NLP Inspired Training Mechanics For Modeling Transient Dynamics

Lalit Ghule
Office of CTO
Ansys Inc.
Canonsburg, PA 15317
lalit.ghule@ansys.com

Rishikesh Ranade
Office of CTO
Ansys Inc.
Canonsburg, PA 15317
rishikesh.ranade@ansys.com

Jay Pathak
Office of CTO
Ansys Inc.
San Jose, CA 95134
jay.pathak@ansys.com

Abstract

In recent years, Machine learning (ML) techniques developed for Natural Language Processing (NLP) have permeated into developing better computer vision algorithms. In this work, we use such NLP-inspired techniques to improve the accuracy, robustness and generalizability of ML models for simulating transient dynamics. We introduce teacher forcing and curriculum learning based training mechanics to model vortical flows and show an enhancement in accuracy for ML models, such as FNO and UNet by more than 50%.

1 Introduction

Numerical simulations are proving to be of paramount importance across different avenues of industrial and research development. These simulations are performed by solving partial differential equations, which are represented on a discretized computational domain using finite difference or finite volume methods. These methods provide accurate predictions, but they are computationally very expensive. As a result, researchers in the deep learning community have devised many different models to learn physics behind these engineering problems using supervised learning methods, that determine the input to output mapping [1, 2, 3, 4, 5] or unsupervised learning methods, that embed physical laws into loss functions to compute PDE solutions [6, 7, 8, 9]. These physics-informed methods provide a unique benefit over most approaches by imposing initial and boundary conditions in the optimization process. Even though these surrogate models perform reasonably well, they suffer from error accumulation, especially in the extrapolation regime. The accumulation of error is worse for transient problems because the deep learning models diverge during inference. Many researchers have tried to solve this problem by proposing different operator networks like Fourier Neural Operator (FNO) [1], Multiscale Neural operator [10], Koopman operator [11] etc. but the error accumulation problem for long time range predictions is still prevalent.

This problem is very common in the field of natural language processing. In the models that translate one language to another, error-prone predictions in the beginning lead to a completely different output during sequential rollouts. To tackle this problem, researchers in this field have come up with several approaches. William et al. [12] proposed an algorithm named Teacher forcing in their paper on training recurrent neural networks back in 1989. This approach suffers from over-generalization on training data and performs worse during inference. Later in 2015, Bengio et al. [13] proposed a new training mechanism in their paper, "scheduled sampling for sequence prediction", also known as Curriculum Learning.

Taking inspiration from the natural language processing field, we demonstrate the effectiveness of these methods in the field of numerical simulations. The most common way to train transient models is to roll out the whole trajectory using its own predictions right from the beginning of the training process and calculating loss over all the predictions [1]. In this study, we show that Teacher Forcing

and Curriculum Learning techniques yield better results as compared to the regular training procedure. Specifically, we show that Curriculum Learning outperforms all other approaches during inference.

2 Method

During training a model for any transient (time-series) problem, we predict a sequence of outputs and calculate the loss between predictions and ground truth. Let's consider an example, shown in Eq. 1, of a general transient system, where we can map solutions from $t_1, t_2, \ldots, t_n$ to t_{n+1} using a neural network.

$$t'_{n+1} = \Theta\left(t_n, t_{n-1}, t_{n-2}, \ldots, t_1\right) \tag{1}$$

Here Θ represents the parameters of the model. To predict the next time step, t_{n+2}, we pass in the previous n steps consisting of its own prediction. In Eq. 2, the solution prediction of t_{n+2} inputs the latest prediction at t'_{n+1} as well as the remaining n-1 historical solutions. This approach is also used by Zongyi Li et al., in their work on Fourier Neural Operators [1].

$$t'_{n+2} = \Theta\left(t'_{n+1}, t_n, t_{n-1}, t_{n-2}, \ldots, t_2\right) \tag{2}$$

During training time, as the ground truth for t_{n+1} is available, we can use the ground truth solution, as shown in Eq. 3. As a result, the model always gets the correct sequence of previous n time steps during training. This approach of handling time history is known as Teacher Forcing [12].

$$t'_{n+2} = \Theta\left(t_{n+1}, t_n, t_{n-1}, t_{n-2}, \ldots, t_2\right) \tag{3}$$

The models trained with this approach often perform better than the approach used in Eq. 2. This is because the training in the previous approach is less robust due to noisy loss formulations resulting from error accumulation over long-time ranges. However, a model trained with teacher forcing never learns to correct itself as it never sees its own prediction during training. This results in divergence from the correct behavior for long-time predictions.

