Machine Learning Surrogate Models for Landau Fluid Closure

Chenhao Ma, a) Ben Zhu, b) Xue-qiao Xu, and Weixing Wang
1) Princeton Plasma Physics Laboratory
2) Lawrence Livermore National Laboratory

(Dated: 26 September 2019)

The first result of applying the machine/deep learning technique to the fluid closure problem is presented in this letter. As a start, three different types of neural networks (multilayer perceptron (MLP), convolutional neural network (CNN) and two-layer discrete Fourier transform (DFT) network) were constructed and trained to learn the well-known Hammett-Perkins Landau fluid closure in configuration space. We found that in order to train a well-preformed network, a minimum size of training data set is needed; MLP also requires a minimum number of neurons in the hidden layers equal to the degrees of freedom in Fourier space despite training data is fed in configuration space. Out of three models DFT performs the best for the clean data most likely due to the existence of nice Fourier expression for Hammett-Perkins closure but it is least robust with respect to input noise. Overall, with appropriate tuning and optimization, all three neural networks are able to accurately predict Hammett-Perkins closure and reproduce the inherit nonlocal feature, suggesting a promising path to calculate more sophisticated closures with the machine/deep learning technique.

The fluid closure problem in plasma physics is probably as old as plasma physics itself. The problem arises when deriving fluid equations through the chains of moment equations for kinetic theories, the resulting lower order moment equations always contain higher order moments. To truncate the moment hierarchy, a proper closure is thus required to approximate this higher order moment from existing lower order moments for microscopic descriptions, which is conventionally constructed by phenomenological constitutive relations. Moment closure hierarchies for kinetic theories are important and active scientific researches in fluid dynamics, plasma physics, neuroscience, and so on. The widely used Spitzer-Harr amplitude and wave-vectors in Fourier space.

On a separate note, machine/deep learning technique has made a significant progress and started to emerging physical science research recently. Inside fusion plasma physics community, models have been trained to assist experimental data analysis, carry out real-time modeling of neutral beam injection, aid operation control (e.g., disruption prediction), accelerate evaluation of the core turbulent transport fluxes and the pedestal structure, design better ICF experiments and so on. Here we are interested in another kind of machine learning applications – generating machine learning surrogate models of phenomena for microscopic descriptions that can be used within models for fluid simulations, yielding better and faster solutions. An interesting, open question is thus raised – is it possible to get a machine-learned closure model for plasma physics? Or in other words, can a neural network be trained with existing theories, or data from simulations to learn the closure which could be further applied to fluid simulations? If the answer is positive, then how to build and train such a model? What is the performance both in terms of accuracy and speed-up?

As a proof of principle, we started with training neural networks to learn Hammett-Perkins closure – it is relatively simple (thus understandable), analytically solvable (thus easy to evaluate errors), yet non-trivial as it represents certain aspects of the general closure problems (e.g., the nonlocal feature explained below). Furthermore, even though it is simple in Fourier space, it is very difficult or impossible to be implemented in fluid simulation codes, such as BOUT++, in configuration space due to the particular discretizations, geometry and boundary conditions. In such circumstances training Hammett-Perkins closure in configuration space is desired.

The Hammett-Perkins closure gives the relation between temperature and heat flux perturbation in Fourier space as

\[
\tilde{\hat{q}}_k = -n_0 \sqrt{\frac{8}{\pi}} v_i \tilde{T}_k |k|.
\]

Transforming Eq.(1) from Fourier space back to configuration space yields

\[
\tilde{\hat{q}}(x) = -n_0 \sqrt{\frac{8}{\pi^3}} v_i \int_0^\infty dx' \left[ \tilde{T}(x + x') - \tilde{T}(x - x') \right] / x'.
\]

---

a) Electronic mail: cma@pppl.gov
b) Electronic mail: zh12@lbnl.gov
The convolution in Eq. (2) suggests that the heat flux $\hat{q}(x)$ in configuration space is nonlocal (i.e., the perturbed heat flux $\hat{q}(x)$ at location $x$ relies on not only the local temperature $\hat{T}(x)$ and its gradient $\partial \hat{T}(x)/\partial x$, but also the nearby temperature perturbations $\hat{T}(x \pm x')$). The motivation of this study is to assess neural networks’ capability of recovering nonlocal feature. It shall be noted that in Fourier space the expression of Hammett-Perkins closure is linear and local; thus, training a neural network for Eq. (1) becomes trivial. Therefore, in this letter, we focus on training neural networks to learn Hammett-Perkins closure Eq. (2) in configuration space. At the same time, we try to answer several technical questions, for example, how much data is needed to train a neural network? Is there any constraints on neural network configuration? Which type of neural network performs best and why?

