Journal of Multidisciplinary Engineering Science and Technology (JMEST)
ISSN: 2458-9403
Vol. 10 Issue 4, April - 2023

KYAMOS Software – Artificial Intelligence prediction of compressible fluid flows

[1,2]Antonis P. Papadakis, [1]Aimilios Ioannou, [1]Sofia Nikolaidou and [1,2]Wasif Almady
KYAMOS LTD, 37 Polyneikis Street, Strovolos, 2047, Nicosia, Cyprus
Frederick University, 7. Y. Frederickou, 1036, Palouriotissa, Nicosia, Cyprus

***Abstract*— In this paper, the authors utilize a novel Finite Volume, Total Variation Diminishing (FV-TVD) scheme to firstly simulate a shock wave propagation inside the Sod shock tube. The simulation results are automated such that multiple number of initial pressures and densities are generated as initial conditions randomly, and multiple results are saved to be used as dataset for deep learning training purposes. Secondly, a state-of-the-art UNET architecture model is developed in python using PyTorch that is used to predict the propagation of the shock, and specifically density, pressure, and velocity. The UNET model is trained and tested using a number of filters during the encoding and decoding process. It is shown that the AI predicted results are in very good agreement with the FV-TVD, depicting that the UNET architecture can successfully and accurately predict the fluid behavior in compressible flows with shocks. It was also found that the AI UNET model was able to produce reasonable results with only 100 epochs, which shows the ability of the UNET model to capture the fluid flow dynamics.**

Keywords—Finite Volume, Artificial Intelligence, UNET; Shock Waves; Compressible Flows;

I. Introduction

A. Scope

In this paper, we discuss the computational fluid dynamics industry, and how it is related to artificial intelligence (AI). For this purpose, the general theory behind AI is discussed and the importance of tuning hyperparameters is highlighted. Specific attention is given to the UNET model and how it can be applied to the CFD industry for fluid flow prediction. A literature review on state-of-the-art UNET algorithms follows, with results from a trained and tested UNET model. Finally, conclusions are drawn.

B. CFD

In Computational Fluid Dynamics (CFD) one needs to solve the appropriate set of equations to characterize the flows under consideration. Depending on the type of flow, one can use different methods and techniques, each with its strengths and weaknesses, to accurately capture fluid flow. Many methods are most suited for specific types of flow; hence the choice of method will greatly affect the result. For example, it is well documented that in terms of field simulations, finite element techniques perform well, especially near sharp corners, when used in combination with non-regular elements such as triangles in two-dimensions and tetrahedrals in three-dimensions. Another example is the Lattice Boltzmann (LB) method, which performs well for incompressible flows, porous media, and turbulence due to its ability to treat boundary conditions effectively, accurately, and efficiently. Another example is the Finite Volume (FV) method which guarantees conservation of flux and performs well in compressible-flow scenarios.

Once a model is chosen, other factors will affect the stability and accuracy of the results, such as the discretization method used to solve a problem. In all methods, there is a time-evolving geometry and flow, hence one needs to discretize the spatial and temporal dimensions, to solve the differential equations that dictate the flow. Differential equations in fluid flows usually include macroscopic variables such as density, velocity, pressure, and energy, which are calculated in a spatiotemporal evolution.

The results are obtained in files at regular intervals, for all grid points, hence a time evolution of the macroscopic quantities can be obtained. In CFD, there are primitive variables and conserved quantities. Primitive variables, as mentioned above, are individual variables that measure a specific quantity. Conserved quantities are momentum (instead of velocity) and energy (instead of pressure).

There are various types of boundary conditions, however the most widely used are the Dirichlet boundary conditions, which set a fixed value of a quantity on the boundary, the Neumann boundary condition which guarantees that the partial derivate of a quantity normal to the boundary will be constant, the no-slip boundary condition which sets the velocities of the flow in all directions to zero and is usually applied to solid surfaces, and the outflow boundary condition which forces zero gradient normal to the boundary to guarantee outflow behavior. Each boundary condition can be implemented in a different way in each of the methods being developed such as within the finite difference, finite element, finite volume, or Lattice Boltzmann methods, depending on the mathematical formulation of each method.

