# Encoder-decoder neural network for solving the nonlinear Fokker-Planck-Landau collision operator in XGC

M.A. Miller, R.M. Churchill, C.S. Chang, R. Hager

**Abstract**

An encoder-decoder neural network has been used to accelerate the numerical solving of a partial integro-differential equation, the Fokker-Planck-Landau collision operator. This is part of the governing equation in the massively parallel particle-in-cell code, XGC, which is used to study turbulence in fusion energy devices. The neural network emphasizes physics-inspired learning, where it is taught to respect physical conservation constraints of the collision operator by including them in the training loss, along with the L2 loss. The run time for the current Picard iterative solver of the operator is $O(n^2)$, where n is the number of plasma species. As the XGC1 code begins to attack problems including a larger number of species, the collision operator will become expensive computationally, making the neural network solver even more important, especially since the training only scales as $O(n)$. A wide enough range of collisionality has been considered in the training data to ensure the full domain of collision physics is captured. Eventual work will include expansion of the network to include multiple plasma species.

## 1 Introduction

As high performance computing (HPC) initiatives bring the advent of the exascale era, numerical simulations become capable of providing prediction with high levels of accuracy. In fusion energy, it is hard to model entire devices at a time given the complexity of multi-scale interactions. Calculating these interactions is computationally expensive, especially in the case of gyrokinetic codes, which solve for the motion of particles on the smallest of scales. XGC (X-point Gyrokinetic Code) is a massively parallel hybrid Lagrangian particle-in-cell gyrokinetic code focused on simulating the highly nonlinear edge region of fusion devices [1, 2]. An expensive part of the code is the calculation of the collision of the five-dimensional (5D) particles. XGC solves the nonlinear Fokker-Planck-Landau (FPL) collision operator, seen in equation 1, on a two-dimensional grid [3, 4]. At every time step, particles tracked in 3D configuration space and 2D velocity coordinates are histogrammed into velocity particle distribution functions $f$, on regular, 2D velocity grids at each configuration space vertex. A Picard iteration scheme is then used to solve the collision operator on the 2D velocity grid at each configuration space vertex, with a typical number of configuration space vertices in a simulation on the order of 5-10 million.

$$\frac{df_a}{dt} = \sum_b C_{ab}(f_a; f_b') = -\sum_b \frac{e_a^2 e_b^2 ln\Lambda_{ab}}{8\pi\epsilon_0^2 m_a} \nabla_v \cdot 3 \int \mathbf{U} \cdot \left( \frac{f_a}{m_b} \nabla_v' f_b' - \frac{f_b'}{m_a} \nabla_v f_a \right) d^3v' \tag{1}$$

Here, $a$ and $b$ denotes the species. $f_a$ and $f_b$ are the particle distribution functions of the species, $e_{a,b}$ is the charge, $m_{a,b}$ is the mass $ln\Lambda_{ab}$ is the Coulomb logarithm, and $\mathbf{U}$ is a tensor that is a function of the vector $\mathbf{v} - \mathbf{v}'$. The run time for the current Picard iteration solver of the collision operator is $O(n^2)$, where n is the number of plasma species. As the XGC1 code begins to attack problems including a larger number of species, the collision operator will become the dominant computation.

This paper proposes using machine learning algorithms to learn the nonlinear transformation of the collision operator, enabling the XGC code to be accelerated. Deep neural networks have proven successful for many computer vision tasks. Some attempts have already been made to extend the frameworks of these networks to solving partial differential equations in complex geometries. Since the FPL collision operator can be expressed as solving $\delta f_a = F(f_a, f_b, ...)$, where $\delta f_a$ is the same size as $f_a$, $f_b$, etc., here we use encoder-decoder deep neural networks used in the common computer vision task of semantic segmentation.

## 2 Computer Vision Technique for Collision Operator

Semantic segmentation is a computer vision task of labeling each pixel with the object class that pixel belongs to, e.g. marking all pixels that belong to a car in an image. For semantic segmentation, the input is an image of dimensions H x W x 3. As each pixel receives a class score, the 3 color channels become C class scores and the output takes the form H x W x C. Since the image has H x W pixels, the output array must contain H x W vectors of C elements. The decision of what a certain object is now made at a pixel level, H x W times. For each of these pixels, there is an associated cross-entropy loss function from which the total overall loss is computed, which is then used in backpropagation to update the overall gradients.

Semantic segmentation techniques are a good starting point for attacking the collision operator prediction problem, specifically because of the similarities in the transformation of spatial structure required. To perform the collision calculation, at every timestep, XGC performs particle to mesh interpolation, creating distribution functions on regular 2D velocity grids of size $N_{v_{perp}}$ x $N_{v_{par}}$ at each configuration grid vertex. This array can be thought of as a set of bins representing the number of particles at particular velocities, both parallel and perpendicular to the magnetic field lines.

