Electronic Supplementary Material (ESI) for Digital Discovery.
This journal is © The Royal Society of Chemistry 2022

Deep Learning Models for the Estimation of Free Energy of Permeation of Small Molecules across Lipid Membranes

Prantar Dutta, Deepak Jain, Rakesh Gupta*, Beena Rai

Physical Sciences Research Area, Tata Research Development and Design Centre, TCS Research,

54-B, Hadapsar Industrial Estate, Pune- 411013, India

* *Correspondence to*: Rakesh Gupta

Email: gupta.rakesh2@tcs.com, Phone: + 91-20-66086422

Supporting Information

S1. Martini Bead Types

The Martini 2 coarse-grained force field defines 18 bead types divided into four categories — polar (P), nonpolar (N), apolar (C), and charged (Q). The charged beads (four in number) are not considered in the database, so free energy data is available for dimers of the other 14 bead types. Beads of types P and C are assigned integers from 1 to 5 to indicate their polarity (1 being the least polar and 5 being the most). In contrast, N beads are differentiated according to their ability to form hydrogen bonds (a = acceptor, d = donor, da = both, 0 = none). The following table lists the bead types in this study and their corresponding octanol-water partitioning free energy reported by Bereau and Kremer [1].

Table S1: The octanol-water partitioning free energy (ΔG_{OW}) of neutral Martini beads at 298 K

Bead Type	ΔG_{OW} (kcal/mol)
C1	3.4
C2	3.3
C3	3.0
C4	2.4
C5	1.7
N0	1.0
Na	0.6
Nd	0.6
Nda	0.6
P1	-0.5
P2	-0.9

P3	-2.1
P4	-2.2
P5	-2.1

S2. Alternate Training-Validation-Test Data Splits

To build our models, we performed a 90:10 train-test split, followed by 90:10 train-validation split of the training portion from the first split. Thus, starting from our dataset of 630 systems, we obtained 510 training examples, 57 validation examples, and 63 test examples. The results presented in the main paper are based on a particular training-validation-test split. However, to ensure that the deep learning models are not biased towards this specific distribution, we used different random states to generate three additional sets of training, validation, and test data. Both neural network architectures discussed in this study— Model-L and Model-LA were trained and evaluated on these sets, along with the baseline deep neural network (DNN) model. Table S2 shows the models' performance metrics on the new sets, referred to as Set_1, Set_2, and Set_3. We observe no notable change in the model performances on shuffling the training, validation, and test data. Hence, the models are robust and reliable.

Table S2: Mean Absolute Error (MAE) (in kcal/mol) and Coefficient of Determination (R^2) of deep learning models on the training, validation, and test sets for the three additional data splits

Model	**Data Split**	**Performance Metrics**					
		Training Set		**Validation Set**		**Test Set**	
		MAE	**R^2**	**MAE**	**R^2**	**MAE**	**R^2**
Model-L	Set_1	0.34	0.991	0.45	0.985	0.50	0.985
	Set_2	0.31	0.993	0.44	0.986	0.45	0.986
	Set_3	0.38	0.990	0.41	0.986	0.52	0.981
Model-LA	Set_1	0.24	0.995	0.36	0.990	0.43	0.988
	Set_2	0.22	0.996	0.32	0.993	0.43	0.988
	Set_3	0.24	0.996	0.37	0.989	0.45	0.987

DNN	Set_1	0.88	0.944	0.90	0.943	1.06	0.931
	Set_2	0.88	0.946	1.10	0.928	0.98	0.938
	Set_3	0.87	0.947	0.88	0.944	0.98	0.938

S3. Molecules in Trimer-Test Set

The following table lists the 50 molecules of three neutral Martini beads selected for additional testing and model validation. These molecules were selected from a dataset of 694 trimers reported by Hoffmann et al. [2].