To model fluid flows, there are different types of equations that need to be solved. For example, in the case of inviscid flows, one only needs to solve the Euler equations, which are the Navier-Stokes equations, without the presence of any viscous terms.

In problems where viscosity does not affect the flow, one does not need to solve the Navier-Stokes, but only the simplified form- the Euler equations. For this purpose, within KYAMOS software, in the Graphical User Interface, the user is asked to choose the type of flow to be analyzed and hence choose beforehand the right solver model to run their simulations. The fact that one needs to specify the type of simulation model allows the deployment of dedicated and optimized solvers, specific to the type of flow. In contrast, a generic model would need to include multiple methods, making the solution setup procedure non-optimal and ineffective.

In every simulation, initial conditions are always necessary to dictate the initial behavior of the flow. These initial conditions can drastically influence the state of the system at later times, and together with the boundary conditions, will determine the overall behavior of the flow.

In CFD simulations, one can use regular grids, which makes the location of centroids, faces and volumes easier to determine, however it suffers from the ability to reconstruct sharp corners and surfaces. On the contrary, one can use non-regular meshes which allows the capturing of non-uniform obstacles easier, but includes a randomness in mesh-connectivity and centroid, face and volume recognitions. Each method has its advantages and disadvantages. There is no ideal way to perform simulations as there is always dependency on the geometry in question and the type of flow. In case there are sharp corners that affect the flow, non-uniform grids may be a wiser option, and where smooth surfaces exist, the regular grid may be the most preferred method.

C. *Introduction to AI*

To predict fluid flows, one can solve the differential equations for the problem in question. However, that usually takes long waiting times to obtain the results. In some cases, it is imperative that results are obtained instantly so that immediate action can be taken, or even changes to the input of the simulation that can affect the results. Even though Graphical Processing Units (GPUs) and Central Processing Units (CPUs) are becoming increasingly faster, we are currently unable to provide real live simulations for medium-sized problems, let alone for large scale problems. To be able to achieve real live simulations, we need a groundbreaking alternative method, and this can be achieved using AI.

AI allows the pretraining of a model with various conditions beforehand and once the model is ready and available, it can be used by several users to run under multiple changed parameters. One prerequisite is that during the training of the model, these variations were indeed considered and thus included in the training procedure.

II. AI THEORY

A. *Introduction*

A deep learning AI model is usually trained by going through two distinct phases, the training phase, and the testing phase. Usually, the pretrained cases are 70% and above, with the validation cases ranging from 30% down to 10%. During the training, one must deal with learning rates which dictate the speed at which the artificial intelligence network is learning. A very high learning rate will try to quickly balance, however may also cause instabilities and deviations from the solution due to the large steps involved with optimization towards finding the optimal solution. On the other hand, a very low learning rate, which is ideal, will result in long waiting times due to the multitude of steps that need to be executed, and will sometimes make the training process unproductive. Hence appropriate learning rates must be chosen for each AI model to be trained such that it quickly reaches the optimized solution, without compromising the stability of the result. AI usually includes neurons that possess a weight and a bias, and these weights and biases need to be trained in a manner such that they produce the desired result.

A neural network consists of layers that have a specific purpose, with each layer having several neurons that have a specific purpose to achieve. There is no golden rule on how many layers are necessary and how many neurons on each layer must be included to achieve the desired result. Hence a trial-and-error design is necessary to find the minimum required neural network that can produce the predicted results. Each neuron in the layer needs to be trained by adjusting its weight and bias, and this is achieved by using a forward-backward propagation error correction. Specifically, a random guess is brought forward, and a prediction is made according to the current random weights and biases and interconnections of the neurons. The predicted result, which in the beginning will be highly deviated from the actual result is compared with the actual solution, which is readily available from already obtained numerical simulations; in this case CFD simulations. Hence, a direct comparison between the predicted and the actual results is made and the difference between the two results is calculated. The ultimate task is to have this model produce ideally a zero error. The error uses an optimizer to calculate the error.