XGC creates distribution functions for each particle type simulated. In this case, we consider electrons and a single ion species. The collision operator takes in these distribution functions and performs the collision calculation according to the FPL equations and outputs the change in the distribution function of species $a$, labeled $\delta f_{a,col}$. For example the single ion species $\delta f_{i,col}$ is:

$$\delta f_{i,col} = dt \cdot C_i(f_i, f_e) = dt \cdot [C_{ii}(f_i, f_i) + C_{ie}(f_i, f_e)] \tag{2}$$

This $\delta f_{a,col}$ has the same configuration space size as the original distribution function, much like semantic segmentation techniques, which preserve the size of an input image.

One main difference between the collision operator and typical semantic segmentation is $\delta f_{a,col}$ is continuous, such that it's a regression problem instead of a classification problem. We use a mean-squared error loss (or L2) as part of the loss function:

$$L_{L2} = \frac{1}{N} \sum_{i=1}^{N} (\delta f_{XGC} - \delta f_{ML})^2 \tag{3}$$

where N is the number of configuration space vertices. The L2 loss, however, does not take into consideration the conservation laws of the collision operator. For two species, $a$ and $b$, these are [5]:

$$\int \phi_a(\mathbf{v}) C_{ab}(f_a, f_b)\, d^3v = -\int \phi_b(\mathbf{v}) C_{ba}(f_b, f_a)\, d^3v \tag{4}$$

where $\phi(\mathbf{v}) = \{1, m\mathbf{v}, \frac{1}{2}mv^2\}$ represent the conservation of mass, momentum, and energy of the collision operator.

We enforce these physical conservation constraints in our neural network by including them as regularization terms in the loss:

$$L_{cons} = \sum_{i=1}^{3} \lambda_i \sum_a \langle \phi_a^i\, \delta f_{a,ML} \rangle \tag{5}$$

This equation incorporates equation 4 into the loss, seeking to minimize the change in these quantities due to the collision operator. Here, $\phi_a^i = \{1, m_a\mathbf{v}, \frac{m_a v^2}{2}\}$ for density, momentum, and energy conservation, and $\langle\rangle$ represents the numerical velocity integral. The quantity $\lambda_i$ represents a hyperparameter scaling factor that allows for modification of how heavily each conservation property is weighed in the overall loss function. This in turn affects how much the network corrects for each of the conservation errors compared to the L2 error.

## 3 Training Setup and Architecture

A dataset of the output $\delta f$ of the Picard iteration scheme was gathered using the existing collision kernel, comprising over 2.5 million samples of the 32x31 velocity-space grid distribution functions. Using the distri-

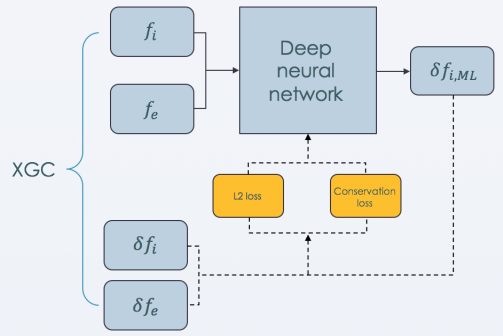


Figure 1: Schematic of the training processes used for collision prediction

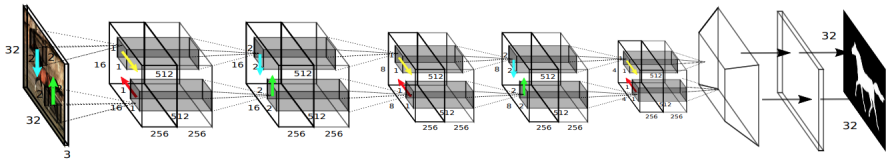


Figure 2: Schematic of ReSeg, a neural network architecture used for semantic segmentation

bution function for both the ions and the electrons allows for calculation of the actual $\delta f$ within the XGC collision kernel. These three sets of data, $f_i$, $f_e$, and $\delta f_i$ can then be used for training the neural network according to the schematic depicted in figure 1.