Table S3: Molecules of the trimer-test set along with their free energy of permeation

Molecule	Free Energy (kcal/mol)	Molecule	Free Energy (kcal/mol)	Molecule	Free Energy (kcal/mol)
C1-C1-Nda	-6.66	C3-Na-C1	-5.13	Nd-N0-Nda	4.24
C1-C5-Nd	-3.46	C4-C1-C4	-7.95	Nd-Na-C1	-0.60
C1-N0-Na	-1.71	C4-C3-Na	-2.99	Nda-C2-Nd	-0.21
C1-N0-P2	0.26	C4-C3-Nda	-2.97	Nda-C3-C3	-3.96
C1-N0-P4	2.32	C5-C1-Nd	-3.59	Nda-P4-C2	4.35
C1-P5-C4	0.92	C5-C1-P5	1.44	P1-C1-Na	0.26
C2-C1-C3	-10.58	C5-C4-Nd	-1.22	P1-C3-C2	-3.59
C2-C2-P1	-4.42	N0-C1-C2	-7.14	P1-N0-C1	-0.60
C2-C3-Nd	-4.68	N0-C1-P1	-0.86	P1-Nd-C5	3.54
C2-Na-C5	-2.83	N0-C5-P1	2.38	P1-Nda-C4	2.80
C2-Nd-P1	1.04	N0-P5-C2	4.05	P2-C3-Nd	2.62
C2-Nda-Nda	-0.03	Na-C1-C1	-6.66	P2-P5-C1	6.61
C2-P4-C4	0.68	Na-C2-P3	2.88	P3-C1-N0	1.17
C2-P5-P5	10.4	Na-C5-C3	-2.11	P5-C1-P1	5.34
C3-C1-Nda	-5.32	Na-Nda-P2	7.25	P5-C4-C2	1.32
C3-C1-P3	-2.27	Nd-C2-C1	-6.06	P5-N0-C4	5.61
C3-C4-C3	-7.45	Nd-C3-P4	4.54		

S4. Unravelling LSTM and Attention Mechanism

Recurrent neural networks (RNNs) are a special class of artificial neural network equipped to handle sequential data. Unlike ordinary neural networks where all inputs and outputs are independent of each other, RNNs handle data where the output at a particular time step in a sequence depends on the inputs at previous time steps as well. For example, in natural language processing tasks like text prediction, it is necessary to "memorize" all the previous words in a sentence for predicting the next word. RNNs achieve this 'memory' using a hidden state at each time step such that the output of a particular time step depends both on the input at that time step and the hidden state from the previous time step. Contrary to traditional neural networks where each hidden layer has its own weights and biases, RNNs use common weights and biases across all hidden layers such that they theoretically collapse into a single recurrent layer.

Despite their power in handling sequences, RNNs suffer from the vanishing gradient problem which limits its ability to learn long-term dependencies. As the gradients are backpropagated through time, the derivatives become very small, and training is not effective. LSTMs are a variant of RNN which tackles this problem by maintaining a cell state for each cell and using logical structures called for adding or removing information from the cell state. Figure S1 shows a typical LSTM cell with its three gates— input, forget, and output. The equations for the three gates are as follows:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

where i_t, f_t, and o_t represent the input, forget, and output gates respectively, *W*'s and *b*'s are the weights and biases of the gate neurons, σ is the sigmoid function, h_{t-1} is the output of the previous cell (at time step *t-1*), and x_t is the input at the current time step. The cell state and output are calculated as follows:

$$\tilde{C}_t = \tanh\left(W_c[h_{t-1}, x_t] + b_c\right)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$h_t = o_t * \tanh(C_t)$$

where $\tilde{C}_t$ is the candidate value for cell state and C_t is the cell state at time t, W_c and b_c represent the weight and bias, and h_t is the cell output. The sigmoid activation function in the three gates produce values between 0 and 1, which determines the extent to which information is added or removed from the cell state. The cell output depends on the current input and the output and cell state of the previous cell. A detailed discussion on RNN and LSTM can be found the tutorial by Sherstinsky [3].