The error is backward propagated through simple mathematics and changes the weights and biases of neurons, setting them a step closer to the optimized solution. To calculate the error, the best widely used method is the Adams optimizer which is widely used in AI models. One needs to mention that the error corrector ensures that the backward correction is in the right direction towards the solution, and not deviating away from the solution.

B. *Hyperparameters*

The hyperparameters are parameters of a machine learning algorithm that are predetermined, prior to training and affect greatly the performance and behavior of the training algorithm. Such hyperparameters can be the learning rate, batch size, number of epochs, regularization strength, and the architecture of the neural network, which includes the number of layers of the network, as well as the number of neurons being used in each of the layers. Since these hyperparameters have a great impact on the behavior and performance of the optimization algorithm, it is important to tune these parameters, and most times, this exercise is performed through trial and error. This process is called hyperparameter tuning and is achieved by systematically varying the hyperparameters and evaluating the performance using a validation dataset. This will inevitably involve multiple operations which are time consuming, however it is necessary for achieving the desired result. Each dataset will have its own optimized hyperparameters and there is no single solution for all problems. There is a need for better ways to be able to adjust these values and further research should be conducted on this topic, since it would greatly impact AI simulations. It is usually an iterative process, being time-consuming and computationally expensive, as it often involves training and evaluating multiple models with different hyperparameter configurations. However, hyperparameter tuning is a crucial step in building effective machine learning models and improves the performance of the algorithm and leads to more accurate predictions.

C. Batches

In order not to go back and forth in every sample with error correction, the trained model is split into batches. These are groups of results that are used to find an average error. After each batch, a backward correction is performed for the weights and the biases. Batches are subsets of the training dataset that are processed as one block during the training process. Instead of training the dataset for each sample by adjusting the weights and biases, which can be counterproductive, one can be more efficient by training the weights and biases at more irregular times, achieving more or less the same result, without the cost of frequently going back and forth adjusting the weights and biases. The batch size is a hyperparameter that will depend on the size and complexity of the dataset, the available hardware resources, as well as the complexity of the neural network architecture. A large batch size will result in faster training times, but may result in overfitting or memory issues, as the network may start to memorize the training set, rather than learning generalizable patterns. On the other hand, smaller batch sizes will result in slower training times, but more stable training and better generalization performance as the network is forced to learn from a more diverse set of examples in each batch. As a rule of thumb, it is always best to run as large batch sizes as possible, and to ensure that overfitting does not occur.

D. Epochs

In neural networks, epoch is defined as one complete iteration usage of the whole training dataset. Multiple epochs occur when the whole training dataset is used multiple times to train the model until the result is optimized. It can also be considered as a full pass through all the training dataset. The number of epochs will greatly affect the accuracy and stability of the predicted results and must be tuned accordingly such that we do not overtrain or undertrain the model. In this case, two scenarios may occur, those of underfitting or overfitting. Underfitting occurs when the number of epochs is far from the optimized and the weights and biases are not given enough chance to optimize themselves. In the case of overfitting, by running many more epochs than required, the network may end up memorizing the training dataset and hence being unable to predict differentiable results on new data. To avoid overfitting or underfitting the data, it is important to monitor the performance of the network on a validation set during the training process and stop the training process when the performance on the validation set starts to degrade. Practically speaking, most of the time, the optimal number of epochs is obtained through trial and error. The ideal epoch number will be affected by the network architecture, the complexity and size of the dataset and the choice of the optimization algorithm to be used, as well as learning rates to achieve the best possible performance.

III. UNET Theory

A. *Introduction*

The UNET AI model was proposed by Olaf Ronneberger, Philipp Fischer, and Thomas Brox at the University of Freiburg in Germany in 2015. It is an image segmentation technique that partitions the image into multiple segments, with each segment corresponding to a different region of the image. It was used by the authors in medical imaging applications to segment anatomical structures or identify abnormalities within the body. The basis of UNET is a convolutional neural network, hence it consists of convolutional layers and no fully connected layers. This absence of fully connected layers allows the network to be trained fast and efficiently and can handle images of variable size. It consists of two main parts, the part of the encoder and the part of the decoder. The encoder processes the input image through a number of convolutional layers, with the sole purpose is to gradually decrease the spatial resolution of the image, while at the same time, increasing the number of feature maps. Hence, the output of the encoder encompasses a set of feature maps that can capture the high-level semantic information of the image, for example image features. Then the image is passed to the decoder, where it takes the feature maps and output of the encoder and uses up-sampling in various stages, such as to reconstruct the original size of the input image, by using transposed convolutional layers.

