Multi-Resolution Active Learning of Fourier Neural Operators

Shibo Li SHIBO@CS.UTAH.EDU
Kahlert School of Computing
University of Utah

Xin Yu XINY@CS.UTAH.EDU
Kahlert School of Computing
University of Utah

Wei Xing W.XING@SHEFFIELD.AC.UK
School of Mathematics and Statistics
University of Sheffield

Robert M. Kirby KIRBY@CS.UTAH.EDU
Kahlert School of Computing, Scientific Computing and Imaging Institute
University of Utah

Akil Narayan AKIL@SCI.UTAH.EDU
Department of Mathematics, Scientific Computing and Imaging Institute
University of Utah

Shandian Zhe ZHE@CS.UTAH.EDU
Kahlert School of Computing
University of Utah

Abstract

Fourier Neural Operator (FNO) is a popular operator learning framework, which not only achieves the state-of-the-art performance in many tasks, but also is highly efficient in training and prediction. However, collecting training data for the FNO is a costly bottleneck in practice, because it often demands expensive physical simulations. To overcome this problem, we propose Multi-Resolution Active learning of FNO (MRA-FNO), which can dynamically select the input functions and resolutions to lower the data cost as much as possible while optimizing the learning efficiency. Specifically, we propose a probabilistic multi-resolution FNO and use ensemble Monte-Carlo to develop an effective posterior inference algorithm. To conduct active learning, we maximize a utility-cost ratio as the acquisition function to acquire new examples and resolutions at each step. We use moment matching and the matrix determinant lemma to enable tractable, efficient utility computation. Furthermore, we develop a cost annealing framework to avoid over-penalizing high-resolution queries at the early stage. The over-penalization is severe when the cost difference is significant between the resolutions, which renders active learning often stuck at low-resolution queries and inferior performance. Our method overcomes this problem and applies to general multi-fidelity active learning and optimization problems. We have shown the advantage of our method in several benchmark operator learning tasks.

1. Introduction

Operator learning is emerging as an important topic in scientific machine learning. It intends to estimate function-to-function mappings and can serve as a useful surrogate model for many

physical simulation related applications, such as weather forecast (Pathak et al., 2022), control (Bhan et al., 2023), engineering design (Liu et al., 2023) and inverse problems (Kaltenbach et al., 2022). One popular and representative approach is the Fourier neural operator (FNO) (Li et al., 2020d), which uses fast Fourier transform (FFT) to fulfill a linear transform in the functional space. The Fourier layers in the FNO actually define a global spectral convolution. The FNO not only shows state-of-the-art performance in many tasks, but also is highly efficient in training and prediction.

Despite the advantages, collecting training data for the FNO can be a severe bottleneck in practice because it often requires many physical simulations (*e.g.*, running numerical solvers), which is known to be computationally expensive. To reduce the cost, one can consider multi-resolution data. The low-resolution data is cheap to obtain (typically computed with rough meshes) but the provided output function samples are quite inaccurate (large bias). On the contrary, high-resolution data offers accurate output function samples, yet is much more costly to generate (from dense meshes). Although with substantial difference in quality, the low and high resolution examples share the same underlying physics and are strongly correlated. Hence, one can still expect using multi-resolution data to well train the FNO while reducing the data cost.

However, blindly collecting examples at different resolutions is hardly optimal in both cost saving and learning efficiency. To reduce the data cost to the greatest extent while optimizing the learning efficiency, we propose MRA-FNO, a novel multi-resolution active learning method for FNO, which can dynamically select the best input function and resolution each time, at which to generate new examples. The major contributions of our work are summarized as follows.

- **Probabilistic Multi-Resolution FNO.** We first extend the FNO to integrate multi-resolution training data. To capture the influence of the resolution choice on the predictive distribution, we append a resolution embedding to the samples of the input function. After the FNO layers, we create two branches: one generates the predictive mean of the target function and the other the predictive variance. In this way, the predictive mean and variance are up to not only the input function samples but also the resolution choice. We then use Monte-Carlo ensemble learning (Lakshminarayanan et al., 2017) to fulfill effective uncertainty quantification, which is critical for utility evaluation and active learning.
- **Active Learning.** To optimize the learning efficiency while reducing the data cost as much as possible, we maximize the utility-cost ratio to select the best training input and resolution at each step, where the utility is measured by mutual information. The strategy is similar to the state-of-the-art multi-fidelity active learning and Bayesian optimization methods (Li et al., 2022b; Takeno et al., 2020; Li et al., 2020a), but there are two severe challenges. The first challenge is that the computation of the utility function is analytically intractable and costly. We use moment matching to approximate the predictive distribution as a multi-variate Gaussian. We then leverage the structure of the covariance matrix, and apply the matrix determinant lemma to fulfill efficient, closed-form mutual information calculation. The second challenge is that, directly maximizing the utility-cost ratio as in the prior methods, tends to make the active learning stuck at low-resolution queries and inferior performance. This is due to that when the data is few (at the early stage), the mutual information measurement for examples at different resolutions is close. The value of high-resolution examples is actually under-estimated or over-penalized by the large cost. To address this problem, we propose a flexible cost annealing framework, which initializes the same cost for every resolution. The cost for each resolution is scheduled to gradually converge to the true cost with the accumulation of data. When the data is enough and the mutual information can reflect the true potential of each example, our active

learning returns to maximizing the benefit-cost ratio. In this way, our method can flexibly leverage high-resolution examples at the early stage to enable continuous improvement. Our framework applies to general multi-fidelity learning and optimization problems.

- **Experimental Results.** We evaluated MRA-FNO with four benchmark operator learning tasks, based on Burger's, Darcy flow, nonlinear diffusion and Navier-Stoke equations. On fixed training datasets, our multi-resolution FNO shows better or very close prediction error as compared to the standard FNO. Both the prediction accuracy and test log likelihood is much larger than applying other popular Bayesian inference methods, including Monte-Carlo dropout, stochastic gradient Langevin dynamics and variational inference, showing that our ensemble inference provides much better uncertainty quantification. During the course of each active learning experiment, MRA-FNO consistently shows much better prediction accuracy with the same accumulated data cost, as compared with random queries, core-set active learning, and our framework with dropout inference.

2. Background

Operator Learning. Suppose our goal is to learn a function-to-function mapping $\psi : \mathcal{H} \to \mathcal{Y}$, where $\mathcal{H}$ and $\mathcal{Y}$ are two function spaces (*e.g.*, Banach spaces). The training dataset comprises pairs of discretized input and output functions, $\mathcal{D} = \{(\mathbf{f}_n, \mathbf{y}_n)\}_{n=1}^N$, where each $\mathbf{f}_n$ are samples of a function $f_n \in \mathcal{H}$, and $\mathbf{y}_n$ are samples of $\psi[f_n] \in \mathcal{Y}$. All the input and output functions are discretized (sampled) at a set of evenly-spaced locations, *e.g.*, a 64×64 mesh in the 2D spatial domain $[0, 1] \times [0, 1]$.

