Recurrent Neural Network-based Model for Accelerated Trajectory Analysis in AIMD Simulations

A Preprint

Mohammad Javad Eslamibidgoli
School of Computing Science
Simon Fraser University
8888, University Drive
Burnaby, BC, Canada, V5A 1S6
meslamib@sfu.ca

Mehrdad Mokhtari
Department of Computational Physics
Simon Fraser University
8888, University Drive
Burnaby, BC, Canada, V5A 1S6
mokhtari@sfu.ca

Michael H. Eikerling
IEK-13, Institute of Energy
and Climate Research,
Forschungszentrum Jülich GmbH,
52425 Jülich, Germany
m.eikerling@fz-juelich.de

October 3, 2019

ABSTRACT

The presented work demonstrates the training of recurrent neural networks (RNNs) from distributions of atom coordinates in solid state structures that were obtained using *ab initio* molecular dynamics (AIMD) simulations. AIMD simulations on solid state structures are treated as a multi-variate time-series problem. By referring interactions between atoms over the simulation time to temporary correlations among them, RNNs find patterns in the multi-variate time-dependent data, which enable forecasting trajectory paths and potential energy profiles. Two types of RNNs, namely gated recurrent unit and long short-term memory networks, are considered. The model is described and compared against a baseline AIMD simulation on an $Ir_{0.75}W_{0.25}O_2(100)$ slab. Findings demonstrate that both networks can potentially be harnessed for accelerated statistical sampling in computational materials research.

Keywords RNN · LSTM · GRU · Time-series · AIMD simulation

1 Introduction

Although the primary methods were developed in the 1950s to 80s, breakthroughs in machine learning have occurred only recently, with many interesting applications such as in computer vision [1], speech and language technology [2], self-driving cars [3], recommender systems [4], financial predictions [5], and many more [6]. This is mainly owed to recent advances in neural network architectures and algorithms [7], escalating growth of data for model training [8], powerful parallel computer processing, enhanced frameworks for implementation [9] and, of course, larger involvement from the scientific community as well as industry in the field.

Machine learning is well-established in areas like bioinformatics as a computational method to analyze and interpret biological data [10, 11]. Deep neural network architectures such as multi-layered networks, convolutional neural networks (CNNs), recurrent neural networks (RNNs) and memory networks are the most used architectures in this area [12]. Machine learning has also shown to be promising in theoretical chemistry [13] mainly to speed up the

discovery of novel compounds or materials [14, 15, 16, 17], as well in terms of predicting the potential energy surface of chemical structures [18, 19].

Extensive research is focused on developing inter-atomic potentials [20] or performing AIMD simulations [21, 22]. Brockherde *et al.* employed the kernel ridge regression training algorithm based on an external Gaussian potential to generate the machine learned potential-density map of a given molecule [18]; their proposed approach starts with the construction of descriptors, which encode the structure of each molecule from a training data-set. The kernel matrix is then developed from a kernel function to represent the correlation of descriptors to each other [18]. The data-set in their work is composed of 2000 DFT optimized structures.

Gomez-Bombarelli *et al.* [15] have recently used variational autoencoders to transform the discrete representation of molecular structures into a continuous latent space, from which a decoder neural network converts these continuous representations back into a discrete molecular representation. This approach allows designing new molecules and enables efficient exploration and optimization of the chemical spaces of compounds [15]. Chmiela *et al.* advanced a symmetrized gradient-domain machine learning model, in which one can create a molecular force field of intermediate-sized organic compounds with the same accuracy as high-level *ab initio* calculations only using limited samples (of size 1000) of AIMD calculated trajectories [23]. In another work, Xie and Grossman proposed a crystal graph convolutional neural network framework to provide a universal representation of crystal materials and an accurate prediction of DFT calculated properties of various crystal structures and compositions [24].

This paper pioneers the use of RNN-based models in predicting trajectory paths and potential energy profiles of solid-state structures in AIMD simulations. The main purpose of AIMD simulation is to generate as many as possible statistically independent configurations of the system under study at affordable computational costs. This requires extensive simulation runs and sufficiently long intervals between configurations which are used in thermodynamic analyses to ensure that they are statistically uncorrelated [25]. We show that RNN-based algorithms can generate trajectory paths as accurately as AIMD simulations, but in a much shorter time. The presented approach treats interactions between atoms as temporary correlations; thus, the problem of predicting the trajectory of atoms reduces to a multi-variate time-series forecasting problem.

Time-series forecasting has become a groundbreaking research topic over the past couple of years; a significant example is the seq2seq model proposed by Google to make multi-step sequence predictions in order to solve the machine translation problem [26]. An important class of RNN architectures designed for sequence-to-sequence problems is called the encoder-decoder LSTM, which comprises an encoder that receives input signals and returns a fixed-length internal representation as a vector, and a decoder that interprets this vector and uses it to predict the output signal [27]. This approach is powerful as not only we can have an input of univariate time-series data, but also, we can encode the multi-variate features of a time-series data-set. Various other types of RNNs have been employed for such predictions such as long short-term memory (LSTM) [28], bidirectional long short-term memory (BLSTM) and mixture density network (MDN) approaches [29, 30], and gated recurrent units (GRU) [31].

