

MODEL AGGREGATION: MINIMIZING EMPIRICAL VARIANCE OUTPERFORMS MINIMIZING EMPIRICAL ERROR

THÉO BOURDAIS^{†,*} AND HOUMAN OWHADI[†]

ABSTRACT. Whether deterministic or stochastic, models can be viewed as functions designed to approximate a specific quantity of interest. We propose a data-driven framework that aggregates predictions from diverse models into a single, more accurate output. This aggregation approach exploits each model’s strengths to enhance overall accuracy. It is non-intrusive—treating models as black-box functions—model-agnostic, requires minimal assumptions, and can combine outputs from a wide range of models, including those from machine learning and numerical solvers. We argue that the aggregation process should be point-wise linear and propose two methods to find an optimal aggregate: Minimal Error Aggregation (MEA), which minimizes the aggregate’s prediction error, and Minimal Variance Aggregation (MVA), which minimizes its variance. While MEA is inherently more accurate when correlations between models and the target quantity are perfectly known, Minimal Empirical Variance Aggregation (MEVA), an empirical version of MVA—consistently outperforms Minimal Empirical Error Aggregation (MEEA), the empirical counterpart of MEA, when these correlations must be estimated from data. The key difference is that MEVA constructs an aggregate by estimating model errors, while MEEA treats the models as features for direct interpolation of the quantity of interest. This makes MEEA more susceptible to overfitting and poor generalization, where the aggregate may underperform individual models during testing. We demonstrate the versatility and effectiveness of our framework in various applications, such as data science and partial differential equations, showing how it successfully integrates traditional solvers with machine learning models to improve both robustness and accuracy.

1. Introduction

Many challenges in scientific computing and Machine Learning (ML) involve approximating functions using models. These models can vary significantly, ranging from numerical solvers that integrate differential equations to various ML methods and other approximation techniques. Their performance can differ widely across different benchmarks, as seen in the Imagenet leaderboards [24]. In some cases, no single model outperforms others across all scenarios; each has its own strengths and weaknesses. For instance, the Intergovernmental Panel on Climate Change (IPCC) report on model evaluation [15] highlights numerous evaluation metrics, showing that different models excel in different areas. When multiple models are available to predict the same quantity of interest, a key challenge arises: how can their predictions be combined most effectively? Ensembling ML models is a well-established practice with many successful applications, such as bagging, boosting, and Mixture of Experts (MoE) [28]. Gradient boosting, in particular, is widely regarded for its effectiveness, combining simplicity and accuracy in data analysis [7]. Most ensembling methods focus on training strategies that build and combine multiple models [28]. However, methods for aggregating predictions from existing models are less developed and often rely on simplistic techniques like averaging [13]. This highlights the need for more advanced approaches to aggregate predictions to better exploit diverse models’ unique strengths.

In this work, we present a method for aggregating predictions from any model, with minimal assumptions about their nature, treating them as arbitrary input-output functions, whether deterministic or stochastic. Our approach broadens the scope of model aggregation beyond machine learning and data analysis, enabling the integration of diverse methods (e.g., in scientific computing) aimed at estimating the same target quantities. In Section 3, we demonstrate that effective aggregation should be point-wise linear. To achieve this, we explore training the aggregate by directly minimizing its empirical error, a method we call Minimal Empirical Error Aggregation (MEEA). However, as we illustrate through pathological examples in Sections 3.2.1 and 3.2.2, as well as in the computations presented in Section 3.3, MEEA may simply turn individual models into features for interpolating the target and thereby suffer from overfitting and poor generalization. To address these shortcomings, we propose a new approach, Minimal Empirical Variance Aggregation (MEVA), which aggregates models by estimating their errors, as detailed in Section 4. Figure 1 provides an overview of the aggregation methods discussed. We validate our approach on a data science task in Section 5.1 and two operator learning tasks in Section 5.1. In all cases, MEVA outperforms direct error minimization, resulting in an aggregate model that is more robust and effective than any individual model.

2. Related work

Ensemble methods combine multiple models to improve accuracy. They can be broadly categorized into fusion and selection [28]. In the fusion setting, models are trained on the entire input space, and their output is combined, often using a weighted average. Examples include Bagging [9], random forests

[†]Department of Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA 91125, USA
E-mail addresses: theo.bourdais@caltech.edu, owhadi@caltech.edu

*Corresponding author

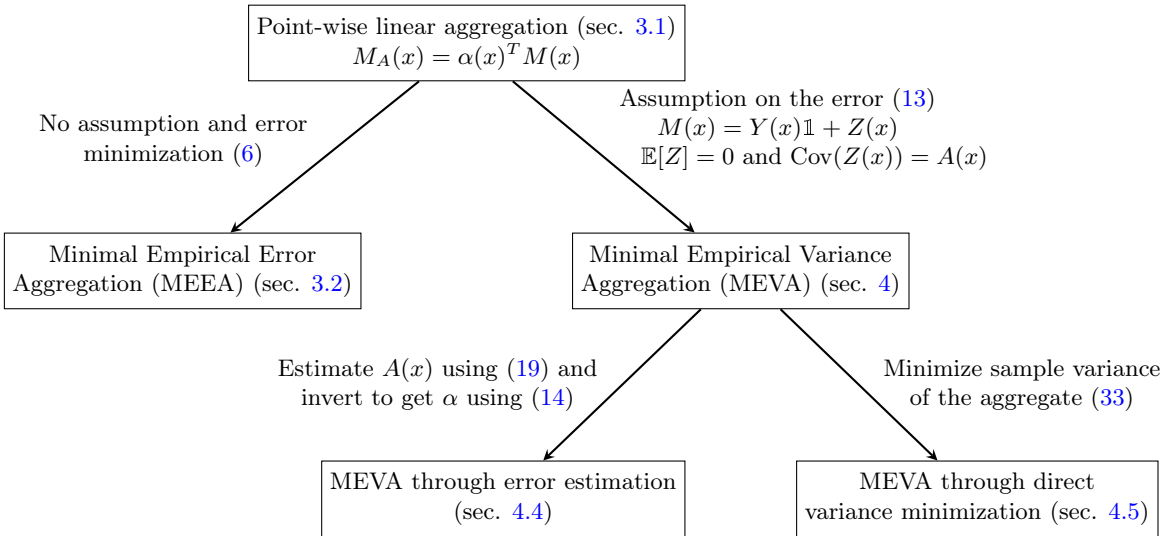


FIGURE 1. The different empirical aggregation methods presented in this paper and the main equations to justify them.

[10], and boosting [26]. In selection methods, local experts are trained, and their output is selected based on whether the input belongs to their domain of expertise. Mixture of Experts (MoE) [21] are selection methods where multiple experts and a gating model that selects responses are jointly trained. While inspired by these methods, our approach assumes the models are already trained and focuses solely on their aggregation. Consequently, our framework must be able to aggregate both strong and weak models. In the ensembling literature, weak models perform slightly better than random chance and are simple to train, while strong models perform better but are more expensive to train [25].

The simplest way to combine multiple model predictions is by averaging them, as in [14]. More generally, aggregations often use a linear combination of the predictions, which will be theoretically justified, especially in Gaussian cases, in section 3.1. The combination weights may be constant, as in gradient boosting [26], or output-dependent. In that case, they can be set by a probabilistic model [23] or be trained. [18] uses trainable MoE layers to combine features of different experts.

A growing trend is to apply statistical inference/ML methods in scientific computing [22, 6, 12, 8]. In this work, we use a statistical/ML framework to aggregate methods, even if they may not be inherently inference/ML-based.

3. The minimal error aggregation

In this section, we motivate the use of aggregation techniques with an ideal case (section 3.1), as well as the use of a specific form of aggregation in equation (2) and a minimal error aggregation loss (6). However, we will see, through two pathological examples in section 3.2 and a computation in the Gaussian Process (GP) case in section 3.3, that this direct loss (6) should not be used.

3.1. Best linear aggregate when model correlations are known. An effective way to derive an aggregation method is to define it as optimal with respect to a specific loss function. If the target function $Y(x)$ and the models $M_i(x)$ are viewed as random variables on the same probability space, a commonly used loss function closely related to concepts like conditional expectation is the Mean Square Error (MSE). In this context, an aggregated model $M_A(x)$ can be defined as any measurable function of the models that minimizes the expected squared error:

$$M_A(x) := \underset{f \text{ measurable}}{\operatorname{argmin}} \mathbb{E} [(Y(x) - f(M_1(x), \dots, M_n(x)))^2] = \mathbb{E}[Y(x) | M_1(x), \dots, M_n(x)], \quad (1)$$

where the second equality follows from the L_2 characterization of conditional expectation. While computing the conditional expectation can be intractable [1], this computation reduces to solving a linear system in the Gaussian case. Specifically, if the vector $(Y(x), M_1(x), \dots, M_n(x))$ is Gaussian with mean 0 (see remark below), then the conditional expectation in the equation above is a linear combination of the models' outputs. [23] uses this assumption to aggregate Gaussian processes. [4] demonstrates that any consistent and rational aggregation of models must be a weighted average that is point-wise linear. Finally, many ensembling methods (random forest [10], MoE [27]) also use a linear combination of models. Motivated by this, we restrict our aggregation approach to a pointwise linear combination of the models:

$$M_A(x) = \alpha^*(x)^T M(x) \quad (2)$$

With this simplification, aggregation can be achieved by determining the Minimal Error Aggregation (MEA) $\alpha^*(x)$ such that:

$$\alpha^*(x) = \operatorname{argmin}_{\alpha \in \mathbb{R}^n} \mathbb{E} \left[(Y(x) - \alpha^T M(x))^2 \right] \quad (3)$$

By definition, the MEA is the best possible point-wise linear aggregation. A simple computation yields the following form for the aggregation

$$M_A(x) = \mathbb{E} \left[Y(x) M(x)^T \right] \mathbb{E} \left[M(x) M(x)^T \right]^{-1} M(x) \quad (4)$$

This equation reveals that the MEA requires knowledge of both the correlation matrix of the models, $\mathbb{E} [M(x)M(x)^T]$, as well as the correlation between the models and the target $\mathbb{E} [Y(x)M(x)]$. In the next section, we will demonstrate that the aggregated model can achieve exceptional performance when these correlations are perfectly known.

