MULTI-OBJECTIVE OPTIMIZATION VIA EQUIVARIANT DEEP HYPERVOLUME APPROXIMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Optimizing multiple competing objectives is a common problem across science and industry. The inherent inextricable trade-off between those objectives leads one to the task of exploring their Pareto front. A meaningful quantity for the purpose of the latter is the hypervolume indicator, which is used in Bayesian Optimization (BO) and Evolutionary Algorithms (EAs). However, the computational complexity for the calculation of the hypervolume scales unfavorably with increasing number of objectives and data points, which restricts its use in those common multi-objective optimization frameworks. To overcome these restrictions we propose to approximate the hypervolume function with a deep neural network, which we call DeepHV. For better sample efficiency and generalization, we exploit the fact that the hypervolume is scale-equivariant in each of the objectives as well as permutation invariant w.r.t. both the objectives and the samples, by using a deep neural network that is equivariant w.r.t. the combined group of scalings and permutations. We evaluate our method against exact, and approximate hypervolume methods in terms of accuracy, computation time, and generalization. We also apply and compare our methods to state-of-the-art multi-objective BO methods and EAs on a range of synthetic benchmark test cases. The results show that our methods are promising for such multi-objective optimization tasks.

1 INTRODUCTION

Imagine, while you are reviewing this paper at the breakfast table you remember that you quickly need to drive over to the office to welcome your new collaborators. As an experienced reviewer, who knows how to find typos on the spot, and an excellent driver, who always drives fast, but safe, you are confident in your abilities on each of those tasks. So you continue reading this paper while driving. Suddenly you realize that you need to split focus. You face the unavoidable trade-off between driving fast while only glancing at the paper and driving slowly but properly processing each point of the paper. Since this is not your first ride, you learned over time how to transition between these competing objectives while still being optimal under those trade-off constraints. Since you don't want to stop the car you decide to drive as fast as possible as safety allows while still making some progress in reading. You safely arrive at the office, where you pitch to your collaborators the idea of building self-driving cars that can also automatically review papers. The question remains of how to train an artificial intelligence to learn to excel in different, possibly competing, tasks or objectives and make deliberate well-calibrated trade-offs between them, whenever necessary.

Simultaneous optimization of multiple, possibly competing, objectives is not just a challenge in our daily routines, it also finds widespread application in many fields of science. For instance, in machine learning (Wu et al., 2019; Snoek et al., 2012), engineering (Liao et al., 2007; Oyama et al., 2018), and chemistry (O'Hagan et al., 2005; Koledina et al., 2019; MacLeod et al., 2022; Boelrijk et al., 2021).

The challenges involved in this setting are different from the ones in the *single objective* case. If we are only confronted with a single objective function f the task is to find a point $\mathbf{x}^*$ that maximizes f:

$$\mathbf{x}^* \in \operatorname*{argmax}_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}), \tag{1}$$

only by iteratively proposing input points $\mathbf{x}_n$ and evaluating f on $\mathbf{x}_n$ and observing the output values $f(\mathbf{x}_n)$. Since the usual input space $\mathcal{X} = \mathbb{R}^D$ is uncountably infinite we can never be certain if the

finite number of points $\mathbf{x}_1, \ldots, \mathbf{x}_N$ contain a global maximizer $\mathbf{x}^*$ of f. This is an instance of the *exploration-vs-exploitation* trade-off inherent to Bayesian optimization (BO) (Snoek et al., 2012), active learning, (Burr, 2009) and reinforcement learning(Kaelbling et al., 1996).

In the *multi-objective* (MO) setting, where we have $M \geq 2$ objective functions, $f_1, \ldots, f_M$, it is desirable, but usually not possible to find a single point $\mathbf{x}^*$ that maximizes all objectives at the same time:

$$\mathbf{x}^* \in \bigcap_{m=1}^{M} \operatorname*{argmax}_{\mathbf{x} \in \mathcal{X}} f_m(\mathbf{x}). \tag{2}$$

A maximizer $\mathbf{x}_1^*$ of f_1 might lead to a non-optimal value of f_2, etc. So the best we can do in this setting is to find the set of *Pareto points* of $F = (f_1, \ldots, f_M)$, i.e. those points $\mathbf{x} \in \mathcal{X}$ that cannot be improved in any of the objectives f_m, $m = 1, \ldots, M$, while not lowering the other values:

$$\mathcal{X}^* := \{\mathbf{x} \in \mathcal{X} \,|\, \nexists\, \mathbf{x}' \in \mathcal{X}.\, F(\mathbf{x}) \prec F(\mathbf{x}')\}\,, \tag{3}$$