For simplicity we assume plasma has uniform density $n_0$, simulation domain $\mathbf{x} \in [0, 2\pi]$, resolution $n_x = 128$ with periodic boundary condition. To prepare the data set for model training, validation and testing, a series of fluctuating temperature profile $\{\hat{T}_i(x)\}$ is first generated with

$$\hat{T}_i(x) = \sum_{k=1}^{k_{\text{max}}} A_{ik} \sin(kx + \phi_{ik}),$$

(3)

where uniformly distributed $A_{ik}$ and $\phi_{ik}$ are randomly chosen in the range $0 \leq A_{ik} \leq 1$ and $0 \leq \phi_{ik} < 2\pi$; the corresponding $\hat{q}_i(x)$ is then calculated with Eq. (1) by Fourier transform. Fig. 1 shows one pair of $\hat{T}(x)$ (blue) and $\hat{q}(x)$ (red). Finally, paired temperature and heat flux profiles $\{\hat{T}_i(x), \hat{q}_i(x)\}$ are re-normalized to fit in between $-1$ to $1$,

$$\hat{T} = \frac{\hat{T}}{T_{\text{max}}}, \quad \hat{q} = \frac{\hat{q}}{q_{\text{max}}},$$

(4)

where $T_{\text{max}}$ and $q_{\text{max}}$ are the maximum absolute value of $\hat{T}_i$ and $\hat{q}_i$ in the entire data set, respectively.

In this study, three types of neural networks were constructed with Keras interface, and TensorFlow backend as shown in Figure 2. The first type of neural network is conventional multilayer perceptron (MLP). It contains one input layer, two hidden layers and one output layer. Either ReLU or tanh function is chosen as the activation function in MLP except at the last layer where linear function is applied. The second type of neural network is convolutional neural network (CNN). It has an input layer, double convolution-pooling layers, and followed by two convolution layers, double upsampling-convolution layers. In CNN the activation function is ReLU for hidden layers, and linear for output layer. The third type of neural network is a two-layer discrete Fourier transfer (DFT) neural network. All three networks use standard Adam optimization algorithm, mean-squared-error (MSE) as the loss function for training and validation. However, in the following discussion, we primarily use mean-absolute-error (MAE) and the mean of MAE distribution $\epsilon$ as the error metric since it is more intuitive (i.e., no square-root operation is needed to evaluate the relative error).

Neural networks are known to be “data-consuming”, i.e., the performance of the neural networks strongly depends on the amount of training data. A well trained neural network shall also be neither overfit nor underfit, i.e., has a minimum discrepancy between training and validation loss. The first task is therefore trying to find
out how much data is adequate for this simple 1D closure problem by comparing the training and validation loss. We started to train the MLP model with increasing size of training data $n_{\text{sample}}$.

Fig. 3 illustrates the training history of MLP with $n_{\text{sample}} = 100$ and $n_{\text{sample}} = 10,000$. Here epoch in the horizontal axis means the number of times that the entire training data set is passed to the neural network. In both cases, training and testing MAE start at similar value. During the training process, the training MAE decreases much faster than testing MAE for $n_{\text{sample}} = 100$ case; and the discrepancy between training and testing MAE at the converged phase indicates that MLP is overfitted, suggesting the training data set is insufficient. For the $n_{\text{sample}} = 10,000$ case, the training MAE and testing MAE have similar values almost over the whole training process. We hence concluded that at least $n_{\text{sample}} = 10,000$ training data is required for the 1D closure problem at this resolution for MLP. Fig. 3 also shows that the trained model tends to converge faster with a larger data set: when $n_{\text{sample}} = 100$, model converges near epoch=150; while for $n_{\text{sample}} = 10,000$, it converges near epoch=60.

