FESC-Net: A Deep Learning Network for the Fusion of Electronic Structure Calculation Fidelities

Wei .W. Xing[a,d], Akeel A. Shah[b,*], Guohao Dai[c], Hong Qiu[c]

[a]School of Integrated Circuit Science and Engineering, Beihang University, 37 Xueyuan Road, Haidian District, Beijing, 100191,China
[b]School of Energy and Power Engineering, Chongqing University, 174 Shazhengjie, Shapingba, Chongqing 400044, China.
[c]School of Automation and Mechanical Design, Shenzhen University, Shenzhen, China
[d]Beihang Hangzhou Innovation Institute Yuhang, Xixi Octagon City, Yuhang District, Hangzhou 310023, China

Keywords: Multi-fidelity models; Autoregressive model; High-dimensional inputs; Electronic structure calculations

1. Introduction

Electronic structure theory methods for the prediction of molecular properties in chemical compound space form a hierarchy in terms of accuracy and computational costs. The Hartree-Fock (HF) method approximates the many-electron wave function using a single Slater determinant of single-electron wave functions, thereby neglecting electronic correlations and overestimating energies. Post-HF methods attempt to correct at various levels the inadequacies of HF. Configuration interaction (CI) incorporates two or more determinants, depending on the number of excitations allowed. Other notable methods include coupled cluster (CC) theory and M**o**ller-Plesset (MP) perturbation theory. All of these and other post-HF wave function methods add massive additional computational costs to the $O(N^3)$ cost of HF. As a result, they are restricted to small system sizes. Semi-empirical methods, such as CNDO/INDO [**?**], AM1 [**?**], ZINDO **?**], follow the same framework but at a considerably reduced computational cost afforded by neglecting or approximating the time-consuming two-electron integrals. The low accuracy is not acceptable for most applications, but these methods remain important for large-scale systems in which higher levels of theory are not possible.

The most ubiquitous electronic-structure methodology is Kohn-Sham (KS) density functional theory (DFT),

[*]Corresponding Author. E-mail: akeelshah@cqu.edu.cn; Tel: +86 178 8023 0871

with exchange and correlation energy functionals approximated by a hierarchy of models [?]. The basis of DFT is the Hohenberg-Kohn theorems [?], which permit circumvention of the problem of approximating the many-body wave function directly. Although it often fails in the presence of strong electronic correlations and non-local van der Waals interactions [?], DFT strikes a good balance between accuracy and computational costs. The scaling of DFT is $O(N^3)$, significantly lower than those of advanced wave function based methods, such as CCSD(T) [?] ($O(N^7)$) and MP2 [?] ($O(N^5)$). KS-DFT constructs a fictitious system of non-interacting electrons with electron density equal to that of the ground-state density $n(\mathbf{r})$ of the real system. At the heart of the standard KS-DFT approach is the exchange-correlation potential $v_{xc}[n](\mathbf{r})$, which arises as the functional derivative of an exchange-correlation energy $E_{xc}[n]$, a component of the total energy that is introduced by separating out the non-interacting kinetic energy functional and the Hartree energy.

The major approximation in KS-DFT concerns the form of $E_{xc}[n]$, the exact form of which is not known. The simplest approximation of all are the local density approximations (LDA), which approximate $E_{xc}[n]$ based on a homogeneous electron gas of density $n(\mathbf{r})$ [? ?]. Generalised gradient approximation (GGA) functionals such as PBE [?] depend locally on $n(\mathbf{r})$ and $\nabla n(\mathbf{r})$. Meta-GGA further incorporate the Laplacian $\nabla^2 n(\mathbf{r})$, e.g., TPSS [?], which has shown some show improvements over PBE in terms of atomisation and surface energies [?]. Using exact (Hartree-Fock) exchange energy as a functional of the KS orbitals in combination with local correlations leads to what are termed hybrid functionals. Perhaps the best known is the B3LYP functional, which combines an LDA exchange with a gradient correction, a GGA correlation, and a LDA correlation, with fitting of key parameters to molecular data [? ? ? ?]. The PBE0 functional, on the other hand, combines the PBE exchange with the Hartree-Fock exchange in a 3:1 ratio, together with the PBE correlation term [?]. In fact, there is a vast array of functionals available, with new functionals continually being developed. More sophisticated functionals, going from LDA to GGAs to meta-GGAs to hybrid functionals, typically improve accuracy but at the expense of considerable additional computational costs.

The computational resources and times required for very accurate predictions has motivated the use of approximate methods, beyond the semi-empirical methods described above. Notably, machine learning has been used in various ways to replace all or certain aspects of the calculations. The most obvious approach is to directly learn the map between the system and chosen outputs, such as atomization energies [**? ? ?**]. A smaller number of attempts have been made to find mappings between the charge density and the functional contributions to the total energy, as well as mappings between the external potential and the charge density [**? ?**]. The motivation is often a machine learning approximation of the kinetic energy functional to speed up the KS-DFT calculations [**? ?**], but there are key challenges in evaluating the functional derivative. An alternative is instead to focus on the density of states [**? ?**] or local density of states [**? ?**]. A key ingredient of all machine learning approaches for such systems is in characterising the inputs. A number of methods have been employed, including Coulomb matrices [**? ?**] or eigenvalues of Coulomb matrices [**? ?**], bags of bonds (BoB) [**?**], smooth overlap of atomic positions (SOAP) [**?**], or generalized symmetry functions [**?**].

Another main route for reducing the computational burden of complex models is multi-fidelity modelling, which combines models of different fidelity in such a way that it reduces reliance on high-fidelity results [**? ? ? ?**]. In contrast to pure machine learning approaches, multi-fidelity methods have receive almost no attention for electronic structure calculations, despite being one of the main surrogate modelling paradigms. Multi-fidelity models typically involve the construction of surrogate models based on the underlying models at different fidelity [**? ?**], or in a smaller number of cases, by including corrections to the low-fidelity results based on a limited number of observations at a higher fidelity [**? ? ? ?**]. This is particularly relevant for electronic structure methods since it may not always be advisable to replace the entire modelling approach with a data-driven counterpart.

Recently, Tran et al. [**?**] developed a multi-fidleity model taking as low and high fidelities the Spectral Neighbor Analysis Potential (SNAP) ML-IAP25 [**?**] and DFT models, respectively, with applications to MD simulations of ternary random alloys. The authors used the classical autoregressive (AR) model of **?**], which exploits correlations between low and high fidelity samples and assumes a linear relationship between the two to acquire accurate solutions without full reliance on the high-fidelity model. The same method was used by **?**

] to learn lattice energies of crystals, with a force field method used as the low-fidelity model and either GGA or hybrid DFT as the high fidelity model.

AR was improved upon by **?**], who developed the nonlinear autoregressive model (NAR) to enhance the predictive power of AR by placing a GP prior over the unknown mapping between fidelities. The major drawbacks to this approach is that it requires expensive sampling methods to sample from the posterior and the high fidelity GP takes the low fidelity output as an input. A tractable alternative was developed in by Xing et al. [**?**], based on a residual structure and GP priors over the residuals. This method leads to an analytical posterior and it was shown to outperform the other main GP based approaches, including deep GPs. Stochastic collocation [**?**] is also another prominent multi-fidelity method. In this method, the coefficients of a low-fidelity projection onto an approximating subspace are used as the coefficients for the high-fidelity equivalent (it is also a form of interpolation). As shown in **?**], it is nothing more than a GP prediction with a linear kernel. The obvious drawback of this method is that it relies on out-of-sample executions of the full low-fidelity model for making predictions, which is a serious drawback for extensive explorations of compound spaces.

In this paper, we develop a new multi-fidelity model specific to electronic structure calculation outputs, e.g. the ground state energy or highest/lowest un/occupied molecular orbital (HOMO/LUMO) energy. This new method relies on extracting features from the Coulomb matrix representations of the molecules, which together with the low fidelity output are fed into a convolutional neural network to predict the high fidelity output. The method is applied to the GDB-13 **?**] database, which stores a large volume of data on drug-like molecules with up to 13 heavy atoms. Over 7000 organic molecules appear in the database, so this other GDB sets have frequently been used for assessing machine learning methods **? ?**]. We show that our new method outperforms AR, NAR, ResGP and SC by a considerable margin on both the HOMO/LUMO and the polarizability.