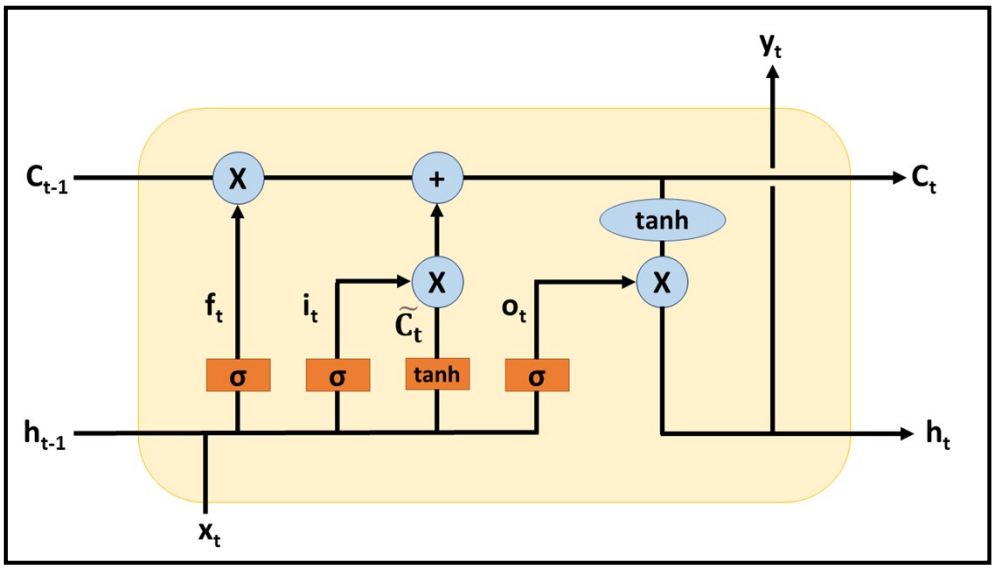

Figure S1: A typical LSTM cell with inputs, outputs, and gates

Attention mechanism in deep learning developed as a technique of quantifying the interdependence between the elements of an input sequence with the output and with each other. It is generally used as a part of a larger encoder-decoder architecture for handling sequences. Various types of attention mechanisms have been proposed over the last few years, and thus arriving at a generalized definition is difficult. Here, we focus on the Bahdanau attention mechanism [4] tailored to our problem, which is different than the machine translation problem for which it was applied. The first step of our implementation involves computing the alignment scores from the hidden states of the previous LSTM layer using a simple single-layer feedforward neural network. Then, the attention weights are calculated by applying the SoftMax

function on the alignment scores. Finally, the context vector is obtained by performing a weighted sum of the attention weights with the hidden states. In the original paper, the alignment scores are calculated using decoder output at the previous time step, in addition to the hidden states, and the context vector is fed to the decoder. As our problem do not involve a decoding step, we only use the hidden states for computing alignment score, and the context vector is fed to a dense layer. Figure S2 shows a magnified view of how the LSTM-Attention block works in Model-LA. The input to the model is a 350 (time steps/snapshots from the trajectory) x 8 (features) matrix for each target free energy value. The matrix passes through the LSTM layer producing the hidden states, which then enter the attention layer to generate the context vector. The context vector is processed through the dense layer, and a final output neuron predicts the free energy.