Remark. In the general case with non-zero mean, the aggregation of a Gaussian vector would be affine. Moreover, other methods, such as boosting [26], use affine aggregation instead. Since we do not assume anything on the models, affine coefficients can be recovered by adding an extra constant model, i.e. by a linear aggregation of $\tilde{M}(x) = (M(x), 1)$. In section 3.2, we will see that this can lead to catastrophic failure for empirical error minimization. Section 4.4 shows that empirical variance minimization provides a more robust approach for estimating bias.

3.1.1. Aggregating PDE solvers in the Gaussian setting. We consider here a best-case scenario for MEA. [12] introduces a Gaussian processes framework for solving Partial Differential Equations (PDE)der the Laplace equation as a simple example:

$$\begin{cases} -\Delta u(x) = f(x) & \text{for } x \in [-1, 1]^2 \\ u(x) = g(x) & \text{for } x \in \partial([-1, 1]^2) \end{cases} \quad (5)$$

Place a Gaussian prior $\xi \sim \mathcal{N}(0, K)$ on the solution where K is a Radial Basis Function (RBF) kernel.

Given interior collocation points $X_i \in [-1, 1]^2$ and boundary collocation points $X_j^b \in \partial([-1, 1]^2)$, we will approximate the solution u with the Gaussian process estimator $\hat{u} = \mathbb{E} [\xi \mid -\Delta \xi(X) = f(X), \xi(X^b) = g(X^b)]$. Fixing the boundary collocation points, sample 100 sets of 60 interior collocation points $X^{(k)}$ to obtain 100 different models of the solution of the PDE, that we will denote

$$\hat{u}^{(k)} := \mathbb{E} \left[\xi \mid -\Delta \xi(X^{(k)}) = f(X^{(k)}), \xi(X^b) = g(X^b) \right].$$

Collocation points are sampled according to the uniform distribution in the domain and independently across points and sets. Since the conditional means $\mathbb{E} [\xi \mid -\Delta \xi(X^{(k)}), \xi(X^b)]$ are also Gaussian, we can employ representer formulas for aggregating these models according to (4) by taking $Y(x) = \xi(x)$ and $M_k(x) = \mathbb{E} [\xi(x) \mid -\Delta \xi(X^{(k)}), \xi(X^b)]$. Since the resulting approach reduces to nested kriging [23], (4) is not only the best (in mean squared error) linear aggregate. It is also the best non-linear aggregate of all models. Figure 2 shows the results of this experiment. While the individual models are far from approximating the solution because of the low number of collocation points chosen, the aggregate 2d is very close to the true solution. It performs significantly better than any individual models used in the aggregation. We may also note that averaging over all models does not reach the same level of accuracy. This example shows that there can be massive improvements in aggregation performance from a simple method, such as averaging, by optimally incorporating information about covariances across models and between models and the target.

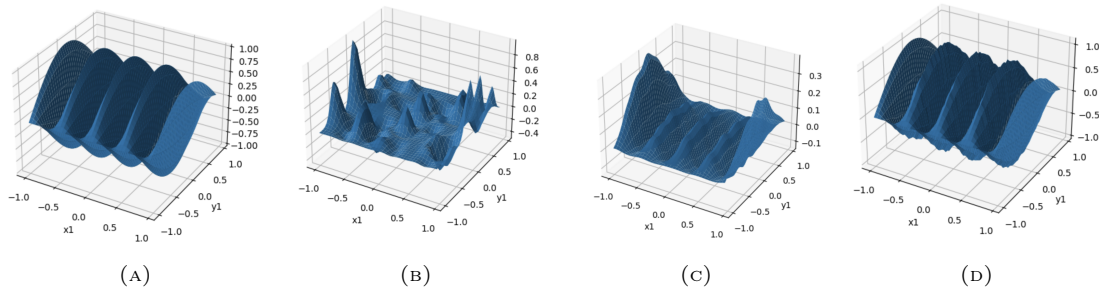


FIGURE 2. (a) Real solution of the PDE (b) One of the models aggregated (c) Uniform average of all models (d) Proposed aggregate (4)

3.2. Data-driven aggregation. We will now consider the situation where the quantities $\mathbb{E}[Y(x)M(x)]$ and $\mathbb{E}[M(x)M(x)^T]$ required for the best linear aggregate (4) may be ill-defined (e.g., because the underlying models are not stochastic) or difficult to estimate. In this situation, a natural alternative approach is to estimate the aggregation coefficients α^* appearing in (2) directly from data by a minimizer $\hat{\alpha}$ of a (possibly regularized) empirical version of the loss (3):

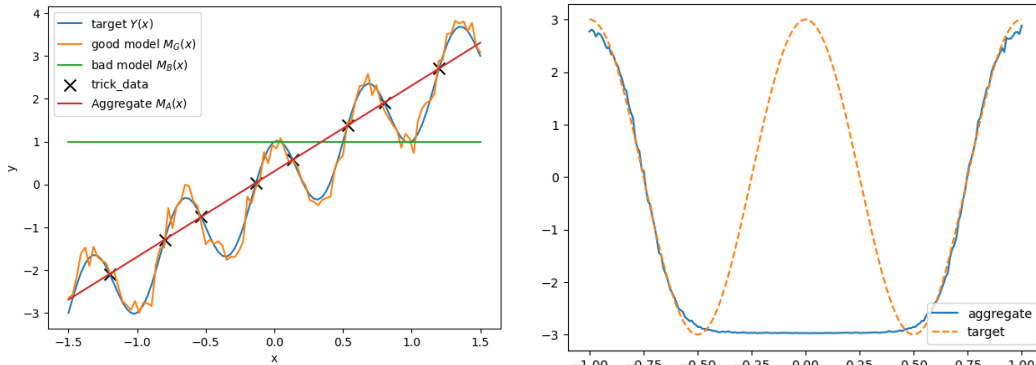
$$\hat{\alpha} = \operatorname{argmin}_{a \in \mathcal{H}} \sum_{i=1}^N |Y^i - a(X^i)^T M(X^i)|^2 + \lambda \|a\|_{\mathcal{H}}^2, \quad (6)$$

where the $(X^i, Y^i := Y(X^i))$ are N data points; \mathcal{H} is a set of functions, such as a Reproducing Kernel Hilbert Space (RKHS), a set of functions obtained via gradient boosting or from a neural network architecture; $\lambda \geq 0$ and $\|a\|_{\mathcal{H}}^2$ is a regularizing norm (e.g., an RKHS norm). We call this aggregation Minimal Empirical Error Aggregation (MEEA). While this transition from expected loss to empirical loss may seem well-founded, it can introduce a significant loss of information. The following section showcases two pathological examples that exhibit unexpected behavior due to this loss of information. In both cases, we set up a regression problem with a target $Y(x) \in \mathbb{R}$, for $x \in \mathbb{R}$. We have two types of models: a good model M_G and bad models M_B . Using data, we try to aggregate these models and showcase why the empirical loss (6) may not be suitable for ensembling models from data.

3.2.1. A dubious trend. This first example is a basic aggregation with linear coefficients. Figure 3a represents the results of this experiment. We pick:

$$\begin{cases} Y(x) = 2x + \cos(3\pi x) \\ M_G(x) = Y(x) + \epsilon(x) \text{ where } \epsilon(x) \sim \mathcal{N}(0, 0.2) \\ M_B(x) = 1 \end{cases} \quad (7)$$

We also choose a linear aggregation, which means that we will find a_G, a_B, b_G, b_B s.t. $\hat{\alpha}(x) = (a_G x + b_G, a_B x + b_B) \in \mathbb{R}^2$. Finally, we pick some trick data $X = (0.8 - 2/3 * 4, 0.8 - 2/3 * 3, \dots, 0.8, -0.8, -0.8 + 2/3 * 1, \dots, -0.8 + 2/3 * 4)$ so that the $Y(X^i) = Y^i$ form a line. We may see that M_G has a lower error than M_B on each X^i by evaluating the models at these data points. Thus, we expect an aggregation method to use the good model primarily in the aggregation. However, the opposite behavior is observed. Minimizing the loss (6) with $\lambda = 0$, we obtain that the coefficient on the good model is 0, so M_G is completely ignored, which is counterintuitive. Instead, the aggregate uses the bad model, M_B , to directly estimate Y by linear regression. This shows that the MEEA simply employs models as features to interpolate the data points instead of weighting each model according to its estimated accuracy.