2. Data sets

This dataset is a subset of GDB-13 (a database of nearly 1 billion stable and synthetically accessible organic molecules) composed of all molecules of up to 23 atoms (including 7 heavy atoms C, N, O, and S), totalling 7165 molecules. This dataset features a large variety of molecular structures such as double and triple bonds, cycles, carboxy, cyanide, amide, alcohol and epoxy. The molecule descriptors are Coulomb matrices, defined as

$$C_{ii} = \frac{1}{2} Z_i^{2.4} \tag{1}$$

$$C_{ij} = \frac{Z_i Z_j}{|R_i - R_j|} \tag{2}$$

where Z_i is the nuclear charge of atom i and R_i is its position. The Coulomb matrix has built-in invariance to translation and rotation of the molecule. The dataset comprises the inputs in the form of 23×23 Coulomb matrices and various properties calculated using different levels of theory. Of these properties polarizability and HOMO/LUMO data are available at multiple fidelities. In the first case, DFT results with a PBE0 functional are available as the high fidelity and SCS data are available as the (more approximate) low fidelity data. For the HOMO/LUMO, the values are available using ZINDO (low fidelity) and PBE0 as well as GW. For the given data set, the latter two give approximately the same level of accuracy, so either can be taken as the high fidelity data.

3. Multi-fidelity model

We are interested in solutions to electronic structure calculations, focusing on any general (scalar) quantity of interest $y(X)$, e.g., the ground state energy, as a function of inputs X that characterise the molecule, e.g., a Coulomb matrix. To obtain a high-fidelity approximation $y^h(X)$ for $y(X)$, we may use a method with a high level of theory, while lower-fidelity approximations $y^l(\boldsymbol{\xi})$ can be obtained using low(er) levels of theory. The multi-fidelity dataset takes the form $\{\mathbf{X}^f, \mathbf{y}^f\}$, where $\mathbf{X}^f \in \mathbb{R}^{N_f \times l \times m}$ is a multi-dimensional array of inputs X_n, $n = 1, \ldots, N_f$, of size $l \times m$ and $\mathbf{y}^f = (y^f(X_1), \ldots, y^f(X_{N_f}))^T$ is a vector of N_f corresponding solutions at fidelity level $f = l, h$. We use $f = h$ to denote the high fidelity and $f = l$ the low fidelity data. In line with the common setting for multi-fidelity emulation [**? ?**], the high-fidelity training inputs are assumed to be a subset of the low fidelity inputs, i.e., $\mathbf{X}^h \subset \mathbf{X}^l$.

The goal of this paper is to accurately approximate one or more quantities $y^h(X)$ from an high-fidelity (high level of theory) electronic structure calculation by using data from the high-fidelity and corresponding low-fidelity model. This is achieved through a combined feature engineering and deep learning approach. In a first network, the inputs (Coulomb matrices) and eigenvalues as well as singular values of the input are mapped to a multi-dimensional feature space, followed by a mapping from the feature space to the low-fidelity output. In a second network, the low-fidelity prediction and scalar features obtained from the input and its eigenvalues/singular values are mapped to the high fidelity output.

3.1. Feature enhanced deep learning for multi-fidelity electronic structure calculations: FEDMEC

We first introduce some basic terminology to aid the reader before presenting the architecture of the proposed network. A dense layer of a network contains a specified number k of nodes/neurons, into which are fed linear combinations of the l features x_i emerging from the preceding layer. These linear combinations (called activations a_i) involve unknown weights w_{ij} connecting feature x_i with node j. They undergo a nonlinear transformation via an activation function $f(\cdot)$ (usually the same for each activation) to produce outputs that are fed to the following layer. The entire operation and the output $\mathbf{o}$ can be written compactly as:

$$\mathbf{o} = \mathbf{f}(\mathbf{a}) = (f(a_1), \dots, f(a_k))^T; \quad \mathbf{a} = \mathbf{W}_1\mathbf{x} \tag{3}$$

in which the matrix $\mathbf{W}_1 \in \mathbb{R}^{k\times l}$ contains the weights (to be learned during training), $\mathbf{a} = (a_1, \dots, a_k)^T$ and $\mathbf{x} = (x_1, \dots, x_l)^T$. Since the transformation connects every feature x_i to to every neuron in the layer, the layer is termed dense or fully-connected. A common form of regularisation is to randomly set some of the outputs of the layer (dropout) by setting $\mathbf{f}(\mathbf{a}) \mapsto \text{diag}(r_1, \dots, r_k)\mathbf{f}(\mathbf{a})$, in which diag$(\cdot, \dots, \cdot)$ denotes a diagonal matrix and $r_j \in \{0, 1\}$, $j = 0, \dots, k$, are independent samples from a Bernoulli distribution. This operation is equivalent to removing those neurons j for which $r_j = 0$.

A convolutional layer acts in a quite different manner, and can involve a much smaller number of unknown parameters. The input to a convolutional layer is a multidimensional (dimension ≥ 1) array, e.g., $\mathbf{L} \in \mathbb{R}^{H\times W\times D}$ of size size $H \times W \times D$, with entries $L_{i,j,k}$. It is operated upon by a kernel (also called a filter or window) of smaller size, e.g., $\mathbf{K} \in \mathbb{R}^{F\times F\times D}$ with entries $K_{i,j,k}$ (which are unknown hyperparameters). The so-called channel dimension D must be the same as that of the input. The convolution operation $*$ is defined such that the m-th component of the result $\mathbf{A} = \mathbf{L} * \mathbf{K}$ is given by:

$$A_{l,m} = \sum_{i=1}^{H}\sum_{j=1}^{W}\sum_{p=1}^{D} K_{i,j,p} L_{l+i-1,m+j-1,p} \tag{4}$$

The kernel essentially scans an $F \times F \times D$ segment of the input and the convolutions operation above takes sums of element-wise products of the kernel and segment entries. The kernel moves through the image repeating this process in an ordered manner and the number of locations by which the kernel shifts between each scan is called the stride. Larger strides can be taken in order to produce smaller sized outputs from the convolution operation. In the case of (4), a stride of 1 is used. In order to increase or preserve the size of the convolution output in relation to the input, fibres of zeros can be spliced into the input, a process referred to as padding. Typically, multiple kernels are employed, which can be in stacked in multidimensional array, e.g., N kernels of size $F \times F \times D$, yielding $\mathbf{K} \in \mathbb{R}^{F\times F\times D\times N}$.

The convolution result $\mathbf{A}$ is passed through an activation function to produce an output $\mathbf{O} = \mathbf{f}(\mathbf{A})$. The output is frequently subjected to pooling (also called downsampling) as a form of regularization, in which a filter or kernel of a specified size scans segments of $\mathbf{O}$ to combine the entries in some manner, e.g., such as taking the maximum value of the entries (max pooling) or average their values (average pooling).

The basic architectures of the proposed networks are illustrated in Figure **??**. Precise details are provided in the next section when the results are presented. Three features are selected for the input to series of convolutional and dense layers to learn the selected low-fidelity output. To obtain the features, we first perform an eigendecomposition of the Coulomb matrix $C \in \mathbb{R}^{23\times 23}$, namely, we solve:

$$C\mathbf{v} = \lambda\mathbf{v} \tag{5}$$

for the eigenvector/eigenvalue pairs $\mathbf{v}_n, \lambda_n$, $n = 1, \ldots, 23$. In practise, the eigenpairs are obtained from a singular value decomposition (SVD) of C rather than the direct eigendecomposition, since the SVD is more stable numerically.

The eigenvalues are then passed through one or more dense layers (the final layer having m^2 neurons) to learn an $m \times m$ feature $\mathcal{F}_1$. Similarly, an $m \times m$ feature $\mathcal{F}_2$ is learned by passing the singular values $\sqrt{\lambda_n}$, $n = 1, \ldots, 23$ through one or more dense layers. The third $m \times m$ feature is learned by passing the Coulomb matrix C through one or more convolutional layers followed by a dense layer after flattening with m^2 neurons. Each convolutional layer is followed by a max pooling layer.