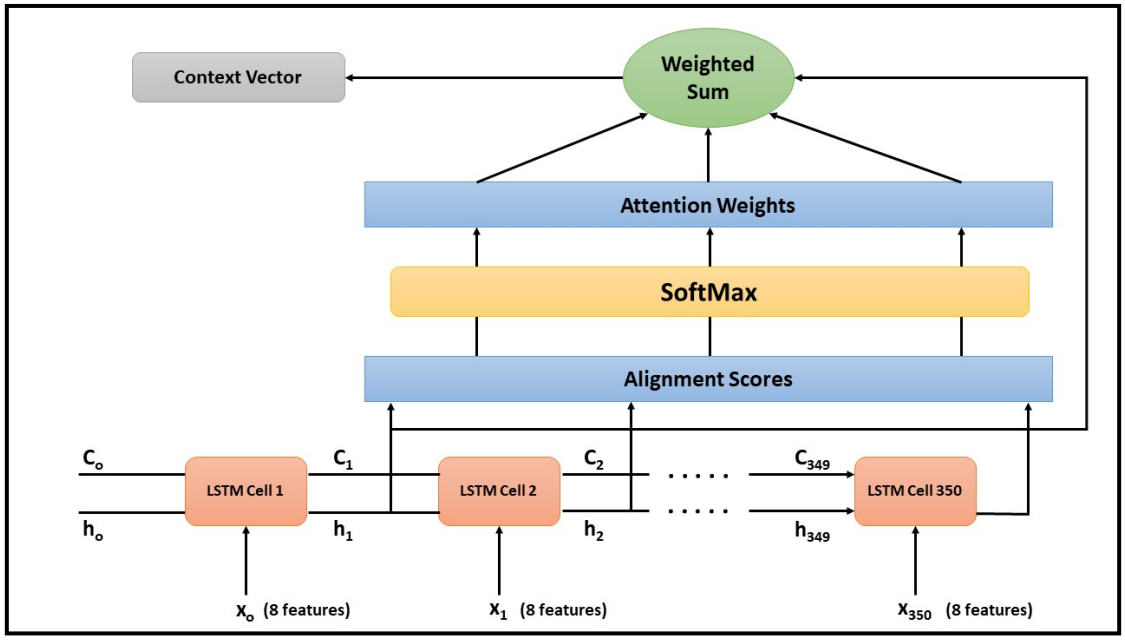

Figure S2: A magnified view of the LSTM-Attention block in Model-LA

S5. Loss History Plots

The loss history plots for Model-L and Model-LA during the training process are shown in Figure S3. Although the losses almost reach converge with a few hundred epochs, we continued to train the model for 1000 epochs to observe the training behaviour. The model with the lowest validation loss was chosen for both cases.

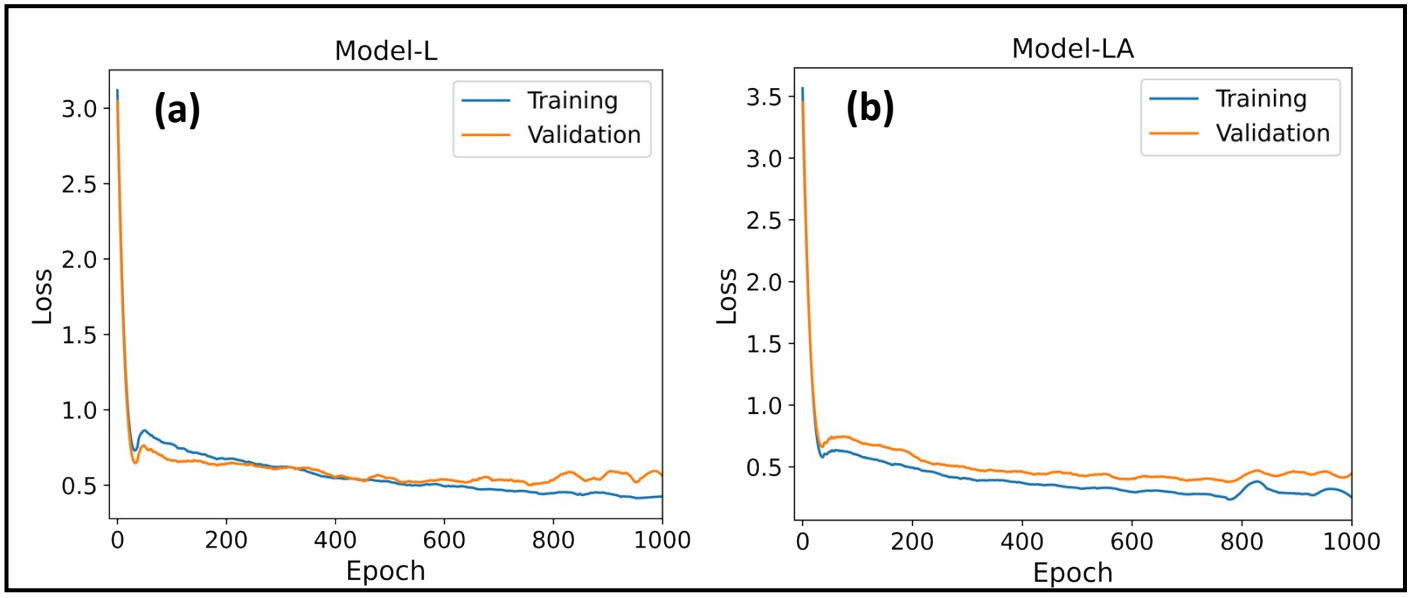

Figure S3: Loss history plots of (a) Model-L and (b) Model-LA during the training process

References

[1] T. Bereau and K. Kremer, "Automated Parametrization of the Coarse-Grained Martini Force Field for Small Organic Molecules," *Journal of Chemical Theory and Computation,* vol. 11, no. 6, pp. 2783-2791, 2015.

[2] C. Hoffmann, R. Menichetti, K. H. Kanekal and T. Bereau, "Controlled exploration of chemical space by machine learning of coarse-grained representations.," *Physical Review E,* vol. 100, no. 3, p. 33302, 2019.

[3] A. Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network," *Physica D: Nonlinear Phenomena,* vol. 404, 2020.

[4] D. Bahdanau, K. Cho and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," in *International Conference on Learning Representations (ICLR)*, 2015.