(A) Pathological example described in section 3.2.1 (B) Pathological example described in section 3.2.2

FIGURE 3. Pathological examples. Both share the same structure. Top: the models, truth, and the aggregate. Bottom: coefficients $\hat{\alpha}$

3.2.2. Unknown region. In this second example illustrated in Figure 3b we select

$$\begin{cases} Y(x) = 3 \cos(2\pi x) \\ M_G(x) = Y(x) + \epsilon(x) \text{ where } \epsilon(x) \sim \mathcal{N}(0, 0.3) \\ M_B^1(x) = 3 \\ M_N^2(x) = -3 \end{cases} \quad (8)$$

Write $M = (M_G, M_B, M_N)^T$ for the vector defined by the three models. It is common for aggregation methods, such as MoE, to constrain $\hat{\alpha}$ to be a convex combination, which we implement here by seeking an aggregate of the form $M_A(x) = \operatorname{Softmax}(\hat{\alpha}(x))^T M(x)$ where $\operatorname{Softmax}(\hat{\alpha}(x))_k := \frac{\exp(\hat{\alpha}_k(x))}{\sum_{j=1}^3 \exp(\hat{\alpha}_j(x))}$. To

identify the functions $\hat{\alpha}_k$ we sample $N = 100$ points X_i uniformly in $[-1, -0.5] \cup [0.5, 1]$ leaving the region $[-0.5, 0.5]$ with no data and we then minimize

$$\min_{\alpha_1, \alpha_2, \alpha_3 \in \mathcal{H}_\kappa} \sum_{i=1}^N (Y(X_i) - \text{Softmax}(\hat{\alpha}(X_i))^T M(X_i))^2 + \lambda \sum_{k=1}^3 \|\alpha_k\|_\kappa^2 \quad (9)$$

where κ is an RBF kernel and $\|\cdot\|_\kappa$ is its associated RKHS norm. Using the representer theorem, we obtain that minimizers of (9) are of the form $\hat{\alpha}_k(x) = \sum_{i=1}^N \kappa(x, X_i) v_i^k$ where the coefficients $v_k \in \mathbb{R}^N$ minimize the reduced optimization problem

$$\min_{v_1, v_2, v_3 \in \mathbb{R}^N} \sum_{i=1}^N (Y(X_i) - \text{Softmax}(\hat{\alpha}(X_i))^T M(X_i))^2 + \lambda \sum_{k=1}^3 v_k^T \kappa(X, X) v_k \quad (10)$$

where $\kappa(X, X)_{i,j} = \kappa(X_i, X_j)$. In this example, the good model is consistently better than the bad models over the entire dataset. Nevertheless, our aggregate fits the dataset perfectly. Still, it fails to learn how the different models behave, leading to poor performance outside the dataset despite having a model with good performance everywhere. This pathological behavior arises because the aggregate fails to account for model errors in its approximation of Y .

3.3. Aggregation using vector-valued Gaussian processes and matrix-valued kernels. We will now model aggregation as vector-valued Gaussian process regression to elucidate the pathologies observed in the two abovementioned examples. Under this vector/matrix-valued generalization of GPs/kernels [2] is provided in appendix A. We choose a matrix-valued kernel $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}^{n \times n}$, and write \mathcal{H}_K for its associated Reproducing Kernel Hilbert Space (RKHS). Given the data $(Y(X_1), M(X_1)), \dots, (Y(X_N), M(X_N))$ we approximate the aggregation coefficients in (2) with

$$\hat{\alpha} = \operatorname{argmin}_{\alpha \in \mathcal{H}_K} \sum_{i=1}^N [Y(x_i) - \alpha(x_i)^T M(x_i)]^2 + \lambda \|\alpha\|_{\mathcal{H}_K}^2. \quad (11)$$

(11) then yields the following aggregate (see Appendix B):

$$M_A(x) = \hat{\alpha}(x)^T M(x) = \tilde{k}(x, X) (\tilde{k}(X, X) + \lambda I)^{-1} Y \quad (12)$$

where $\tilde{k}(x, y) = M(x)^T K(x, y) M(y)$. This is also the GP regressor $M_A = \mathbb{E}[\xi | \xi(X) = Y + \mathcal{N}(0, \lambda I)]$ with prior $\xi \sim \mathcal{N}(0, \tilde{k})$. This observation is crucial as it shows that the model values are used as a feature for the regression, as expected. However, it demonstrates that the Mean Empirical Error Aggregate (MEEA) only tries to regress Y without considering the underlying models' accuracy. This observation is counterintuitive as we expect model aggregation to leverage the individual models to approximate the target function better rather than merely regressing directly to the target. This behavior is particularly evident in pathological examples, illustrating the limitations of the direct regression approach.

4. Aggregation through error estimation: the Minimal Variance Aggregate (MVA)

We now introduce a different strategy for aggregating models. Since, as observed in section 3.1, pointwise linear aggregation has many benefits, we will still employ the pointwise linear aggregation formula (2) but modify the underlying stochastic model to take into account model errors.

4.1. Modelling the error. Our models, represented in the vector M , are assumed to be unbiased estimators of Y . Thus, the errors of each model are interpreted as follows:

$$M(x) = Y(x) \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} + Z(x) \quad \text{where} \quad \begin{cases} \mathbb{E}[Z(x)] = 0 \\ \text{Cov}[Z(x)] = A(x) \end{cases} \quad (13)$$

where $\mathbb{1} := (1, \dots, 1)^T$ is the all ones vector. Under this assumption, the unbiased linear estimate of $Y(x)$ given $M(x)$ with minimal variance, which we call the Minimal Variance Aggregate (MVA) is:

$$M_A(x) = \frac{\mathbb{1}^T A(x)^{-1} M(x)}{\mathbb{1}^T A(x)^{-1} \mathbb{1}} \quad (14)$$

This aggregation uses the information of the precision matrix to decide how to combine the models. If the models are indeed unbiased, it is equivalent to the Minimal Error Aggregation (MEA). Despite summing to 1, the coefficients $\alpha(x)$ are not necessarily positive, giving added flexibility in case models have negative conditional correlations.

Unbiased estimators. Since the estimators are unbiased, any linear combination with weights $\alpha_i(x)$ s.t. $\sum_{i=1}^n \alpha_i(x) = 1$ is unbiased as well. Thus, the aggregate we choose here is the lowest variance one, i.e., the Best Linear Unbiased Estimate (BLUE).

Unlike the MEA, which directly minimizes the L_2 loss, this formulation uses an additional assumption, Gaussian errors $Z(x)$ with zero mean, to obtain the best aggregation through a variance minimization problem. This variance minimization is done in two steps: first, estimate model errors covariances $A(x)$; second, aggregate the models using the error estimates and equation (14). It has several advantages:

- It is a probabilistic model of our aggregation. Unlike L_2 loss minimization, this approach explicitly models the error, allowing for more refined reasoning in specific applications, such as in PDE aggregation.
- It is a pointwise linear aggregation which is required for consistency [3].
- By reframing the aggregation problem as an error estimation challenge, we force the aggregation approach to consider a model effective if it closely approximates the target.
- With L_2 loss minimization, the models and the aggregator aim to estimate the target directly. As a result, the quality of the aggregate is inherently tied to the performance of the aggregator. However, when the aggregator focuses solely on the errors without direct observation of the target, the task becomes effectively divided between target estimation and error estimation.

The following sections describe how to perform Minimal Variance Aggregation (MVA) when the error covariance matrix $A(x)$ is not known, thus defining the Minimal Empirical Variance Aggregate (MEVA) in section 4.4.

Independent errors. If models errors are independent then $A(x)$ is diagonal and

$$M_A(x) = \frac{\sum_{i=1}^n p_i M_i(x)}{\sum_{i=1}^n p_i} \text{ where } A^{-1}(x) = \text{Diag}(p_1(x), \dots, p_n(x)) \quad (15)$$

Since $A(x)$ is a covariance matrix, it is positive semi-definite and $p_i(x) \geq 0$, so (15) is a convex combination. This is to be contrasted with (14) where the combination may be non convex if model errors are correlated.

4.2. Approximating the covariance matrix. To compute our aggregation, we must compute the covariance matrix of the error for all $x \in \mathcal{X}$ from the data. This poses several challenges. First, we must approximate a symmetric matrix everywhere on the domain, needing $n(n+1)/2$ coefficients to aggregate n models. Moreover, we must ensure this matrix is always positive-definite. To simplify the approximation and to easily enforce the positive-definite condition, we suppose that for all x , $A(x)$ can be diagonalized using a fixed eigenbasis P . Then, its positive eigenvalues are approximated by $e^{\lambda_i(x)}$, where the λ_i are functions to be regressed from the data. Thus, we approximate $A(x)$ with

$$\hat{A}(x) = P^T \begin{pmatrix} e^{\lambda_1(x)} & 0 & \dots \\ 0 & \ddots & 0 \\ \vdots & 0 & e^{\lambda_n(x)} \end{pmatrix} P \text{ where } \begin{cases} PP^T = I_n \\ \lambda_i \text{ are regressors} \end{cases} \quad (16)$$

This formulation has the advantage of symmetrizing the covariance and precision matrix estimation, as the log-eigenvalues of both matrices are opposite. This is desirable because while we are interested in the precision matrix, the covariance matrix is easier to estimate.

We will now estimate the covariance matrix. Let Z^1, \dots, Z^N be i.i.d. samples of the random variable $Z \in \mathbb{R}^n$ such that $\mathbb{E}[Z] = 0$. An unbiased estimate of $\text{Cov}[Z]$ is

$$\hat{A} = \frac{1}{N} \sum_{i=1}^N Z^i (Z^i)^T \quad (17)$$

Observe that \hat{A} can be identified as the minimizer of the following loss involving the Frobenius norm $\|\cdot\|_F$:

$$\hat{A} = \underset{\Sigma \in \mathbb{R}^{n \times n}}{\text{argmin}} \sum_{i=1}^N \|\Sigma - Z^i (Z^i)^T\|_F^2 \quad (18)$$

We will now regularize this formulation to approximate the model errors covariance matrix for all values of x . Writing e^i for the vector with entries defined as the sample errors $(e^i)_k = M_k(X^i) - Y(X^i)$, we identify the functions λ_i as

$$\lambda = \underset{l \in \mathcal{H}}{\text{argmin}} \sum_{i=1}^N \|P^T \text{Diag}(e^{l_k(X^i)})P - e^i (e^i)^T\|_F^2 + a \|l\|_{\mathcal{H}}^2 \quad (19)$$

$$= \underset{l \in \mathcal{H}}{\text{argmin}} \sum_{i=1}^N \sum_{k=1}^n \left(e^{l_k(X^i)} - (P e^i)_k^2 \right)^2 + a \|l\|_{\mathcal{H}}^2 \quad (20)$$

Here, we denote the entries of the vector-valued function l by l_k , and \mathcal{H} represents the space of vector-valued functions selected to approximate the log-eigenvalues. This space could be a RKHS associated with a particular kernel, a neural network with a specific architecture, or any other normed function space. The notation $\|\cdot\|_{\mathcal{H}}$ refers to the regularization norm intrinsic to that space, such as the RKHS norm or the L_2 -norm of the weights and biases in a neural network.