Zhao et al. [30] proposed a BLSTM-MDN approach for three-point shot prediction in basketball games. They have utilized a Python library called Hyperopt [32] for hyperparameter self-tuning during model training. Faster convergence rate and more accurate prediction are the superior features of their approach in comparison to CNNs and MDNs. In terms of predicting the trajectories, their proposed model can produce new trajectory samples beyond predicting those stemming from the real data. Another example reported is predicting amazon spot price using a three-layer LSTM-based network, composed of two LSTM layers and one dense neural network layer, by Baughman et al. [33]. Their model has revealed a reduction in mean square error (MSE) of 60% up to 95% for different training-sets relative to the well-known Autoregressive Integrated Moving Average (ARIMA) model [34].

Sagheer and Kotb [35] employed deep LSTM (DLSTM) recurrent networks for univariate time series forecasting of petroleum production. To find the optimum architecture of DLSTM and optimal selection of hyperparameters, a genetic algorithm was used in their work. For evaluation purposes, the ARIMA model [34], the Vanilla RNN model [36], the deep GRU model [37], the Nonlinear Extension for linear Arps decline model [38], and the Higher-Order Neural Network model [39] were employed. Using root mean square error and root mean square percentage error [40] measures, the proposed deep LSTM approach, outperforms the other standard models.

In the following, we first describe the GRU and LSTM methodologies followed by the computational details of our AIMD simulations for data-set preparation. Next, we propose our two-layer network model for trajectory predictions of atoms in solid-state structures which consists of a GRU or LSTM layer and one dense layer using sigmoid activation function to condense output from the previous layer. We evaluate this network architecture by comparing the accuracy of trajectory predictions against AIMD simulations.
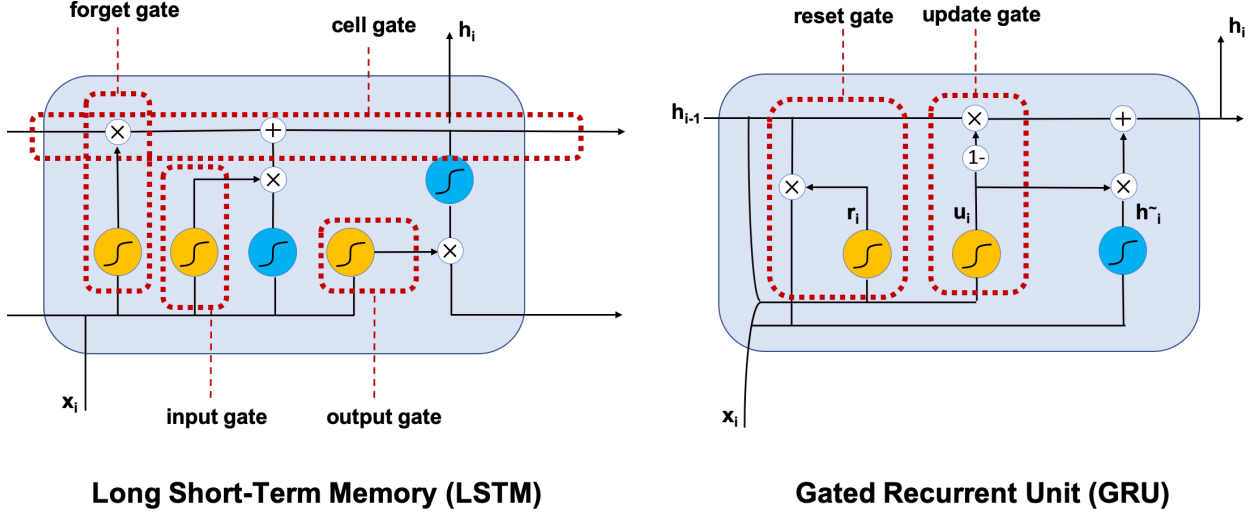
Figure 1: The LSTM and GRU blocks. Yellow and blue circles represent sigmoid and tanh functions, respectively. White circles show point-wise operations.

2 Methodology and Data Preparation

2.1 LSTM/GRU networks

RNNs receive as input a sequence of signals with no predetermined size limit and utilize their internal states (so-called as memories) for processing the input. The LSTM model, as a special kind of RNN, was proposed by Sepp Hochreiter et al. in 1997 [41], to solve the vanishing gradient issue associated with RNNs [6, 42]. The main problem is, by proceeding to the lower layers thorough back propagation, the gradient or partial derivative of the cost function with respect to the layer's weight, W, becomes vanishingly small, preventing the weights from updating their values [43]. To address this problem, LSTMs employ a gating mechanism, composed of three gates, which give the model permission to pass or forget the data, as shown in Figure 1 [41]. The proposed gates establish a new relationship between data and forget previous relationships over a particular time span. Thus, the gating mechanism enables the previous input signals to affect the current signal at a given time while remaining unaffected by those signals far apart [41]. Gated Recurrent Unit (GRU) is a form of LSTMs introduced in 2014 [37]. In what follows, we only present the architecture of a GRU which is similar to that of a LSTM model.

