Supplementary Material for Accelerated Modelling of Interfaces for Electronic Devices using Graph Neural Networks

Pratik Brahma[1*], **Krishnakumar Bhattaram**[1*], **Sayeef Salahuddin**[1,2]

[1] Department of Electrical Engineering and Computer Sciences
University of California Berkeley
[2] Materials Science Division, Lawrence Berkeley National Laboratory
{pratik_brahma, Krishna Bhattarai, sayeef}@berkeley.edu

1 Neural Network Architecture

This section discusses the architecture of our graph neural network used to predict atomic forces, density of states, and injection velocity. This neural network architecture combines SchNet[S4] and SpookyNet[S7]. For completeness, we describe each neural network layer here and the acronyms present in Figure 2 of the main text.

Atom Embeddings The atomistic device structure is described by the atomic numbers of all atoms $Z = (Z_1, ..., Z_N)$ and their atomic positions $R = (\vec{r}_1, ..., \vec{r}_N)$, where N is the total number of atoms in the device. An atom embedding is a random vector initialized by an embedding layer depending on the atom type.

$$\mathbf{x}_i = \mathbf{a}_{Z_i} \quad \mathbf{a}_{Z_i} \in \mathbb{R}^F \tag{S1}$$

In the above equation, F is the dimension of the embedding space.

Linear Layer A linear layer (**lin**) is defined by:

$$\mathbf{x}_i^o = \mathbf{lin}(\mathbf{x}_i) = W\mathbf{x}_i + \mathbf{b} \tag{S2}$$

Linear Layer with Shifted Softplus Activation A linear layer with a shifted soft plus activation (**lssp**) is defined as:

$$\begin{aligned} \mathbf{x}_i^o &= \mathbf{lssp}(\mathbf{x}_i) = \mathbf{ssp}(W\mathbf{x_i} + \mathbf{b}) \\ \mathbf{ssp}(x) &= \log\left(0.5e^x + 0.5\right) \end{aligned} \tag{S3}$$

Radial Basis Function Neural Network Firstly, we calculate distance vectors ($\vec{r}_{ij}$) to all neighbors of every atom within a given cutoff r_c. The radial basis functions take the inter-atomic distance vector ($\vec{r}_{ij}$) as inputs and generate features that capture the symmetries in the atomistic device structure. These functions are defined as follows:

$$\mathbf{g}_s(\vec{r}) = \begin{bmatrix} {}_0g_0^0 \\ \vdots \\ {}_{K-1}g_0^0 \end{bmatrix} \tag{S4}$$

* These authors contributed equally to this work

Preprint. Under review.

$$\vec{\mathbf{g}}_p(\vec{r}) = \begin{bmatrix} {}_0g_1^{-1} & {}_0g_1^{0} & {}_0g_1^{1} \\ \vdots & \vdots & \vdots \\ {}_{K-1}g_1^{-1} & {}_{K-1}g_1^{0} & {}_{K-1}g_1^{1} \end{bmatrix} \tag{S5}$$

$$\vec{\mathbf{g}}_d(\vec{r}) = \begin{bmatrix} {}_0g_2^{-2} & {}_0g_2^{-1} & {}_0g_2^{0} & {}_0g_2^{1} & {}_0g_2^{2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ {}_{K-1}g_2^{-2} & {}_{K-1}g_2^{-1} & {}_{K-1}g_2^{0} & {}_{K-1}g_2^{1} & {}_{K-1}g_2^{2} \end{bmatrix} \tag{S6}$$

$$ {}_kg_l^m = \rho_k(\|\vec{r}_{ij}\|) \cdot Y_l^m(\vec{r}_{ij}) \tag{S7}$$

$Y_l^m(\vec{r}_{ij})$ represents the spherical harmonics features. K represents the total number of radial basis function network features.

$$\rho_k(r) = b_{k,K-1}\left(\exp(-\gamma r)\right) \cdot f_{cut}(r) \tag{S8}$$

$$b_{k,K-1}(x) = \binom{K-1}{k} x^k (1-x)^{K-1-k} \tag{S9}$$

$$f_{cut}(r) = \begin{cases} \exp\left(-\frac{r^2}{(r_{\text{cut}}-r)(r_{\text{cut}}+r)}\right), & r < r_{\text{cut}} \\ 0, & r \geq r_{\text{cut}} \end{cases} \tag{S10}$$

In the above equation, $b_{k,K-1}$ are the Bernstein polynomials. We use a Gaussian filter cutoff (f_{cut}) to emulate decreasing contribution to output properties with increasing neighbor distance.