Initial training was begun on U-Net [6], one of the networks that performed best on semantic segmentation techniques earlier on. Once initial results were satisfactory, more sophisticated networks were investigated. In the end, the ReSeg network was chosen. It combines Convolutional (CNN) and Recurrent Neural Networks (RNN) to perform semantic segmentation [7] and with minor tweaks, to replicate the collision operator. ReSeg begins with a CNN, in this case VGG16. This CNN downsamples the input distribution function, producing a grid that is 256 planes deep. ReSeg then stacks two layers of ReNet, each of which is composed of four RNNs that sweep over the distribution function. Two layers, interspersed with ReLU activations upsample the data to a 32 x 31 x 100 grid. For the purposes of this problem, the last layer of the network includes a convolutional kernel of size 1 x 1 x 100 with stride 1, which outputs the desired 32 x 31 x 1 distribution function representing $\delta f$. For this problem, there are over 2.6 million parameters in the network.

A schematic of the network architecture can be seen in figure 2. Though it is eventually important to explore a range of devices, initial training was begun on the JET tokamak. Stochastic Gradient Descent (SGD) with momentum was used primarily as the optimizer, though RAdam, a variant of Adam that reduces the variance of adaptive learning rates [8], also showed success. Training was done for 100 epochs, with a validation set done twice an epoch. The training and validation results are shown in figure 3, with the L2 and conservation error shown separately.

# 4 Initial Training and Validation

Figure 3 shows the training loss over time for 100 epochs of ReSeg training. The learning rate was adjusted using a combination of warmup and manual learning rate decay every 10 epochs. Note that the "Training" and "Validation" plots include the $\lambda_i$ from equation 5, while the "Conservation" plot does not. It is clear that after around 70 epochs, the loss begins to plateau and learning stagnates. This implores a finer tuning of the hyperparameters, or perhaps a revision of the optimization procedure, specifically in the weights assigned to each quantity in the loss function ($\lambda_i$ in equation 5). Figure 4 illustrates the difference in learning of the structure of the $\delta f$ for two different $\psi_n$, a normalized radial coordinate, which lie in two different regimes of collisionality. The lower the $\psi_n$, meaning the closer the plasma is to the core, the easier it is for the network

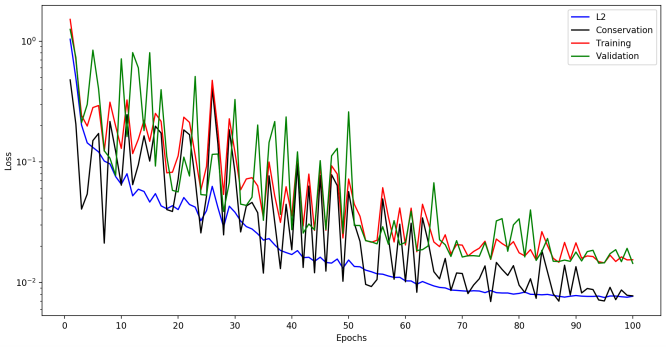


Figure 3: Initial training results on JET data for 100 epochs and for four toroidal planes

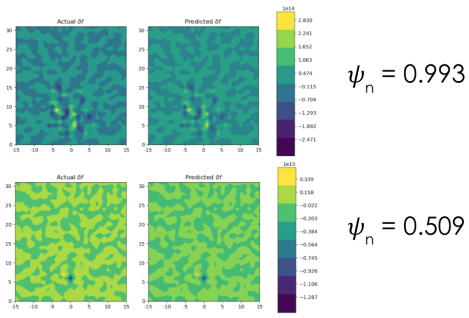


Figure 4: Comparison of $\delta f_{col}$ prediction in two different regimes of collisionality