In a GRU model, as shown in Figure 1, a reset gate, r, and an update gate, u, are the main gates. Concatenating the new input with the preceding memory is enabled by the reset gate. The update gate, on the other hand, determines the portion of the preceding memory to be kept. Denote the hidden state of the GRU at time step i by h'_i; in a GRU structure, the activation h'_i at time i is a linear interpolation between the previous activation h'_{i-1} and the candidate activation (h_i). An update gate u_i governs this linear interpolation, and the candidate activation (h_i) depends on a reset gate r_i. The exact formulation is as follows

$$h'_i = (1 - u_i)h'_{i-1} + u_i\widetilde{h}_i, \tag{1}$$

$$u_i = \sigma(W_u x_i + U_u h'_{i-1}), \tag{2}$$

$$\widetilde{h}_i = tanh(W_u)x^i + U(r_i \odot h'_{i-1}), \tag{3}$$

$$r_i = \sigma(W_r x_i + U_r h'_{i-1}) \tag{4}$$

Here, vector x_i represents the input. The update gate u_i in Equation 2 decides how much of each dimension of the hidden state is retained and how much should be updated with the transformed input x_i from the current time step.

2.2 Data preparation

Vienna *ab initio* Simulation Package (VASP 5.2) was used to perform the DFT calculations [44]. Exchange-correlation effects were incorporated within the generalized gradient approximation (GGA), using the functional by Perdew,
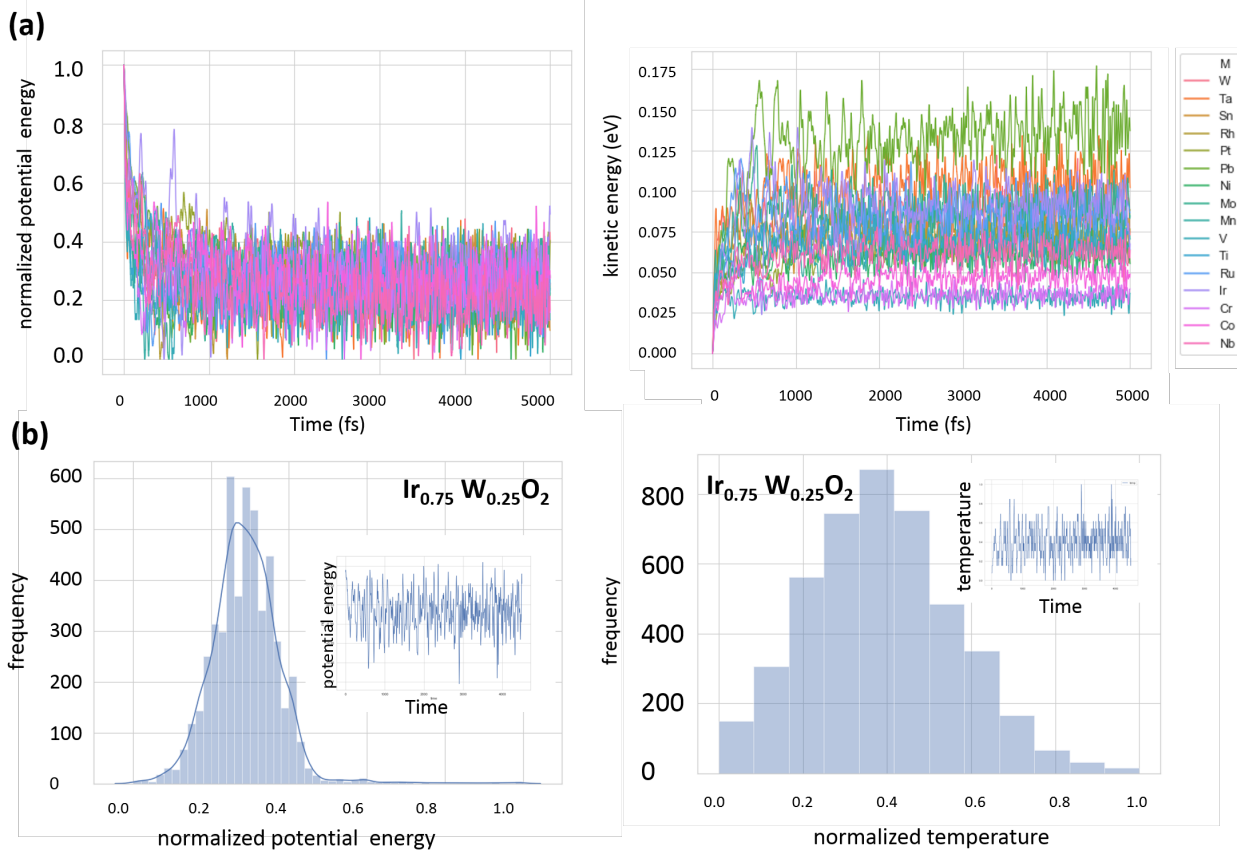
(a)

(b)

Figure 2: Data prepared based on our AIMD simulations over 5000 fs of the rutile $Ir_{0.75}M_{0.25}O_2$ (100) surfaces, where M is Ir, Ni, Ti, V, Cr, Co, Nb, Ru, Rh, Sn, Ta, Pt, Pb, Mn, W, Mo. (a) potential energy and kinetic energy profiles for all systems over AIMD simulation (b) distribution of potential energy and temperature for $Ir_{0.75}W_{0.25}O_2$ during the AIMD simulation.

