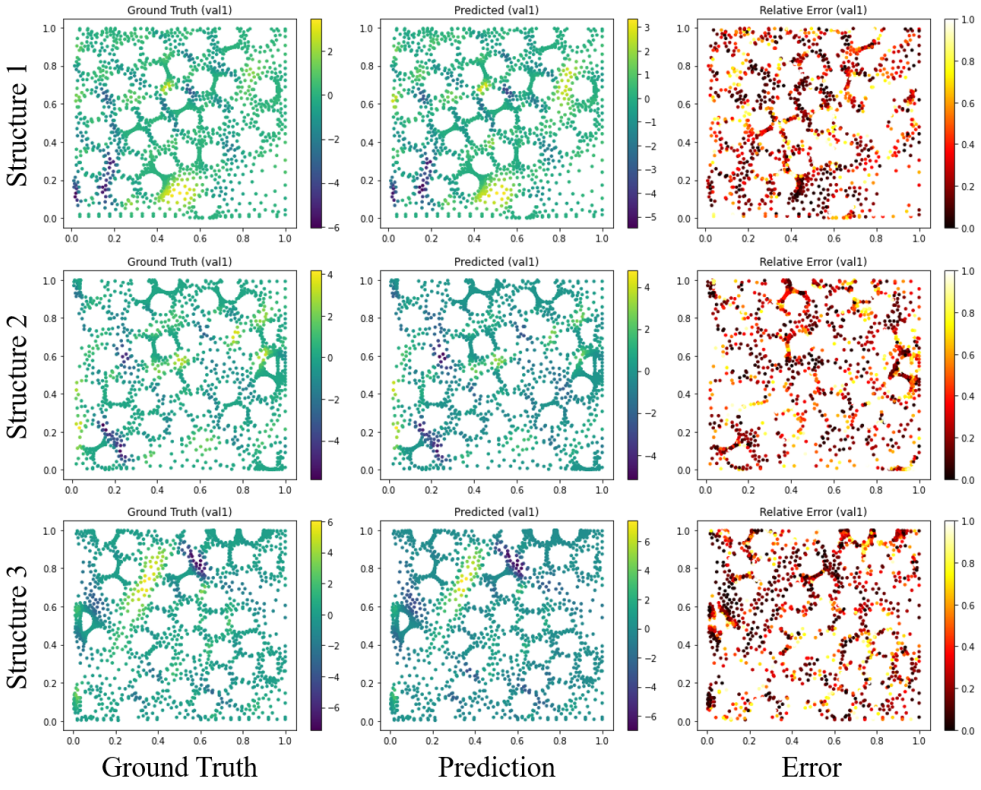

Figure 9. Prediction Results of Transformer on Vertical Velocity

References

[1] M. Cheng, F. Fang, C. C. Pain, and I. Navon. Data-driven modelling of nonlinear spatio-temporal fluid flows using a deep convolutional generative adversarial network. *Computer Methods in Applied Mechanics and Engineering*, 365:113000, 2020.

[2] C. Cowen-Breen, Y. Wang, S. Bates, and C.-Y. Lai. Euler operators for mis-specified physics-informed neural networks. In *ICML 2024 AI for Science Workshop*.

[3] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[4] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

[5] 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*, 378:686–707, 2019.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[7] H. Wu, H. Luo, H. Wang, J. Wang, and M. Long. Transolver: A fast transformer solver for pdes on general geometries. *arXiv preprint arXiv:2402.02366*, 2024.

[8] C. Yang, X. Yang, and X. Xiao. Data-driven projection method in fluid simulation. *Computer Animation and Virtual Worlds*, 27(3-4):415–424, 2016.

7. Contributions and Acknowledgments

The physics-informed block of Unet is from my previous work, and it is re-implemented here. I acknowledge Stanford Sherlock Computing Center for the GPU resources.