Local Interaction Block The local interaction block generates messages with the inter-atomic distance vectors as inputs. These messages contain the local chemical environment information of each atom. The equations are as follows:

$$\mathbf{c}_i = \mathbf{lssp}(\mathbf{x}_i) \tag{S11}$$

$$\mathbf{s}_i = \sum_{j\in\mathcal{N}(i)} \mathbf{lin}_s(\mathbf{x}_j) \odot (\mathbf{G}_s \mathbf{g}_s(\vec{r}_{ij})) \tag{S12}$$

$$\vec{\mathbf{p}}_i = \sum_{j\in\mathcal{N}(i)} \mathbf{lin}_p(\mathbf{x}_j) \odot (\mathbf{G}_p \vec{\mathbf{g}}_p(\vec{r}_{ij})) \tag{S13}$$

$$\vec{\mathbf{d}}_i = \sum_{j\in\mathcal{N}(i)} \mathbf{lin}_d(\mathbf{x}_j) \odot (\mathbf{G}_d \mathbf{g}_s(\vec{r}_{ij})) \tag{S14}$$

$$\mathbf{l}_i = \mathbf{lssp}\left(\mathbf{c}_i + \mathbf{s}_i + (\mathbf{P}_1\vec{\mathbf{p}}_i)^T (\mathbf{P}_2\vec{\mathbf{p}}_i) + \left(\mathbf{D}_1\vec{\mathbf{d}}_i\right)^T \left(\mathbf{D}_2\vec{\mathbf{d}}_i\right)\right) \tag{S15}$$

In the above equations, $\mathcal{N}(i)$ represents the neighbors of atom i within a cutoff r_c. $\mathbf{G}_s, \mathbf{G}_p, \mathbf{G}_d \in \mathbb{R}^{F\times K}$ are matrices which calculates the linear combinations of features $\mathbf{s}_i \in \mathbb{R}^F$, $\vec{\mathbf{p}}_i \in \mathbb{R}^{F\times 3}$ and $\vec{\mathbf{d}}_i \in \mathbb{R}^{F\times 5}$, where F is the number of filters. The matrices $\mathbf{P}_1, \mathbf{P}_2, \mathbf{D}_1$ and $\mathbf{D}_2$ provide linear projections for the rotationally equivariant features $\vec{\mathbf{p}}_i$ and $\vec{\mathbf{d}}_i$. In the end, a message l_i is generated, which contains the local environment information to update the state vector of atom i.

Message Update Using the generated message l_i for atom i, we update the state vector ($\mathbf{x}_i^t$) by using the following equation:

$$\mathbf{x}_i^{t+1} = \mathbf{lssp}\left(\mathbf{lssp}\left(\mathbf{x}_i^t\right) + \mathbf{l}_i\right) \tag{S16}$$

Energy and Atomic Forces Output After performing the message update phase T times, we use the final state vectors x_i^T to predict the desired property, assuming that the global property is obtained from the summation of its local parts.

$$R^{\text{global}} = \sum R^{\text{local}} = \sum_i f_{NN}\left(\mathcal{A}_i\right) \tag{S17}$$