Burke, and Ernzerhof (PBE) [45]. All calculations were performed based on the Projector Augmented Wave (PAW) method [46]. According to the PAW method, core electrons were kept frozen and replaced by pseudopotentials (Ir, M, O, H) and valence electrons (Ir: 6s2 5d7; O: 2s2 2p4; H: 1s1) are expanded in a plane wave basis set up to an energy cut-off of 400 eV. Geometry optimization studies were terminated when all forces on ions were less than 0.05 eV^{-1}.The Brillouin zone was sampled with the Monkhorst Pack scheme [47] using $6 \times 6 \times 8$ k-points.

For the general raw data, Co, Cr, Ir, Mn, Mo, Nb, Ni, Pb, Pt, Rh, Ru, Sn, Ta, Ti, V and W were substituted for metal M in the $Ir_xM_{1-x}O_2$ bulk and the unit-cell was optimized, to account for the strain effects, from which the (100) facet was generated. IrO_2 considered in this study has a tetragonal structure with lattice parameters $a = b = 4.51$ Å, $c = 3.18$ Å. AIMD simulations were performed in the microcanonical ensemble using the Verlet algorithm with a time step of 1 fs. The simulations were initialized using configurations obtained from the DFT-based energy minimization. The number of k-points in AIMD simulations was decreased to $2 \times 2 \times 1$.

Trajectory raw data for all atoms were extracted from the VASP output using Visual Molecular Dynamics [48] and reshaped for neural network training. No missing data-points were present as expected from AIMD simulations. However, 4 or 5 atoms at the boundaries out of the total 36 atoms in the unit-cell were found to be moving from one side of the cell to the other during the simulation, which was due to the boundary conditions applied in the solid state simulation. We identified these atoms and removed their data from the data-set.

We used Python 3.5, TensorFlow 1.12.0 [49] and Keras 2.1.6-tf [50], pandas 0.24.1 [51], sklearn [52], matplotlib [53] and seaborn for graphics.

4

Table 1: List of hyperparameters

Hyperparameters	Min	Max	Default Values)
Learning rate	10^{-4}	5×10^{-3}	10^{-3}
Batch size	128	512	256
Number of units for GRU and LSTM	200	300	250
Sequence length	50	200	150
Epochs	20	20	20
Steps per epoch	100	200	100
L2 regularization parameter	0.0001	0.001	0.001
Train-test splitr	80%-20%	90%-10%	80%-20%

3 Results and Discussion

We first generated trajectory data from AIMD simulations over 5000 fs on rutile $Ir_{0.75}M_{0.25}O_2$ (100) surfaces as baseline model system. Figure 2 (a) shows the normalized potential energy and the kinetic energy of 16 different simulated systems for various metals. After about 500 fs, the slab system was equilibrated. Figure 2 (b) shows the distribution of potential energy and temperature for $Ir_{0.75}W_{0.25}O_2$ (100) over the last 4500 fs of simulation time, which follows a Gaussian type distribution.

Our model consists of a GRU or LSTM layer and a subsequent dense layer to combine output from the previous layer. As input, the RNN-model receives trajectory data for all atoms in the unit-cell. Moreover, we include data for the potential energy profile as well as the temperature fluctuations over the simulation run. As output, in addition to the x, y, z coordinates of the considered atom, we also aim to predict the fluctuations in the potential energy profile. The predictions are then compared with the AIMD calculated values.

One test atom for prediction was chosen from the data-set. We resampled the target data for this atom by shifting it negatively by 200 fs (around 5% of the training-set), meaning that we shifted the target data points to a configuration 200 fs in the past to let the model predict the future trajectory of the test atom. We used 80% of the AIMD time steps for the training-set and the rest for the test-set. The range of feature values was scaled in the data preprocessing step. To train the RNN, we created batches of shorter sequences of input data selected randomly from the training-set. To prevent overfitting, L2 regularization was performed. RNN was created in Keras [50] and GRU or LSTM were added to the first layer. This layer returns a sequence of 250 values as input for the dense layer which in turn outputs 4 vectors. The sigmoid activation function was used to output the values between 0 and 1 which is justified as we expect the output values to be in the same range as the training-set. RMSprop optimizer was used to compile the model. The values for the hyperparameters used are shown in Table 1. We used mean squared error (MSE) as loss function which required to be minimized during the model training:

$$MSE = \frac{\sum_{i=1}^{n}(y_i - y_i^p)^2}{n} \qquad (5)$$

Figure 3 shows the predicted output sequences, namely x, y, z coordinates and the energy profile of our GRU- and LSTM-based models compared with the ground truth from AIMD simulation. As shown for the training-set, both models predicted the fluctuations of the coordinates with high accuracy. This was expected as the train-data was seen by the models during the training phase. The prediction, however, is not accurate for the first few time steps because the models have not received enough input. Likewise, for the potential energy the fluctuations were also predicted well on the training data. In addition, both models' predictions were reasonably accurate on the test-set, although some peaks were not matched with the AIMD trajectories and there seems to be a small deviation.