Curriculum Learning [13], an advanced variation of Teacher Forcing algorithm, is a mix of the approaches stated in the Eqs. 2 and 3. In this approach, we randomly decide whether to use t_{n+1} (target) or t'_{n+1} (prediction) for future predictions during each training epoch. Fig. 1 explains the curriculum learning approach in more detail. Let e be the ratio of number of targets used to the length of the sequence. In the initial phase of the training (when e=1), there is a heavier emphasis on using the target as shown in the Fig. 1(a), where all future predictions are computed from historical ground truth solutions. In the middle phase of the training, e begins to decrease and the number of predicted solutions used for future predictions are more. For example, in Fig. 1b the prediction of t'_7 depends on the previous prediction t'_6. Lower e values corresponds to use of more predicted solutions for future predictions. Finally, as shown in Fig. 1c, during the end phase of training all future predictions are computed from previously predicted solutions. The slow transition from using targets to predictions in future predictions allows for stable training and accurate models. Moreover, the training imitates the actual inference process and hence, improves the robustness of the model

3 Experiments

In this work, we use a publicly available dataset to demonstrate the effectiveness of Curriculum Learning technique. The dataset solves the 2D vorticity equation derived from Navier-Stokes for a viscous, incompressible fluid on a unit torus. The viscosity is 1e-3 and the initial condition is solved for 50 timesteps on a 64 x 64 structured grid. More information about the dataset can be found in the work by Zyongi et al. [1]. The dataset has 5000 samples. 4000 samples are used for training, 500 samples are used for validation and 500 samples are used for testing. Out of 50 timesteps, only the first 40 are used for training. The last 10 timesteps are used to check the extrapolation performance.

There are two models used, a UNet model [14] and FNO 2D time model [1]. These models are trained with three different schemes.

- The whole roll-out using model predictions (This is similar to the training mechanics used in [1])
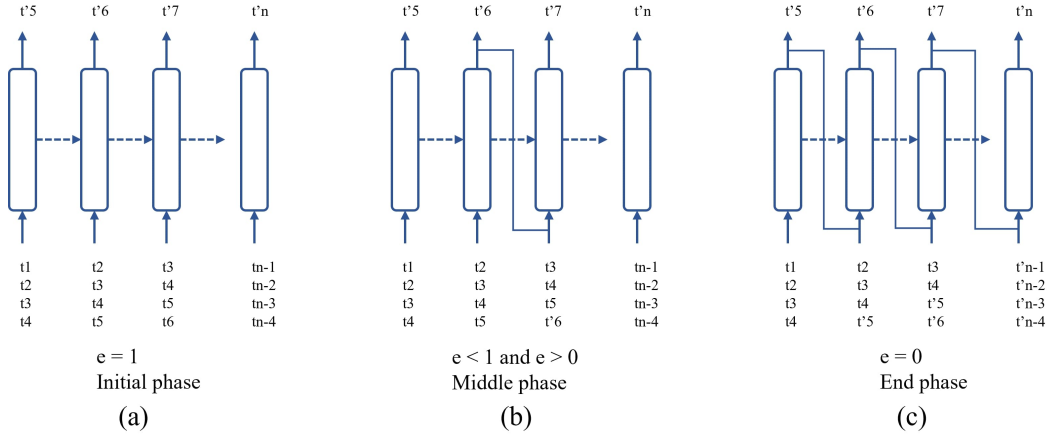- Teacher forcing

t'5   t'6   t'7   t'n   t'5   t'6   t'7   t'n   t'5   t'6   t'7   t'n

t1   t2   t3   tn-1   t1   t2   t3   tn-1   t1   t2   t3   t'n-1
t2   t3   t4   tn-2   t2   t3   t4   tn-2   t2   t3   t4   t'n-2
t3   t4   t5   tn-3   t3   t4   t5   tn-3   t3   t4   t'5   t'n-3
t4   t5   t6   tn-4   t4   t5   t'6   tn-4   t4   t'5   t'6   t'n-4

$e = 1$ $e < 1$ and $e > 0$ $e = 0$
Initial phase   Middle phase   End phase

(a)   (b)   (c)

Figure 1: The model training starts with e=1, where predictions are made based on ground truth (a). In the middle phase, (b), the model sometimes takes ground truth and sometimes take its own prediction. This phase helps model to correct itself based on its own prediction. In the end, (c), the model is trained only with its own prediction. The horizontal dotted lines indicate that the same model weights are used for the next prediction.