Fourier Neural Operators (FNO). Given a discretized input function $\mathbf{f}$, the FNO first applies a feed-forward network (FFN) over each element of $\mathbf{f}$ and the associated sampling location to lift the input to a higher-dimensional channel space. Then the FNO introduces a Fourier layer to perform a linear transform and nonlinear activation in the functional space,

$$v(\mathbf{x}) \leftarrow \sigma\left(\mathcal{W}v(\mathbf{x}) + \int \kappa(\mathbf{x} - \mathbf{x}')v(\mathbf{x}')\mathrm{d}\mathbf{x}'\right)$$

where $v(\mathbf{x})$ in the R.H.S is the input function to the Fourier layer and in the L.H.S the output function, $\kappa(\cdot)$ is the integration kernel and $\sigma(\cdot)$ is the activation. Based on the convolution theorem $\int \kappa(\mathbf{x} - \mathbf{x}')v(\mathbf{x}')\mathrm{d}\mathbf{x}' = \mathcal{F}^{-1}\left[\mathcal{F}[\kappa] \cdot \mathcal{F}[v]\right](\mathbf{x})$ where $\mathcal{F}$ and $\mathcal{F}^{-1}$ are the Fourier and inverse Fourier transform, respectively, the Fourier layer performs fast Fourier transform (FFT) over v, multiplies it with the discretized kernel in the frequency domain, and then performs inverse FFT. The local linear transform, $\mathcal{W}v(\mathbf{x})$, is performed by standard convolution (as in convolution nets). Due to the usage of FFT, the computation of the Fourier layer is highly efficient. After several Fourier layers, another FFN is applied channel-wisely to project back and make the final prediction. The training is typically done by minimizing an L_2 loss, $\Theta^* = \mathrm{argmin}_\Theta \frac{1}{N}\sum_{n=1}^N \|\mathbf{g}_n - \psi_{\mathrm{FNO}}(\mathbf{f}_n; \Theta)\|$, where Θ are the model parameters, including the discretized kernel in the frequency domain, standard convolution parameters in each Fourier layer, and the parameters of the FNN's for channel lifting and projection.

3. Probabilistic Multi-Resolution FNO

Despite the advantages of the FNO, the training data collection can be a severe bottleneck for practical usage, because it typically requires many expensive physical simulations. To reduce the

cost, we consider using multi-resolution data, which combines accurate yet expensive high-resolution examples with inaccurate (large bias) yet cheap-to-generate low-resolution examples. We then propose an active learning approach to lower the data cost to the fullest extent while reaching a high learning efficiency. To this end, we first propose a probabilistic FNO that can effectively integrate multi-resolution training examples and perform posterior inference.

Specifically, suppose a multi-resolution dataset is given, $\mathcal{D} = \{(\mathbf{f}_n, \mathbf{g}_n, r_n)\}_{n=1}^N$ where r_n denotes the resolution of the n-th example. We have R different resolutions in total ($1 \leq r_n \leq R$). For example, on a 2D spatial domain $[0,1] \times [0,1]$, we might have two resolutions, 16×16 and 128×128. To explicitly model the influence of the resolution choice on the prediction, we introduce an embedding $\mathbf{e}_r$ to represent each resolution $r \in [1, R]$. In our experiment, we set $\mathbf{e}_r$ to a one-hot encoding. We have also tried other embeddings, such as positional encodings (Vaswani et al., 2017). The performance is close. We apply an FFN to every element of $\mathbf{f}_n$, the corresponding sample location $\mathbf{x}_n$, and the embedding $\mathbf{e}_{r_n}$ to obtain a new representation $\widehat{\mathbf{f}}_n$, where each

$$[\widehat{\mathbf{f}}_n]_j = \text{FNN}([\mathbf{f}_n]_j, \mathbf{x}_j, \mathbf{e}_{r_n}). \tag{1}$$

Next, we use standard Fourier layers to perform successive linear and nonlinear transforms in the functional space. Denote by $\mathbf{v}_n$ the output (discretized) function. We then create two branches. One branch applies an FNN in each channel to project $\mathbf{v}_n$ back to the target dimension and output the predictive mean, $\boldsymbol{\mu}_\Theta(\mathbf{f}_n, \mathbf{e}_n)$ where Θ denote the model parameters. The other branch performs a standard convolution and then an FNN to output the predictive variance in the log domain, $\eta_\Theta(\mathbf{f}_n, \mathbf{e}_n)$. We then use a Gaussian likelihood to model the observed (discretized) output function,

$$p(\mathbf{g}_n|\mathbf{f}_n, r_n) = \mathcal{N}\left(\mathbf{g}_n|\boldsymbol{\mu}_\Theta(\mathbf{f}_n, \mathbf{e}_{r_n}), \exp\left(\eta_\Theta(\mathbf{f}_n, \mathbf{e}_{r_n})\right) \cdot \mathbf{I}\right). \tag{2}$$

As we can see, both the predictive mean and variance are not only dependent on the input $\mathbf{f}_n$ but also up to the resolution choice r_n. In this way, our model can capture the influence of the resolution choice on the predictive distribution. One can naturally expect a larger predictive variance at low resolutions, reflecting more uncertainty. On the other, the prediction at high resolutions should be more confident and gives a smaller predictive variance. Our model is illustrated in Appendix Fig. 5.

Next, we use Monte-Carlo ensemble learning[1] (Lakshminarayanan et al., 2017) to fulfill effective posterior inference. Specifically, we randomly initialize the model parameters Θ, and maximize the log likelihood to obtain one point estimate via stochastic mini-batch optimization,

$$\Theta^* = \underset{\Theta}{\text{argmax}} \sum_{n=1}^{N} \log\left[\mathcal{N}\left(\mathbf{g}_n|\boldsymbol{\mu}_\Theta(\mathbf{f}_n, \mathbf{e}_n), \exp\left(\eta_\Theta(\mathbf{f}_n, \mathbf{e}_n)\right)\mathbf{I}\right)\right]. \tag{3}$$

We independently repeat this procedure for M times, and obtain an ensemble of the point estimates of the model parameters, $\{\Theta_1^*, \ldots, \Theta_M^*\}$. We then construct a discrete posterior approximation of the model parameters, $p(\Theta|\mathcal{D}) \approx \frac{1}{M}\sum_{m=1}^M \delta(\Theta - \Theta_m^*)$, where $\delta(\cdot)$ is the Dirac delta measure. Given a test input function $\mathbf{f}$ and the resolution embedding $\mathbf{e}$, the predictive distribution of the output function is therefore a Gaussian mixture,

$$p(\mathbf{y}(\mathbf{f}, \mathbf{e})|\mathcal{D}) = \frac{1}{M}\sum_{m=1}^{M} \mathcal{N}\left(\mathbf{y}|\boldsymbol{\mu}_{\Theta_j^*}(\mathbf{f}, \mathbf{e}), \eta_{\Theta_j^*}(\mathbf{f}, \mathbf{e}) \cdot \mathbf{I}\right). \tag{4}$$

1. we do not introduce adversarial samples as in (Lakshminarayanan et al., 2017). We empirically found little help with such samples.