Figure 4 compares the generated partial autocorrelation function (PACF) of the potential energy calculated with AIMD simulations with those predicted by GRU and LSTM models. In time-series, the partial autocorrelation indicates the correlation between an instance with those at prior time steps having the relationships of intervening instances removed. Mathematically, partial autocorrelation function values are the coefficients of an autoregression on lagged values of the time-series as given by,

$$Y_t = a_0 + a_1 Y_{t-1} + ... + a_N Y_{t-N} \qquad (6)$$

As shown in Figure 4, the PACF from AIMD simulation and the models indicate that after around 20 fs the energy values become uncorrelated. This particularly shows the ability of GRU and LSTM models to predict uncorrelated configurations which can be used directly for the calculations of thermodynamic properties.

In Figure 5, we calculated the point-by-point absolute percentage error $((|y_i - y_i^p|)/y_i \times 100)$ of GRU and LSTM models against the AIMD simulation. For both the training and test sets, the percentage error of the GRU model for x,
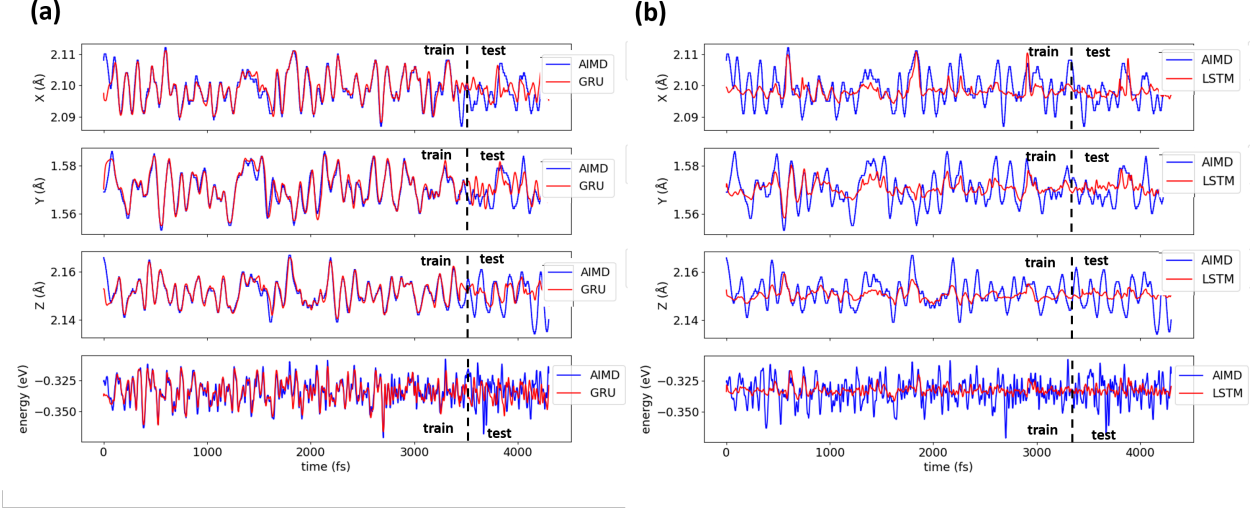
5

Figure 3: Predictions of the trajectory of the Ir atom in $Ir_{0.75}W_{0.25}O_2$ (100) slab and total potential energy by (a) GRU and (b) LSTM compared to AIMD truth.
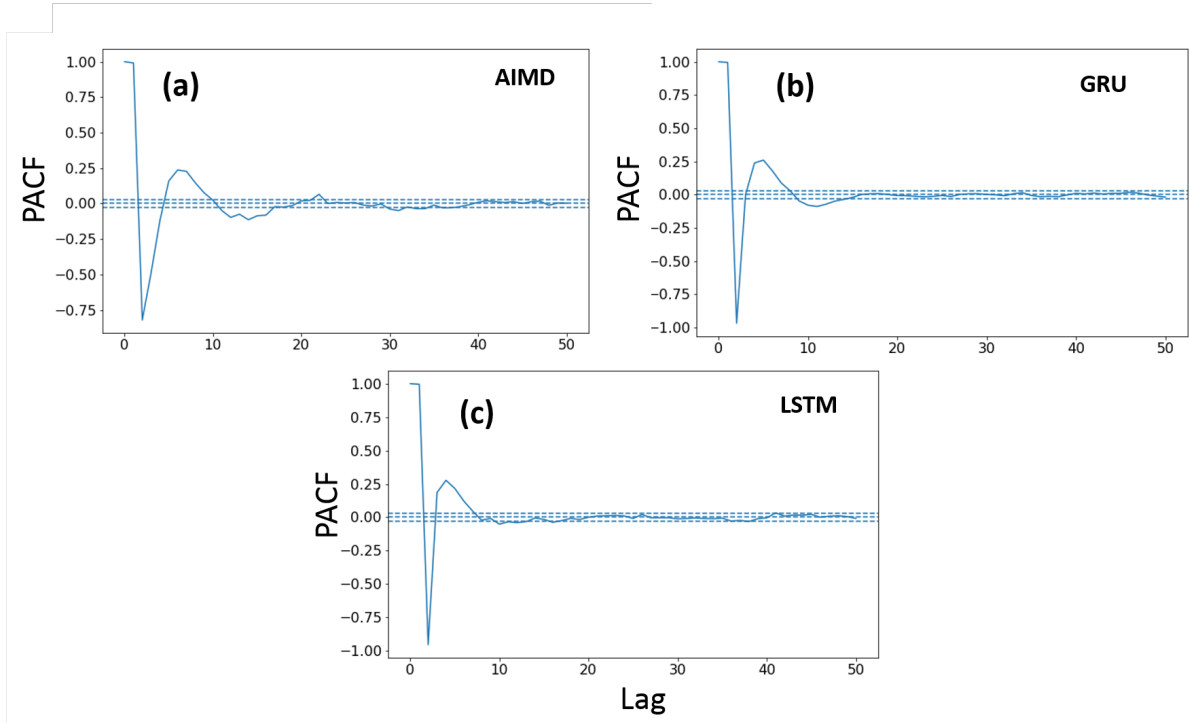