- Curriculum Learning

During training, the model takes the first 10 timesteps as input and predicts the next one timestep. The model is rolled out for all the remaining 30 time steps before the gradient descent step. At the time of inference, the model is rolled out on all the 40 timesteps in the same fashion. The mean squared error loss is calculated on the whole sequence of 30 timesteps predictions. The model is trained for 500 epochs with Adam optimizer. The initial learning rate is 0.001 and is subsequently halved after every 100 epochs. For Fourier Neural Operator model, the hyperparameters used are same as mentioned [1]. All experiments are run on a single Nvidia Tesla A100 GPU. The decay scheme used for e, in Curriculum Learning, is linear.

4   Results

Table 1 shows the relative L2 norm on the test set. The numbers in the table are averaged over 500 test samples. The first technique gives satisfactory results. The Teacher forcing models show better results. The FNO model improves by 32.5%. On the other hand, UNet shows a remarkable improvement of 50%.

Table 1: Relative L2 norm comparison for different solution approaches applied to UNet and FNO

Solution approaches	FNO 2D time	UNet
Roll-out using model predictions	0.046	0.082
Teacher Forcing	0.031	0.041
Curriculum Learning	0.025	0.027

The Teacher Forcing improves the performance across all models. Curriculum Learning further improves the error rate. Curriculum Learning outperforms all other approaches. The FNO model improves by 52% over the baseline. It achieves the best performance. UNet also improves by 67% over its baseline. It may be observed that the UNet model is benefitted the most. With general training mechanics, the FNO model is inherently better in mapping transient problems than UNet, but with curriculum learning the results from UNet and FNO are very close.

The following graphs in Fig. 2 show the roll-out error on the test set. It is evident that the roll-out error improves with Curriculum Learning. The error seems to rise with a significantly smaller rate compared to corresponding baselines. Also, in the extrapolation region (time steps beyond 40), the model tends to behave reasonably well. As the data is unseen for the model, the error rate increases in this region for both models
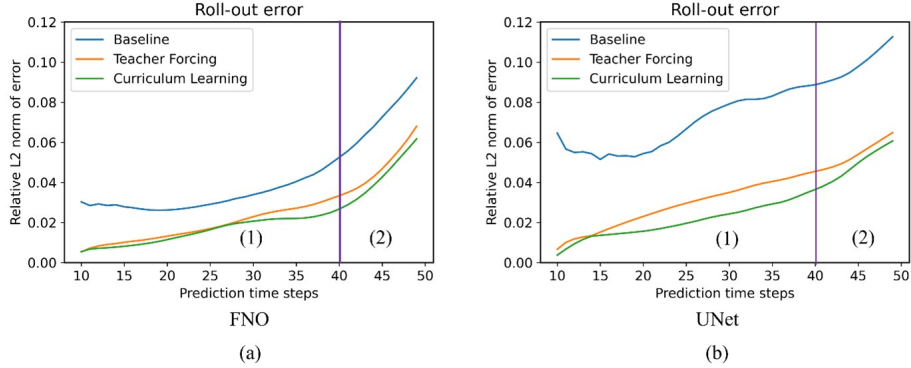
Figure 2: Roll-out error. The graphs (a) and (b) shows the average error over 500 test samples for FNO and UNet model respectively. The vertical line at time step number 40 represents the separation between interpolation and extrapolation region. (1) shows interpolation region whereas (2) shows extrapolation region.