4. Multi-Resolution Active Learning

Now, we present our multi-resolution active learning algorithm. To optimize the learning efficiency while lowering the data cost as much as possible, at each step, we maximize a utility-cost ratio (as the acquisition function) to determine the most valuable input function and its resolution, at which we query a new example. Specifically, we prepare a pool of candidate input functions $\mathcal{P}$. Denote by λ_r the cost of generating the output function at resolution $r \in [1, R]$. We have $\lambda_1 < \ldots < \lambda_R$. To measure the value of an example with input function $h \in \mathcal{P}$ and resolution r, we consider two utility functions. The first one follows (Li et al., 2022b) and quantifies the information the example can bring to predict at the highest resolution R,

$$u(h, r) = \mathbb{I}(\mathbf{y}(\mathbf{h}^r, \mathbf{e}_r), \mathbf{y}(\mathbf{h}^R, \mathbf{e}_R)|\mathcal{D}) \tag{5}$$

where $\mathcal{D}$ is the current training dataset, $\mathbb{I}(\cdot, \cdot)$ is the mutual information, $\mathbf{h}^r$ and $\mathbf{h}^R$ are function h discretized at resolution r and R, respectively, and $\mathbf{e}_r$ and $\mathbf{e}_R$ are the corresponding resolution embeddings. The utility function (5) only considers how the example can improve the prediction for the same input function. To model its benefit in improving the prediction for other input functions, we follow (Li et al., 2022a) to consider a second utility function $u(h, r) = \mathbb{E}_{p(h')}[\mathbb{I}(\mathbf{y}(\mathbf{h}^r, \mathbf{e}_r), \mathbf{y}(\mathbf{h}'^R, \mathbf{e}_R)|\mathcal{D})]$, where $h' \in \mathcal{H}$ and $p(h')$ is a distribution over $\mathcal{H}$. The expectation usually does not have a closed-form, and we therefore draw A functions, $h'_1, \ldots, h'_A \sim p(h')$, and adopt the Monte-Carlo approximation,

$$\widehat{u}(h, r) = \frac{1}{A} \sum_{l=1}^{A} \mathbb{I}(\mathbf{y}(\mathbf{h}^r, \mathbf{e}_r), \mathbf{y}(\mathbf{h}'^R_l, \mathbf{e}_R)|\mathcal{D}). \tag{6}$$

4.1 Efficient Utility function Computation

The utility function in both (5) and (6) demands we compute the mutual information between a pair of predictions from our model. The computation is challenging in that (1) those predictions are typically high-dimensional (*e.g.*, a 100×100 resolution corresponds to $10K$ dimensional outputs), and (2) the mutual information is analytically intractable due to the Gaussian mixture predictive distribution in (4). To address this problem, we observe that for any two predictions $\mathbf{y}_1$ and $\mathbf{y}_2$

$$\mathbb{I}(\mathbf{y}_1, \mathbf{y}_2|\mathcal{D}) = \mathbb{H}(\mathbf{y}_1|\mathcal{D}) + \mathbb{H}(\mathbf{y}_2|\mathcal{D}) - \mathbb{H}(\mathbf{y}_1, \mathbf{y}_2|\mathcal{D}). \tag{7}$$

Denote by $(\mathbf{f}_1, \mathbf{e}_1)$ the discretized input function and resolution embedding for $\mathbf{y}_1$ and by $(\mathbf{f}_2, \mathbf{e}_2)$ for $\mathbf{y}_2$. We first use moment matching to approximate the predictive distributions of $\mathbf{y}_1$, $\mathbf{y}_2$ and $\widehat{\mathbf{y}} = [\mathbf{y}_1; \mathbf{y}_2]$ as multi-variate Gaussian distributions, and we can thereby compute each entropy with a closed form. Specifically, let us first consider $\widehat{\mathbf{y}}$. According to (4), we can derive that $p(\widehat{\mathbf{y}}|\mathcal{D}) = \frac{1}{M}\sum_{m=1}^{M} \mathcal{N}(\widehat{\mathbf{y}}|\boldsymbol{\rho}_j, \boldsymbol{\Lambda}_j)$, where $\boldsymbol{\rho}_j = [\boldsymbol{\mu}_{\Theta^*_j}(\mathbf{f}_1, \mathbf{e}_1); \boldsymbol{\mu}_{\Theta^*_j}(\mathbf{f}_2, \mathbf{e}_2)]$ and $\boldsymbol{\Lambda}_j = \operatorname{diag}\left(\eta_{\Theta^*_j}(\mathbf{f}_1, \mathbf{e}_1)\mathbf{I}, \eta_{\Theta^*_j}(\mathbf{f}_2, \mathbf{e}_2)\mathbf{I}\right)$. The mean and covariance (first and second moments) are

$$\mathbb{E}(\widehat{\mathbf{y}}|\mathcal{D}) = \frac{1}{M}\sum_{m=1}^{M} \boldsymbol{\rho}_m, \quad \operatorname{cov}(\widehat{\mathbf{y}}|\mathcal{D}) = \frac{1}{M}\sum_{m=1}^{M}\left(\boldsymbol{\Lambda}_m + \boldsymbol{\rho}_m\boldsymbol{\rho}_m^\top\right) - \mathbb{E}(\widehat{\mathbf{y}}|\mathcal{D})\mathbb{E}(\widehat{\mathbf{y}}|\mathcal{D})^\top. \tag{8}$$

Via moment matching, we construct a multi-variate Gaussian approximation,

$$p(\widehat{\mathbf{y}}|\mathcal{D}) \approx \mathcal{N}(\widehat{\mathbf{y}}|\mathbb{E}(\widehat{\mathbf{y}}|\mathcal{D}), \operatorname{cov}(\widehat{\mathbf{y}}|\mathcal{D}))$$

which is the best approximation in the exponential family in the sense of Kullback Leibler divergence (Bishop and Nasrabadi, 2006). Accordingly, the entropy can be computed with a closed-form, $\mathbb{H}(\widehat{\mathbf{y}}) = \frac{1}{2}\log\det\left[\mathrm{cov}(\widehat{\mathbf{y}}|\mathcal{D})\right] + \text{const}$.

However, since $\widehat{\mathbf{y}}$ is high-dimensional, computing the log determinant of its huge covariance matrix is extremely expensive or even infeasible. To address this problem, we observe that

$$\mathrm{cov}(\widehat{\mathbf{y}}|\mathcal{D}) = \mathbf{\Lambda} + \frac{1}{M-1}\sum_{m=1}^{M}\left(\boldsymbol{\rho}_m - \mathbb{E}(\widehat{\mathbf{y}}|\mathcal{D})\right)\left(\boldsymbol{\rho}_m - \mathbb{E}(\widehat{\mathbf{y}}|\mathcal{D})\right)^\top \tag{9}$$

where $\mathbf{\Lambda} = \mathrm{diag}\left(\frac{1}{M}\sum_{m=1}^{M}\eta_{\Theta_j^*}(\mathbf{f}_1,\mathbf{e}_1)\cdot\mathbf{I}, \frac{1}{M}\sum_{m=1}^{M}\eta_{\Theta_j^*}(\mathbf{f}_2,\mathbf{e}_2)\cdot\mathbf{I}\right)$ is a diagonal matrix, and the second term in the R.H.S is actually the empirical covariance matrix over $\{\boldsymbol{\rho}_m\}$. We can further derive that $\mathrm{cov}(\widehat{\mathbf{y}}|\mathcal{D}) = \mathbf{\Lambda} + \mathbf{B}\mathbf{B}^\top$, where $\mathbf{B} = \frac{1}{\sqrt{M-1}}[\boldsymbol{\rho}_1 - \mathbb{E}(\widehat{\mathbf{y}}|\mathcal{D}), \ldots, \boldsymbol{\rho}_M - \mathbb{E}(\widehat{\mathbf{y}}|\mathcal{D})]$, which includes M columns. We then use the matrix determinant lemma (Harville, 1997) to compute,

$$\log\det\left[\mathrm{cov}(\widehat{\mathbf{y}}|\mathcal{D})\right] = \log\det\left[\mathbf{\Lambda} + \mathbf{B}\mathbf{B}^\top\right] = \log\det[\mathbf{\Lambda}] + \log\det[\mathbf{I} + \mathbf{B}^\top\mathbf{\Lambda}^{-1}\mathbf{B}]. \tag{10}$$

The first log determinant is over the diagonal matrix $\mathbf{\Lambda}$, and the complexity is linear in the dimension of $\widehat{\mathbf{y}}$. The second log determinant is computed over an $M \times M$ matrix. Since M is the size of the ensemble and is very small (we take $M = 5$ in our experiments), the computation is highly efficient. It is straightforward to use a similar method to compute $\mathbb{H}(\mathbf{y}_1|\mathcal{D})$ and $\mathbb{H}(\mathbf{y}_2|\mathcal{D})$ in (7).