Figure 4: Partial autocorrelation function for the potential energy profile obtained from AIMD, GRU and LSTM.

y, z coordinates were found to be less than 0.9%, with a mean error of around 0.1%. The mean percentage error for potential energy predicted by GRU was 1.3%, standard deviation was 1.5% and the maximum error was found as 9.4%. Regarding the LSTM model, the maximum percentage error for the coordinates was found as around 1% with the mean error and standard deviation values of around 0.1-0.2%. The calculated mean error for the potential energy predicted by LSTM was 2.1% with standard deviation of 1.7% and maximum value of 9.3%. Even though the calculated error shows slightly more accurate predictions for the GRU model, LSTM seems to be more robust in terms of dealing with the overfitting issue as the error values for both train and test sets lie in pretty much the same range.
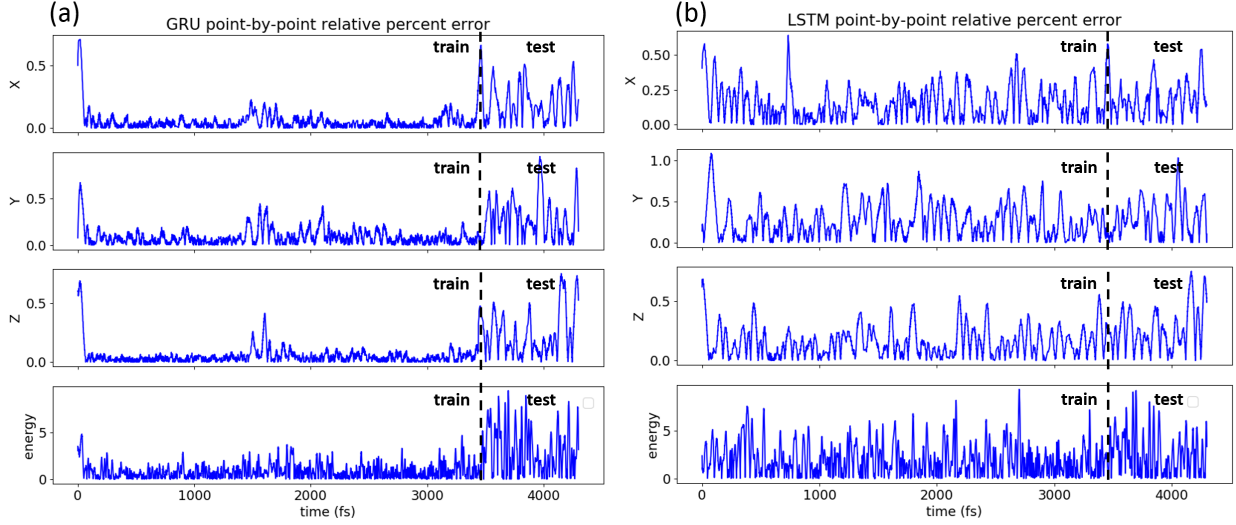
Figure 5: Absolute percentage error of the two models