In machine/deep learning, hyperparameter refers to a parameter that is set before the training process, for example, the activation function selection, hidden layer configuration, optimization algorithms and so on. Choosing correct hyperparameter is critical for model performance. Here, we explored the impact of different activation functions. Fig. 4 shows the MAE of different type of neural network with different activation functions and different $n_{\text{sample}}$ used in the training. In general discrepancy between training MAE and testing MAE decreases when $n_{\text{sample}}$ increases from 10 to 1,000,000, except for MLP with $n_{\text{sample}} = 10$ when the training MAE is 6th order of magnitude smaller than testing MAE. As discussed above, the apparent departure is an indicator that model is either overfit or underfit. In this case, MLP is clearly overfit as $n_{\text{sample}}$ is too small and the resulting model poorly predicts the heat flux for testing data. On the other hand, the near-perfectly overfit with $n_{\text{sample}} = 10$ result proves that fully connected MLP is indeed able to capture the nonlocal effect of Hammett-Perkins closure Eq.\(2\). When $n_{\text{sample}}$ increases, the training MAE increases since model can no longer overfit all the training data, meanwhile, the testing MAE decreases, until they converge to the same value when $n_{\text{sample}}$ is large enough. Test results also show that the MLP with tanh function converges with less data at $n_{\text{sample}} = 100$, and gives a smaller testing MAE than the MLP with ReLU function when $n_{\text{sample}} \lesssim 1,000$. This is likely due to tanh function suffers from the vanishing gradient problem which prevents the neural network from further training even with more data. Thus, MLP with ReLU has better performance when training data is larger (e.g., $n_{\text{sample}} > 10,000$ in our test).

The width and depth of neural networks (i.e., the numbers of neurons in the hidden layers $n_{\text{neuron}}$ and the number of hidden layers) are also crucial hyperparameters in a neural network. Here we first tested the impact of $n_{\text{neuron}}$ on the testing MAE for a two hidden layers MLP. Fig. 5 shows the testing MAE varies with $n_{\text{neuron}}$ for different number of modes $k_{\text{max}}$ in the system. Because the resolution is set to be $n_x = 128$ for $\bar{T}$ and $\bar{q}$, the maximum number of allowable Fourier modes in the system is 64. We scanned the $k_{\text{max}}$ in Eq.\(3\) from 8 to 64 by generating separated data sets, and training the neural network with $n_{\text{sample}} = 10,000$. The sudden dropping of MAE for all cases demonstrates that there is a threshold of minimum number of neurons required in each hidden layer $n_{\text{neuron}}$ in order to accurately recover Hammett-Perkins
closure. The threshold is approximately
\[ n_{\text{th}}^\text{neuron} = 2k_{\text{max}}. \] (5)
When \( n_{\text{neuron}} < n_{\text{th}}^\text{neuron} \), the testing MAE is always larger than 0.1; while when \( n_{\text{neuron}} \geq n_{\text{th}}^\text{neuron} \) the testing MAE rapidly reduces an order of magnitude or more. The threshold \( 2k_{\text{max}} \) coincides the degrees of freedom of the input \( \tilde{T}(x) \) in Fourier space representation – that is, amplitude \( A_k \) and phase shift \( \phi_k \) are randomly chosen for each \( k \), suggesting MLP is able to learn the Fourier space information despite the training data is fed in configuration space. This also implies that if the input data contains full-spectrum information, then \( n_{\text{neuron}} \) shall be at least equal to \( n_x \) in order to resolve all Fourier modes. It should also be noted that based on our tests, increasing \( n_{\text{neuron}} \) well beyond \( n_{\text{th}}^\text{neuron} \) does not necessary improving model performance dramatically especially for the low \( k_{\text{max}} \) cases. As for the depth of a model, no significant performance improvement was found as the number of hidden layers was gradually increasing from 1 to 6 (\( \epsilon = 2.89 \times 10^{-3} \) to \( \epsilon = 2.69 \times 10^{-3} \)). Therefore, we concluded (1) the minimum number of neurons of hidden layers depends on the input training data – in general, \( n_{\text{neuron}} \) should be a little bit larger than \( n_{\text{th}}^\text{neuron} = 2k_{\text{max}} \) where \( k_{\text{max}} \) is the maximum wave number in the input data, and (2) a two-hidden-layer MLP is sufficient to accurately calculate the heat flux \( \tilde{q}_i(x) \) for \( \tilde{T}_i(x) \) given \( n_{\text{neuron}} \geq n_{\text{th}}^\text{neuron} \). Unless noted otherwise, results in the rest of the letter were obtained with training data set with \( k_{\text{max}} = 8 \) and if applicable, a two-hidden-layer MLP with \( n_{\text{neuron}} = 64 \).