Rediscovering softmax. Taking $P = I_n$ in the proposed formulation our aggregation reduces to

$$M_A(x) = \text{Softmax}(-\lambda(x))^T M(x) \quad (21)$$

Where $\text{Softmax}(-\lambda(x))_i = e^{-\lambda_i(x)} / \sum_{k=1}^n e^{-\lambda_k(x)}$. This formulation is very common, particularly for Mixtures-of-Experts (MoE).

Underdetermination of MEA compared to MVA. Notice that MEA uses only one equation at each datapoint, namely $Y(X^k) \approx \alpha(X^k)^T M(X^k)$. For a given $Y(X^k)$ and $M(X^k)$, there are many values of $\alpha(X^k)$ satisfying this constraint, indicating that the system is underdetermined. In such a situation, the regularization will play a crucial role. On the other hand, MVA uses as many equations as there are models at each data point: $e^{\lambda_k(X^i)} \approx (Pe^i)_k^2$. This makes λ better determined and less reliant on properly choosing the regularization to achieve good results.

4.3. Choosing the matrix P . We have explored two options for the matrix P :

- $P = I$, the identity matrix, is the simplest choice.
- P as the orthonormal basis that diagonalizes the empirical covariance matrix $C := \frac{1}{N} \sum_{i=1}^N e_i e_i^T$.

In the experiments presented below, we only show results for $P = I$. This choice is straightforward and delivers accuracy comparable to or better than using the eigenvectors of C .

We also observed that using the empirical covariance matrix can render the method unstable if the models are highly correlated. Indeed, consider, for instance, two models whose errors are significantly correlated, leading to the following empirical covariance matrix:

$$C := \begin{pmatrix} 1.1 & 0.9 \\ 0.9 & 1 \end{pmatrix} \quad (22)$$

In this case, the second eigenvector of C , $P_2 \approx \frac{1}{\sqrt{2}}(0.97, -1.02)$, results in $\mathbb{1}^T P_2 \approx -0.04$, which effectively subtracts the two models. This eigenvector is associated with a small eigenvalue (0.15), indicating that the difference between the two models has a low variance on average. Since equation (14) depends on the inverse of the estimated covariance matrix, a small estimated variance eigenvalue $e^{\lambda_2(x)}$ leads to $\alpha(x)$ having large coefficients with opposing signs. In this scenario, we observed that the aggregation method can become sensitive to small errors and thereby suffer from poor generalization.

4.4. The Minimal Empirical Variance Aggregate (MEVA) through error estimation. To summarize, we have defined the Minimal Variance Aggregate (MVA) in (14) by assuming the models are unbiased. To estimate this aggregate from data, we must approximate the error covariance matrix $A(x)$ for all x . To do so, we choose an orthonormal matrix P as described in section 4.3, and a set of functions \mathcal{H} to approximate the log-eigenvalues of $A(x)$ using equation (19) as described in section 4.2. Combining these three sections, the error estimation-based Minimal Empirical Variance Aggregate (MEVA) is defined as

$$\lambda = \underset{l \in \mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^N \sum_{k=1}^n \left(e^{l_k(X^i)} - (Pe^i)_k \right)^2 + a \|l\|_{\mathcal{H}}^2 \quad (23)$$

$$\alpha(x)^T = \frac{\mathbb{1}^T P^T D(x) P}{\mathbb{1}^T P^T D(x) P \mathbb{1}} \text{ where } D(x) = \text{Diag}[\exp(-\lambda_k(x)), k = 1, \dots, n] \quad (24)$$

$$M_A(x) = \sum_{i=1}^n \alpha_i(x) M_i(x) \quad (25)$$

where $(e^i)_k = M_k(X^i) - Y(X^i)$.

Adding a bias correction. Assuming the models $M(x)$ to be unbiased is necessary to obtain a linear aggregate, as adding a bias correction will result in an affine aggregation of the form $M_A(x) = \alpha^T(x)M(x) + \beta(x)$. This assumption may, however, be violated in practice, and in this case, bias will be interpreted as variance. In a context with limited data and little knowledge of the models and target, it would be difficult to know if an error can be attributed to bias or variance. In the case where it is clear models are biased and interpreting it as variance hurts accuracy, we may modify our method by accounting for the bias

$$M(x) = Y(x)\mathbb{1} + \tilde{Z} \text{ where } \begin{cases} \mathbb{E}[\tilde{Z}(x)] = \mu(x) \\ \text{Cov}[\tilde{Z}(x)] = A(x) \end{cases} \quad (26)$$

In such a model, we must, in addition to λ , train μ . Using the definition of \tilde{Z} , we know the loss for μ is of the form $\sum_{i=1}^N (\mu(X^i) - e^i)^T A^{-1}(X^i) (\mu(X^i) - e^i)$. Taking $P = I$, this bias-corrected aggregation would be:

$$\tilde{\lambda}, \tilde{\mu} = \underset{l, m \in \mathcal{H}_1 \times \mathcal{H}_2}{\operatorname{argmin}} \sum_{i=1}^N \sum_{k=1}^n \left[\left(e^{l_k(X^i)} - (e_k^i - m_k(X^i))^2 \right)^2 + e^{-l_k(X^i)} \left(m_k(X^i) - e_k^i \right)^2 \right] \quad (27)$$

$$+ a \|l\|_{\mathcal{H}_1}^2 + b \|m\|_{\mathcal{H}_2}^2 \quad (28)$$

$$\tilde{\alpha}(x) = \text{Softmax}(-\tilde{\lambda}(x)) \quad (29)$$

$$\tilde{M}_A(x) = \sum_{i=1}^n \tilde{\alpha}_i(x) (M_i(x) - \tilde{\mu}_i(x)) \quad (30)$$

where $\mathcal{H}_1, \mathcal{H}_2$ are two spaces of functions used to approximate the log-eigenvalues of $A(x)$ and the bias of the models.

In the Boston housing dataset example in section 5.1, we tried to implement this bias-corrected aggregation, and it did not yield better results than the simpler method MEVA. While we do not prescribe its use in general, we will not consider this model in the rest of the paper.

4.5. An alternative, direct Minimal Empirical Variance Aggregation loss. We observed in section 4.1 that our approach can be interpreted as minimizing the aggregate’s variance, as we first estimate the errors of the models and then derive an aggregation using (14). In our model (13), any combination α s.t. $\mathbb{1}^T \alpha = 1$ is an unbiased estimator (and any unbiased estimator has to satisfy this property). To get our aggregation, we chose the best linear unbiased estimator (BLUE), by minimizing the variance of $\alpha(x)^T M(x)$. Under the model (13), $Cov[M(x)] = A(x)$, so our aggregate is

$$\alpha(x) = \operatorname{argmin}_{\nu \in \mathbb{R}^n, \sum_{i=1}^n \nu_i = 1} \nu^T A(x) \nu \quad (31)$$

Instead of estimating $A(x)$ and inverting this estimation to obtain the minimal variance aggregate as in (14), we propose using an empirical version of the above loss. Specifically, let the empirical covariance matrices A_i be defined as:

$$A_i := \begin{cases} P^T \operatorname{Diag}((Pe^i)_k^2, k = 1.., n) P & \text{for general } P \\ \operatorname{Diag}((Y^i - M_k(X^i))^2, k = 1.., n) & \text{if } P = I_n \end{cases} \quad (32)$$

and write \mathcal{H}_1 for a normed subspace of the space of unbiased aggregators $\{u : x \mapsto u(x) \in \mathbb{R}^n \text{ s.t. } \sum_{i=1}^n u_i(x) = 1, \forall x\}$. Then the proposed alternative approach identifies an aggregator $\tilde{\alpha}$ by minimizing the sample variance of the aggregate:

$$\tilde{\alpha} = \operatorname{argmin}_{u \in \mathcal{H}_1} \sum_{i=1}^N u(X^i)^T A_i u(X^i) + a \|u\|_{\mathcal{H}_1}^2 \quad (33)$$

We employ this strategy in the Boston dataset example (section 5.1). Here, \mathcal{H}_1 is a set of vector-valued functions parameterized by a neural network, where the output is constrained to have positive entries summing to one via a softmax layer. The term $\|u\|_{\mathcal{H}_1}^2$ is an L^2 -norm regularizer applied to the network’s weights and biases. This method is both straightforward and practical.

5. Experiments

We present three experiments to showcase the effectiveness of our method. The code can be found in the paper’s [repository](#).