The three $m \times m$ features are concatenated into a $3 \times m \times m$ multi-dimensional array, which is fed through one or more convolutional layers (the first layer is followed by a max pooling layer) and a final layer with one neuron to predict the low fidelity output. The activation functions in all dense layers are the Parametric ReLU (PReLU) function. In a second network, the eigenvalues and separately the singular values are passed through one or more dense layers with a final layer containing a single neuron. The scalar features emerging from these layers can then be spliced with the low-fidelity data to form a 3-dimensional array, which is then passed through a series of dense layers. In all layers, the activation function is the PReLU. Both networks were trained with the ADAM algorithm [**?**].

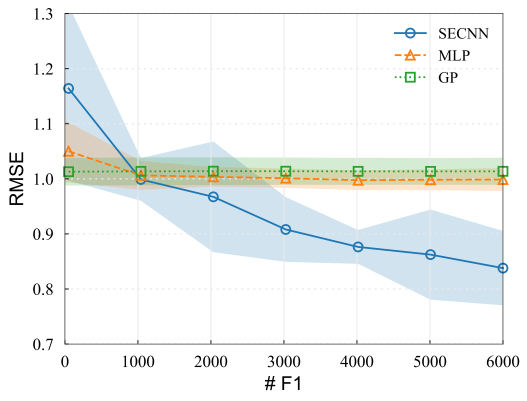
We name this method *Fidelity fusion for Electronic Structure Calculations Network* (FESC-Net). In section 4 we conduct simulations on the data set to demonstrate the superiority of FESC-Net over existing state-of-the-art methods. AR, NAR and ResGP are briefly described in the Appendix. The precise details (number of layers, neurons, kernels, strides, epochs, parameters associated with the learning algorithm) are provide below. Next we analyse the complexity of FESC-Net.

3.2. Complexity and Scalability

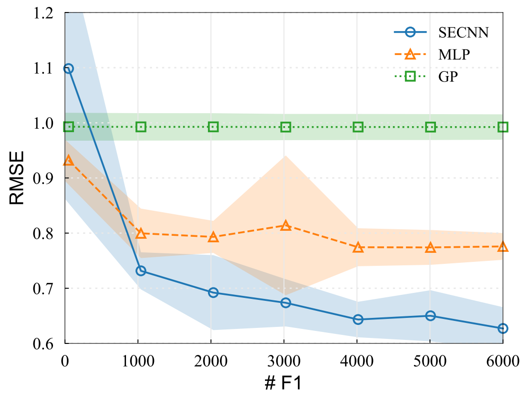
4. Experiments

We assess the performance of FESC-Net in three steps. We first illustrate the accuracy of the low-fidelity prediction, compared to standard regression methods. This step allows for tuning of the first network (number of layers, sizes of the layers, number of epochs, and other hyperparameters). We assess the performance of the second network given low-fidelity information, i.e., with the exact low-fidelity data at hand (as required in stochastic collocation) and compare to other multi-fidelity approaches. The second network is tuned during this step. Finally, we assess and compare the performance of the combined networks forming FESC-Net to predict high-fidelity data and compare to other approaches.

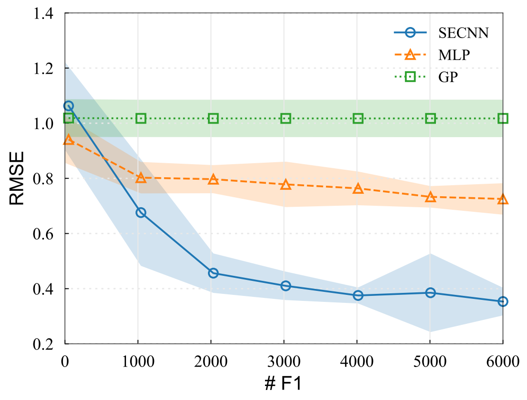
4.1. Predictions of low-fidelity results with deep learning



(a)

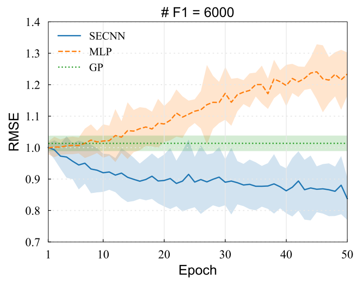


(b)

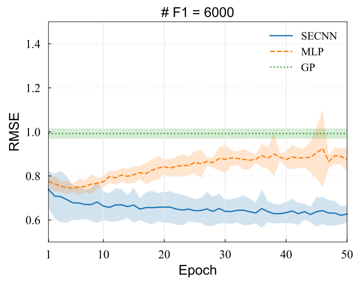


(c)

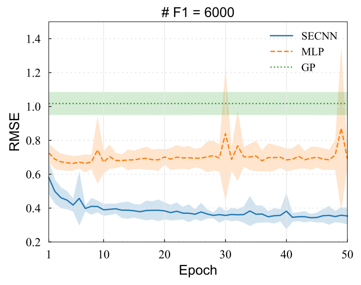
Figure 1: RMSE values for the low-fidelity predictions using SECNN, an MLP and a GP, based on different numbers of low-fidelity training points. (a) HOMO, (b) LUMO, (c) polarizability.



(a)

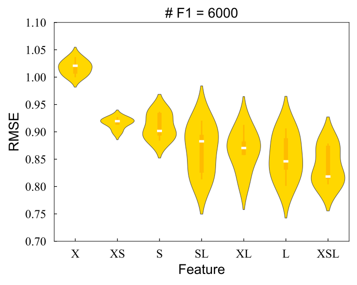


(b)

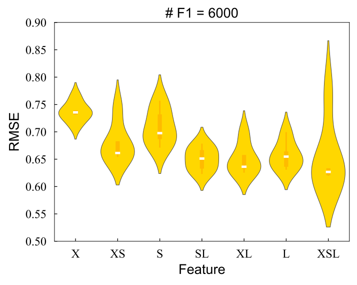


(c)

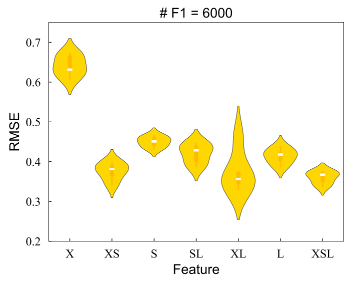
Figure 2: Experiment results of low-fidelity Predictions. (a) Comparisons of the root mean square error (RMSE) for FESC-Net, MLP, and GP, with different amount of F1 training points. (b) Comparisons of RMSE at different training epochs for FESC-Net, MLP, and GP. (c) Comparisons of RMSE for the put forward model using different combinations of features as input, to approximate F1 data. Here, X denotes the origin input data; S denotes the singular value extracted from X. L denotes the eigenvalues extracted from X. (e.g. SL represents that the model takes both singular value and eigenvalues as input data)



(a)



(b)



(c)

Figure 3: Experiment results of low-fidelity Predictions. (a) Comparisons of the root mean square error (RMSE) for FESC-Net, MLP, and GP, with different amount of F1 training points. (b) Comparisons of RMSE at different training epochs for FESC-Net, MLP, and GP. (c) Comparisons of RMSE for the put forward model using different combinations of features as input, to approximate F1 data. Here, X denotes the origin input data; S denotes the singular value extracted from X. L denotes the eigenvalues extracted from X. (e.g. SL represents that the model takes both singular value and eigenvalues as input data)

We first approximate the relationship between the input matrix and the low-fidelity data using the first network and compare it to a multi-layer perceptron (MLP) and a Gaussian process (GP) model. We used three outputs with different combinations to test the methods thoroughly. The three outputs are: (i) HOMO (low fidelity: Zindo; high fidelity: PBE0); (ii) LUMO (low fidelity: Zindo; high fidelity: GW); (iii) polarizability (low fidelity: SCS; high fidelity: PBE).

A range of architectures were tested to find the best performing network, which is now described in detail. The eigenvalues and singular values were each passed through two dense layers, to map them to 6×6 features. In both cases, there are 23 input neurons, a hidden layer with 16 neurons and layers of size 36 containing the features. The Coulomb matrix is passed through two convolutional layers: the first with 16 kernels of size 2×2, stride of 1, and a padding of 1 and the second layer with 1 kernel of size 3×3, a stride of 1, and a padding of 1. The spliced $3 \times 6 \times 6$ features are passed through 2 convolutional layers. The first layer has 16 kernels of size $3 \times 3 \times 3$ with a stride and padding of 1. In the second layer there is one kernel of size 3×3, a stride of 1 and no padding. 50 epochs were found to be sufficient for training with a fixed learning rate of 0.001.