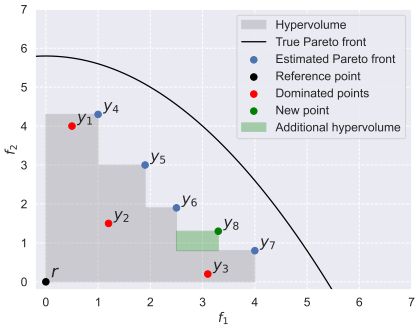

Figure 1: Illustration of Pareto front and Hypervolume.

where $\mathbf{y} \preceq \mathbf{y}'$ means that $y_m \leq y'_m$ for all $m = 1, \ldots, M$ and $\mathbf{y} \prec \mathbf{y}'$ that $\mathbf{y} \preceq \mathbf{y}'$, but $\mathbf{y} \neq \mathbf{y}'$. In conclusion, in the *multi-objective* setting we are rather concerned with the *exploration of the Pareto front*:

$$\mathcal{P}^* := F(\mathcal{X}^*) \subseteq \mathbb{R}^M =: \mathcal{Y}, \tag{4}$$

which often is a $(M-1)$-dimensional subspace of $\mathcal{Y} = \mathbb{R}^M$. Success in this setting is measured by how "close" the *empirical Pareto front*, based on the previously chosen points $\mathbf{x}_1, \ldots, \mathbf{x}_N$:

$$\hat{\mathcal{P}}_N := \{F(\mathbf{x}_n) \,|\, n \in [N],\, \nexists\, n' \in [N].\, F(\mathbf{x}_n) \prec F(\mathbf{x}_{n'})\} \subseteq \mathcal{Y}, \tag{5}$$

is to the 'true' Pareto front $\mathcal{P}^*$, where $[N] := \{1, \ldots, N\}$. This is illustrated in Fig. 1, where the blue points form the empirical Pareto front and where the black line depicts the true Pareto front.

Since the values $F(\mathbf{x}_n)$ can never exceed the values of $\mathcal{P}^*$ w.r.t. $\prec$ one way to quantify the closeness of $\hat{\mathcal{P}}_N$ to $\mathcal{P}^*$ is by measuring its *hypervolume* $\mathrm{HV}_{\mathbf{r}}^M(\hat{\mathcal{P}}_N)$ inside $\mathcal{Y}$ w.r.t. a jointly dominated reference point $\mathbf{r} \in \mathcal{Y}$. This suggests the multi-objective optimization strategy of picking the next points $\mathbf{x}_{N+1} \in \mathcal{X}$ in such a way that it would lead to a maximal improvement of the previously measured hypervolume $\mathrm{HV}_{\mathbf{r}}^M(\hat{\mathcal{P}}_N)$. This is illustrated in Fig. 1, where the hypervolume (i.e., the area in 2D) is shown for the empirical Pareto front (blue dots). Adding an additional point to the empirical Pareto front (y_8) in Fig. 1), increases the hypervolume indicated by the green area.

Unfortunately, known algorithms for computing the hypervolume scale unfavorably with the number of objective functions M and the number of data points N. Nonetheless, finding a fast and scalable computational method that approximates the hypervolume reliably would have far-reaching consequences, and is an active field of research. Computation of the hypervolume has complexity of $O(2^N NM)$ when computed in a naive way, however more efficient exact algorithms have been proposed such as WFG ($O(N^{M/2} \log N)$,While et al. (2012)) and HBDA ($O(N^{M/2})$, Lacour et al. (2017)). However, these computational complexities are still deemed impractical for application in EAs (Tang et al., 2020), where computational overhead typically is required to be low. Also in BO, where one is typically less concerned with the computational overhead, faster hypervolume methods would be beneficial. For instance, the state-of-the-art expected hypervolume improvement (qEHVI) (Daulton et al., 2020) is dependent on many hypervolume computations, greatly restricting its use to the setting of high M and N. In addition, the authors of the recently proposed MORBO (Daulton et al., 2022), which is current state-of-the-art in terms of sample-efficiency and scalability to high M and N, identified the computational complexity of the hypervolume as a limitation.