5.1. Aggregation on the Boston housing dataset. The Boston housing dataset [17] is a popular benchmarking dataset for Machine Learning (ML) applications. It contains $N = 506$ samples of 14 variables relating to housing prices in Boston, the task being to predict the median value of a home based on factors such as criminality or tax rates. We split this dataset into train, validation, and test sets. To evaluate our method, we train several standard ML methods on the train set: linear regression, decision tree, random forest, Support Vector Regression (SVR), k-neighbors, and gradient boosting. We then train our aggregation on the validation set before testing on the test set. To get a fair comparison, we also trained the ML methods on the train and validation sets combined so that they see the same amount of data as the aggregate. We first implement our aggregation as described in section 4.2, using GP regressors, i.e., we place a Gaussian prior $\lambda_i \sim \mathcal{N}(0, \kappa)$ on the unknown λ_i and we compute their MAP estimator given available data. This is equivalent to minimizing (19) over $\lambda_i \in \mathcal{H}_\kappa$ where \mathcal{H}_κ is the RKHS defined by the kernel κ . After experimenting with many kernels, the best we found uses the Matérn kernel $\kappa_{\text{Matérn}}(u, v, \rho) = \left(1 + \frac{\sqrt{3}\|u-v\|}{\rho}\right) \exp\left(-\frac{\sqrt{3}\|u-v\|}{\rho}\right)$, and is defined as

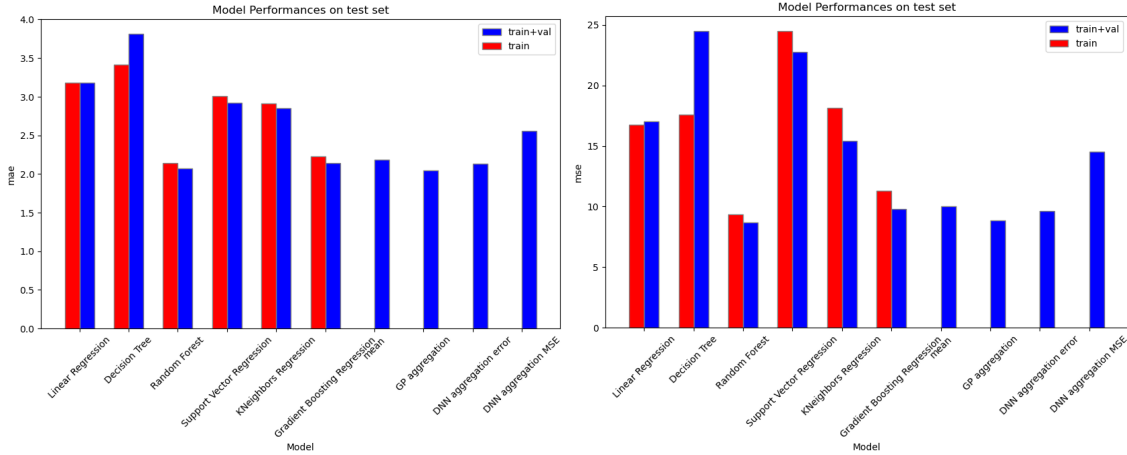
$$\kappa(x, y) = \kappa_{\text{Matérn}}(x, y, \rho_1) + \kappa_{\text{Matérn}}(M(x), M(y), \rho_2) \quad (34)$$

with adequate choice of ρ_1, ρ_2 . This formulation takes into account both the input x and the prediction $M(x)$ to output the aggregation coefficients $\alpha(x)$.

The results of this experiment are shown in figure 4. The aggregate, denoted *GP aggregation*, performs better than all individual models being aggregated. Furthermore, it performs better in Mean Absolute Error (MAE) than any ML model when fed the same amount of data. However, this performance increase is small and is not seen in the MSE. We also plot the mean aggregation, which performs well in this situation.

We conduct a second experiment to demonstrate the effectiveness of our proposed method compared to direct MSE minimization, as defined by the loss function in equation (6). In this experiment, we use a small, fully connected neural network¹ and train it using two different loss functions: the direct MSE described in section 3, and the variance minimization loss outlined in section 4.5. The results, presented in figure 4, highlight the impact of the chosen loss function. Since the only difference between the two approaches is the loss function, the results clearly illustrate the advantages of error estimation. Specifically,

¹See details in the paper’s [repository](#)



(A) Comparison in Mean Absolute Error (MAE)

(B) Comparison in Mean Squared Error (MSE)

FIGURE 4. Comparisons of the performance of the different models on the Boston housing dataset. Red bars are the performance of models trained using the training set and used in the aggregation. Blue bars show models trained on the training and validation set (*train+val*) to get a fair comparison with the aggregations, which also use training and validation sets. The aggregation does not use the models trained on *train+val*.

the minimal error aggregation (denoted *DNN aggregation MSE* in the figure) performs significantly worse compared to our minimal variance aggregation (denoted *DNN aggregation error* in the figure). It also performs worse than the two best models aggregated by a large margin.

5.2. Aggregation of PDE solvers. In this section, we introduce the approach for combining Partial Differential Equation (PDE) solvers through model aggregation. These examples highlight the flexibility of our method, demonstrating its ability to aggregate a wide range of models, including both traditional and machine learning (ML) methods. By incorporating classical PDE solvers into the aggregation framework, we lower the accuracy threshold necessary for the aggregated solution to outperform individual models. Notably, this threshold is significantly below the typical 1% relative error benchmark targeted by most ML-based PDE solvers. We will begin by formulating the aggregation problem within the operator learning context. Following that, we present two experimental case studies that showcase the effectiveness of our approach.

5.2.1. Operator Learning in the Aggregation Context. Consider the Laplace equation with zero Dirichlet boundary conditions on the domain $\Omega = [0, 1]^2$ as an illustrative example of a PDE:

$$\begin{cases} -\Delta u^\dagger(\omega) = f(\omega) & \text{for } \omega \in \Omega \\ u^\dagger(\omega) = 0 & \text{for } \omega \in \partial\Omega \end{cases} \quad (35)$$

where $f \in \mathcal{F} \subset L^2(\Omega)$ and $u^\dagger \in \mathcal{Y} \subset H^2(\Omega) \cap H_0^1(\Omega)$ is the solution. The solution operator maps the source term f to the solution u^\dagger . In this context, the solution operator is defined as:

$$S: \begin{cases} \mathcal{F} & \rightarrow \mathcal{Y} \\ f & \mapsto u^\dagger \end{cases} \quad (36)$$

This operator is often approximated using a PDE solver, which can be based on methods like finite element analysis [5] or spectral methods [16]. Alternatively, the operator can be learned directly from pairs of input/output solutions, a supervised learning task known as operator learning [19]. Since both PDE solvers and machine learning methods approximate operators, we can naturally frame our aggregation task as an aggregation of multiple operators (or PDE solvers), denoted M_1, \dots, M_n . The aggregated operator M_A is then expressed as:

$$M_A(f) = \sum_{i=1}^n \alpha_i(f) M_i(f) \in \mathcal{F} \quad (37)$$

where α is an operator that maps \mathcal{F} to $L^2(\Omega, \mathbb{R}^n)$.

Remark: Although we presented the most straightforward aggregation operator, $M_A(f, M(f)) = \sum_{i=1}^n \alpha_i(f) M_i(f)$, this is not the only possible approach. In the context of operators, there exists many alternatives that remain pointwise linear with respect to the models. For instance, another possible approach is to aggregate Fourier coefficients:

$$\tilde{M}_A(f) = \mathcal{T}^{-1} \left(\sum_{i=1}^n \alpha_i(f) \mathcal{T}(M_i(f)) \right) \quad (38)$$

where \mathcal{T} represents the Fourier transform. While we explored this option in the following examples, it did not perform as well as the simpler aggregation method outlined above and is thus not included in this paper.

5.2.2. The operator aggregation loss. To get the loss for our problem, we will first state the general loss in the operator learning setting, then modify it for increased performance.

Since the aggregation loss (19) is defined for real-valued objectives with an abstract input $x \in \mathcal{X}$, we must specify \mathcal{X} in the context of operator learning. We use $\mathcal{X} = \{(\omega, f) \in \Omega \times \mathcal{F}\}$, which means $x = (\omega, f)$ is a choice of point in space and source term. We define $Y(x) := S(f)(\omega) = u^\dagger(\omega) \in \mathbb{R}$. Given a grid $(\omega^1, \dots, \omega^{N_1}) \in \Omega$ and a set of input functions (f^1, \dots, f^{N_2}) , we get the $N_1 \times N_2$ datapoints $X^{i,j} := (\omega^i, f^j)$ for our aggregation. Adapting (19) to this setting, with $P = I$ yields

$$\lambda = \operatorname{argmin}_{l \in \mathcal{H}} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \sum_{k=1}^n \left(e^{l_k(X^{i,j})} - (Y(X^{i,j}) - M_k(X^{i,j})) \right)^2 + a \|l\|_{\mathcal{H}}^2 \quad (39)$$

For reference, (43) and (44) recall how to obtain the aggregation from λ .

Continuous limit: The above loss (39) is a discretization, using the grid $(\omega^1, \dots, \omega^{N_1})$, of the loss:

$$\min_{l \in \mathcal{H}} \sum_{j=1}^{N_2} \sum_{k=1}^n \left\| e^{l_k(f^j)} - (Y(f^j) - M_k(f^j)) \right\|_{L_2}^2 + a \|l\|_{\mathcal{H}}^2 \quad (40)$$

where $\|g\|_{L_2}^2 = \int_{\Omega} g(\omega)^2 d\omega$. This is important to note as operator learning problems must be viewed as discretized versions of the continuous limit [11]. In practice, it is necessary to properly treat cases where the discretization is nonuniform or differs between samples.

When solving PDEs, the focus is often on the order of magnitude of the error, i.e., its logarithm. Therefore, we may adapt the loss function (39) to account for this. Specifically, observe that if the regularization parameter a is small, then functions λ obtained from regressors that can interpolate arbitrary data (e.g., GPs with universal kernels) may result in a near-zero loss and $e^{\lambda_k(X^{i,j})} \rightarrow (Y(X^{i,j}) - M_k(X^{i,j}))^2 := e_k^{i,j}$ as $a \rightarrow 0$.

Thus, we can linearize the exponential around $\log[e_k^{i,j}]$ to get a linearized loss:

$$\lambda_l = \operatorname{argmin}_{l \in \mathcal{H}} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \sum_{k=1}^n \left(l_k(X^{i,j}) - \log[e_k^{i,j}] \right)^2 (e_k^{i,j})^4 + a \|l\|_{\mathcal{H}}^2 \quad (41)$$

This linearized loss shows that samples for which the error is small are down-weighted. This means that instances where the models perform well are ignored, and the aggregation would focus on instances where they perform badly. Instead, we use the following more sensitive loss (42), which, in practice, gives good results on these problems. We also recall the process for obtaining α and the aggregate M_A from λ_s

$$\lambda_s = \operatorname{argmin}_{l \in \mathcal{H}} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \sum_{k=1}^n \left(l_k(X^{i,j}) - \log \left[(Y(X^{i,j}) - M_k(X^{i,j}))^2 \right] \right)^2 + a \|l\|_{\mathcal{H}}^2 \quad (42)$$

$$\alpha(x) = \alpha(\omega, f) = \operatorname{Softmax}(-\lambda_s(x)) = \operatorname{Softmax}(-\lambda_s(\omega, f)) \quad (43)$$

$$M_A(x) = M_A(\omega, f) = \sum_{i=1}^n \alpha_i(x) M_i(x) \quad (44)$$

In both examples, we use the Fourier Neural Operator (FNO) [20] to learn the operator l .