In the above equation, f_{NN} is the neural network that predicts the local property from the local atomic environment $\mathcal{A}_i$ for every atom i. Under this assumption, we predict the total energy as a summation of local atomic energies as follows:

$$E^{gs} = \sum_i E_i^{gs} = \sum_i \textbf{lssp}\left(x_i^T\right) \tag{S18}$$

A linear layer with shifted softplus activation transforms the state vector of every atom i to its local atomic energy contribution E_i^{gs}. Since the graph neural network is composed of smooth differentiable functions, the atomic force on all atoms can be calculated by differentiating the total ground state energy output with respect to their atomic positions.

$$\vec{\mathbf{F}}_i\left(Z_1, ..., Z_N, \vec{\mathbf{r}}_1, ..., \vec{\mathbf{r}}_N\right) = -\frac{\partial E_{gs}}{\partial \vec{\mathbf{r}}_i}\left(Z_1, ..., Z_N, \vec{\mathbf{r}}_1, ..., \vec{\mathbf{r}}_N\right) \tag{S19}$$

Density of States Output In this section, we provide brief details on how the neural network converts the state vector $\mathbf{x}_i^T$ to its local contribution of $D(E)$ and $J_x(E) = v_x(E)D(E)$. Notably, the Density of States (DOS) needs vector-valued outputs; as a result, the readout neural network has to predict multiple output values. The vector is generated by sampling $D(E)$ and J_x in some energy windows. The training convergence deteriorates with increased points in the energy space of DOS. Thus, reducing the dimension of the DOS and J_x vector becomes necessary using Principal Component Analysis (PCA) [S1]. The training dataset containing the DOS vectors is first normalized, such that the i^{th} energy window is $\mathbf{y}_i = \mathbf{d}_i - \bar{\mathbf{d}}$, where $\bar{\mathbf{d}}$ is the mean of the whole dataset. Subsequently, the eigenvalues (λ_p) and the eigenvectors ($\mathbf{u}_p$) of the covariance matrix $\mathbf{S} = \mathbf{Y^T Y}$ ($\mathbf{Y}$ are matrices with columns as $\mathbf{y_i}$) are found:

$$\mathbf{S u_p} = \lambda_p \mathbf{u_p} \tag{S20}$$

$$d \approx \sum_{p=1}^{P}\left(\mathbf{y}^T \mathbf{u}_p\right)\mathbf{u}_p + \sum_{p=1}^{P}\left(\bar{\mathbf{d}}^T \mathbf{u}_p\right)\mathbf{u}_p = \sum_{p=1}^{P} \alpha_p^T \mathbf{u}_p \tag{S21}$$

The highest P eigenvalue vectors are chosen to reconstruct the DOS pattern in energy space from the PCA basis. Thus, the neural network is then trained to predict only the P PCA coefficients as follows:

$$\text{DOS} = \sum_{i=1}^{N} \text{LDOS}\left(\mathcal{A}_i\right) = \sum_{i=1}^{N}\sum_{p=1}^{P} \alpha_{p,i}^T \mathbf{u}_p \tag{S22}$$

$$\alpha_{p,i} = \textbf{lssp}\left(\mathbf{x}_i^T\right) \tag{S23}$$

The above equation **lssp** is a neural network layer whose output dimensions are P and $\alpha_{p,i}$ are the predicted PCA coefficients by the neural network for atom i. Notably, in this case, the output neural network predicts the local density of states (LDOS) for every atom, whose summation gives us the total DOS of the atomistic device. The same procedure is used to predict the current density $J_x(E) = v_x(E)D(E)$ where $v_x(E)$ is the bandstructure velocity and $D(E)$ is the density of states. With the two predicted properties, we can calculate the injection velocity of the transistor as follows [S2]:

$$v_{inj} = \frac{\int dE J_x(E) f(E + U - E_f)}{\int dE D(E) f(E + U - E_f)} \tag{S24}$$

$$N_{inv} = \int dE D(E) f(E + U - E_f) \tag{S25}$$

$f(x)$ is the fermi distribution function, U is the electrostatic potential at the source-channel barrier, and E_f is the fermi level of the source. Consequently, we can obtain drain current through the transistor channel as $I_D = q N_{inv} v_{inj}$