4.2 Cost Annealing

In practice, directly maximizing the utility-cost ratio $\frac{u(h,r)}{\lambda_r}$ or $\frac{\widehat{u}(h,r)}{\lambda_r}$ (see (5) and (6)) tends to make the active learning stuck at low-resolution queries and inferior performance, especially when the cost discrepancy is significant between the low and high resolutions. This is because at the early stage, the training data is few, and the mutual information does not differ much for candidates at different resolutions. In other words, the scales are close. Consequently, the potential of high-resolution examples is under-estimated or over-penalized by the large cost, and the active learning keeps selecting low-resolution examples, which actually hinder the model improvement.

To overcome this problem, we propose a cost annealing method. We schedule a dynamic cost assignment for each resolution. Denote by $\widehat{\lambda}_r(t)$ the cost schedule for resolution r at step t. For convenience, we normalize the true cost into $[0, 1]$, *i.e.*, each $\lambda_r \in [0, 1]$ and $\sum_{r=1}^{R}\lambda_r = 1$. We set

$$\widehat{\lambda}_r(t) = \frac{\lambda_r}{1 + (R\lambda_r - 1)c(t)}, \tag{11}$$

where $c(t)$ is a decaying function such that $c(0) = 1$ and $c(\infty) = 0$. For example, we can use

$$c(t) = \exp(-\alpha t), \quad \text{or} \quad c(t) = 2(1 - s(\alpha t)), \tag{12}$$

where $s(\cdot)$ is the sigmoid function and α controls the decay rate. We can see that all $\widehat{\lambda}_r(0) = \frac{1}{R}$ and $\lim_{t\to\infty}\widehat{\lambda}_r(t) = \lambda_r$. At each step t, we select the input and resolution by maximizing the acquisition function, $\frac{u(h,r)}{\widehat{\lambda}_r(t)}$ or $\frac{\widehat{u}(h,r)}{\widehat{\lambda}_r(t)}$. In this way, at the early stage when the data is few and the mutual information does not differ much, our method avoids over-penalizing high-resolution examples, and

Algorithm 1 MRA-FNO (M, $\mathcal{P}$, T, $\{\lambda_r\}_{r=1}^R$)

1: Learn the probabilistic multi-resolution FNO from an initial dataset $\mathcal{D}$ with the ensemble size M.
2: **for** $t = 1 \dots T$ **do**
3: Based on the cost schedule (11), select the input function $h_t \in \mathcal{P}$ and the resolution r_t by

$$h_t, r_t = \underset{h \in \mathcal{P}, 1 \le r \le R}{\operatorname{argmax}} \frac{\beta(h, r)}{\widehat{\lambda}_r(t)}$$

where $\beta(h, r)$ is the utility function that can take (5) or (6).
4: Query the discretized output function $\mathbf{y}_t$ at h_t with resolution r_t.
5: Remove h_t from $\mathcal{P}$.
6: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{h}_t, \mathbf{y}_t, r_t)\}$ where $\mathbf{h}_t$ is the discretized h_t at resolution r_t.
7: Re-train the probabilistic multi-resolution FNO on $\mathcal{D}$
8: **end for**

promote their queries to ensure continuous model improvement. With the accumulation of data, the mutual information is more and more capable of reflecting the true value of new examples, the active learning returns to maximizing the ideal utility-cost ratio to select the input function and resolution at which to query. Our method is summarized in Algorithm 1.

Algorithm Complexity. The time complexity of each active learning step is $\mathcal{O}(|\mathcal{P}|RM^2d)$ where $|\mathcal{P}|$ is the size of the candidate pool, and d is the output dimension at the highest resolution. The space complexity is $\mathcal{O}(Md)$, which is to store the predictive distribution (for any input function) and the parameter estimates in the ensemble.

5. Related Work

Operator learning is a fast-growing research area. A variety of operator learning methods have been developed, most of which are based on neural networks and henceforth called neural operators. For example, along with FNO, a simple low-rank neural operator (LNO) (Li et al., 2020d) was proposed to employ a low-rank decomposition of the operator's kernel. Li et al. (2020b) proposed GNO that uses Nystrom approximation and graph neural networks to approximate the function convolution. In (Li et al., 2020c), a multipole graph neural operator (MGNO) is developed, which uses a multi-scale kernel decomposition to achieve linear complexity in computing the convolution. Gupta et al. (2021) developed a multiwavelet-based operator learning model that represents the operator's kernel with fine-grained wavelets. Another popular approach is the Deep Operator Net (DeepONet) (Lu et al., 2021), which combines a branch net over the input functions and a trunk net over the sampling locations to predict the target function values. A more stable and efficient version, POD-DeepONet was proposed in (Lu et al., 2022), which replaces the branch net with the POD (or PCA) bases computed from the training data. (Seidman et al., 2022) used a nonlinear combination (*e.g.*, a feed-forward network) of the branch net and trunk net outputs to approximate the target function. A survey of neural operators is given in (Kovachki et al., 2023). The recent works have also developed kernel operator learning approaches (Long et al., 2022; Batlle et al., 2023).

Active learning is a classical machine learning topic. The recent research focuses on the active learning of deep neural networks. For example, in (Gal et al., 2017), Monte-Carlo (MC) Dropout (Gal and Ghahramani, 2016) was used to generate the posterior samples and compute the acquisition function. (Geifman and El-Yaniv, 2017; sen) used core-set search to query diverse and representative

	Relative L_2 Error		NLL	
Method	**Burgers**	**Darcy**	**Burgers**	**Darcy**
FNO	$\mathbf{0.0575 \pm 0.0031}$	0.0891 ± 0.0078	NA	NA
FNO-Dropout	0.0791 ± 0.0035	0.1038 ± 0.0056	176.84 ± 16.11	4447.92 ± 63.08
FNO-SGLD	0.0804 ± 0.0049	0.0933 ± 0.0074	223.48 ± 15.74	3683.45 ± 83.49
FNO-SVI	0.1182 ± 0.0056	0.0946 ± 0.0041	391.61 ± 10.59	4027.71 ± 73.96
MRA-FNO	$\mathbf{0.0586 \pm 0.0042}$	$\mathbf{0.0876 \pm 0.0059}$	$\mathbf{44.57 \pm 3.62}$	$\mathbf{1167.73 \pm 29.13}$

Table 1: Prediction accuracy in non-active learning. NLL stands for Negative Log Likelihood. The results were averaged from five runs.

examples, which are shown to be particularly effective for convolution neural nets. Other examples include (Gissin and Shalev-Shwartz, 2019; Ducoffe and Precioso, 2018) for adversarial active learning, (Ash et al., 2019) using the gradient magnitude to represent the uncertainty and to query new examples, *etc*. Recently, (Li et al., 2022b) proposed the first multi-fidelity active learning approach, which dynamically queries multi-fidelity simulation examples to train a surrogate model that predicts PDE solutions from PDE parameters. (Li et al., 2022a) further developed a batch multi-fidelity active learning algorithm with budget constraints. The key difference is that these works aim to learn a mapping from the PDE parameters (low-dimensional input) to the solution (high-dimensional output), and they employ an auto-regressive architecture to combine examples of multiple fidelities. Their stochastic variational inference is inferior in posterior approximation and uncertainty quantification for operator learning. We therefore develop another posterior inference approach based on ensemble learning, which turn out to be much more effective. We accordingly develop an efficient method for utility function computation. In addition, we discovered the over-penalization problem during the active learning, which was never discovered in these previous works. We proposed a novel and flexible cost annealing framework to overcome the problem. The most recent work (Pickering et al., 2022) proposed an active learning approach for DeepONet. The goal is to query examples that can facilitate the discovery and forecast of rare events. The work does not consider multi-resolution examples and their varying costs. Hence, the goal, model estimation, acquisition function design and computation are all very different from our work.