In Fig. 3, we show some of the snapshots of the output produced by the UNet model for a test sample. The figure shows targets, baseline, and curriculum learning outputs for certain time steps. In the figure, circle (A) represents the approximate location in time and space where the prediction starts to deviate from the ground truth. This deviation accumulates and diverges from the actual trajectory as evident in circle (B). The curriculum learning output also has some deviation in the beginning, but it sufficiently corrects itself and does not deviate vastly from the target. One thing to note here is that the last 2 time steps are from the extrapolation region which the model never learned during the training. This indicates that the model's performance significantly improves in the extrapolation region using this training mechanism.
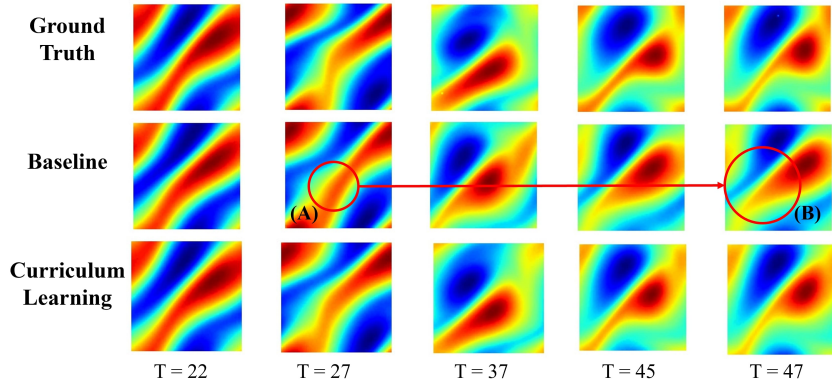

Figure 3: Error accumulation visualization. The figure shows the evolution of the solution for the baseline and Curriculum learning.

5   Conclusion and Future work

This work shows the effectiveness of Curriculum Learning in learning PDEs for better performance during inference. It also shows that the techniques from other fields like Natural Language Processing can be beneficial in deep learning for numerical simulations. The simple change in the training mechanics help in better generalization and extrapolation as the model gradually learns to correct itself during training using its own predictions. This is possible because the training objective slowly gets similar to the inference. We expect that this method can be used for any generalized transient problems to improve the overall performance for any model architecture. Future work includes trying out different decay schemes like exponential or inverse sigmoid for curriculum learning. The investigation of the effect of number of epochs on the convergence could be useful in case of

challenging datasets. The relationship between the learning rate decay scheme and the performance improvement with this method needs to be studies for stable convergence.

6 Broader Impact

Even though this work grew out of solving Navier-Stokes 2D vorticity equation, the broader level point is that we can leverage the proposed method for any transient system. The method independently provides an extra bit of accuracy and can be used to accelerate numerical simulation-based deep learning research for tackling error propagation in other fields of science and technology.

References

[1] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[2] Saakaar Bhatnagar, Yaser Afshar, Shaowu Pan, Karthik Duraisamy, and Shailendra Kaushik. Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics*, 64(2):525–545, 2019.

[3] Yinhao Zhu and Nicholas Zabaras. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, 366:415–447, 2018.

[4] Xiaoxiao Guo, Wei Li, and Francesco Iorio. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 481–490, 2016.

[5] Rishikesh Ranade, Chris Hill, Lalit Ghule, and Jay Pathak. A composable machine-learning approach for steady-state simulations on high-resolution grids. *arXiv preprint arXiv:2210.05837*, 2022.

[6] Leah Bar and Nir Sochen. Unsupervised deep learning algorithm for pde-based forward and inverse problems. *arXiv preprint arXiv:1904.05417*, 2019.

[7] Jonathan D Smith, Kamyar Azizzadenesheli, and Zachary E Ross. Eikonet: Solving the eikonal equation with deep neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 59(12):10685–10696, 2020.

[8] Maziar Raissi, Paris Perdikaris, and George 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.

[9] Rishikesh Ranade, Chris Hill, and Jay Pathak. Discretizationnet: A machine-learning based solver for navier–stokes equations using finite volume discretization. *Computer Methods in Applied Mechanics and Engineering*, 378:113722, 2021.

[10] Björn Lütjens, Catherine H Crawford, Campbell D Watson, Christopher Hill, and Dava Newman. Multiscale neural operator: Learning fast and grid-independent pde solvers. *arXiv preprint arXiv:2207.11417*, 2022.

[11] Kaushik Balakrishnan and Devesh Upadhyay. Stochastic adversarial koopman model for dynamical systems. *arXiv preprint arXiv:2109.05095*, 2021.

[12] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[13] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28, 2015.

[14] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.