We also trained one classical deep learning model – the convolutional neural network (CNN). A CNN consists of an input and an output layer, as well as multiple hidden layers. We constructed our CNN with convolution layer, pooling layer and up-sampling layer. The output of convolution layer is the sliding dot product of its input and its filters where the filters detect specific type of local features of the input. The weights of a filter is shared in the whole domain, which is reasonable assuming spatial symmetry. Pooling layer down-samples the input by combining multiple neurons in the previous layer to one neuron in the next layer, while up-sampling layer repeats one neuron in the previous layer to multiple neurons in the next layer. As shown in Fig. 4 CNN appears not able to perfectly overfit the training data with \( n_{\text{sample}} = 10 \), but eventually has a similar performance as MLP when the training and testing MAE converged at \( n_{\text{sample}} = 100,000 \). This is likely due to CNN is “deeper” and has more complicated architecture compared to MLP, it requires more data to converge.

Due to the nature of closure problem, the choice of activation function of the output layer is essential to get correctly trained MLP and CNN. Because heat flux profile has both positive and negative values, ReLU is no longer suitable as the activation function of the last layer since it only outputs in \([0, \infty)\). Fig. 6 shows an example that the CNN fails to predict the output \( \tilde{q} \) when ReLU is chosen as the activation function of the output layer. An appropriate activation function of the last layer is linear function. When the activation function is switched from ReLU to linear, the CNN correctly predicts the output \( \tilde{q} \) from the input \( \tilde{T} \).

The last type of neural network we tested is a two-layer discrete Fourier transfer (DFT) neural network. DFT is different than MLP and CNN in the sense that it is intentionally designed to solve Fourier transform type problem. It has no hidden layer – only the input and output layers; the activation function is linear for all neurons; and it trains transformation matrix \( \mathcal{W} \). For instance, in DFT the relation between input \( \tilde{T}(x) \) and output \( \tilde{q}(x) \) is \( \tilde{q} = \mathcal{W} \tilde{T} \). Recall the definition of forward and inverse discrete Fourier transform
\[ X_k = \mathcal{F} x_n, \quad x_n = \mathcal{F}^{-1} X_k, \] (6)
where \( x = \{x_n\} \) and \( X = \{X_k\} \) are the signals in configuration and Fourier space, \( \mathcal{F} \) and \( \mathcal{F}^{-1} \) are forward and inverse Fourier transfer matrix. The idea behind DFT neural network is training weight matrix \( W \) to be close to the analytical form of forward transform matrix \( \mathcal{F} = \{F_{jk}\} = \{\omega^j k / \sqrt{N}\} \) or inverse transform matrix \( \mathcal{F}^{-1} = \{F_{jk}^{-1}\} \) where \( N \) is total sample in one period (i.e., \( n_x \) in our setup). Back to Hammett-Perkins closure Eq.\(^7\), it can be rewritten in DFT matrix form

\[
\bar{q}(x) = \mathcal{F}^{-1} \mathcal{K} \mathcal{F} \bar{T}(x),
\]

(7)

where \( \mathcal{K} = \{k_{jk}\} = \{-i \text{sign}(k) \delta_{jk}\} \) is the matrix form of \(-ik/|k|\). Therefore training a DFT neural network to learn Hammett-Perkins closure in configuration space is equivalent to learn weight matrix \( W = \mathcal{F}^{-1} \mathcal{K} \mathcal{F} \). One would hence expect that DFT shall have a good performance due to the existence of analytical expression (Eq.\(^7\)) for Hammett-Perkins closure.