6. Experiment

6.1 Prediction Accuracy on Fixed Training Data

We first examined if our probabilistic multi-resolution FNO can achieve good prediction accuracy and uncertainty calibration. To this end, we tested with two benchmark operator learning tasks, one is based on a Burgers' equation and the other a Darcy flow equation. For *Burgers*, we aim to learn a mapping from the initial condition to the solution at time $t = 1$, while for *Darcy*, the goal is to learn a mapping from the coefficient function to the solution. We considered two resolutions for each task. The details are provided in Section A in Appendix.

We randomly generated 200 examples for each resolution to obtain a training set. We randomly generated another 200 examples at the highest resolution as the test set. We compared with the standard FNO (point estimation), FNO trained via MC Dropout (FNO-Dropout) (Gal and Ghahramani, 2016), stochastic gradient Langevin dynamics (FNO-SGLD) (Welling and Teh, 2011), and stochastic variational inference (FNO-SVI) (Kingma and Welling, 2013). For all the methods, we set the mini-batch size to 20, the learning rate to 10^{-3}, and use ADAM optimization and Cosine Annealing schedule. We used the FNO implementation from the original authors (`https://github.com/`

Task	Resolution	Cost Ratio
Burgers	33, 129	$1 : 41.2$
Darcy	32×32, 128×128	$1 : 38.3$
Darcy3	32×32, 64×64, 128×128	$1 : 21.3 : 38.3$
Diffusion	32×32, 64×64, 128×128	$1 : 4.7 : 17.6$
NS	16×16, 64×64	$1 : 7$

Table 2: Resolution and cost ratio for each active learning task. The cost is measured by the average running time for solving the PDEs (100 runs) at the corresponding resolution.

`neuraloperator/neuraloperator`). We tuned the dropout rate from [0.1, 0.5]. For SGLD and SVI, we assigned a standard Gaussian prior over the model parameters. For SVI, we employed a fully factorized Gaussian posterior approximation. We repeated the training and test procedure for five times, and examined the average relative L_2 error, the average negative log likelihood (NLL), and their standard deviation on the test datasets. The results are reported in Table 1.

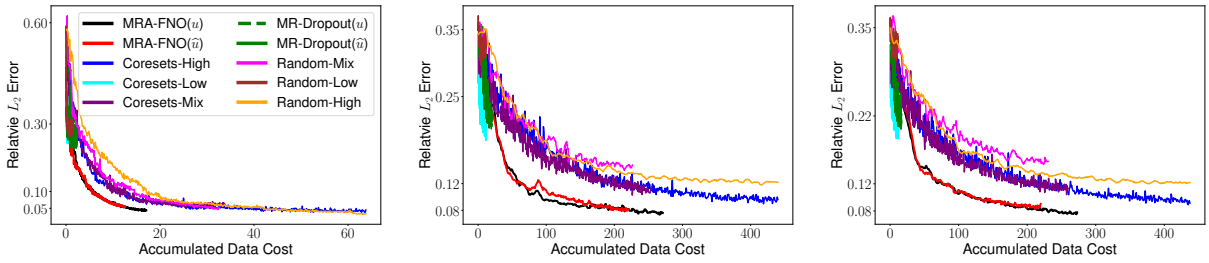
We can see that the our model (MRA-FNO) achieves the relative L_2 error significantly smaller than the competing methods in all the cases, except that in Burger's equation, the L_2 error of MRA-FNO is slightly worse than the standard FNO. More important, MRA-FNO consistently outperforms all the probabilistic versions of FNO by a large margin in test log likelihood. Hence, not only does our model give superior prediction accuracy, our ensemble posterior inference also enables much better uncertainty quantification.

6.2 Active Learning Performance

Next, we evaluated the active learning performance of MRA-FNO. In addition to the tasks in Section 6.1, we considered two more PDEs, one is a nonlinear diffusion equation, and the other is a 2D Navier-Stokes (NS) equation used in (Li et al., 2020d). For each task, we considered two resolutions. We leave the details in Section A of Appendix. In addition, we tested active learning on the same *Darcy* problem as in Section 6.1 with three resolutions. We summarize the data acquiring cost at different resolutions in Table 2. As we can see, the cost discrepancy is large among different resolutions.

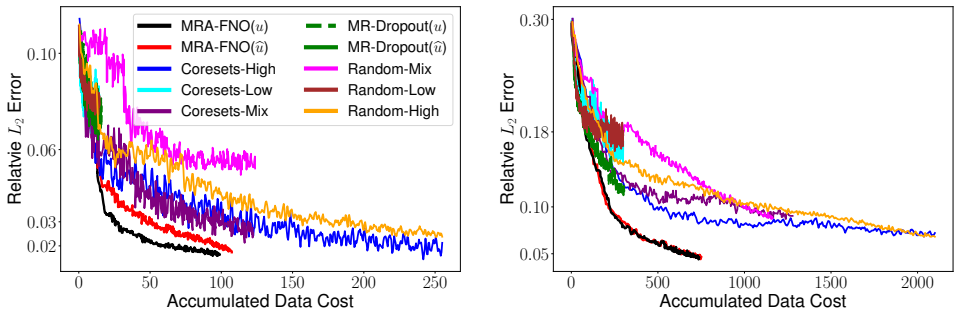
We compared with the following active learning methods for FNO. (1) Random-Low/High, randomly selecting an input function from the candidate pool, and querying the example at the lowest/highest resolution. (2) Random-Mix: randomly selecting both the input and resolution. (3) Coreset-Low/High: we used the coreset active learning strategy (sen) to select the input function that maximizes the minimum distance to the existed examples, according to the output of the last Fourier layer as the representation. We fixed the resolution to be the lowest or the highest one. (4) Coreset-Mix, the same coreset active learning strategy as in (3), except that we allow querying at different resolutions. We interpolate the representation to the highest resolution to compute the distance. (5) MR-Dropout: we used MC dropout to perform posterior inference for FNO, and then used the same acquisition function(s), computation method, and annealing framework as in our approach to identify the input function and resolution. (6) MR-PredVar: we averaged the predictive variance of each output function values as the utility function, and the remaining is the same as our approach.

For every active learning experiment, we randomly generated 10 examples for each resolution to obtain an initial dataset. We randomly generated 990 input functions at the highest resolution, which



(a) *Burgers* (b) *Darcy* (c) *Darcy3*

Figure 1: Relative L_2 error *vs.* accumulated data cost. Each method ran 500 active learning steps. Note that different methods can end up with different total data cost (after running the same number of steps).