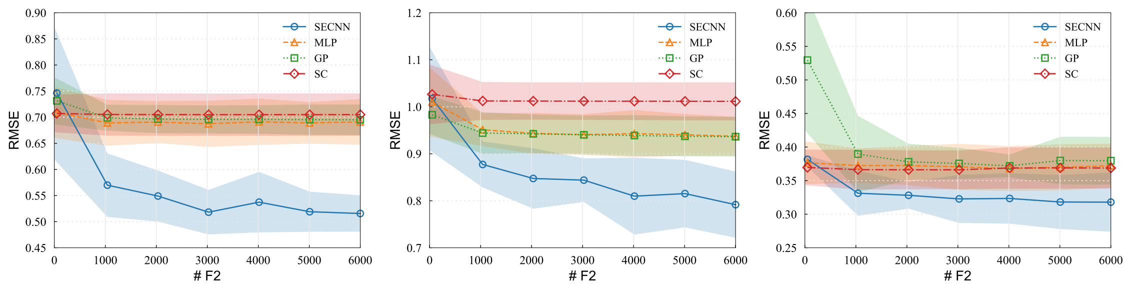
For the GP method, an ARD kernel was used together with a zero mean function (the data was centred) taking as input the vectorised Coulomb matrix. The best performing MLP has five layers with an input layer of size $529 = 23 \times 23$, and subsequent layers of size 100, 25, 4 and 1. The number of epochs was 1 for the HOMO data, 3 for the LUMO and 5 for the polarizability, since it suffered from overfitting beyond these numbers, and the learning rate was fixed at 0.001. For each method, we conducted five tests with random shuffling of data and show the range of errors obtained.

The number of low-fidelity training points was varied with a fixed number of 1081 test points. As can be seen in Fig. 12(a) for the HOMO data, the FESC-Net method outperforms the MLP and GP model in terms of the RMSE when the number of low-fidelity (F1) data points exceeds 1000. For the LUMO and polarizability results in Figs. 12(b),(c), the FESC-Net RMSE is lowest at a smaller number of training points, especially in the latter case in which it is the lowest for ca. 500 training points and higher. With an increasing number of F1 data points, there is a continuous decline in the RMSE, whereas the other methods fail to improve in the case of the HOMO data. For the LUMO and polarizability, the MLP performance also improves, although not as dramatically as that of FESC-Net. At 3000 training points, the RMSE for FESC-Net is ca. 10 %, 17% and 50% lower for the HOMO, LUMO and polarizability compared to the MLP, which is the second most accurate method, by a considerable margin compared to the GP in the last two cases.

For all three outputs, Figs. 12(b)-(c) shows the evolutions of the RMSE against the number of epochs for the FESC-Net and MLP (the GP value will not change since it uses the training data only once in a maximum likelihood estimate). In all cases, the MLP suffers from overfitting beyond a low number of epochs, whereas FESC-Net continues to improve, which demonstrates that the FESC-Net is less prone corruption by noise in the data than the MLP.

To demonstrate the significance of each added feature, an ablation study was conducted, with 6000 F1 training points. In terms of the specific network structure, we only need to change the number of channels after feature concatenation if the number of features is reduced. There are 8 different combinations of the 3 features. For each combination, the model was run 5 times with training data selected randomly to ensure the robustness of the results. The RMSE is plotted in Figs. 12(c)-(c), from which we see a decline in the median RMSE as the number of features is increased. In the case of the polarisability, the median is marginally lower using only the eigenvalues and Coulomb matrix, but the spread of errors is greater.

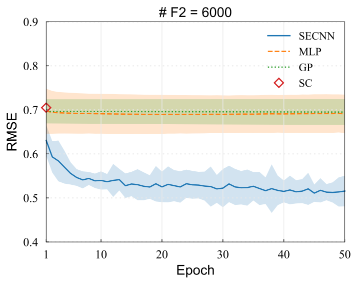
4.2. High-fidelity predictions with low-fidelity Information



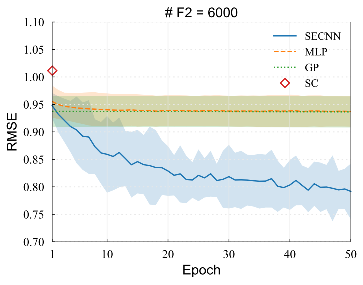
(a) (b) (c)

Figure 4: Experiment results of high-fidelity Predictions. (b) Comparisons of the RMSE for FESC-Net, MLP, GP and SC, with different amount of F2 training points. (b) Comparisons of the RMSE at different training epochs for FESC-Net, MLP, GP, SC. (b) Comparisons of RMSE for the put forward model using different combinations of features as input, to approximate F2 data. Here, S denotes the singular value extracted from origin input data (X). L denotes the eigenvalues extracted from X. Yl denotes the low-fidelity(F1) data.

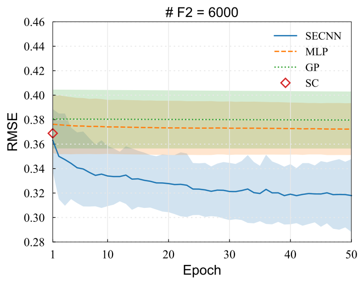
To predict the high-fidelity output given the low-fidelity result, the second network is employed as described



(a)

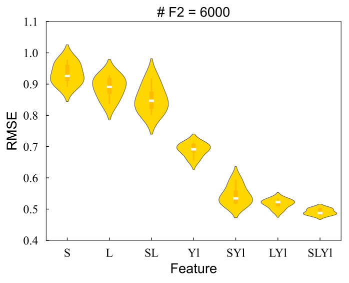


(b)

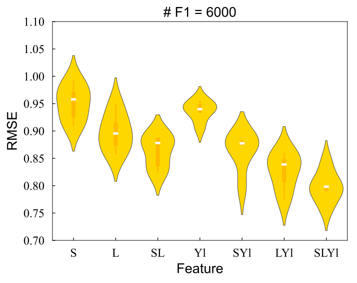


(c)

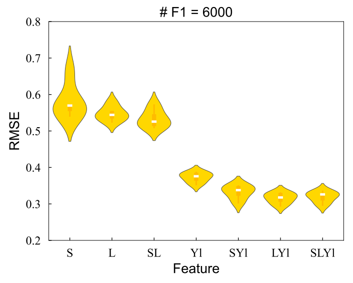
Figure 5: Experiment results of high-fidelity Predictions. (b) Comparisons of the RMSE for FESC-Net, MLP, GP and SC, with different amount of F2 training points. (b) Comparisons of the RMSE at different training epochs for FESC-Net, MLP, GP, SC. (b) Comparisons of RMSE for the put forward model using different combinations of features as input, to approximate F2 data. Here, S denotes the singular value extracted from origin input data (X). L denotes the eigenvalues extracted from X. Yl denotes the low-fidelity(F1) data.



(a)



(b)



(c)

Figure 6: Experiment results of high-fidelity Predictions. (b) Comparisons of the RMSE for FESC-Net, MLP, GP and SC, with different amount of F2 training points. (b) Comparisons of the RMSE at different training epochs for FESC-Net, MLP, GP, SC. (b) Comparisons of RMSE for the put forward model using different combinations of features as input, to approximate F2 data. Here, S denotes the singular value extracted from origin input data (X). L denotes the eigenvalues extracted from X. Yl denotes the low-fidelity(F1) data.

earlier. Both the singular values and eigenvalues are passed through three dense layers of sizes 10, 5 and 1 to obtain scalar features, which are combined with the low-fidelity data point. The concatenated input is passed through four fully-connected layers of sized 20, 20 and 10, with an output layer of size 1. The results are compared with an MLP, a GP model and stochastic allocation (SC). The MLP consisted of four fully connected layers of sizes 10, 20, 10, and 1, with an input layer of size 1, into which the low-fidelity output is fed. Both networks were run with 50 epochs and a learning rate of 0.001. The GP model has the same structure as in the previous experiment but with a concatenated Coulomb matrix and low-fidelity output as the input. For each method, we conducted five tests with random shuffling of data and averaged the errors.