To allow the network to propagate fine-grained details from the encoder to the decoder, the decoder incorporates skip connections by concatenating the feature maps from the encoder with the corresponding feature maps from the decoder at each stage of the up-sampling process. This enables the decoder to refine the segmentation boundary based on the detailed information from the encoder, ensuring that no features have been lost during the encoding process.

UNET has a unique ability to deal with small training datasets by using data augmentation, that involves applying various image transformations to the input image and the corresponding segmentation masks, such as rotations, flips, and scaling. This generates new samples of the original dataset, and it can learn more features of the images, where they wouldn't be without this process.

A Softmax activation function that is used in the output layer of the UNET network, produces a probability distribution over the different classes at each pixel location, which allows UNET to handle multi-class segmentation problems. This is another advantage of the UNET model, in cases when a pixel in the image belongs to several different types of classes, and not just one. Summarizing, UNET is a very versatile AI model which is highly effective in image segmentation. It is based fully on convolutional principles, avoiding fully connected layers. The utilization of skip connections enables it to learn robust features and refine the segmentation boundary based on detailed information from the input image. Its two main advantages are its ability to handle small datasets and multi-class segmentation problems.

B. *UNET in CFD*

Even though UNET has been used as a neural network architecture in a range of applications, it has also been used in Computational Fluid Dynamics (CFD) for simulating the fluid flow and heat transfer phenomena in various engineering applications. UNET has recently gained increased popularity due to its ability to segment and analyze complex geometries and predict the flow around structures efficiently in terms of speed and accuracy.

As mentioned before, UNET's ability to segment and extract important macroscopic features from two and three-dimensional simulations such as speeds, pressure, temperatures, energy, and velocities, emanates from its ability to handle large and complex datasets, as well as its flexibility in handling multi-class segmentation problems. Examples of sophisticated complex problem solving, is its ability to analyze complex flow structures, such as vortices, wakes and turbulent jets.

To analyze these structures, the UNET model can be trained to segment the flow field into different regions based on the characteristics of the flow, such as velocity magnitude, direction, and vorticity. This can provide valuable insights into the dynamics of the flow and help identify areas of high turbulence or energy dissipation.

UNET is also good in identifying hotspots or regions of high heat transfer in heat transfer phenomena, since they would appear as a feature on the image of a temperature distribution result, with direct application in the optimization and design of heat exchangers, combustion systems, etc.

UNET overall is able to capture effectively complex flows and heat transfer phenomena by handling large and complex datasets, while at the same time handling multi-class segmentation problems and can be an ideal choice for CFD analyzing and optimizing phenomena.

IV. Literature review on UNET

The authors used supervised neural networks to predict two-dimensional velocities and pressure fields in laminar flows around arbitrary shapes. The aforesaid fields were obtained via a CFD solver for the Navier-Stokes equations. The data-trained U-net architectures were tested based on their predictive ability for unseen shapes, utilizing ad hoc error functions. The results do not show a clear advantage for predictions of field maps, but the U-nets do indeed produce reasonably accurate results [1].

Vinuesa et al. [2] depict that machine learning is rapidly becoming a core technology in scientific computing, and especially computational fluid dynamics. They discuss applications of AI to accelerate direct numerical simulations, to improve turbulence closure modeling and to develop enhanced reduced-order models. The emerging promising areas of machine learning are also discussed, and the potential limitations of AI are discussed as well, including the UNET model used by the authors of this paper.