Figure 6 shows the spatial distribution of the atoms in the unit-cell over the sampling run, which approximately follows a normal shape. Moreover, the temperature fluctuation and potential energy profile used as input signals were also found to be of Gaussian type. Therefore, it is not surprising to observe highly accurate predictions by the GRU and LSTM models on this data-set, as such algorithms tend to learn more easily from this kind of distribution.
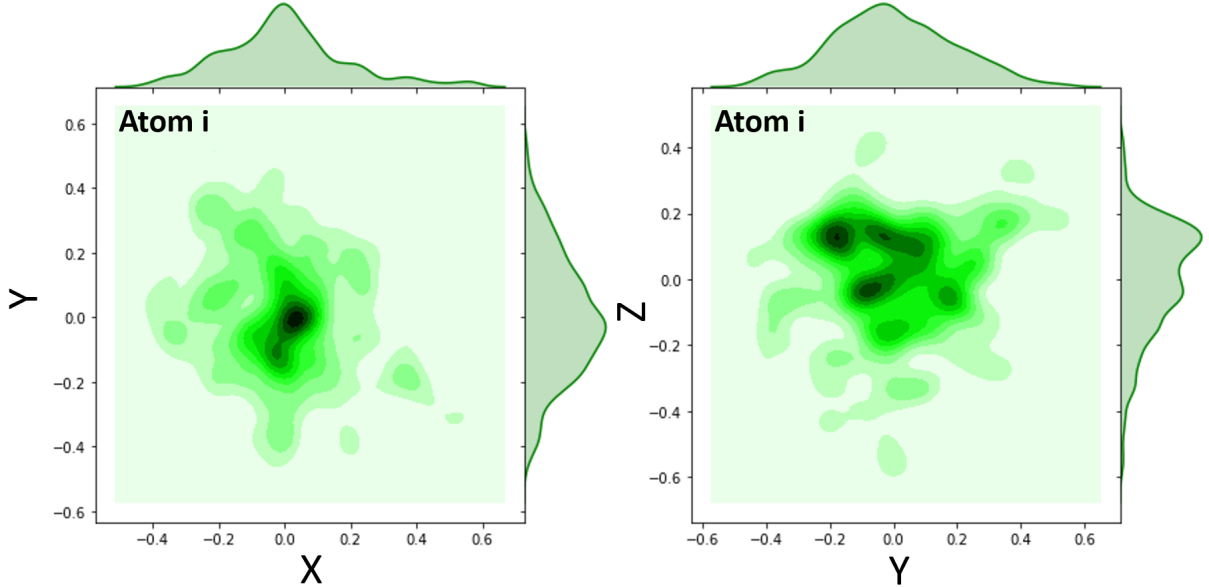
Figure 6: Density plot of coordinates of (a) all atoms (b) randomly chosen atom i, over the AIMD simulation run.

Overall this study suggests that statistical sampling of a solid-state system can be swiftly performed with GRU/LSTM, thus, RNN-based models can be potentially employed as complementary tools AIMD simulations in order to markedly reduce computational costs. For future studies, these models will be tested on more complicated systems, simulating how a system behaves out of equilibrium, e.g. during relaxation times, phase transitions or defect formation.

7

4 Conclusions

In this work, we explored two types of recurrent neural networks, namely gated recurrent unit and long short-term memory networks, for the fast prediction of trajectory and potential energy profiles in *ab initio* molecular dynamics simulations. We found that these algorithms can learn from the distribution of the coordinates of atoms over the simulation run. We treated the problem as a multi-variate time-series and referred the interactions between the atoms to temporary correlations. Our model is compared against the baseline AIMD simulation. Findings suggest that both GRU and LSTM networks can be considered as alternative models to AIMD.

5 Acknowledgments

References

[1] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.

[2] Santosh K Gaikwad, Bharti W Gawali, and Pravin Yannawar. A review on speech recognition technique. *International Journal of Computer Applications*, 10(3):16–24, 2010.

[3] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[4] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. Wide & deep learning for recommender systems. *CoRR*, abs/1606.07792, 2016.

[5] Rick L Wilson and Ramesh Sharda. Bankruptcy prediction using neural networks. *Decision support systems*, 11(5):545–557, 1994.

[6] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[7] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[8] Paul Zikopoulos, Chris Eaton, et al. *Understanding big data: Analytics for enterprise class hadoop and streaming data*. McGraw-Hill Osborne Media, 2011.

[9] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[10] Davide Chicco. Ten quick tips for machine learning in computational biology. *BioData mining*, 10(1):35, 2017.

[11] Leyi Wei, Yijie Ding, Ran Su, Jijun Tang, and Quan Zou. Prediction of human protein subcellular localization using deep learning. *Journal of Parallel and Distributed Computing*, 117:212–217, 2018.

[12] Jie Hou, Badri Adhikari, and Jianlin Cheng. Deepsf: deep convolutional neural network for mapping protein sequences to folds. *Bioinformatics*, 34(8):1295–1303, 2017.

[13] Alán Aspuru-Guzik, Roland Lindh, and Markus Reiher. The matter simulation (r) evolution. *ACS central science*, 4(2):144–152, 2018.

[14] Benjamin Sanchez-Lengeling and Alán Aspuru-Guzik. Inverse molecular design using machine learning: Generative models for matter engineering. *Science*, 361(6400):360–365, 2018.

[15] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[16] Tran Doan Huan, Rohit Batra, James Chapman, Sridevi Krishnan, Lihua Chen, and Rampi Ramprasad. A universal strategy for the creation of machine learning-based atomistic force fields. *NPJ Computational Materials*, 3(1):37, 2017.

[17] Ying Li, Hui Li, Frank C Pickard IV, Badri Narayanan, Fatih G Sen, Maria KY Chan, Subramanian KRS Sankaranarayanan, Bernard R Brooks, and Benoît Roux. Machine learning force field parameters from ab initio data. *Journal of chemical theory and computation*, 13(9):4492–4503, 2017.

[18] Felix Brockherde, Leslie Vogt, Li Li, Mark E Tuckerman, Kieron Burke, and Klaus-Robert Müller. Bypassing the kohn-sham equations with machine learning. *Nature communications*, 8(1):872, 2017.

[19] Stefan Chmiela, Alexandre Tkatchenko, Huziel E Sauceda, Igor Poltavsky, Kristof T Schütt, and Klaus-Robert Müller. Machine learning of accurate energy-conserving molecular force fields. *Science advances*, 3(5):e1603015, 2017.