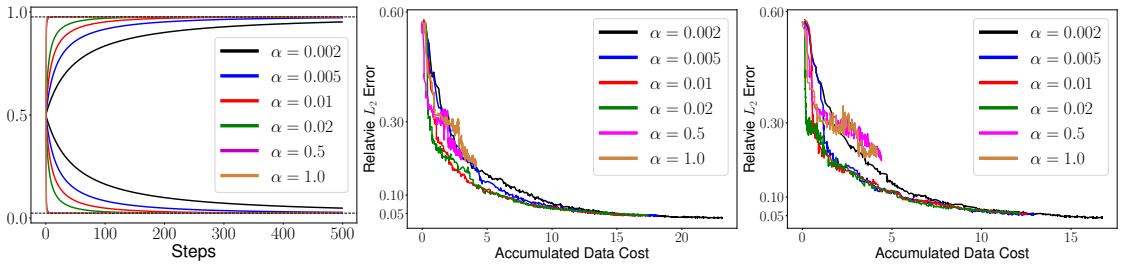


(a) *Diffusion* (b) *Navier Stoke (NS)*

Figure 2: Relative L_2 error *vs.* accumulated data cost.

we used as the candidate pool for active learning. If one example is queried at a lower resolution, the input function is downsampled accordingly. We randomly generated another 200 examples at the highest resolution for testing. We then ran active learning with each method. For our method and MR-Dropout, we tested two annealing schedules, one is based on the exponential decay and the other sigmoid decay; see (12) . We tuned the decaying rate α from $\{0.002, 0.005, 0.01, 0.02, 0.5, 1.0\}$. We ran 500 active learning steps (queries) for all the experiments except for the *NS* problem, we ran 300 steps. We examined the relative L_2 error of each method *vs.* the accumulated data cost. To avoid cluttered figures, we show the result of our method with the exponential-decay-based schedule in Fig. 1 and 2, and the result of using the sigmoid decay and MR-PredVar in Fig. 6 in Appendix.

Prediction Accuracy. As we can see, at the beginning, the performance of each method is identical or very close. As the active learning progresses, MRA-FNO improves rapidly and constantly. It soon achieves a superior prediction accuracy to all the competing methods, and consistently outperforms them during the remaining course of the active learning. Accordingly, MRA-FNO can reach the smallest prediction error under the same data cost, or use the least data cost to achieve the same performance. We empirically observed that using the utility function (5) or (6), denoted by MRA-FNO (u) and MRA-FNO ($\widehat{u}$), respectively, result in close performance, except that on the diffusion problem, MRA-FNO (u) appears to be better. This might be because the Monte-Carlo approximation in (6) (we set $A = 5$) still has a significant gap from the true expectation under the distribution over a functional space. It is worth noting that both Random-Low and Coreset-Low were quickly trapped at large prediction errors. It therefore shows only using low-resolution examples, the



(a) Cost schedule (Exp) (b) MRA-FNO (u) (c) MRA-FNO ($\widehat{u}$)

Figure 3: The influence of the cost schedule on active learning. We report the result with the exponential decay; see (12). The larger α, the faster the schedule converges to the true cost.

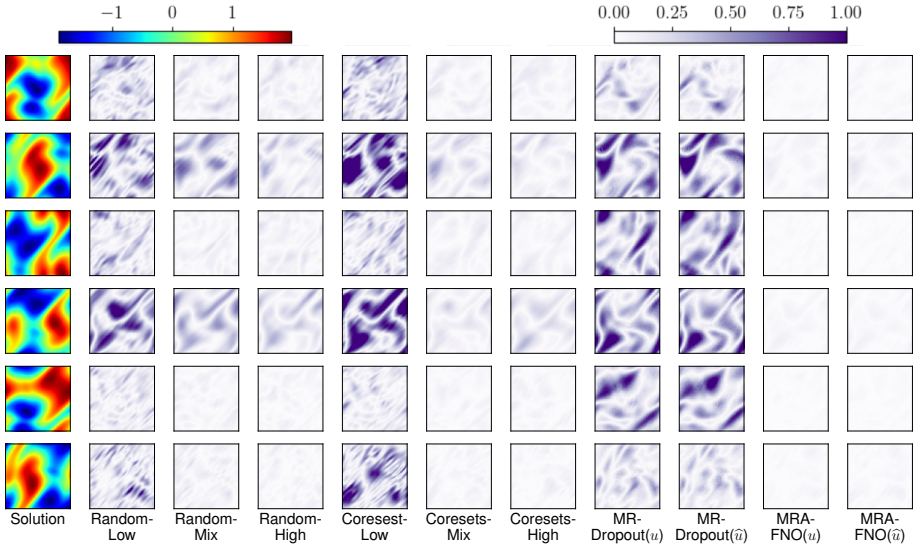


Figure 4: Point-wise error on *NS*.

predictive performance will soon meet a bottleneck and can hardly improve, though the data cost grows very slowly. On the other hand, Random-High and Coreset-High enables steady improvement because they only query high-resolution examples at each step. However, the data cost accumulation is much greater, *e.g.*, Fig. 1b and 1c. In addition, the performance of MR-Dropout tends to stuck at large prediction errors early, especially in *Burgers*, *Darcy* and *Darcy3*. We observed that MR-Dropout mainly selected low-resolution examples. This might be because the uncertainty quantification by dropout is not reliable for FNO, and even using our annealing framework cannot correct its bias. From Fig. 6 of Appendix, we can see that the performance of MRA-FNO with the sigmoid-based cost schedule is close to that with exp-based schedule (see (12)), except in *Darcy3*, the exp-based schedule shows a slight yet consistent advantage. Interestingly, MR-PredVar outperforms the other competing methods in all the cases, confirming the importance of effective uncertainty quantification in utility evaluation (it also uses our ensemble posterior inference). While MR-PredVar achives close performance to our method in *Burgers*, in all the other cases, MR-PredVar is apparently worse. This might be because MR-PredVar ignores the (strong) correlation between the output function values, and hence the quality of utility evaluation is worse. All these results have demonstrated the advantage of our multi-resolution active learning approach.

Influence of Cost Schedule. Next, we investigated how the cost annealing schedule influences the active learning. To this end, we used the exponential decay function in our schedule, and varied the decaying rate $\alpha \in \{0.002, 0.005, 0.01, 0.02, 0.5, 1.0\}$. We show the cost schedule for different choices of α in Fig. 3a. We then run MRA-FNO on *Burgers* with 500 steps. The L_2 relative error *vs.* the accumulated data cost is reported in Fig. 3b and 3c. We can see that when α is too small, *e.g.*, $\alpha = 0.002$, though the active learning ensures steady improvement of the prediction accuracy, the data cost is suboptimal. To obtain the same performance, a too small α consumes a much bigger data cost, or under the same cost, it gives worse performance. The reason is that the convergence of the cost annealing is too slow; see Fig. 3a. Even when the mutual information has become sufficiently discriminative, the cost assignments for different resolutions are still not far, which actually over-penalize low-resolution examples and lead to a selection bias toward high-resolution examples. Another extreme is to use a too big α, *e.g.*, $\alpha = 0.5$ and $\alpha = 1.0$. In such case, the schedule will converge to the true cost very fast, even at the early stage when data is few. Accordingly, the high-resolution examples are soon over-penalized, making the learning stuck at low-resolution queries. The prediction accuracy is fluctuating yet hard to increase substantially. On the contrary, an appropriate decay rate in between, *e.g.*, $\alpha = 0.01$ and $\alpha = 0.02$, can sidestep these problems, and lead to superior performance in both cost saving and prediction accuracy.