In another paper [3], the question "if, how and why an AI can learn about turbulence" is addresses by using the results of Direct Numerical Simulations (DNS) regarding turbulent channel flow and applying deep learning techniques. The analysis is focused on wall data, with the assumption that a multilayer nonlinear network can express the local heat flux normal to the wall by including pressure fluctuations and shear stresses, quantities based on which a prediction of the local heat flux is done via convolutional neural networks (CNNs). The authors find a very high correlation (0.980) between the DNS and CNN results. Similar correlation coefficients were observed even at Reynolds number three times larger than the trained Reynolds number of 180. During post training, the gradient maps of the network gave insight to essential inputs for correct predictions of the local heat flux, as well as for the spatial relationship of this quantity and nearby input fields. An investigation of the usefulness of the model to describe turbulence was also performed with some evidence that it may provide some insight.

CFD simulations for engineering problems are computationally demanding and the most common deep learning methods are not precise enough. A novel deep U-shaped network-long short-term memory (U-Net-LSTM) framework is proposed for fast hydrodynamic prediction. The results agree with CFD simulations with good predictive ability. Comparison with traditional methods showed significantly lower errors and computational power demands are highly reduced, making the method attractive [4].

The authors propose a modified U-Net neural network architecture that utilizes multigrid methods (U-Net-MG). Using this method, the work shows a reduction in the Root Mean Square Error (RMSE) between 20%-70% compared to the unmodified U-Net model and an improved predicting ability of velocity and pressure fields in the case of canonical fluid dynamics for flows past a stationary cylinder, in 2 out-of-phase moving cylinders and past an oscillating airfoil. Both methods, however, have a lower than 1% RMSE [5].

High computational times and memory-demand of CFD solvers have pushed researchers towards neural network, deep learning algorithms, trained by CFD data. In this work, the authors modelled incompressible laminar flow around objects in two dimensions over generated interpolated grid feature data using a deep U-Net model. The results showed a good agreement between the CFD solver FEATool and the U-Net model, also showing accurate representation of the pressure and velocity fields for the tested shapes [6].

In this work, the Navier-Stokes equations are solved with direct numerical simulation, as well as a large-eddy simulation to better model two-dimensional turbulence using deep learning. Referring to both simulation methods, a numerical method that has a ten times higher resolution and is forty to eighty times faster computationally, while maintaining the accuracy of common finite volume or finite difference methods is presented. Thus, as the authors suggest, this machine learning approach opens the door to large-scale modelling processes [7].

This paper proposes a novel AI method that uses deep learning to model fluid flow both locally and globally, while incorporating velocity and pressure, as well as adjustable boundary conditions. The authors claim that their method -namely “CFDformer” performs better than U-Net and TransUNet methods and is able to represent pressures and velocities in flows different than the ones that were used to train the model [8].

In this work, the authors try to bypass the computational cost of CFD simulations by employing Convolutional Neural Networks (CNNs) as well as U-Net. The model is trained offline with a very computationally demanding data processing. Post-training evaluations of neural networks on scenarios used to train the model requires very low computational power. Fluid flow around shapes is explored where the error of the approximations is within 3% margin. Generalization of the model to new data yields good results with an average error of 10% margin [9].

The authors in this paper [10] try to predict steady fluid flows around numerus fixed cylinders using deep learning and machine learning. The model takes as input the cylinder arrangement and outputs the x- and y- components of the velocity fields. Testing the model shows accurate prediction of the flow when the scenario to be modelled has a similar number of cylinders as the ones used to train the model. When a scenario with smaller number of cylinders is tested, the model does not perform well. The authors suggest that this model can be generalized and performs well when a larger number of cylinders is used.

In another work [11], the authors propose a network and training strategy through data augmentation. The procedure uses a symmetric expanding path for precise localization and a contracting path which captures context. The award-winning strategy outperforms existing best practices in the field of biomedicine by less-than-one-second segmentation of a 512x512 image on a modern GPU.

In the context of geological data, this work [12] proposes a deep learning surrogate time-dependent model for to predict two-dimensional dynamic multiphase flow, which incorporates a residual U-net and an autoregressive procedure, with each step of the algorithm being dependent on the previous. The authors suggest that their model, while working with fewer parameters, can achieve similar or improved predictions with also fewer training data.