Remark on the choice of FNO. We also explored a kernel-based approach for operator learning, as detailed in [6]. Although operator learning using a Matérn kernel yielded good results, FNO offered better performance with minimal hyperparameter tuning for the examples discussed in this section.

5.3. The Laplace equation. Our first example is the Laplace equation (35). We use six simple solvers as models: a finite difference method, two finite differences with an inhomogeneous grid (one denser for $x < 0.4$, the other for $x > 0.4$), a spectral method, and a GP solver. Our data is generated by sampling² random parameters f_{max}, μ_0, μ_1, R for functions of the form

$$u(x, y) = -\sin(\pi x) \cdot \sin(\pi y) \cdot \sin \left(f_{max} \cdot \exp \left(- \left[\begin{array}{c} x - \mu_0 \\ y - \mu_1 \end{array} \right]^T \cdot R \cdot \left[\begin{array}{c} x - \mu_0 \\ y - \mu_1 \end{array} \right] \right) \right) \quad (45)$$

Six hundred pairs of functions ($f_i := \Delta u_i, u_i$) are sampled, of which 500 are used for training and 100 for testing. Figure 5 shows an example of such a pair. These functions and the models are evaluated on a grid of 100×100 . The aggregation is performed by FNO, which takes input f and the models' outputs and outputs the aggregation coefficients' logits λ . When training a FNO with parameters $\theta \in \Theta \subset \mathbb{R}^{N_\theta}$, we find the minimizer of (42) with

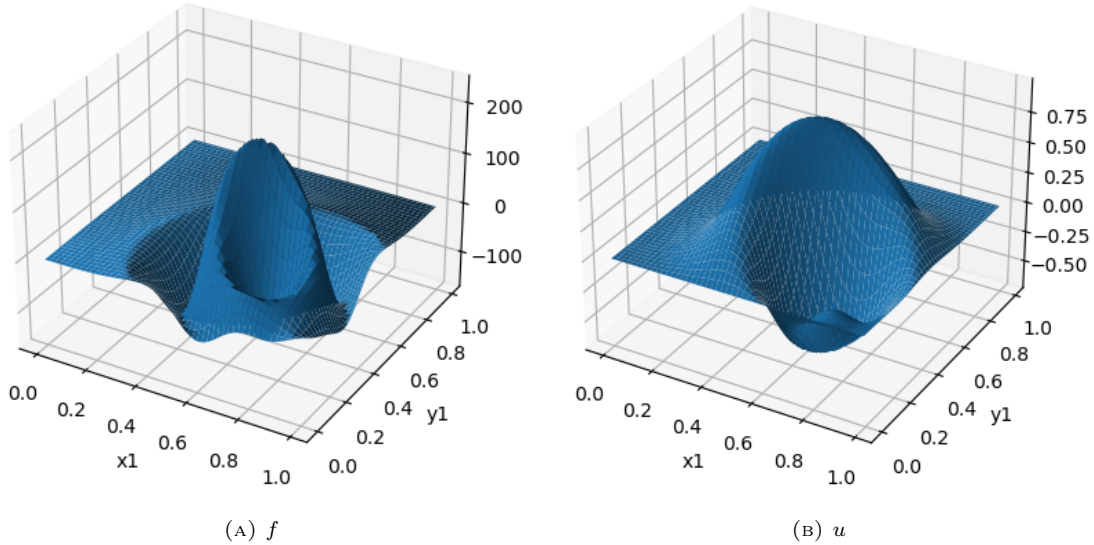
$$\lambda(f) = \operatorname{FNO}(\theta)(f, M_1(f), \dots, M_n(f)) \text{ for some } \theta \quad (46)$$

This is equivalent to setting $\mathcal{H} = \{\operatorname{FNO}(\theta)(\cdot, M_1(\cdot), \dots, M_n(\cdot)) \text{ for } \theta \in \Theta\}$ in (42), with $\|\operatorname{FNO}(\theta)\|_{\mathcal{H}} = \|\theta\|_2$.

²See details in the paper's [repository](#)

Method	Geometric Mean of MSE (\log_{10})
Aggregate	-6.282
FDM	-5.523
Spectral	-4.988
GP	-4.739
FDM (Asymmetric Right)	-4.685
FDM (Asymmetric Left)	-4.699

TABLE 1. Result of Laplace equation experiment


 FIGURE 5. A pair (f, u) s.t. $-\Delta u = f$ and $u(\partial\Omega) = 0$, sampled using (45)

Apart from the GP solver, all other methods perform similarly, with the finite difference method being the best overall. The GP solver, implemented with a fixed length scale, has two regimes. When the function u_i is not evolving too fast (i.e., when f_{max} is not too large), the kernel is well specified, and the GP solver obtains the best performance. On other occasions where f_{max} is larger, the method fails and has a substantial error. Thus, the aggregation challenge here is to use the GP when performing well but ignore it when it fails. Because of the large error in this failure case, we observe that the mean is a poor aggregation that performs worse than all models aggregated. The result of this experiment is shown in figure 7a. Our aggregate is consistently among the best performers on each sample and the best-performing model for many samples. On average, our aggregate performs one order of magnitude better than the models aggregated, as shown in table 1. An example of model outputs, errors, α , and final aggregation is shown in figure 6. As a comparison, we also train FNO using a direct MSE loss of the aggregation, akin to (6). We observe that this method fails as the aggregation converges to a constant, i.e., $\alpha(f, \omega) \approx (1, 0, 0, 0, 0) \forall f, \omega$. The model picked is the best-performing model, the finite difference method. We believe that this convergence to a fixed value, without combining the different models to get a better one as observed in our method, is due to the absence of the log term that we added in the sharp loss (42). As we observed in section 5.2.2, without this log it is difficult for a loss to differentiate between 10^{-3} and 10^{-9} error. Furthermore, while learning the log-error instead of the error is a simple modification from (39) to (42), it is difficult to see where this log would be added in a loss like (6). Finally, we observe that traditional PDE solvers are successfully combined with an ML method (the GP solver). Moreover, it is combined with FNO to get accuracies that FNO itself cannot reach if it tries to learn the operator directly.

5.4. Burger’s equation. We now turn to a non-linear PDE, Burger’s equation with periodic boundary condition and finite time interval:

$$\begin{cases} \partial_t u(t, x) + \partial_x(\frac{1}{2}u^2(t, x)) = \nu \partial_{xx} u(t, x) & \text{for } x, t \in [0, 1]^2 \\ u(0, x) = f(x) & \text{for } x \in [0, 1] \\ u(t, 0) = u(t, 1) & \text{for } t \in [0, 1] \end{cases} \quad (47)$$

We choose $\nu = 2.10^{-3}$ and pick initial conditions f_i as samples from the Gaussian process $\mathcal{N}(0, K_{\exp \sin^2})$ where $K_{\exp \sin^2}(x, y) = \exp\left(-\frac{2 \sin^2(\pi|x-y|)}{l^2}\right)$ and $l = 1.5$. With this set of parameters, the initial conditions

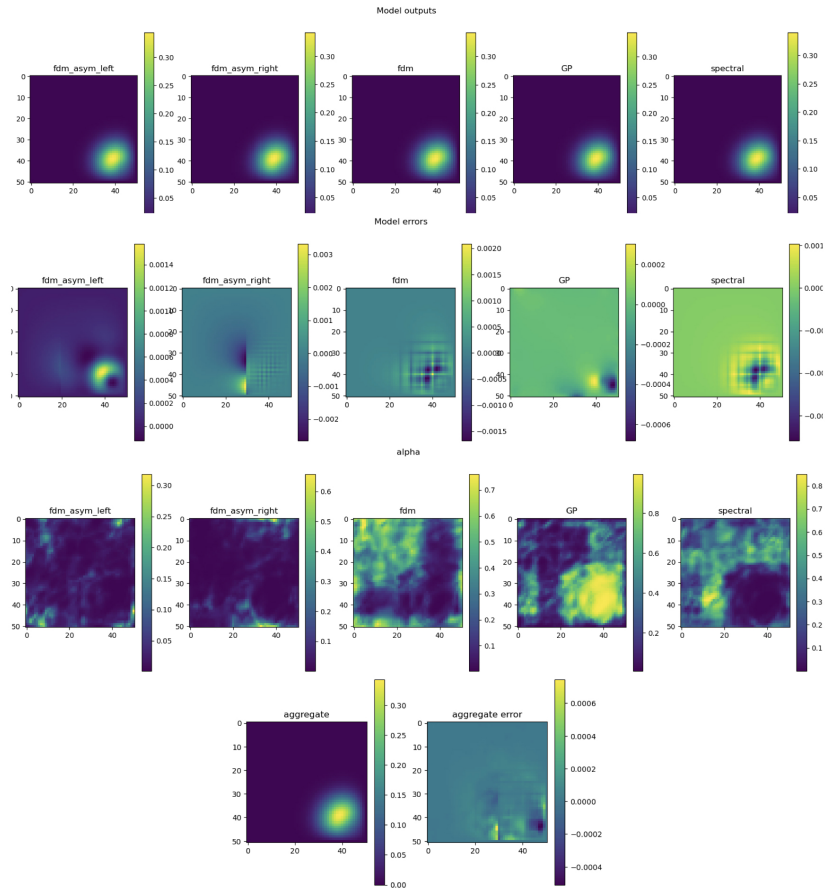


FIGURE 6. Example of aggregation for the Laplace equation (section 5.3)