Point-wise Error. Finally, we investigate the local errors of the prediction. We randomly selected six test cases for *NS* and *Diffusion*. We examined the post-wise error of each method after active learning. We show the results in Fig. 4 and Appendix Fig. 7. We can see that the point-wise error of MRA-FNO is quite uniform across the domain and is close to zero (white). By contrast, the other methods exhibit large errors in many local regions. Together these results have shown that MRA-FNO not only gives a superior global accuracy, but locally better recovers individual output function values.

7. Conclusion

We have presented MRA-FNO, a multi-resolution active learning method for Fourier neural operators. On several benchmark operator learning tasks, MRA-FNO can save the data cost substantially while achieving superior predictive performance. Currently, the selection of the decay rate in our cost annealing framework is done by manual tuning/cross-validation. In the future, we plan to develop novel methods, such as reinforcement learning, to automatically determine the best rate.

8. Acknowledge

We thank Andrew Stuart for valuable discussion and suggestions.

References

Jordan T Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. In International Conference on Learning Representations, 2019.

Pau Batlle, Matthieu Darcy, Bamdad Hosseini, and Houman Owhadi. Kernel methods are competitive for operator learning. arXiv preprint arXiv:2304.13202, 2023.

Luke Bhan, Yuanyuan Shi, and Miroslav Krstic. Operator learning for nonlinear adaptive control. In Learning for Dynamics and Control Conference, pages 346–357. PMLR, 2023.

Christopher M Bishop and Nasser M Nasrabadi. Pattern recognition and machine learning, volume 4. Springer, 2006.

Melanie Ducoffe and Frederic Precioso. Adversarial active learning for deep networks: a margin based approach. arXiv preprint arXiv:1802.09841, 2018.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In international conference on machine learning, pages 1050–1059, 2016.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In International Conference on Machine Learning, pages 1183–1192, 2017.

Yonatan Geifman and Ran El-Yaniv. Deep active learning over the long tail. arXiv preprint arXiv:1711.00941, 2017.

Daniel Gissin and Shai Shalev-Shwartz. Discriminative active learning. arXiv preprint arXiv:1907.06347, 2019.

Gaurav Gupta, Xiongye Xiao, and Paul Bogdan. Multiwavelet-based operator learning for differential equations. Advances in neural information processing systems, 34:24048–24062, 2021.

David A Harville. Matrix algebra from a statistician's perspective. Springer Book Archive-Mathematics, 1997.

Sebastian Kaltenbach, Paris Perdikaris, and Phaedon-Stelios Koutsourelakis. Semi-supervised invertible deeponets for bayesian inverse problems. arXiv preprint arXiv:2209.02772, 2022.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.

Nikola B Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew M Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. J. Mach. Learn. Res., 24(89):1–97, 2023.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. Advances in neural information processing systems, 30, 2017.

Shibo Li, Wei Xing, Robert Kirby, and Shandian Zhe. Multi-fidelity bayesian optimization via deep neural networks. Advances in Neural Information Processing Systems, 33:8521–8531, 2020a.

Shibo Li, Jeff M Phillips, Xin Yu, Robert Kirby, and Shandian Zhe. Batch multi-fidelity active learning with budget constraints. Advances in Neural Information Processing Systems, 35:995–1007, 2022a.

Shibo Li, Zheng Wang, Robert Kirby, and Shandian Zhe. Deep multi-fidelity active learning of high-dimensional outputs. In International Conference on Artificial Intelligence and Statistics, pages 1694–1711. PMLR, 2022b.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. arXiv preprint arXiv:2003.03485, 2020b.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. Advances in Neural Information Processing Systems, 33:6755–6766, 2020c.

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, Anima Anandkumar, et al. Fourier neural operator for parametric partial differential equations. In International Conference on Learning Representations, 2020d.

Ziyue Liu, Yixing Li, Jing Hu, Xinling Yu, Shinyu Shiau, Xin Ai, Zhiyu Zeng, and Zheng Zhang. Deepoheat: Operator learning-based ultra-fast thermal simulation in 3d-ic design. arXiv preprint arXiv:2302.12949, 2023.

Da Long, Nicole Mrvaljevic, Shandian Zhe, and Bamdad Hosseini. A kernel approach for pde discovery and operator learning. arXiv preprint arXiv:2210.08140, 2022.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. Nature machine intelligence, 3(3):218–229, 2021.

Lu Lu, Xuhui Meng, Shengze Cai, Zhiping Mao, Somdatta Goswami, Zhongqiang Zhang, and George Em Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. Computer Methods in Applied Mechanics and Engineering, 393:114778, 2022.

Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. arXiv preprint arXiv:2202.11214, 2022.

Ethan Pickering, Stephen Guth, George Em Karniadakis, and Themistoklis P Sapsis. Discovering and forecasting extreme events via active learning in neural operators. Nature Computational Science, 2(12):823–833, 2022.

Jacob Seidman, Georgios Kissas, Paris Perdikaris, and George J Pappas. Nomad: Nonlinear manifold decoders for operator learning. Advances in Neural Information Processing Systems, 35:5601–5613, 2022.

Shion Takeno, Hitoshi Fukuoka, Yuhki Tsukada, Toshiyuki Koyama, Motoki Shiga, Ichiro Takeuchi, and Masayuki Karasuyama. Multi-fidelity Bayesian optimization with max-value entropy search and its parallelization. In International Conference on Machine Learning, pages 9334–9345. PMLR, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, **L**ukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In Proceedings of the 28th international conference on machine learning (ICML-11), pages 681–688, 2011.

Appendix

Appendix A. Operator Learning Task Details

We tested our method with the following operator learning tasks.

- **Burgers**. The first one is based on the Burger's equation,

$$u_t + u_{xx} = \nu u_{xx}, \quad u(x, 0) = u_0(x), \tag{13}$$

  where $(x, t) \in [0, 1]^2$, and $u_0(x)$ is the initial condition, and $\nu = 0.002$ is the viscosity. We aim to learn a mapping from the initial condition to the solution at $t = 1$, namely, $u_0 \to u(x, 1)$. We considered two resolutions, which use 33 and 129 samples for discretization, respectively. We used a parametric form of the input function, $u_0(x) = a \exp(-ax) \sin(2\pi x) \cos(b\pi x)$, and we then randomly sampled $a, b \in [1, 6]$ to obtain the instances.

- **Darcy**. The second task is based on a 2D Darcy flow equation,

$$-\nabla(c(\mathbf{x})\nabla u(\mathbf{x})) = f(\mathbf{x}), \tag{14}$$

  where $\mathbf{x} \in [0, 1]^2$, $f(\mathbf{x}) = 1$ is a constant forcing function, $c(\mathbf{x}) > 0$ is the diffusion coefficient function, and on the boundary, $u(\mathbf{x}) = 0$. We aim to learn the mapping from the coefficient function to the solution, $c \to u$. We employed two sampling resolutions, 32×32 and 128×128. We followed (Li et al., 2020d) to first sample a discretized function from a Gauss random field, and then thresholding the values to be 4 or 12 to obtain the input function.

- **Diffusion**. The third one is based on a nonlinear diffusion PDE,

$$u_t = 0.01u_{xx} + 0.01u^2 + f(x) \tag{15}$$

  where $(x, t) \in (0, 1) \times (0, 1]$, $u(0, t) = u(1, t) = 0$, $u(x, 0) = 0$ and $f(x)$ is the forcing function. The goal is to learn the mapping from the forcing function to the solution, $f \to u$. We employed three resolutions for data acquiring, 32×32, 64×64 and 128×128. We draw samples of f from a Gaussian process with an RBF kernel. Note that to use FNO and MRA-FNO, we replicate the spatial discretization of f along the time dimension (steps).