[20] Ivan Kassal, Stephen P Jordan, Peter J Love, Masoud Mohseni, and Alán Aspuru-Guzik. Polynomial-time quantum algorithm for the simulation of chemical dynamics. *Proceedings of the National Academy of Sciences*, 105(48):18681–18686, 2008.

[21] Volker L Deringer, Noam Bernstein, Albert P Bartók, Matthew J Cliffe, Rachel N Kerber, Lauren E Marbella, Clare P Grey, Stephen R Elliott, and Gábor Csányi. Realistic atomistic structure of amorphous silicon from machine-learning-driven molecular dynamics. *The journal of physical chemistry letters*, 9(11):2879–2885, 2018.

[22] Michael Gastegger, Jörg Behler, and Philipp Marquetand. Machine learning molecular dynamics for the simulation of infrared spectra. *Chemical science*, 8(10):6924–6935, 2017.

[23] Stefan Chmiela, Huziel E Sauceda, Igor Poltavsky, Klaus-Robert Müller, and Alexandre Tkatchenko. sgdml: Constructing accurate and data efficient molecular force fields using machine learning. *Computer Physics Communications*, 2019.

[24] Tian Xie and Jeffrey C Grossman. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Physical review letters*, 120(14):145301, 2018.

[25] Sung Sakong, Katrin Forster-Tonigold, and Axel Groß. The structure of water at a pt (111) electrode and the potential of zero charge studied from first principles. *The Journal of chemical physics*, 144(19):194701, 2016.

[26] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[27] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[28] Michael Schultz and Stefan Reitmann. Prediction of aircraft boarding time using lstm network. In *2018 Winter Simulation Conference (WSC)*, pages 2330–2341. IEEE, 2018.

[29] Tianjun Zhang, Shuang Song, Shugang Li, Li Ma, Shaobo Pan, and Liyun Han. Research on gas concentration prediction models based on lstm multidimensional time series. *Energies*, 12(1):161, 2019.

[30] Yu Zhao, Rennong Yang, Guillaume Chevalier, Rajiv C Shah, and Rob Romijnders. Applying deep bidirectional lstm and mixture density network for basketball trajectory prediction. *Optik*, 158:266–272, 2018.

[31] Rich Wolski, John Brevik, Ryan Chard, and Kyle Chard. Probabilistic guarantees of execution duration for amazon spot instances. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, page 18. ACM, 2017.

[32] James Bergstra, Brent Komer, Chris Eliasmith, Dan Yamins, and David D Cox. Hyperopt: a python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, 8(1):014008, 2015.

[33] Matt Baughman, Christian Haas, Rich Wolski, Ian Foster, and Kyle Chard. Predicting amazon spot prices with lstm networks. In *Proceedings of the 9th Workshop on Scientific Cloud Computing*, page 1. ACM, 2018.

[34] J. Contreras, R. Espinola, F. J. Nogales, and A. J. Conejo. Arima models to predict next-day electricity prices. *IEEE Transactions on Power Systems*, 18(3):1014–1020, Aug 2003.

[35] Alaa Sagheer and Mostafa Kotb. Time series forecasting of petroleum production using deep lstm recurrent networks. *Neurocomputing*, 323:203–213, 2019.

[36] Caglar Gulcehre, Kyunghyun Cho, Razvan Pascanu, and Yoshua Bengio. Learned-norm pooling for deep feedforward and recurrent neural networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 530–546. Springer, 2014.

[37] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[38] Xin Ma and Zhibin Liu. Predicting the oil production using the novel multivariate nonlinear model based on arps decline model and kernel method. *Neural Computing and Applications*, 29(2):579–591, 2018.

[39] N Chithra Chakra, Ki-Young Song, Madan M Gupta, and Deoki N Saraf. An innovative neural forecast of cumulative oil production from a petroleum reservoir employing higher-order neural networks (honns). *Journal of Petroleum Science and Engineering*, 106:18–33, 2013.

[40] Rob J Hyndman and Anne B Koehler. Another look at measures of forecast accuracy. *International journal of forecasting*, 22(4):679–688, 2006.

[41] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[42] Danilo P Mandic and Jonathon Chambers. *Recurrent neural networks for prediction: learning algorithms, architectures and stability*. John Wiley & Sons, Inc., 2001.

[43] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[44] Georg Kresse and Jürgen Furthmüller. Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Physical review B*, 54(16):11169, 1996.

[45] John P Perdew, Kieron Burke, and Matthias Ernzerhof. Generalized gradient approximation made simple. *Physical review letters*, 77(18):3865, 1996.

[46] Peter E Blöchl. Projector augmented-wave method. *Physical review B*, 50(24):17953, 1994.

[47] Hendrik J Monkhorst and James D Pack. Special points for brillouin-zone integrations. *Physical review B*, 13(12):5188, 1976.

[48] William Humphrey, Andrew Dalke, and Klaus Schulten. Vmd: visual molecular dynamics. *Journal of molecular graphics*, 14(1):33–38, 1996.

[49] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[50] François Chollet et al. Keras, 2015.

[51] Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

[52] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[53] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90, 2007.