Sequential Denoising of Quantum Monte Carlo Simulations with Deep Learning.

Pankaj Chouhan

December 2020

1 Introduction

Quantum Monte Carlo simulation refers to group of algorithm which uses Monte Carlo routine to estimate the multi-dimensional integrals that arise in the different formulations of the many-body problem. Many-body problem can describe almost all physical system and their property with good accuracy, where the particles in the system are not moving too "fast" (approx, to speed of light).

Ruining a single simulation for many-body problem involves evaluation of large number of integral via Monte-Carlo, making it a computational expensive process. Also being stochastic in nature, the simulation generates noise, therefore a good approximation of ground truth is achieved when the simulation is run multiple time. At each successive simulation the noise signal diminishes, making the simulation result close to ground truth. With Deep learning, we are trying to accelerate the simulation process by learning how the noise evolve at each time step.

2 Methods

2.1 Problem Formulation

Initial data is obtained by running the QMC simulation for first t timestep. For each time step, the generated data is 3-D, with dimension 101*101*101 along x,y and z axis, yielding t total files. Each files contain noise plus field, but in successive simulation from 0 to t the field information remain the same but the noise decrease at each successive time-step. With the NN model, we will try to learn this temporal evolution of noise along with the spatial noise characteristics. For pre-processing we sliced the 3-D data along z-axis assuming that field and noise characteristics is independent along z-direction. This gives us 101 time-series sequence of depth t, where the i^{th} sequence is made by stacking all the i^{th} slice across all time-steps. These 101*t sequences act as the training data to NN model.

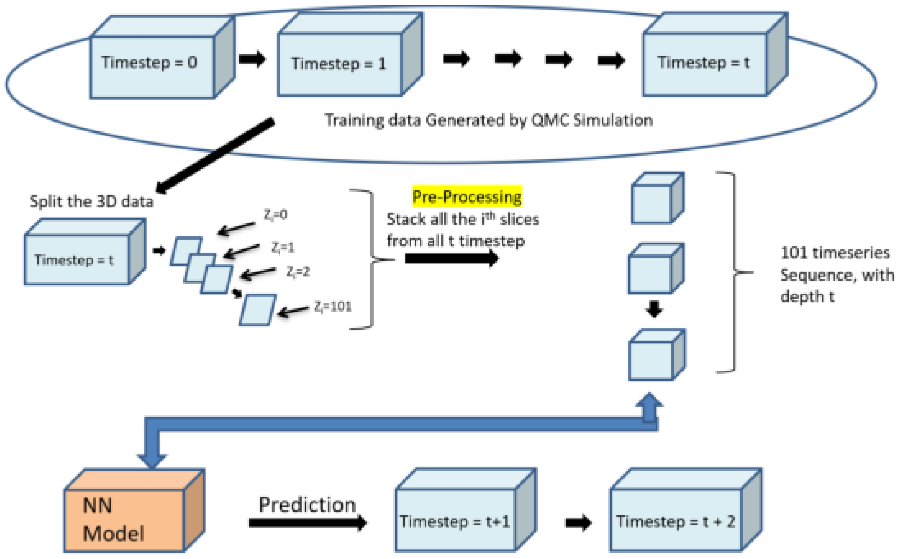

Figure 1: Illustration of the pipeline for data generation, pre-processing, training and prediction

Now since we have data that evolves in time, we can consider the input data as 101 different video sequence. Formulating our problem as video de-noising gave us an advantage because of the extensive research done by deep-learning community on video de-noising and image de-noising. However most of the work in literature is reported where the nature of noise is given, like Gaussian noise[1][2], or either ground truth is available, or some form of prior information is available. For our case we don't know the nature of noise, making it a blind de-noising problem[3][4] which is still a tough challenge in deep learning community. Also the absence of ground truth makes it more difficult to learn the mapping from noise+field to field.

To de-noise a given frame i, the most popular methods adopted in literature is to concatenate the frame i with few frame ahead in time, i+1,i+2 and so on, and few frame backward in time i-1,i-2, and thereafter sent this concatenate data to NN model. Along with this data either the the change in noise between these concatenate frames is provided if ground truth is available or is estimated by using flow estimation technique like optical flow.

To de-noise QMC data we looked into two approaches. The first one uses Convolution Neural Network model to de-noise the spatial component and than clean the data further by performing Noise-to-Noise one shoot learning to de-noise the temporal component. This method is based on the work of Ehret et.al [5]. The other approach uses Convolution Neural Network to learn spatial component along with Long Short Term Memory cells to learn the temporal component.

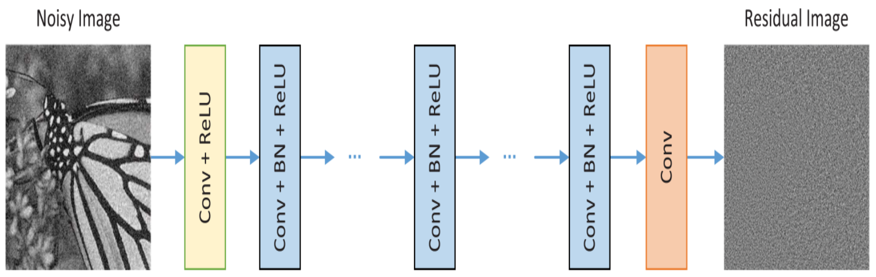

Figure 2: Network architecture of DnCNN, taken from Zhang.et al paper[6].