The accuracy of three different types of neural network, indicated by the probability distribution function of the mean-absolute-error (MAE), is summarized in Fig.\(^7\). All three models are trained with the same training data set with \( n_{\text{train}} = 1,000,000 \), and tested with the same test data set with \( n_{\text{test}} = 10^4 \). Note the test data set is independent of training data set. In this test, both MLP and CNN use ReLU as the activation functions for the hidden layers. MLP and CNN turn out to have similar accuracy with CNN is slightly better than MLP (\( \epsilon_{\text{MLP}} = 9.72 \times 10^{-4} \geq \epsilon_{\text{CNN}} = 7.45 \times 10^{-4} \)), while DFT is 4 orders more accurate than MLP and CNN (\( \epsilon_{\text{DFT}} = 2.69 \times 10^{-8} \)). Authors remark the these models, by no means, have been tuned extensively to get the best performance. In fact, we are almost certain that even better accuracy could be achieved with advanced hyperparameter optimization. Nevertheless, \( 10^{-3} \) level error, perhaps, shall be considered acceptable from practical point of view.

A successful machine learning model should also be robust with respect to the noise presented in simulation and experimental data. We therefore studied the robustness of trained models by adding white noise \( A_{\text{noise}} T_{\text{noise}} \) to the test data set \( T \) and monitoring the model prediction \( \bar{q}_{\text{signal+noise}} \). Here \( A_{\text{noise}} \) is the maximum amplitude of white noise, and the error metric MAE is evaluated by comparing \( \bar{q}_{\text{signal+noise}} \) to the true \( \bar{q} \) with clean \( T \). Fig.\(^8\) shows how MAE varies with \( A_{\text{noise}} \) for three different models. The change of MAE of MLP and CNN has two phases: (1) when \( A_{\text{noise}} \) is smaller than the prediction error of neural network (roughly \( \epsilon \)), MAE does not increase with \( A_{\text{noise}} \) as \( \epsilon \) over-dominates \( A_{\text{noise}} \); (2) when \( A_{\text{noise}} \) is larger than \( \epsilon \), the MAE starts to increase linearly with the noise amplitude \( A_{\text{noise}} \). It also appears that MLP is more resilient to noise, most likely due to the trained network filters out portion of high-\( k \) noise as it only has 64 neurons in the hidden layers. In contrast, DFT seems to capture all levels of white noise because DFT essentially mimics the analytic expression of Hammett-Perkins closure and cannot “filter” any noise. It thus becomes less accurate than MLP or CNN when \( A_{\text{noise}} > 2 \times 10^{-3} \sim \epsilon_{\text{MLP,CNN}} \). Overall, MLP is the most robust model and DFT is the least robust model in the noise test.

Even though MLP and CNN have similar performance for the 1D closure problem, we suspect that CNN will perform much better than MLP for the future 2D and 3D closure problems as convolution layer is even more powerful to handle 2D and 3D data. In principle, the fully connected MLP can also be used to learn the features from 2D and 3D data. However it becomes less practical as the number of neurons and weights are increasing exceptionally with data dimension. For example, the number of weights for 3D data is \( n_x n_y n_z \) for each neuron in the a fully connected layer. On the other hand, the number of weights of a filter in the convolution layer is \( n_f x n_f y n_f z \), independent of the input size. Generally fully connected layer performs better than convolution layer for 1D problem, such as the 1D Hammett-Perkins closure we discussed here. As for DFT, despite it has a superior accuracy comparing to the MLP and CNN, it may lose its advantage when training advanced closures with complicated, if not impossible, analytical forms, e.g., data from the fully nonlinear simulation of Vlasov-Poisson equations. The performance of MLP, CNN and DFT for 2D/3D closure problems are beyond the scope of this study and is left to be explored later.