2 Dataset Generation

This section covers the training datasets used to train our graph neural network architecture. We have two different kinds of datasets: i) Molecular dynamics of the transistor heterostructure for training

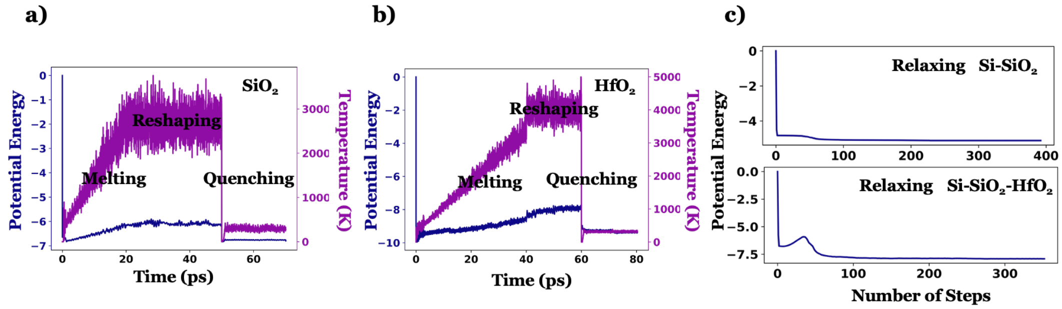

Figure S1: **Molecular Dynamics Trajectory**: a) Melting, Reshaping, and Quenching crystalline silica to obtain its amorphous structure, b) Melting, Reshaping, and Quenching crystalline hafnia to obtain its amorphous structure c)Relaxing the Si-SiO$_2$ interface and relaxing transistor heterostructure

energy and atomic forces and ii) Empirical tight-binding calculations of nanoslab silicon channel to predict the density of states (DOS) and current density (J_x).

2.1 Atomic Forces and Energy

The transistor gate stack consists of crystalline silicon, amorphous silica, and amorphous hafnia. This structure is generated using classical molecular dynamics and the Charge Optimized Many Body (COMB) potential [S5] to estimate the forces between the atoms. The molecular dynamics is performed using a timestep of 0.1 fs.

Generating Amorphous Silica : We start with β-cristobalite silicon oxide crystal structure. The silicon substrate of the transistor heterostructure contains two unit cells in the x and y direction. Therefore, the idea is to melt and reform the silica crystal structure such that the base dimensions of amorphous silica match the base dimensions of the silicon substrate. The initial crystalline silica contains one unit cell in the x and y directions and two unit cells along the z. It is melted using constant number, pressure, and temperature (NPT) dynamics from 300K to 2700K for 20 ps. Subsequently, we reshape the silica melt to match the silicon substrate dimensions using non-equilibrium constant number, volume, and temperature (NVT, nvt/sllod in LAMMPS) dynamics at 2700K for 30 ps. Finally, we quench the melt using a damped force minimizer followed by an NPT annealing at 300K for 20 ps. This whole procedure is shown in Figure S2a).

Generating Amorphous Hafnia : Similar to the previous case, we start with the orthorhombic crystal form of hafnia with, two unit cells in the x, y, and z directions. Following this, the crystal is melted from 300K to 3700K for 20 ps. Subsequently, we reshape the melt so that the base dimensions match the silicon substrate base dimensions using the non-equilibrium NVT process at 3700K for 40 ps. Finally, the melt is quenched using a damped force minimizer followed by an NPT annealing at 300K for 20 ps. This whole molecular dynamics procedure is shown in Figure S2b)

Generating Si-SiO$_2$ interface : To generate the interface, the previously generated amorphous silica is placed on top of the silicon substrate at a distance "d". Then the composite structure is relaxed using a damped force minimizer. The distance "d" is estimated such that the relaxed structure has the lowest energy. The potential energy relaxation is shown in Figure S2c).

Generating Si-SiO$_2$-HfO$_2$ heterostructure : Similar to the interface generation procedure above, the generated amorphous hafnia HfO$_2$ is placed on top of the generated Si-SiO$_2$ interface at a distance "d". Subsequently, the whole heterostructure is relaxed using a damped force minimizer. The distance "d" is estimated such that the relaxed transistor heterostructure has the lowest energy. The potential energy relaxation is shown in Figure S2c).