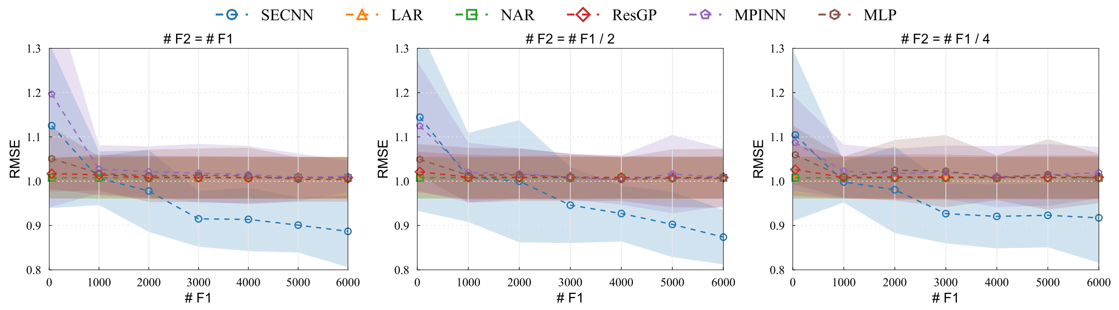
We tested all methods with different numbers of high-fidelity data points (equal to the number of low-fidelity data points). The number of test data points is fixed at 1081. As can be seen in Figs. 6(a)-(c), the proposed network outperforms all other methods in terms of the RMSE when the number of training data points exceeds around 200, with a continual decline as the number of data points is increased. The shaded regions show the spread of errors over the 5 tests. The accuracy gains decline from HOMO to LUMO to polarizability, from around ca. 25 % to ca. 15 % for more than 3000 training points. Figs. 6(a)-(c) show the progress during training (number of epochs) for all methods. The MLP reaches its minimum error after only a small number of epochs, while the proposed network continues to improve.

As with the previous network, an ablation experiment was conducted, with 6000 low-fidelity and high-fidelity training points and 1081 test points. For each of the eight combinations of features we ran the model 5 times and the results are plotted in Figs. 6(c)-(c), from which we see a decline in the RMSE as the number of features is increased, for the HOMO and LUMO, with a less discernible pattern for the polarisability. This demonstrates the value of including the additional features.

4.3. Pure Data-driven High-Fidelity Predictions

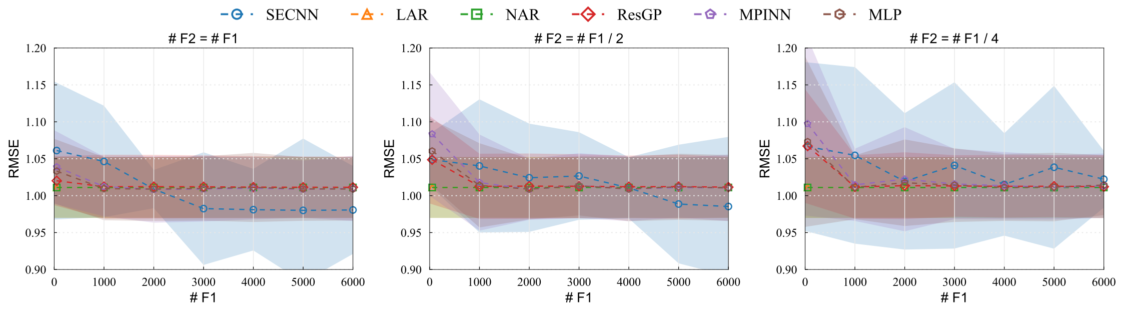
In this section we assess the performance of the full FESC-Net model, and compare it to an MLP, AR, NAR and ResGP. For AR, NAR, and ResGP, we used our own Emukit toolkit (source code:https://github.com/EmuKit/emukit), with the inputs taken to be the vectorised Coulomb matrix. The MLP methods uses the two MLPs described in the previous two subsectionFor each method, we conducted five tests with random shuffling of data.

We first set the number of low and high fidelity training points equal and vary this number, with 1081 test points. The performance of each method can be seen in Fig 13(a), which shows a clearly lower RMSE for FESC-Net compared to the other methods when there are more than 1000 training points. For lower numbers of training points, the GP based methods are superior to both network models. This is consistent with the general rule that network models require large data sets to work well. As the number of training points increases, FESC-Net continues to improve, whereas the other methods show a negligible decline in the RMSE. At 3000 low-fidelity training points, FEMMEC is approximately 10% more accurate than all other methods.



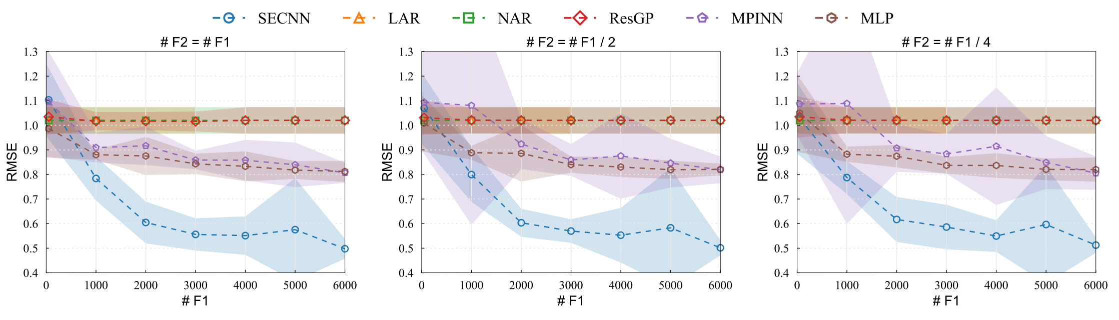
(a) (b) (c)

Figure 7: Experiment results of pure data-drive high-fidelity predictions. (a) Comparisons of the RMSE for FESC-Net, simple MLP and 4 state-of-the-art methods with different amount of F1 data, and the number of F1 data is twice that of F2 data. (b) Comparisons of the RMSE for FESC-Net, simple MLP and 4 state-of-the-art methods with different amount of F1 data, and the number of F1 data is 2 times that of F2 data. (c) Comparisons of the RMSE for FESC-Net, simple MLP and 4 state-of-the-art methods with different amount of F1 data, and the number of F1 data is 3 times that of F2 data. (d) Comparisons of the RMSE for FESC-Net, simple MLP and 4 state-of-the-art methods with different amount of F1 data, and the number of F1 data is 4 times that of F2 data.



(a) (b) (c)

Figure 8: Experiment results of pure data-drive high-fidelity predictions. (a) Comparisons of the RMSE for FESC-Net, simple MLP and 4 state-of-the-art methods with different amount of F1 data, and the number of F1 data is twice that of F2 data. (b) Comparisons of the RMSE for FESC-Net, simple MLP and 4 state-of-the-art methods with different amount of F1 data, and the number of F1 data is 2 times that of F2 data. (c) Comparisons of the RMSE for FESC-Net, simple MLP and 4 state-of-the-art methods with different amount of F1 data, and the number of F1 data is 3 times that of F2 data. (d) Comparisons of the RMSE for FESC-Net, simple MLP and 4 state-of-the-art methods with different amount of F1 data, and the number of F1 data is 4 times that of F2 data.