Authors emphasize that our work here is a proof-of-principle study to demonstrate that appropriate neural networks are capable to calculate Landau fluid closure. It is our hope that the success of training machine-learned Hammett-Perkins closure would gain some confidence on the machine/deep learning approach of closure problem and also guide our future attempts on training more sophisticated closure models. Implementation of such a neural network model in global 3D fluid turbulence codes, for example, BOUT++\(^17\) and GDB\(^18\) is another level of complexity (e.g., nonuniform grid, non-periodic boundary conditions, dependence on other plasma quantities). Further benchmark neural network based Landau fluid code with existing Landau-fluid\(^19\) and Gyro-fluid-mo-
FIG. 7. Probability distribution function (PDF) of absolute error $\epsilon$ for different neural network models when $n_{\text{sample}} = 10^6$. (a) for MLP, (b) for CNN, and (c) for DFT. The mean absolute errors for different NN are $\epsilon_{\text{MLP}} = 9.72 \times 10^{-4}$, $\epsilon_{\text{CNN}} = 7.45 \times 10^{-4}$, $\epsilon_{\text{DFT}} = 2.69 \times 10^{-8}$.

FIG. 8. The MAE with the amplitude of noise. red line for MLP, green line for CNN, and blue line for DFT.

4Z. Chang and J. Callen, Physics of Fluids B: Plasma Physics 4, 1167 (1992).
5M. Umansky, A. Dimits, I. Joseph, J. Omotani, and T. Rognlien, Journal of Nuclear Materials 463, 506 (2015).
6Z. Guo and X.-Z. Tang, Physical review letters 108, 165005 (2012).
7P. Hunana, G. Zank, M. Laurenza, A. Tenerani, G. Webb, M. Goldstein, M. Velli, and L. Adhikari, Physical review letters 121, 135101 (2018).
8L. Wang, B. Zhu, X.-q. Xu, and B. Li, AIP Advances 9, 015217 (2019).
9B. K. Spears, J. Brase, P.-T. Bremer, B. Chen, J. Field, J. Gaffney, M. Kruse, S. Langer, K. Lewis, R. Nora, et al., Physics of Plasmas 25, 080901 (2018).
10D. R. Ferreira and J. Contributors, arXiv preprint arXiv:1811.00333 (2018).
11M. Boyer, S. Kaye, and K. Erickson, Nuclear Fusion 59, 056008 (2019).
12B. Cannas, A. Fanni, P. Sonato, M. K. Zedda, J.-E. contributors, et al., Nuclear Fusion 47, 1559 (2007).
13C. Rea and R. S. Granetz, Fusion Science and Technology 74, 89 (2018).
14I. Kates-Harbeck, A. Svyatkovskiy, and W. Tang, Nature 568, 526 (2019).
15O. Meneghini, S. P. Smith, P. B. Snyder, G. M. Staebler, J. Candy, E. Belli, L. Luo, M. Kostuk, T. Luce, T. Luda, et al., Nuclear Fusion 57, 086034 (2017).
16I. Peterson, K. Hibbird, J. Field, S. Brandon, S. Langer, R. Nora, B. Spears, and P. Springer, Physics of Plasmas 24, 032702 (2017).
17B. Dudson, M. Umansky, X. Xu, P. Snyder, and H. Wilson, Journal of Physics: Conference Series 680, 012012 (2016).
18B. Zhu, M. Franciosco, and B. N. Rogers, Computer Physics Communications 232, 46 (2018).
19A. Dimits, I. Joseph, and M. Umansky, Physics of Plasmas 21, 055907 (2014).
20F. Chollet et al., “Keras,” https://keras.io (2015).
21M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” (2015), software available from tensorflow.org.
22S. S. Haykin et al., Neural networks and learning machines/Simon Haykin. (New York: Prentice Hall,, 2009).
23Y. LeCun, Y. Bengio, and G. Hinton, nature 521, 436 (2015).
24R. Velik, in 2008 International Conference on Computational Intelligence and Security, Vol. 1 (IEEE, 2008) pp. 120–123.
25C. Ma, X. Xu, P. Xi, and T. Xia, Physics of Plasmas 22, 010702 (2015).
26C. Ma, X. Xu, P. Xi, T. Xia, P. Snyder, and S. Kim, Physics of Plasmas 22, 055903 (2015).
27C. Ma and X. Xu, Nuclear Fusion 57, 016002 (2016).