As physics-based simulation in multiphase flows are computationally expensive, in this work [13], the authors have developed an efficient physics-constrained deep learning model in 3D to tackle the issue. By utilizing the predictive abilities in spatial topology of U-Net, the model takes in the properties of the porous media, fluid properties and well controls to produce predictions of the spatio-temporal evolution of pressure and saturation. The three-dimensional problem is first reduced to two-dimensional to be more efficient by lowering computational costs. Calculations of well flow rate are done in a separate postprocessor model by considering the predicted values of the state variables. When contrasted to physics-based simulations, the proposed method is about 1,400 times faster with low average temporal errors for the predicted state variables; 0.27% for pressure and 0.099% for saturation. Well flow rate has an error of less than 5%. The scheme copes with high fidelity fluid flows and can be used efficiently to model computationally expensive inverse problems.

V. AI Simulated Results

In this section, the results are presented from the utilization of the UNET network developed to predict the shock propagation inside a shock tube. To achieve this, 5,000 images from 50 simulations were generated for each of the input and output parameters that are used to train the model. The AI predicted results are in very good agreement with the FV-TVD results.

A direct comparison of the FV-TVD and the UNET is shown below in Fig. 1 and Fig. 2. They depict the density distribution of the shock wave inside the Sod shock tube at time t = 0.42 s, when the initial density and rho are 0.919 kgm^{-2} on the left and 0.183 kgm^{-2} on the right of the diaphragm, and the pressure is 0.929 Pa on the left and 0.166 Pa on the right of the diaphragm. Specifically, Fig. 1 shows the result obtained from an in-house innovative FV-TVD scheme developed by the authors, whereas Fig. 2 shows the predicted results from the UNET AI model. By conducting a direct comparison, it is shown that the UNET model developed is in very good agreement with the predicted FV-TVD results. Hence, we can confidently use AI predictions to calculate the density distribution in a Sod shock tube problem. The UNET model can be extended to work in various other conditions, for example time evolutions, as well as into other initial conditions, boundary conditions and under very different physics. What is important is that the UNET model has shown remarkable accuracy in predicting the fluid flow in this specific case.

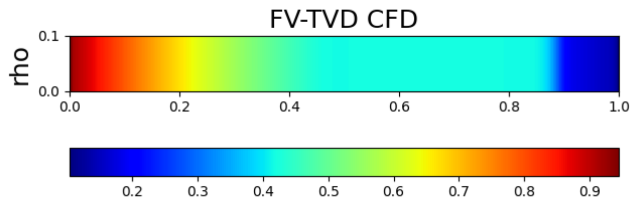

Fig. 1 Density plot for the Sod shock wave tube using the FV-TVD method at time t = 0.42 s.

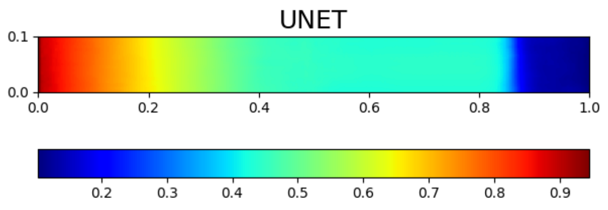

Fig. 2. Density plot for the Sod shock wave tube using the UNET method at time t = 0.42 s.

Fig. 3 below shows the error as a result of the difference in density between the FV-TVD and AI UNET models. It is shown that the error mostly occurs around the shock, where its maximum value is 0.16. This value of the error is very small and can be considered adequate for predicting simulations. By tuning the hyperparameters and feeding the model with a larger dataset, it is expected that more accurate results from the UNET model can be generated.

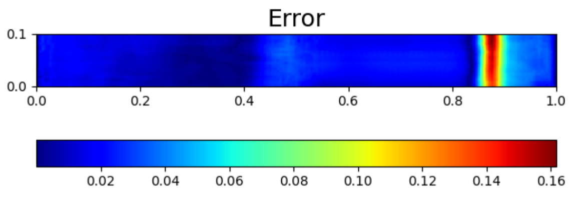

Fig. 3. Density error plot between AI-UNET predicted and real FV-TVD simulations at time t = 0.42 s.