Line 1: Outputs of the different models for a certain input f . Line 2: Errors of each prediction. Line 3: Values of α . Line 4: Aggregate and its error

are infinitely differentiable periodic functions that often, but not always, form a shock within the time frame studied ($t \in [0, 1]$). We use seven different solvers with the same fixed discretization grid. While the first five methods implement the viscous version of Burger's equation directly, for instance, using an explicit scheme or a finite volume method, the penultimate uses a flux limiter. The last one solves the inviscid Burger's equation. With this variety of solvers, we have methods that are accurate in smooth cases when there is no large shock, an intermediate method that may be more robust to shocks, and a method that will never diverge even in the presence of shocks, but will be less accurate in general because it solves a slightly different equation. A good aggregation must be able to recognize artifacts and divergences that may arise around shocks but use the more precise, although more brittle, solvers in cases where they are accurate. To perform the aggregation, similarly to the previous section, we use FNO, to which we feed the models' prediction as a stack of 2-dimensional arrays (one for time and one for space). Specifically, we minimize loss (42) with

$$\lambda = FNO(\theta)(M_1(f), \dots, M_n(f)) \text{ for some } \theta \quad (48)$$

The results are displayed in figure 7b. Our method successfully leverages the predictions of the different models to give a robust and accurate prediction. Those are the behaviors of our aggregation, which can be separated into a few cases. We find easy initial conditions on the left that all models can approximate correctly. We may notice the Riemann method, which solves the inviscid Burger's equation with lower accuracy. Then, on the left half of the graph, the aggregation mainly relies on the TVD solver, the most precise, to achieve excellent accuracy. In the right half, the more complex cases, the aggregation has an accuracy closer to the more robust Riemann solver. In this half, there are cases where the TVD method fails and diverges and is avoided by our aggregation. Overall, the process can identify each solver's strengths and weaknesses and obtain a better average error, as shown in table 2. An example of model outputs, errors, α , and final aggregation is shown in figure 8.

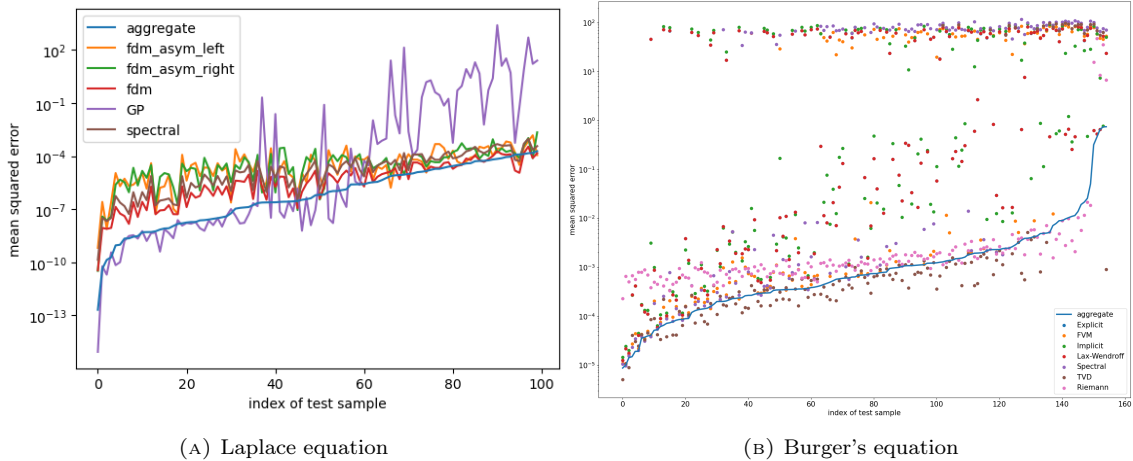


FIGURE 7. log MSE of the different methods for: (7a) the Laplace equation; (7b) Burger’s equation. Samples are sorted by the error of the aggregate

Method	Geometric Mean of MSE (\log_{10})
Aggregate	-3.106
Riemann	-2.734
TVD	-2.568
FVM	-1.228
Spectral	-0.625
Implicit	-0.488
Explicit	-0.455
Lax-Wendroff	-0.455

TABLE 2. Result of Burger’s equation experiment

6. Discussion

We introduced a general data-driven framework for aggregating models with minimal assumptions. As demonstrated in the GP model (Section 3.3) and the two pathological examples (Sections 3.2.1 and 3.2.2), directly training the aggregate model can result in an aggregator that fails to improve upon the performance of individual models. To address this, we introduced a simple assumption of unbiased models in (13) and reformulated the aggregation task as a variance minimization problem (Section 4). In one data science problem (Section 5.1) and two PDE-solving examples (Sections 5.3 and 5.4), our loss function consistently outperformed the minimal error aggregate and led to an aggregation that surpassed the performance of the individual models. These latter two examples also illustrate the flexibility of our method, which can aggregate not only machine learning models but also other types of models, such as PDE solvers, in a data-driven manner.

Our method relies on the availability of unseen data to estimate the error of different models. Therefore, in scenarios where data is limited, splitting the data further to create an unseen validation set may significantly reduce the amount of training data available for the models, potentially impairing overall performance. Additionally, the aggregation task itself can be challenging, and there is no guarantee that the aggregate will always outperform the individual models or simpler approaches, such as averaging, despite the higher computational cost.

Lastly, while we focused on specific aggregation strategies for the regression case, particularly those involving mean squared error, we believe our findings could extend to other aggregation methods. Ideally, the aggregation process should be pointwise linear and supported by a probabilistic model that justifies its structure. Most importantly, the aggregation method should focus on learning the errors of the individual models, rather than the target itself. To achieve this, the training loss must incorporate the models’ errors without access to the true target values.

Acknowledgments

The authors gratefully acknowledge support by the Air Force Office of Scientific Research under MURI award number FA9550-20-1-0358 (Machine Learning and Physics-Based Modeling and Simulation) and by the Department of Energy under award number DE-SC0023163 (SEA-CROGS: Scalable, Efficient and Accelerated Causal Reasoning Operators, Graphs and Spikes for Earth and Embedded Systems). HO also acknowledges support by a Department of Defense Vannevar Bush Faculty Fellowship.

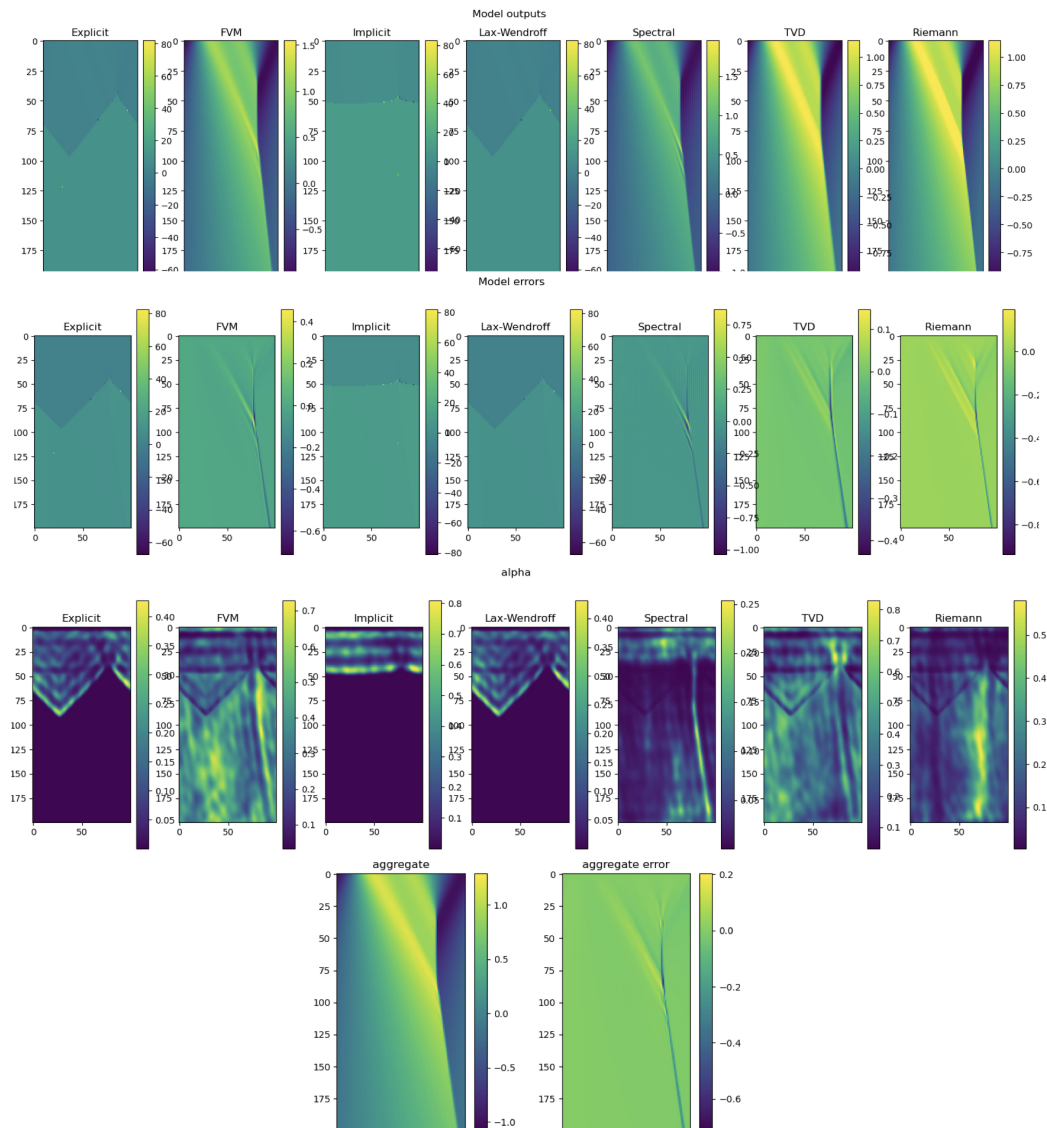


FIGURE 8. Example of aggregation for Burger's equation (section 5.4)

Line 1: Outputs of the different models for a certain input f . Line 2: Errors of each prediction. Line 3: Values of α . Line 4: Aggregate and its error

We may notice that explicit, implicit, and Lax-Wendroff methods all diverged, while the spectral method displays spurious oscillations. The aggregation does not have any of these problems.