Using all the above-generated structures, we obtain a dataset containing around 200k training structures for training energy and molecular forces.

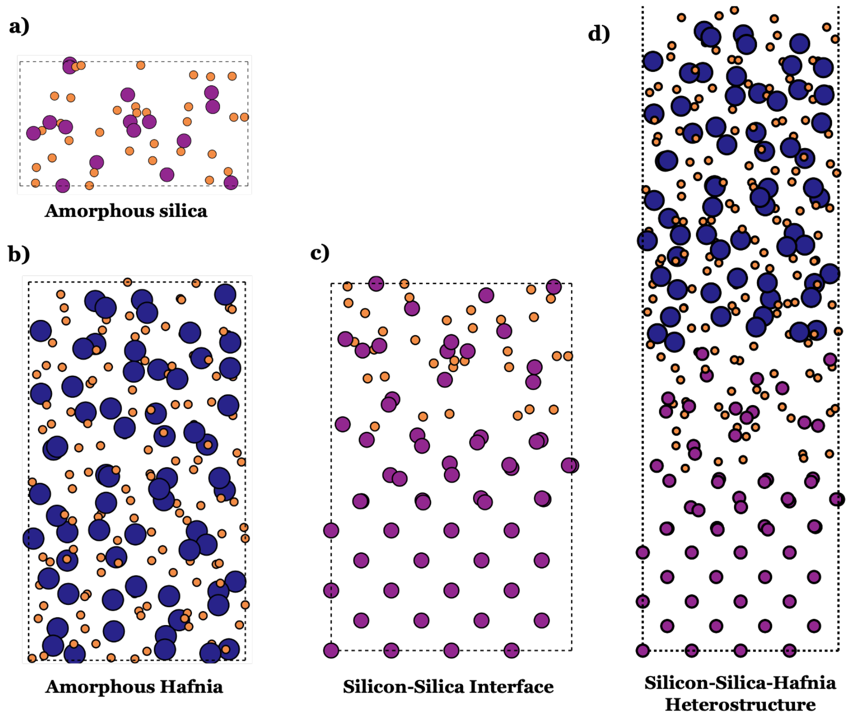

Figure S2: **Generated intermediary structures and final transistor heterostructure**: a) Generated amorphous silica structure, b) Generated amorphous hafnia structure, c) Relaxed silicon-silica structure, d) Relaxed transistor heterostructure

2.2 Density of States and Injection Velocity

Density of states and injection velocity calculations are conducted for the crystalline nanoslab silicon channels. We generate the dataset by varying the strain and thickness of the target structures and simulate electronic structure using a semiempirical tight-binding calculator available in QuantumATK [S6]. Nanoslabs are created by cleaving bulk silicon along the [100] direction, with thicknesses ranging from 5 to 19 layers of silicon, inclusive. Strain is introduced by varying the unit cell size, with unit cell sizes ranging between 0.9 to 1.035 times the unstrained bulk silicon unit cell, with steps of 0.002. The electronic Hamiltonian is constructed using QuantumATK's **SemiEmpiricalCalculator** with the Bassani SiH parameter dataset, with self-consistent calculation turned off. We derive two-dimensional bandstructure defined by a Monkhorst-Pack mesh at 150x150 k-points. Density of states is calculated through the **DensityOfStates** class using the tetrahedral sampling method.

The data from QuantumATK are then post-processed to derive quantities of interest that or to make quantities simpler for training and inference. Bandstructure velocity is calculated through a numerical first-order derivative on bandstructure energies from QuantumATK, which are Gaussian-broadened by 0.01 eV. Density of states energy scales are shifted such that the conduction band minimum E_c is at zero, and values (in eV^{-1}) are normalized by the total number of silicon atoms.

The statistics of the above two generated datasets are shown in Figure S3.