Below in Fig. 4, we present the velocity distribution as a result of the Sod shock tube from the FV-TVD method. Additionally, in Fig. 5, for comparison purposes, we present the predicted results for the velocity based on the UNET AI model. It is shown that by comparing the two graphs, there is adequate agreement between them. Likewise with the density, the pressure also shows the larger differences around the shock.

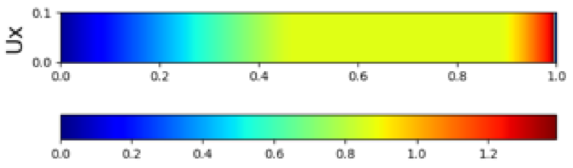

Fig. 2 Velocity plot for the Sod shock wave tube using the FV-TVD method at time t = 0.42 s.

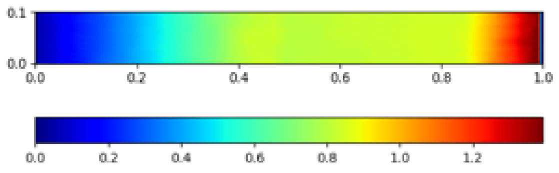

Fig. 3. Velocity plot for the Sod shock wave tube using the UNET method at time t = 0.42 s.

Below, in Fig. 6, the error as a result of the velocity difference between the FV-TVD and AI UNET models is presented. It is shown that the error mostly occurs around the shock, where its maximum value is 0.175. This value of the error is very small, as for the density and can be considered adequate for predicting simulations. Again, by fine tuning the hyperparameters and feeding the model with a larger dataset, it is expected that more accurate results from the UNET model can be generated.

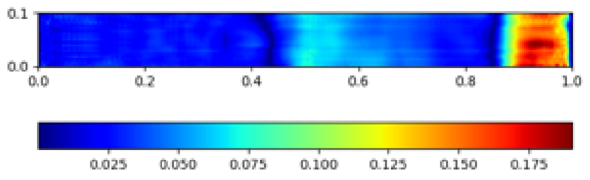

Fig. 4 Velocity error plot between AI-UNET predicted and real FV-TVD simulations at time t = 0.42 s.

Below in Fig. 7, the pressure distribution emanating from the Sod shock tube from the FV-TVD method is presented. Additionally, in Fig. 8, for comparison purposes, we present the predicted results for the pressure distribution based on the UNET AI model. It is shown again that by comparing the two graphs, there is adequate agreement between them.

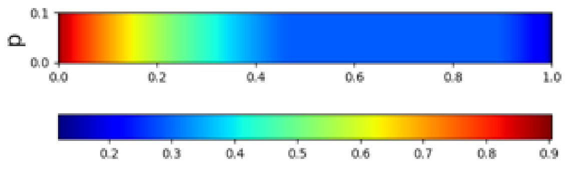

Fig. 5 Pressure plot for the Sod shock wave tube using the FV-TVD method at time t = 0.42 s.

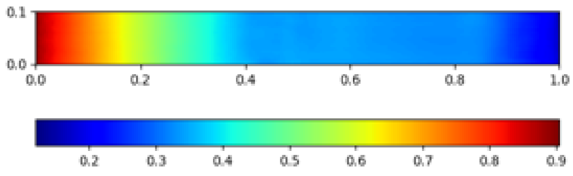

Fig. 6 . Pressure plot for the Sod shock wave tube using the UNET method at time t = 0.42 s.

Finally, in Fig. 9 the error as a result of the difference in pressure between the FV-TVD and AI UNET models is presented. It is shown that the error mostly occurs around the shock, where its maximum value is 0.06. This value of the error is the smallest out of the three distributions, which shows that the UNET AI model can capture the pressure distribution better than the density and velocity.