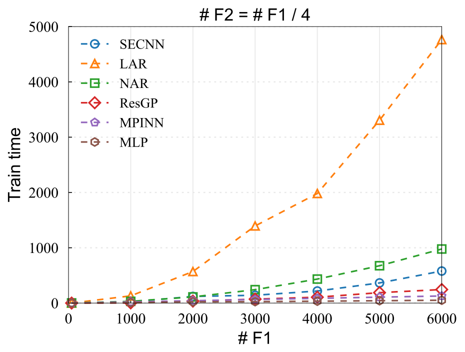


(a) (b) (c)

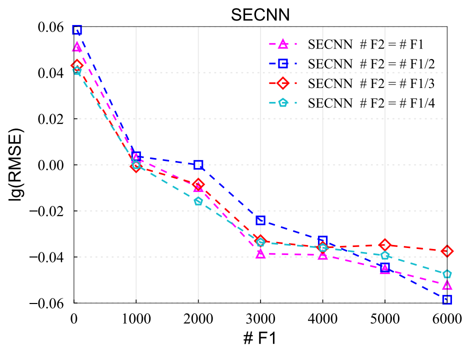
Figure 9: Experiment results of pure data-drive high-fidelity predictions. (a) Comparisons of the RMSE for FESC-Net, simple MLP and 4 state-of-the-art methods with different amount of F1 data, and the number of F1 data is twice that of F2 data. (b) Comparisons of the RMSE for FESC-Net, simple MLP and 4 state-of-the-art methods with different amount of F1 data, and the number of F1 data is 2 times that of F2 data. (c) Comparisons of the RMSE for FESC-Net, simple MLP and 4 state-of-the-art methods with different amount of F1 data, and the number of F1 data is 3 times that of F2 data. (d) Comparisons of the RMSE for FESC-Net, simple MLP and 4 state-of-the-art methods with different amount of F1 data, and the number of F1 data is 4 times that of F2 data.

In practise, the advantage of multi-fidelity methods is that fewer high fidelity results are required, compared to the low-fidelity results. We examined, therefore, the effect of the ratio of high- to low-fidelity training points. 13(b)(c)(d) shows the RMSE for a ratio of 1:2, and 1:4, respectively. The number of test points is1081. These results show that FESC-Net continues to be superior for low-fidelity training point number over 1000, with a similar pattern in the decline in the RMSE as the number of low-fidelity training points increases.

We compare the performance of FESC-Net trained with different amount of F1 and F2 data in Fig 10(b). The RMSE performance is rather steady though the amount of F2 train data is increasing. In addition, the training time of our method is significantly shorter than that of LAR, and NAR, which is plotted in Fig 10(a). This means lower computational costs. When compared to other methods, our method consumes similar running time but achieves better results on RMSE.



(a)



(b)

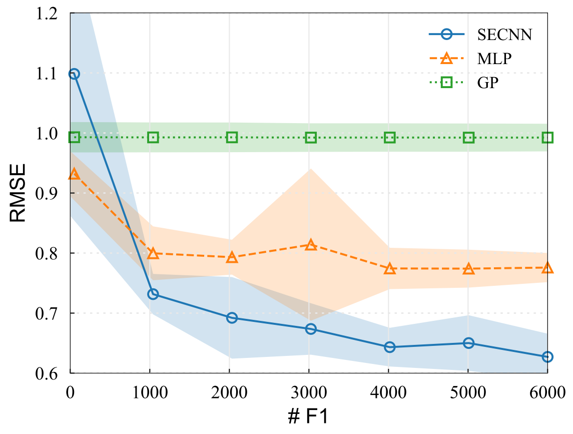
Figure 10: Experiment results of pure data-drive high-fidelity predictions. (a) Comparisons of the training time for FESC-Net, simple MLP and 4 state-of-the-art methods with different amount of F1 data, and the number of F1 data is 4 times that of F2 data. (b) Comparisons of the RMSE for FESC-Net, simple MLP and 4 state-of-the-art methods with different amount of F1 data, and the number of F1 data is 4 times that of F2 data.

4.4. Other Data Sets

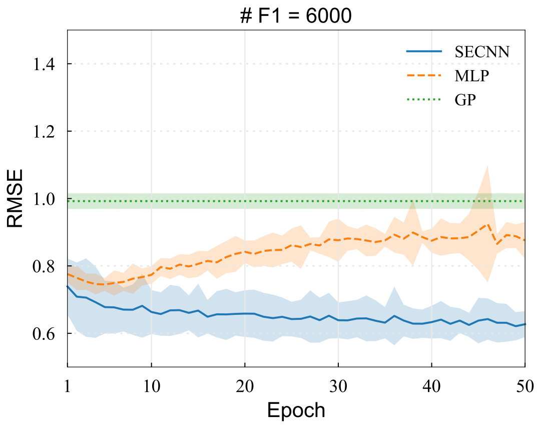
We changed the data set of high and low fidelity and conducted the above experiment again. The experimental results are as follows:

4.4.1. low-fidelity:Zindo lumo high-fidelity: Gw homo

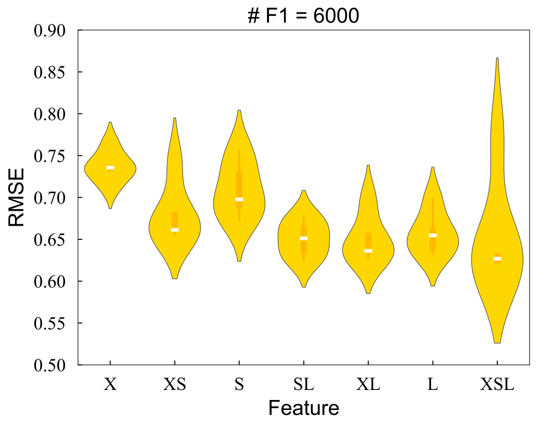
Predictions of low-fidelity results with deep learning.



(a)



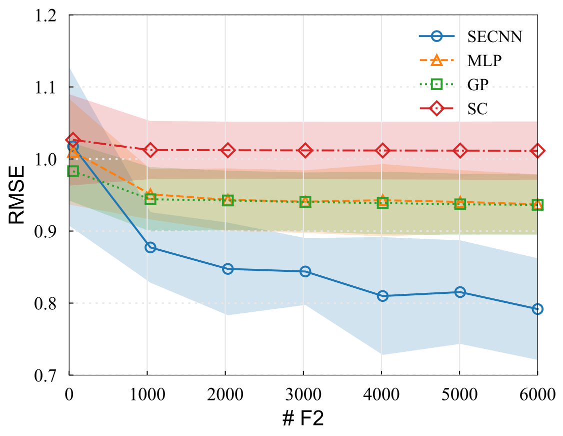
(b)



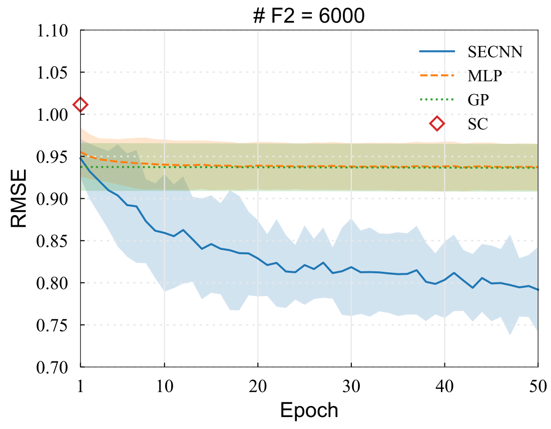
(c)

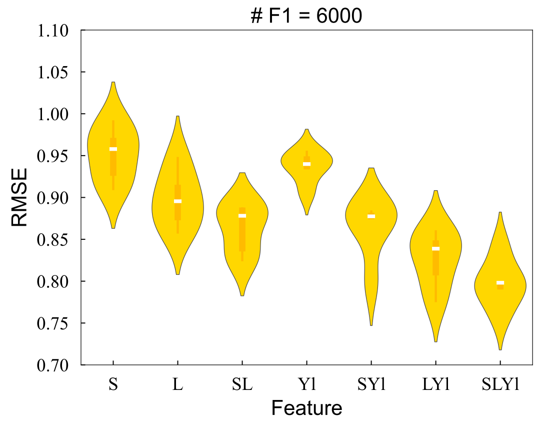
Figure 11: Experiment results of low-fidelity Predictions. (a) Comparisons of the root mean square error (RMSE) for FESC-Net, MLP, and GP, with different amount of F1 training points. (b) Comparisons of RMSE at different training epochs for FESC-Net, MLP, and GP. (c) Comparisons of RMSE for the put forward model using different combinations of features as input, to approximate F1 data. Here, X denotes the origin input data; S denotes the singular value extracted from X. L denotes the eigenvalues extracted from X. (e.g. SL represents that the model takes both singular value and eigenvalues as input data)

High-fidelity predictions with low-fidelity Information.