3 Training Neural Network

This section covers the training loss function of the neural network, the training parameters, and the neural network architecture size.

3.1 Atomic Forces and Energy

The neural network architecture size is depicted in Table 1: The loss function for training the neural

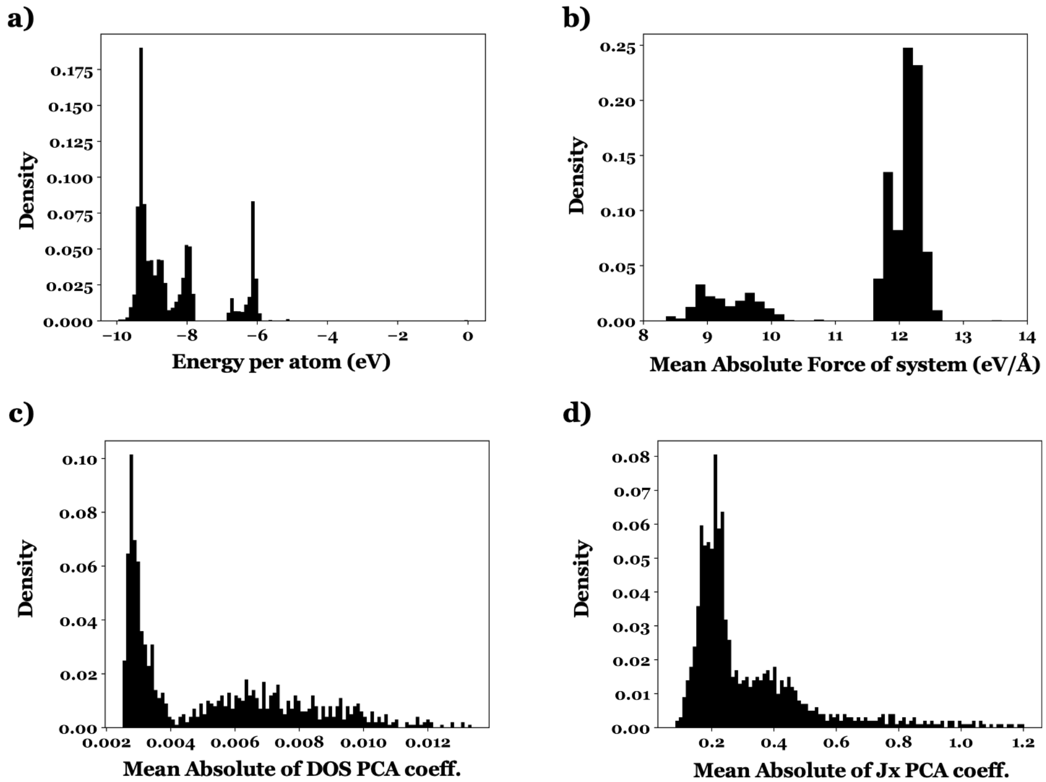

Figure S3: **Statistics of the generated datasets**: (a) Histogram of energy per atom of a structure for the molecular dynamics generated dataset (b) Histogram of mean absolute forces of a structure for the molecular dynamics generated dataset (c) Histogram of mean absolute value of the DOS PCA coefficients for the tight binding generated dataset (d) Histogram of mean absolute value of the J_x PCA coefficients for the tight binding generated dataset

Table 1: Graph Neural Network Size for Molecular Dyanamics

Neural Network Parameter	Value
Cutoff Radius (r_c)	3.0 Å
Number of atomic basis (F)	32
Number of radial basis functions (K)	32
Number of interaction blocks (T)	6

network to predict energy (E_{gs}) and forces $\vec{F}(\vec{\mathbf{R}}_i)$ for a material structure represented by $\vec{\mathbf{R}}_i$ is the weighted mean square error as follows:

$$L = \gamma \sum_i \left(E_\theta^{gs}(\vec{\mathbf{R}}_i) - E^{gs}(\vec{\mathbf{R}}_i) \right)^2 + (1-\gamma) \sum_i \|\vec{F}_\theta^{gs}(\vec{\mathbf{R}}_i) - \vec{F}^{gs}(\vec{\mathbf{R}}_i)\|^2 \tag{S26}$$