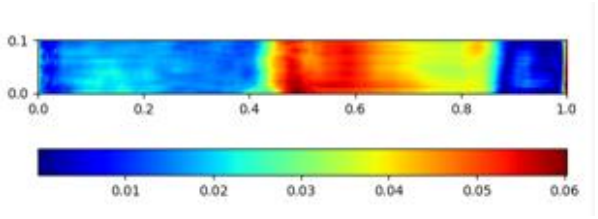

Fig. 7 Pressure error plot between AI-UNET predicted and real FV-TVD simulations at time t = 0.42 s.

VI. Conclusions

In this paper, we provided a direct comparison between the results emanating from an innovative FV-TVD scheme and an AI UNET model. A direct comparison is conducted between the results, calculating the error between expected and predicted results. It was shown that the AI UNET model is more than capable of capturing and predicting the results of fluid flows, i.e., of the density, velocity and pressure with the errors emanating to be acceptable for fluid flow simulations. Further work should be conducted in investigating the effect of the hyperparameters such as of the learning rate, batch size, number of filters, and the number of the input datasets and number of epochs.

Acknowledgment

This work was co-funded by the European Regional Development Fund of the European Union and the Republic of Cyprus through the Research and Innovation Foundation (Project: REALISATION-COMPRESSIBLE-GPU-AI CONCEPT/0521/0045).

References

[1] J. Chen, J. Viquerat, and E. Hachem, "U-net architectures for fast prediction in fluid mechanics," 2019.

[2] R. Vinuesa and S. L. Brunton, "Enhancing computational fluid dynamics with machine learning," *Nature Computational Science,* vol. 2, no. 6, pp. 358-366, 2022.

[3] J. Kim and C. Lee, "Prediction of turbulent heat transfer using convolutional neural networks," *Journal of Fluid Mechanics,* vol. 882, p. A18, 2019, Art. no. A18.

[4] Y. Hou, H. Li, H. Chen, W. Wei, J. Wang, and Y. Huang, "A novel deep U-Net-LSTM framework for time-sequenced hydrodynamics prediction of the SUBOFF AFF-8," *Engineering Applications of Computational Fluid Mechanics,* vol. 16, no. 1, pp. 630-645, 2022/12/31 2022.

[5] Q. T. Le and C. Ooi, "Surrogate modeling of fluid dynamics with a multigrid inspired neural network architecture," *Machine Learning with Applications,* vol. 6, p. 100176, 2021/12/15/ 2021.

[6] T.-T.-H. Le, H. Kang, and H. Kim, "Towards Incompressible Laminar Flow Estimation Based on Interpolated Feature Generation and Deep Learning," *Sustainability,* vol. 14, no. 19, p. 11996, 2022.

[7] D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, and S. Hoyer, "Machine learning–accelerated computational fluid dynamics," *Proceedings of the National Academy of Sciences,* vol. 118, no. 21, p. e2101784118, 2021.

[8] H. Kang *et al.*, "A new fluid flow approximation method using a vision transformer and a U-shaped convolutional neural network," *AIP Advances,* vol. 13, no. 2, p. 025233, 2023.

[9] M. Eichinger, A. Heinlein, and A. Klawonn, "Stationary flow predictions using convolutional neural networks," in *Numerical Mathematics and Advanced Applications ENUMATH 2019: European Conference, Egmond aan Zee, The Netherlands, September 30-October 4*, 2020, pp. 541-549: Springer.

[10] H. Ozaki and T. Aoyagi, "Prediction of steady flows passing fixed cylinders using deep learning," *Scientific Reports,* vol. 12, no. 1, p. 447, 2022/01/10 2022.

[11] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18,* 2015, pp. 234-241: Springer.

[12] Z. Jiang, P. Tahmasebi, and Z. Mao, "Deep residual U-net convolution neural networks with autoregressive strategy for fluid flow predictions in large-scale geosystems," *Advances in Water Resources,* vol. 150, p. 103878, 2021/04/01/ 2021.

[13] B. Yan, D. R. Harp, B. Chen, and R. Pawar, "A physics-constrained deep learning model for simulating multiphase flow in 3D heterogeneous porous media," *Fuel,* vol. 313, p. 122693, 2022.