References

- [1] Nathanael L Ackerman, Cameron E Freer, and Daniel M Roy. On computability and disintegration. *Mathematical Structures in Computer Science*, 27(8):1287–1314, 2017.
- [2] Mauricio A Alvarez, Lorenzo Rosasco, Neil D Lawrence, et al. Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning*, 4(3):195–266, 2012.
- [3] Hamed Hamze Bajgiran and Houman Owahdi. Aggregation of Models, Choices, Beliefs, and Preferences, November 2021. arXiv:2111.11630 [econ, math, stat].
- [4] Hamed Hamze Bajgiran and Houman Owahdi. Aggregation of Pareto optimal models, December 2021. arXiv:2112.04161 [econ, math, stat].
- [5] Klaus-Jürgen Bathe. *Finite Element Method*, pages 1–12. John Wiley & Sons, Ltd, 2008.
- [6] Pau Batlle, Matthieu Darcy, Bamdad Hosseini, and Houman Owahdi. Kernel methods are competitive for operator learning. *Journal of Computational Physics*, 496:112549, 2024.
- [7] Candice Bentéjac, Anna Csörgő, and Gonzalo Martínez-Muñoz. A comparative analysis of gradient boosting algorithms. *Artificial Intelligence Review*, 54(3):1937–1967, March 2021.
- [8] Théo Bourdais, Pau Batlle, Xianjin Yang, Ricardo Baptista, Nicolas Rouquette, and Houman Owahdi. Codiscovering graphical structure and functional relationships within data: A gaussian process framework for connecting the dots. *Proceedings of the National Academy of Sciences*, 121(32):e2403449121, 2024.
- [9] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, August 1996.
- [10] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, October 2001.
- [11] Edoardo Calvella, Nikola B. Kovachki, Matthew E. Levine, and Andrew M. Stuart. Continuum Attention for Neural Operators, June 2024. arXiv:2406.06486 [cs, math].

- [12] Yifan Chen, Bamdad Hosseini, Houman Owhadi, and Andrew M. Stuart. Solving and Learning Nonlinear PDEs with Gaussian Processes, August 2021. arXiv:2103.12959 [cs, math, stat].
- [13] Elizabeth E. Ebert. Ability of a Poor Man’s Ensemble to Predict the Probability and Distribution of Precipitation. *Monthly Weather Review*, 129(10):2461–2480, October 2001. Publisher: American Meteorological Society Section: Monthly Weather Review.
- [14] Romain Egele, Romit Maulik, Krishnan Raghavan, Bethany Lusch, Isabelle Guyon, and Prasanna Balaprakash. AutoDEUQ: Automated Deep Ensemble with Uncertainty Quantification, July 2022. arXiv:2110.13511 [cs].
- [15] Gregory Flato, Jochem Marotzke, Babatunde Abiodun, Pascale Braconnot, Sin Chan Chou, William Collins, Peter Cox, Fatima Driouech, Seita Emori, Veronika Eyering, et al. Evaluation of climate models. In *Climate change 2013: the physical science basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*, pages 741–866. Cambridge University Press, 2014.
- [16] Bengt Fornberg and David M. Sloan. A review of pseudospectral methods for solving partial differential equations. *Acta Numerica*, 3:203–267, 1994.
- [17] David Harrison and Daniel L Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5(1):81–102, March 1978.
- [18] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, L el io Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Th eophile Gervet, Thibaut Lavril, Th eophile Lacroix, and William El Sayed. Mixtral of Experts, January 2024. arXiv:2401.04088 [cs].
- [19] Nikola B. Kovachki, Samuel Lanthaler, and Andrew M. Stuart. Operator Learning: Algorithms and Analysis, February 2024. arXiv:2402.15715 [cs, math].
- [20] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations, May 2021. arXiv:2010.08895 [cs, math].
- [21] Saeed Masoudnia and Reza Ebrahimpour. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42(2):275–293, August 2014.
- [22] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, February 2019.
- [23] Didier Rulli ere, Nicolas Durrande, Fran ois Bachoc, and Cl ement Chevalier. Nested Kriging predictions for datasets with large number of observations, July 2017. arXiv:1607.05432 [stat].
- [24] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, December 2015.
- [25] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, June 1990.
- [26] Robert E. Schapire. The Boosting Approach to Machine Learning: An Overview. In P. Bickel, P. Diggle, S. Fienberg, K. Krickeberg, I. Olkin, N. Wermuth, S. Zeger, David D. Denison, Mark H. Hansen, Christopher C. Holmes, Bani Mallick, and Bin Yu, editors, *Nonlinear Estimation and Classification*, volume 171, pages 149–171. Springer New York, New York, NY, 2003. Series Title: Lecture Notes in Statistics.
- [27] Seniha Esen Yuksel, Joseph N. Wilson, and Paul D. Gader. Twenty Years of Mixture of Experts. *IEEE Transactions on Neural Networks and Learning Systems*, 23(8):1177–1193, August 2012. Conference Name: IEEE Transactions on Neural Networks and Learning Systems.
- [28] Cha Zhang and Yunqian Ma, editors. *Ensemble Machine Learning: Methods and Applications*. Springer New York, New York, NY, 2012.

Appendix A. Vector-valued Gaussian process and matrix-valued kernel

This section is a reminder on vector/matrix-valued GPs/kernels [2]. We define a matrix-valued kernel as a function $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}^{n \times n}$ such that,

- $\forall x, x' \in \mathcal{X}, K(x, x') = K(x', x)^T$
- $\forall x_1, \dots, x_N \in \mathcal{X}, \forall y_1, \dots, y_N \in \mathbb{R}^n,$

$$\sum_{1 \leq i, j \leq N} y_i^T K(x_i, x_j) y_j \geq 0 \quad (49)$$

This generalizes the positivity and symmetry conditions for defining a kernel. Each matrix-valued kernel uniquely defines a zero-mean vector-valued Gaussian process ξ such that :

- $\forall x \in \mathcal{X}, \xi(x) \sim \mathcal{N}(0, K(x, x))$
- $\forall x, x' \in \mathcal{X}, \text{Cov}[\xi(x), \xi(x')] = K(x, x')$

One interesting example of such a vector-valued Gaussian process is the case of n independent Gaussian processes, corresponding to K being diagonal. One can then understand general vector-valued GP as adding correlation to the coordinates, each being its own GP.

Appendix B. Solving (11)

We aim to solve the following optimization problem:

$$\hat{\alpha} = \underset{\alpha \in \mathcal{H}_K}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^N \left[Y^i(x_i) - \alpha(x_i)^T M^i(x_i) \right]^2 + \gamma \|\alpha\|_{\mathcal{H}_K}^2 \quad (50)$$

Applying the representer theorem, we deduce that the solution $\hat{\alpha}$ can be represented as $\hat{\alpha} = (K(\cdot, x_1), \dots, K(\cdot, x_N))V$, where $V \in \mathbb{R}^{nN}$. Additionally, we define the matrix $\mathcal{M} \in \mathbb{R}^{n \times nN}$ as follows:

$$\mathcal{M} = \begin{pmatrix} M_1 \\ \vdots \\ M_N \end{pmatrix}, \quad (51)$$

where each $\mathcal{M}_i = (M(x_i)^T K(x_i, x_1), \dots, M(x_i)^T K(x_i, x_N))$. The optimal vector V is then given by:

$$V = \operatorname{argmin}_{v \in \mathbb{R}^{nN}} \|Y - \mathcal{M}v\|^2 + \lambda v^T \mathcal{K}v \quad (52)$$

where \mathcal{K} is a block matrix such that $\mathcal{K}_{ij} = K(x_i, x_j)$ for $1 \leq i, j \leq N$. Define the matrix $D \in \mathbb{R}^{N \times nN}$ as:

$$D = \begin{pmatrix} M(x_1)^T & 0 & \dots & 0 \\ 0 & M(x_2)^T & \dots & \dots \\ \vdots & \dots & \ddots & \vdots \\ 0 & \dots & \dots & M(x_N)^T \end{pmatrix} \quad (53)$$

Observe that $\mathcal{M} = DK$. Using the matrix identity:

$$(P^{-1} + B^T R^{-1} B)^{-1} B^T R^{-1} = P B^T (B P B^T + R)^{-1} \quad (54)$$

where $P^{-1} = \lambda K$, $B = \mathcal{M}$, and $R = I$, we derive:

$$V = \frac{1}{\lambda} K^{-1} \mathcal{M}^T \left(\frac{1}{\lambda} \mathcal{M} K^{-1} \mathcal{M}^T + I \right)^{-1} Y \quad (55)$$

$$= D^T (DKD^T + I)^{-1} Y \quad (56)$$

It follows that $(DKD^T)_{ij} = M^T(x_i) K(x_i, x_j) M(x_j) = \tilde{k}(x_i, x_j)$. Thus, the final model prediction at a new point x is given by:

$$M_A(x) = M(x)^T K(x, X) V \quad (57)$$

where $K(x, X) = (K(x, x_1), \dots, K(x, x_N))$ so that $\hat{\alpha}(x) = K(x, X) V$. By identifying that $M(x)^T K(x, X) D^T = \tilde{k}(x, X)$, we can conclude:

$$M_A(x) = \tilde{k}(x, X) (\tilde{k}(X, X) + \lambda I)^{-1} Y \quad (58)$$