In the above equation, the variables with subscripts θ are the predicted neural network properties and the remaining variables are ground truth values from the dataset. For our training, we use $\gamma = 0.1$. All the training parameters are provided in table 3. For training energy and atomic forces, the dataset was split into 80-10-10 for training, validation, and testing respectively. All the neural network code was written in PyTorch-CUDA [S3]. Training and inference were performed on an NVIDIA A100 GPU. Finally, we get a mean absolute error of 3.0e-2 eV/Å (0.26 % error) on the training of forces.

Table 2: Graph Neural Network Size for predicting DOS and J_x

Neural Network Parameter	Value
Cutoff Radius (r_c)	5.0 Å
Number of atomic basis (F)	16
Number of radial basis functions (K)	16
Number of interaction blocks (T)	6
Number of PCA coefficients for DOS (P)	200
Number of PCA coefficients for J_x	15

Table 3: Training Parameters

Training Parameter	Value
Optimizer	Adam
Learning Rate	0.01
Learning Rate Schedule	Reduce lr on Plateau
Learning Rate multiplier	0.8
Batch Size	32

3.2 Density of States

Using the neural network architecture in table 2, we train on a weighted mean-squared error loss on the density of states ($D(E)$) and x-component of current density (J^x) as follows:

$$L = \gamma \sum_i \left\| D_\theta(\vec{\mathbf{R}}_i) - D(\vec{\mathbf{R}}_i) \right\|^2 + (1-\gamma) \sum_i \| J^x_\theta(\vec{\mathbf{R}}_i) - J^x(\vec{\mathbf{R}}_i) \|^2 \qquad \text{(S27)}$$

In this case, D_θ and J^x_θ are the reconstructed vector-valued energy-space density of states and injection velocity from the PCA coefficients that the network predicts. We use $\gamma = 0.5$ to equally weight the loss terms, and train on all structures with layer count between 5 to 18 -excluding 13- with lattice constant scaling from 0.9 to 1.035 with steps of 0.002, and validate on structures with 13 and 19 layers with the unscaled lattice constant 1.0 (not included in the training set). All the neural network code was written in PyTorch-CUDA [S3]. All the training parameters are provided in table 3. Training and inference were performed on an NVIDIA A100 GPU. We find a final mean absolute error of 9.0e-4 /eV (0.18% error) for D(E) and 4.9e4 cm/s-eV (0.82% error) for Jx(E).

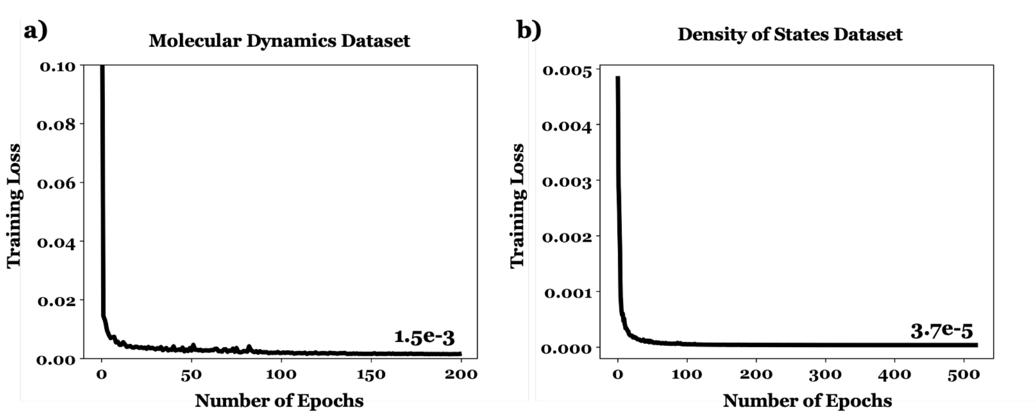

Figure S4: **Training Loss Curve**: The training loss curve for the (a) molecular dynamics dataset and the (b) density of states dataset