- **Navier Stoke (NS)**. The last task is based the a 2D Navier-Stokes (NS) equation used in (Li et al., 2020d). The solution $u(\mathbf{x}, t)$ is the vorticity of a viscous, incompressible fluid, where $\mathbf{x} \in [0, 1]^2$ and $t \in [0, 50]$. We set the viscosity to 10^{-3}. Following (Li et al., 2020d), we considered 40 steps in the time domain. We used the the solution at the first 20 time steps to predict the solution at the next 20 steps. For data collection, we used two resolutions 16×16 and 64×64 in the spatial domain. We sampled the input functions from a Gaussian random field.

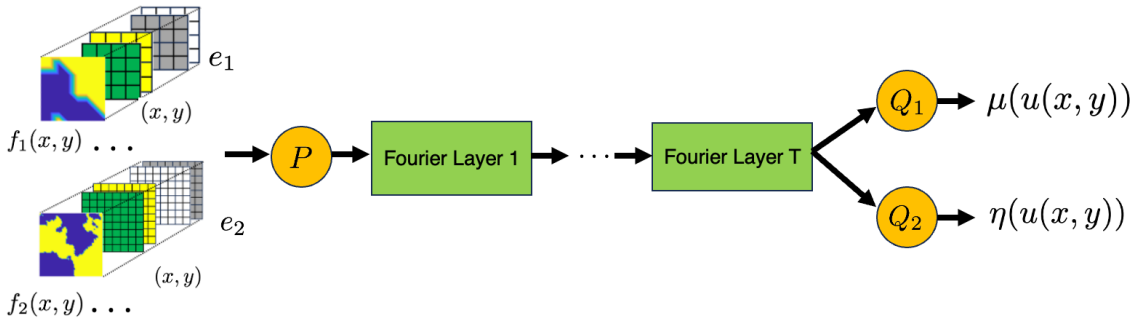
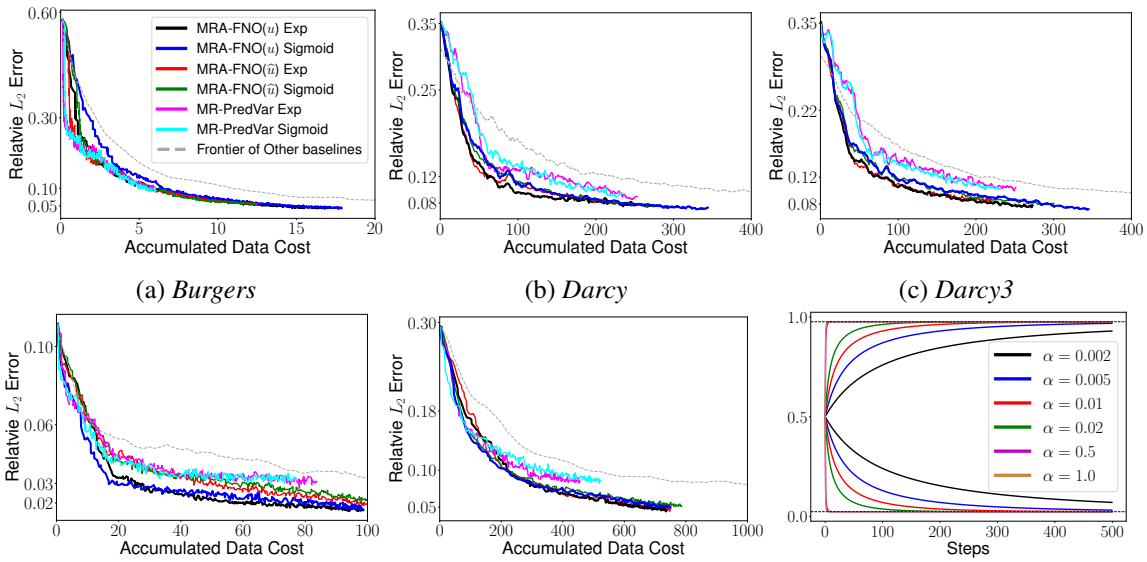


Figure 5: Graphical representation of our probabilistic multi-resolution FNO. Here P is the FFN that lifts the input function to higher-dimensional channel space, Q_1 is the FFN for channel-wise projection and producing the predictive mean, and Q_2 is a convolution net plus another FFN to produce the predictive variance in the log space.



(a) *Burgers* (b) *Darcy* (c) *Darcy3*

(d) *Diffusion* (e) *NS* (f) *Cost Scheduler (Sigmoid)*

Figure 6: Relative L_2 error *vs.* accumulated data cost (a-e) and the cost schedule with a sigmoid-based decay. Each method ran 300 active learning steps for NS, and 500 steps for all the other tasks. Note that different methods can end up with different total data cost (after running the same number of steps).

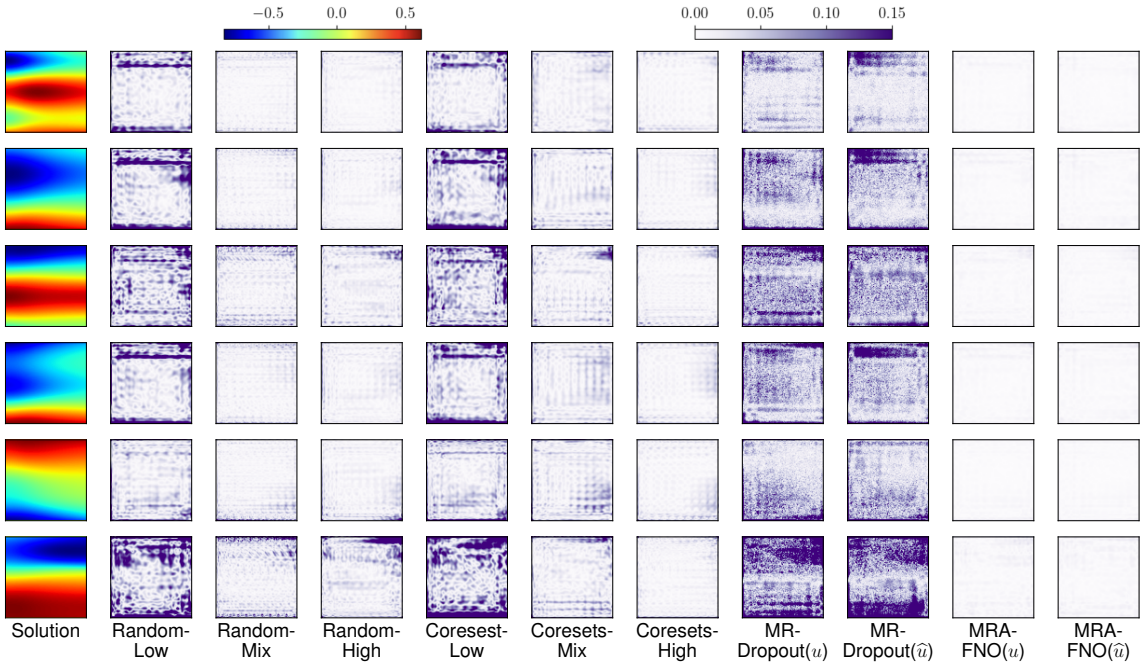


Figure 7: Point-wise error on *Diffusion*.