(a)





(b)

(c)

Figure 12: Experiment results of low-fidelity Predictions. (a) Comparisons of the root mean square error (RMSE) for FESC-Net, MLP, and GP, with different amount of F1 training points. (b) Comparisons of RMSE at different training epochs for FESC-Net, MLP, and GP. (c) Comparisons of RMSE for the put forward model using different combinations of features as input, to approximate F1 data. Here, X denotes the origin input data; S denotes the singular value extracted from X. L denotes the eigenvalues extracted from X. (e.g. SL represents that the model takes both singular value and eigenvalues as input data)

Pure data-driven high-Fidelity predictions.

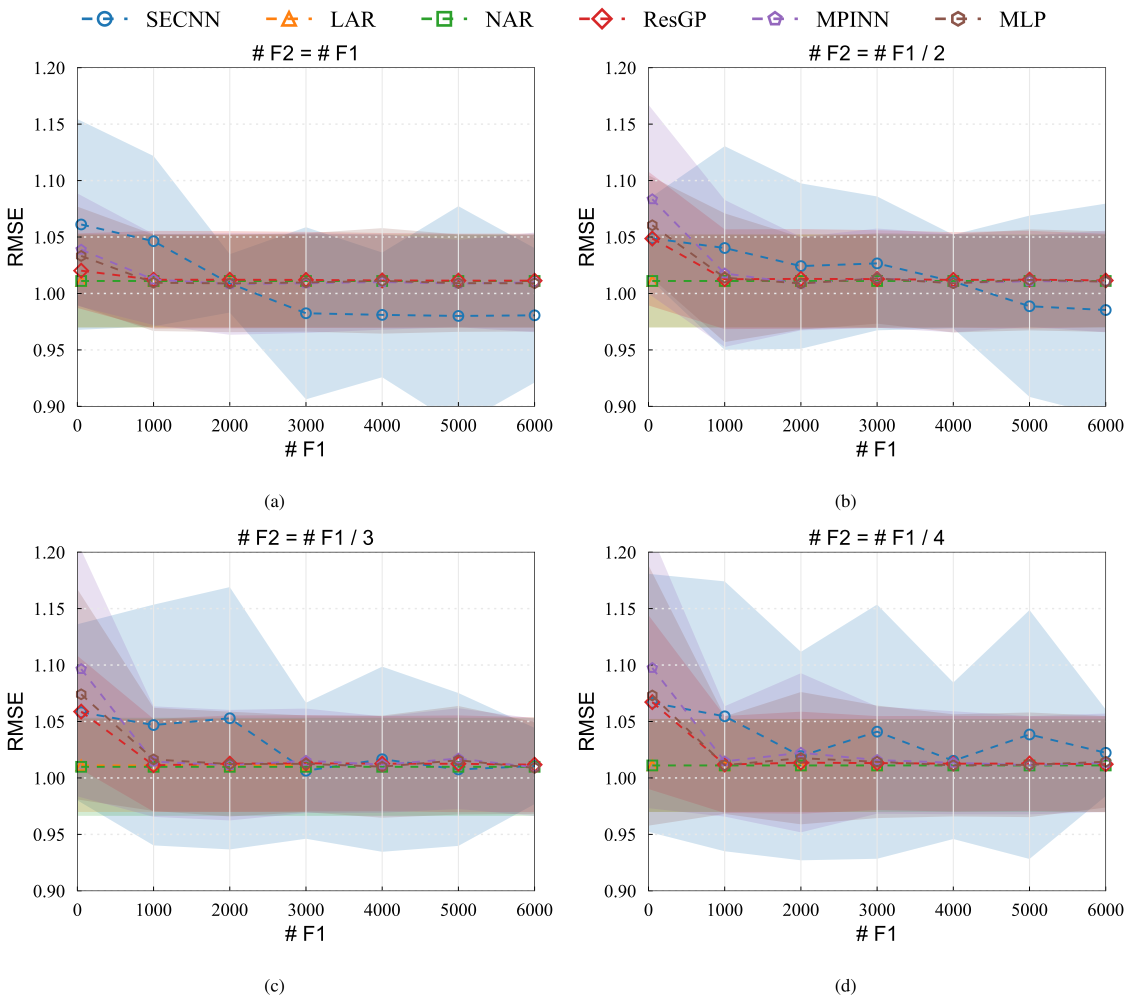


Figure 13: Experiment results of pure data-drive high-fidelity predictions. (a) Comparisons of the RMSE for FESC-Net, simple MLP and 4 state-of-the-art methods with different amount of F1 data, and the number of F1 data is twice that of F2 data. (b) Comparisons of the RMSE for FESC-Net, simple MLP and 4 state-of-the-art methods with different amount of F1 data, and the number of F1 data is 2 times that of F2 data. (c) Comparisons of the RMSE for FESC-Net, simple MLP and 4 state-of-the-art methods with different amount of F1 data, and the number of F1 data is 3 times that of F2 data. (d) Comparisons of the RMSE for FESC-Net, simple MLP and 4 state-of-the-art methods with different amount of F1 data, and the number of F1 data is 4 times that of F2 data.

Supplementary Material

A. Univariate Gaussian process models

A GP model is based on a prior distrubution $y(\mathbf{x}) \sim \mathcal{GP}(y(\mathbf{x}) \mid m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}|\boldsymbol{\theta}))$ over an unknown function $y(\mathbf{x})$, with mean and covariance functions $m(\mathbf{x})$ and $k(\mathbf{x}, \mathbf{x}|\boldsymbol{\theta})$, respectively, containing hyperparameters $\boldsymbol{\theta}$. Here, y is a scalar function and $\mathbf{x}$ is any vector-valued input. Any finite set of function values on $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]^T$, namely $\mathbf{Y} = [y(\mathbf{x}_1), \dots, y(\mathbf{x}_N)]^T$, follows a multivariate Gaussian distribution: $\mathbf{Y} \sim \mathcal{N}(\mathbf{Y} \mid \mathbf{m}, \mathbf{K} + \sigma^2\mathbf{I})$, in which $\mathbf{m} = [m(\mathbf{x}_1), \dots, m(\mathbf{x}_N)]^T$ is the mean vector and $\mathbf{K} = [k(\mathbf{x}_i, \mathbf{x}_j|\boldsymbol{\theta})]_{ij}$, $i, j = 1, \dots, N$, is the covariance matrix.

The mean function is usually considered to be identically zero, after centering the data. The term $\sigma^2\mathbf{I}$ accounts for numerical error with variance σ^2 or model inadequacy. Data from a deterministic simulator is normally considered noise free but $\sigma^2\mathbf{I}$ is often included as a regularization term such that the inversion of $(\mathbf{K} + \sigma^2\mathbf{I})^{-1}$ (see below) is not ill-conditioned [**?** **?**]. Choosing the right kernel function for a specific application is non-trivial. When there is no prior knowledge to guide the choice, the automatic relevance determinant (ARD) kernel:[**?**]

$$k(\mathbf{x}_i, \mathbf{x}_j|\boldsymbol{\theta}) = \theta_0 \exp\left(-[\mathbf{x}_i - \mathbf{x}_j]\mathrm{diag}(\theta_1, \dots, \theta_l)[\mathbf{x}_i - \mathbf{x}_j]^T\right), \tag{A-1}$$

with $\boldsymbol{\theta} = [\theta_0, \dots, \theta_1]^T$, is often used. The ARD kernel can freely capture the influence of each individual input (coordinate of $\mathbf{x}$) on the output. The hyperparameters $\{\tau, \boldsymbol{\theta}\}$ can be estimated by maximizing the log-marginal likelihood:

$$\mathcal{L} = \frac{1}{2}\ln|\mathbf{K} + \sigma^2\mathbf{I}| - \frac{1}{2}\mathbf{Y}^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{Y} - \frac{N}{2}\ln(2\pi). \tag{A-2}$$

The main computational cost is the inversion of $\mathbf{K}$, which is $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$ for time and space complexity, respectively. We can derive the predictive posterior as a function of $\mathbf{x}$ using conditioning rules for Gaussian distributions [**?**]:

$$\begin{aligned} &y(\mathbf{x}) \sim \mathcal{N}\left(y(\mathbf{x}) \mid \mu(\mathbf{x}), v(\mathbf{x})\right) \\ &\mu(\mathbf{x}) = \mathbf{k}(\mathbf{x})^T(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{Y}, \\ &v(\mathbf{x}) = k(\mathbf{x},\mathbf{x}|\boldsymbol{\theta}) - \mathbf{k}(\mathbf{x})^T(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{k}(\mathbf{x}), \end{aligned} \tag{A-3}$$

where $\mathbf{k}(\mathbf{x}) = [k(\mathbf{x},\mathbf{x}_1),\ldots,k(\mathbf{x},\mathbf{x}_N)]^T$ is the vector of covariances between $y(\mathbf{x})$ and the training data $\mathbf{Y}$.