2.2 Model Structure

2.2.1 CNN+Noise2Noise Learning

We are Using DnCNN as our CNN architecture which we adopted from the work of Zhang et.al[6]. The input to DnCNN is noisy observation $y = x + v$, and with use of residual learning DnCNN learn the mapping to spatial noise, $R(y) = v$, and gives us back the clean image $x = y - R(y)$. A DnCNN architecture with depth D has three type of layers. (i) Conv+ReLU: for the first layer, 64 filters of size $3 \times 3 \times$ c are used to generate 64 feature maps, and rectified linear units (ReLU, max(0,·)) are then utilized for nonlinearity. Here c represents the number of image channels, i.e., c = 1 for gray image and c = 3 for color image. (ii) Conv+BN+ReLU: for layers 2 to (D-1), 64 filters of size 3×3×64 are used, and batch normalization is added between convolution and ReLU. (iii) Conv: for the last layer, c filters of size $3 \times 3 \times 64$ are used to reconstruct the output[6]. Fig.2 illustrate the DnCNN architecture.

Cleaned data from DnCNN is than sent to another routine that de-noises the temporal component of noise. The routine consists of creating a optical flow based occlusion mask between frame t and t+1, and based on the mask the weights of DnCNN network is fine tuned N times to learn the right mapping between noisy and clean data. This process is repeated for all the pairs of frame in data. To fine tune DnCNN network a learning rate $5e^{-5}$ and weight decay of $1e^{-5}$ used for Adam optimizer.

2.2.2 CNN+LSTM

For this model a different approach is used where we de-noise spatial and temporal noise component at once. CNN is used for modelling spatial input like images and have shown their importance in field of computer vision but because of it's architecture it performs poorly on time-series based prediction task. LSTM on the other hand performs very well on sequence related task, but won't work with spatial input. To use best of both world, we used CNN + LSTM cells. The CNN Long Short-Term Memory Network (CNN-LSTM) is an LSTM architecture specifically designed for sequence prediction problems with spatial inputs, like

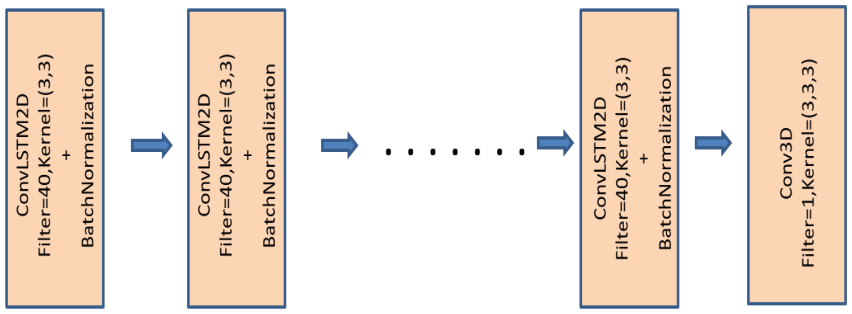

Figure 3: Network architecture of CNN+LSTM.

images or videos.

In this model we have 8 CNNLSTM2D layer with (3,3) kernel and 40 filters followed by batch normalization. The last layer is 3D convolution layer with kernel size (3,3,3). AdeDelta optimization routine with learning rate of 0.01 is used to train the model for 200 epochs.

3 Results

Being a computational expensive process we ran the QMC simulation for 20 time-steps, and uses the last time-step as our ground truth. Since the dimension of data is 101*101*101, a single slice in Z direction is of 101*101 pixel values. Because of the limited data-availability and to reduce the width across which Neural Network has to learn we used patch of 34*34 pixel instead of whole slice to feed into both models of Neural Network. To compare the Noise signal between two images we used PSNR values defined as:

$$PSNR = 10 \log \frac{MAX_I^2}{MSE}$$

where MAX_I is the maximum possible pixel value, and MSE is the mean error defined as:

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2$$

here I is the noise-free m×n monochrome image K is its noisy approximation.

3.1 CNN+Noise2Noise

To train CNN+Noise2Noise we use first 5 time-steps QMC simulation as our training data, and treated the 20th time-step as our ground truth. We used