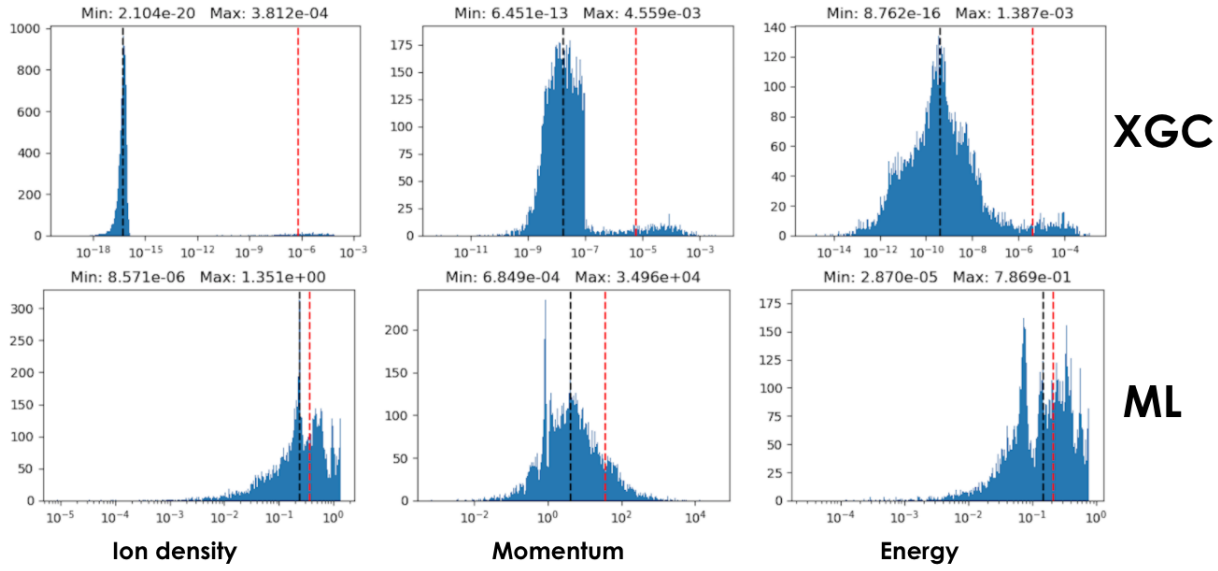


Figure 5: Comparison of conservation properties from XGC and from test set of neural network using machine learning (ML)

to learn.

In order to declare success, it is necessary to look at how well the neural network can solve equation 4, i.e. how well it conserves mass, momentum, and energy. Looking at equation 5, we require each of the terms in the sum to be a minimum. Work still must be done to reduce the quantities in table 5 to the threshold levels in XGC, but even these results show promise for speeding up the collision kernel.

## 5 Conclusions and Further Work

This work presents an encoder-decoder neural network that shows great potential in predicting the non-linear transformation of a particle distribution function arranged in a two-dimensional grid. Through a penalization method, the network enforces the conservation equations, while minimizing L2 error. Though the enforcement of these conservation equations has not yet produced results that meet the convergence criteria in XGC, progress towards this goal has been shown and a significant speedup would already be seen upon re-implementation of a currently trained model to XGC. The network has also proven successful in predicting this transformation for a wide range of training data that spans different regimes of collisionality. In short, this work has proven that machine learning can in fact serve as a useful aid for especially intricate

calculations in numerical work.

One immediate observation is that it would be useful to explore a wider variety of neural network architectures. It could be the case that an architecture that has perhaps not been as effective for image recognition problems could perform very well for the prediction of colliding plasma species. It will also be important in the future to investigate a broader collisionality regime. Training has been done on plasma both in the core and in the edge, which already spans a wide range of collisionality. This can be seen in figure 4, which compares the prediction for points at two different places in the tokamak. Regardless, it would be interesting to see the effects of different levels of collisionality in other devices. It may be the case that a different neural network is needed to be trained for other devices. It is also important to investigate the validity of the neural network between species. Does the same neural network perform equally well for ions and electrons? Especially as the network begins to train on data from other plasma species, would new training need to be done for different species? In the interest of reducing run-time, it is still necessary to make comparisons of run-time of using this neural network for inference in actual XGC production runs. All of these questions need to be addressed before the network can be expected to predict the $\delta f$ reliably enough for use in the XGC collision kernel.

## Acknowledgements

This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Science Undergraduate Laboratory Internship (SULI) program. Support for this work was provided through the Scientific Discovery through Advanced Computing (SciDAC) program funded by the U.S. Department of Energy Office of Advanced Scientific Computing Research and the Office of Fusion Energy Sciences. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725.

## References

[1] S. Ku, C.S. Chang, P.H. Diamond, Nucl. Fusion 49 (2009) 115021.
[2] S. Ku, R. Hager, C.S. Chang, J.M. Kwon, S.E. Parker, J. Comput. Phys. 315 (2016) 467–475.
[3] E.S. Yoon; C.S. Chang, A Fokker-Planck-Landau collision equation solver on two-dimensional velocity grid and its application to particle-in-cell simulation. Phys. Plasmas 2014, 21, 032503.
[4] R. Hager, E.S. Yoon, S. Ku, E.F. D'Azevedo, P.H. Worley, C.S. Chang, J. Comput. Phys. 315 (2016) 644–660.
[5] P. Helander, D.J. Sigmar, Collisional Transport in Magnetized Plasmas, Cambridge University Press, 2005.
[6] O. Ronneberger et al., U-Net: Convolutional Networks for Biomedical Image Segmentation, arXiv:1505.04597, 2015
[7] F. Visin et al., ReSeg: A Recurrent Neural Network-based Model for Semantic Segmentation, arXiv:1511.07053, 2016
[8] L. Liu et al., On the Variance of the Adaptive Learning Rate and Beyond, arXiv:1908.03265v1, 2019