B. Autoregression models based on GPs

Consider a multi-fidelity problem for univariate data, that is $\{\mathbf{X}^f, \mathbf{Y}^f\}_{f=1}^F$, where $\mathbf{Y}^f = [y^f(\mathbf{x}_1),\ldots,y^f(\mathbf{x}_{N_f})]^T \in \mathbb{R}^{N_f}$ is a collection of samples at fidelity f. The condition $\mathbf{X}^f \subset \mathbf{X}^{f-1}$ is required. The general autoregressive formulation for multi-fidelity data is:

$$y^f(\mathbf{x}) = g^f\left(y^{f-1}(\mathbf{x})\right) + \epsilon^f(\mathbf{x}), \tag{B-1}$$

where $g^f\left(y^{f-1}(\mathbf{x})\right)$ is an arbitrary function that maps the low-fidelity results to the high-fidelity results and $\epsilon^f(\mathbf{x})$ (assumed to be Gaussian) is an error term. If we assume a simple linear form for the mapping, i.e., $g^f(y^{f-1}(\mathbf{x})) = c \cdot y^{f-1}(\mathbf{x})$ for some constant c, we recover the classic autoregressive model put forth by **?**]. **?**] proposed the nonlinear autoregression (NAR) formulation by placing a GP prior over the function g^f. To further enhance the model at each fidelity level, Eq. (B-1) is modified as follows:

$$y^f(\mathbf{x}) = g^f\left(\mathbf{x}, y_*^{f-1}(\mathbf{x})\right), \tag{B-2}$$

in which $\epsilon^f(\mathbf{x})$ has been absorbed into g^f, and $y_*^{f-1}(\mathbf{x})$ is the true function value for $\mathbf{x}$ at fidelity $f-1$. This complicated model structure is then simplified using a separable covariance k^f for the GP over $g^f(\mathbf{x}, y_*^{f-1}(\mathbf{x}))$:

$$k^f\left(\mathbf{x},\mathbf{x}', y_*^{f-1}(\mathbf{x}), y_*^{f-1}(\mathbf{x}')\right) = k_\xi^f(\mathbf{x},\mathbf{x}') \cdot k_y^f\left(y_*^{f-1}(\mathbf{x}), y_*^{f-1}(\mathbf{x}')\right), \tag{B-3}$$

where k_ξ^f and k_y^f are valid covariance functions, each with their own hyperparameters. The low fidelity resolution then becomes the input for the high-fidelity GP model, which leads to a concatenating GP structure known as the *deep GP* [**? ?**]. As suggested in [**? ?**], since each fidelity solution is directly treated as the observable latent variable of a deep GP, the model can be trained efficiently by training each fidelity level GP fully independently, as in Eq. (A-2), with the concatenated inputs. However, the forward model, i.e., the predictive posterior at each fidelity f for a test input $\mathbf{x}_*$ is given by:

$$y^f(\mathbf{x}_*) = \int g^f\left(\mathbf{x}, y_*^{f-1}(\mathbf{x})\right) dy_*^{f-1}(\mathbf{x}), \tag{B-4}$$

where $g^f(\mathbf{x}, y_*^{f-1}(\mathbf{x}))$ is a Gaussian predictive posterior with mean and variance given by expressions similar to those in Eq. (A-3). This posterior in general does not have a closed-form solution and requires expensive sampling methods to derive an approximation.

C. Residual Gaussian process model (ResGP)

Again, data $\{\mathbf{X}^f, \mathbf{Y}^f\}_{f=1}^F$, where $\mathbf{Y}^f = [y^f(\mathbf{x}_1), \ldots, y^f(\mathbf{x}_{N_f})]^T \in \mathbb{R}^{N_f}$, is available and $\mathbf{X}^f \subset \mathbf{X}^{f-1}$. Rather than employing the concatenating structure of NAR, in ResGP the high-fidelity GP is decomposed as:

$$y^f(\mathbf{x}) = r^1(\mathbf{x}) + \cdots + r^f(\mathbf{x}) = \sum_{k=1}^{f} r^k(\mathbf{x}), \tag{C-1}$$

for residual functions $r^k(\mathbf{x}) = y^k(\mathbf{x}) - y^{k-1}(\mathbf{x})$ for fidelities $k = 2, \ldots, F$ and with $r^1(\mathbf{x}) = y^1(\mathbf{x})$. An independent GP prior $r^f(\mathbf{x}) \sim \mathcal{GP}(r^f(\mathbf{x}) \mid 0, k^f(\mathbf{x}, \mathbf{x}'|\boldsymbol{\theta}^f) + (\sigma^f)^2)$ is placed over each residual function, leading to:

$$y^F(\mathbf{x}) \sim \mathcal{GP}\left(y^F(\mathbf{x}) \mid 0, \sum_{f=1}^{F} k^f(\mathbf{x}, \mathbf{x}'|\boldsymbol{\theta}^f) + (\sigma^f)^2\right), \tag{C-2}$$

by virtue of the independence assumption. At the lowest fidelity, inputs $\mathbf{X}^1$ and outputs $\mathbf{Y}^1$ along with Eq. (A-2) are used to learn $\{\sigma^1, \boldsymbol{\theta}^1\}$ and therefore obtain the predictive posterior for $r^1(\mathbf{x})$ as in Eq. (A-3). The inputs $\mathbf{X}^2$ and outputs $\mathbf{R}^2 := \mathbf{Y}^2 - \mathbf{Y}^1_{\mathbf{e}_2 \cap \mathbf{e}_1}$ (in which $\mathbf{e}_2 \cap \mathbf{e}_1$ selects those outputs at fidelity 1 corresponding to the inputs $\mathbf{X}^2 \subset \mathbf{X}^1$) are then used to estimate $\{\sigma^2, \boldsymbol{\theta}^2\}$, and obtain the predictive posterior for $r^2(\mathbf{x})$. The procedure can repeated up to fidelity F, and the result can be written compactly as:

$$\begin{aligned}
&y^F(\mathbf{x}) \sim \mathcal{N}\left(y^F(\mathbf{x}) \mid \mu^F(\mathbf{x}), v^F(\mathbf{x})\right) \\
&\mu^F(\mathbf{x}) = \sum_{f=1}^{F} \mathbf{k}^f(\mathbf{x})^T(\mathbf{K}^f + (\sigma^f)^2\mathbf{I})^{-1}\mathbf{R}^f, \\
&v^F(\mathbf{x}) = \sum_{f=1}^{F} [k^f(\mathbf{x},\mathbf{x}|\boldsymbol{\theta}^f) - (\mathbf{k}^f(\mathbf{x}))^T(\mathbf{K}^f + (\sigma^f)^2\mathbf{I})^{-1}\mathbf{k}^f(\mathbf{x})], \\
&\mathbf{k}^f(\mathbf{x}) = [k^f(\mathbf{x},\mathbf{x}_1|\boldsymbol{\theta}^f), \ldots, k^f(\mathbf{x},\mathbf{x}_{N_f}|\boldsymbol{\theta}^f)]^T,
\end{aligned} \tag{C-3}$$

Here $\mathbf{R}^f := \mathbf{Y}^f - \mathbf{Y}^{f-1}_{\mathbf{e}_f \cap \mathbf{e}_{f-1}}$ (in which $\mathbf{e}_f \cap \mathbf{e}_{f-1}$ selects those outputs at fidelity $f-1$ corresponding to the inputs $\mathbf{X}^f \subset \mathbf{X}^{f-1}$), $[\mathbf{K}^f]_{ij} = k^f(\mathbf{x}_i, \mathbf{x}_j|\boldsymbol{\theta}^f)$ is the f fidelity covariance matrix and $\mathbf{k}^f(\mathbf{x})$ is a vector of covariances between $r^f(\mathbf{x})$ and the data $\mathbf{R}^f$.

D. Stochastic collocation