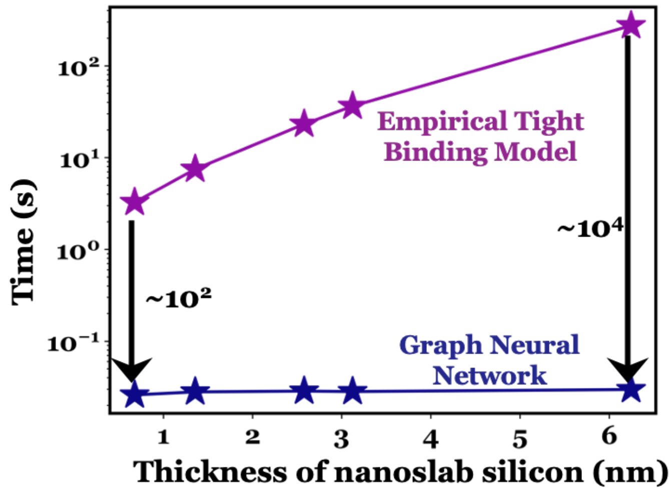

Figure S5: **Scaling of computational time for calculating DOS**: The Graph Neural Network accelerates the empirical tight binding model in the order of 10^2-10^4 for 0.5-6nm of nanoslab silicon thickness

4 Inference Timing

4.1 Atomic Forces and Energy

Molecular Dynamics to generate the dataset were performed using LAMMPS on an Intel Xenon Gold 6330 with 30 parallel threads. All the neural network code was written in PyTorch-CUDA and training and inference were performed on an NVIDIA A100 GPU. The prediction time of the neural network only includes the inference time and not the model loading time.

4.2 Density of States and Injection Velocity

QuantumATK codes for density of states generation were multithreaded and further parallelized across 16 processes on an Intel Xenon Gold 6330 with 112 cores and two threads per core. All the neural network code was written in PyTorch-CUDA and training and inference were performed on an NVIDIA A100 GPU. The prediction time of the neural network only includes the inference time and not the model loading time.

References

[S1] K. Bang, B. C. Yeo, D. Kim, S. S. Han, and H. M. Lee. Accelerated mapping of electronic density of states patterns of metallic nanoparticles via machine-learning. *Scientific Reports*, 11, 12 2021.

[S2] Y. Liu, N. Neophytou, T. Low, G. Klimeck, and M. S. Lundstrom. A tight-binding study of the ballistic injection velocity for ultrathin-body soi mosfets. *IEEE Transactions on Electron Devices*, 55:866–871, 3 2008.

[S3] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[S4] K. T. Schütt, H. E. Sauceda, P. J. Kindermans, A. Tkatchenko, and K. R. Müller. Schnet - a deep learning architecture for molecules and materials. *Journal of Chemical Physics*, 148, 6 2018.

[S5] T. R. Shan, B. D. Devine, T. W. Kemper, S. B. Sinnott, and S. R. Phillpot. Charge-optimized many-body potential for the hafnium/hafnium oxide system. *Physical Review B - Condensed Matter and Materials Physics*, 81, 3 2010.

[S6] S. Smidstrup, K. Stokbro, A. Blom, T. Markussen, J. Wellendorff, J. Schneider, T. Gunst, B. V. v Petr A Khomyakov, U. G. Vej-Hansen, M. Brandbyge, et al. Quantumatk: An integrated platform of electronic and atomic-scale modelling tools. *J. Phys: Condens. Matter (APS)*, 32:015901, 2020. doi: 10.1088/1361-648X/ab4007.

[S7] O. T. Unke, S. Chmiela, M. Gastegger, K. T. Schütt, H. E. Sauceda, and K. R. Müller. Spookynet: Learning force fields with electronic degrees of freedom and nonlocal effects. *Nature Communications*, 12, 12 2021.