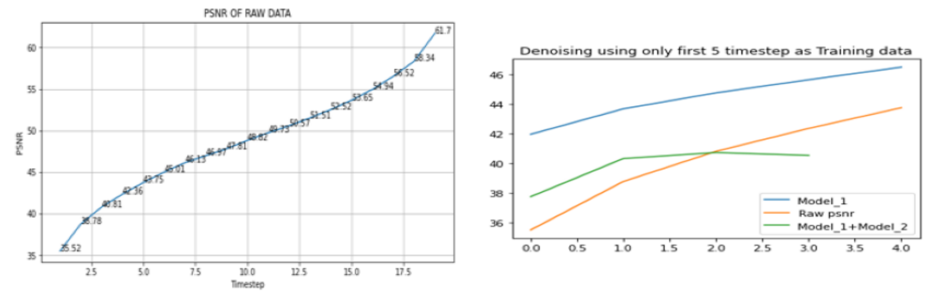

Figure 4: Fig.4a illustrate PSNR values for first 19 time-step w.r.t to 20th time-step. Fig.4b show the result of spatial de-noising by first model(CNN part) and Model1 + Model2(CNN+Noise2Noise). We can see that PSNR value of Model1 (blue line) is greater than PSNR value of raw data orange curve (the first five values of fig 4.a), However when we include noise2noise learning (Green Curve) the PSNR value goes below the raw PSNR showing that CNN+LSTM isn't capable of learning spatial and temporal noise of QMC.

only first five time-steps because our main task is to run the QMC simulation only for first few step and then use NN to predict the future time-steps. Fig.4a represent the PSNR of the noisy input w.r.t ground truth. Fig.4b compare the PSNR value between the result obtained from running CNN model for spatial de-noising (blue curve), results after running the Noise2Noise learning on spatial de-noised data(Green Curve), and raw PSNR data(Orange Curve). It's clear from that figure that CNN network is de-noising the spatial component but Noise2Noise learning is outputting the result that has low PSNR as compared to raw data and is failing the temporal de-noising task. Since our data represent the electron density it has high sparsity, and the relatively difference between the pixel value for noise and field is relatively small making it a difficult task to model. Also the change in noise level between two consecutive frame is so subtle that optical flow isn't be able to track the correct gradient of change in pixel value between successive frames. We believe that bad occlusion mask created by optical flow is the major factor in driving the PSNR value of temporally de-noised data below raw PSNR value.

3.2 CNN+LSTM

For CNN+LSTM we used first 10 time-step as our training data because LSTM architecture needs a good amount of sequential data to capture the data propagation behaviour.

From Fig.5 it's evident that model is able to capture the trend no further than one-step ahead. This model shows some promises but need optimization in terms of number of layers, filter etc. The Model for which the results are reported is still trivial with only 8 CNNLSTM layers and has low learning capacity as compared to requirement present by the learning task. The future

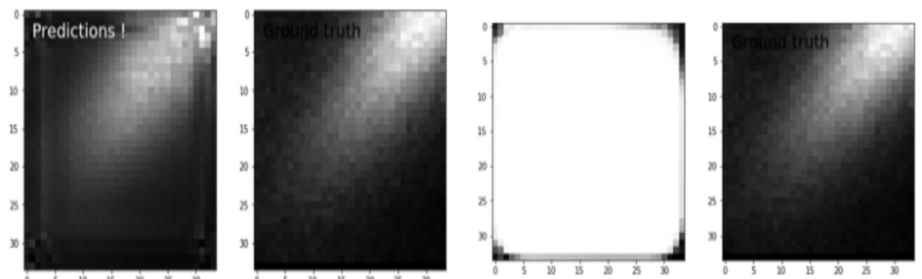

Figure 5: Fig.5a show the one-step ahead predicted result (11th time-step) and Fig.5b show the actual corresponding slice. Fig.5c show the two-step ahead predicted result(12th time-step) and Fig.5d show the actual corresponding slice for 12th time-step. Fig.5a has the characteristics and shape of the actual data. For fig.5c the one-step ahead predicted result is used data while predicting two-step ahead data. From fig.5c you can see the poor quality of prediction.

work for this model is to use a self-optimization routine on NN to find the best model in term of layers, filter, learning rate etc. for this task.

4 Conclusion

Because of high sparsity of training data and only a subtle difference between noise and field this learning task need a highly optimize model to separate noise from field. We tried two models motivated by the different approaches adapted in literature. The first approach successfully de-noise the spatial component of noises using CNN but fails with temporal noise because of the poor quality of occlusion mask created. The other approach use the power of LSTM cell to predicted one-step and two step ahead prediction. This model shows promised but need auto-tuning to find the best model for the learning task. For future work we will revisit the problem statement and use the error file generated in QMC simulation as part of our training data, also instead of using slices we can explore about using the 3D data (3D CNN).

References

[1] Matias Tassano, Julie Delon, Thomas Veit. DVDNET: A FAST NETWORK FOR DEEP VIDEO DENOISING. arXiv:1906.11890v1

[2] Matias Tassano, Julie Delon, Thomas Veit. DVDNET: AFastDVDnet: Towards Real-Time Deep Video Denoising Without Flow Estimation. arXiv:1907.01361

[3] Joshua Batson, Loic Royer. Noise2Self: Blind Denoising by Self-Supervision. arXiv:1901.11365v2

[4] Shi Guo1, Zifei Yan, Kai Zhang, Wangmeng Zuo1, Lei Zhang. Toward Convolutional Blind Denoising of Real Photographs.

[5] Thibaud Ehret, Axel Davy, Gabriele Facciolo, Jean-Michel Morla. Model-blind Video Denoising Via Frame-to-frame Training. arXiv:1811.12766

[6] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 26, NO. 7.