Therefore, efforts have been made to approximate the hypervolume. The method FPRAS (Bringmann & Friedrich, 2010), provides an efficient Monte Carlo (MC) based method to approximate the hypervolume, of complexity $O(NM/\epsilon)$, with the error of $\pm\sqrt{\epsilon}$. In addition, at time of this work, a hypervolume approximation based on DeepSets (Zaheer et al., 2017) was proposed called HV-Net (Shang et al., 2022). HV-Net uses a deep neural network with permutation invariance (i.e. DeepSets)

w.r.t. to the order of the elements in the solution set to approximate the hypervolume of a non-dominated solution set. It showed favorable results on both the approximation error and the runtime on MC and line-based hypervolume approximation methods.

Other methods try to avoid hypervolume computation entirely and use other metrics to guide optimization. Such as ParEgo (Knowles, 2006), which randomly scalarizes the objectives to generate diverse solutions. Other approaches focus on distance metrics regarding the empirical Pareto front (Rahat et al., 2017; Deb et al., 2002a). However, methods using the hypervolume to guide optimization, are typically superior. This is shown both in the setting of EAs and for BO. For example, a plethora of work shows that SMS-EMOA (Beume et al., 2007) is more sample-efficient compared to MO EAs using alternative infill criterion (Narukawa & Rodemann, 2012; Tang et al., 2020). Also, qEHVI and MORBO have strong empirical performance compared to alternatives (Daulton et al., 2020; 2022).

In this paper, we develop DeepHV, a novel hypervolume approximation approach based on deep learning. As an input the model requires a non-dominated solution set after which the model outputs the predicted hypervolume of this solution set. By using specialized layers, DeepHV incorporates mathematical properties of the hypervolume, such as scaling equivariance of the hypervolume w.r.t. the scaling of the objectives in the solution set. In addition, the proposed model is permutation invariant to the order of the elements and objectives in the solution set. Thus leveraging additional symmetry properties of the hypervolume compared to HV-Net. We show that our method allows for development of a single model that can be used for multiple objective cases up to $M = 10$. We obtain improved performance compared to HV-Net on unseen test data. We compare the time performance of our methods with exact methods and an approximate MC method. We show a considerable speed-up that scales favorably with the number of objectives. In addition, to evaluate the useability of our approach, we use DeepHV in the setting of multi-objective BO and EAs on a range of synthetic test functions and compare with state-of-the-art methods and show competitive performance.

2 THE HYPERVOLUME

In this section, we introduce relevant information regarding the definition and symmetry properties of the hypervolume, which will later complement the definition of our method.

Hypervolume - Definition For a natural number $N \in \mathbb{N}$, we put $[N] := \{1, \dots, N\}$. Consider (column) vectors $\mathbf{y}_n \in \mathbb{R}^M$ in the M-dimensional Euclidean space. For a subset $J \subseteq \mathbb{N}$, we abbreviate: $\mathbf{y}_J := [\mathbf{y}_j | j \in J]^\top \in \mathbb{R}^{M \times |J|}$, so $\mathbf{y}_{[N]} := [\mathbf{y}_1, \dots, \mathbf{y}_N] \in \mathbb{R}^{M \times N}$. We then define the ($M$-dimensional) *hypervolume* of $\mathbf{y}_J$ w.r.t. a fixed reference point $\mathbf{r} \in \mathbb{R}^M$ as:

$$\mathrm{HV}_{\mathbf{r}}^M(\mathbf{y}_J) := \mathrm{HV}_{\mathbf{r}}^M(\mathbf{y}_{j_1}, \dots, \mathbf{y}_{j_{|J|}}) := \boldsymbol{\lambda}^M \left(\bigcup_{j \in J} [\mathbf{r}, \mathbf{y}_j] \right), \tag{6}$$

where $\boldsymbol{\lambda}^M$ is the M-dimensional Lebesgue measure, which assigns to a subset of $\mathbb{R}^M$ their M-dimensional (hyper) volume, and where we for $\mathbf{y} = [y_1, \dots, y_M]^\top \in \mathbb{R}^M$ abbreviated the M-dimensional cube as: $[\mathbf{r}, \mathbf{y}] := [r_1, y_1] \times \cdots \times [r_M, y_M]$, if $r_m \leq y_m$ for all $m \in [M]$, and the empty-set otherwise. For fixed N and another point $\mathbf{y} \in \mathbb{R}^M$ we define its *hypervolume improvement* (over $\mathbf{y}_{[N]}$) as:

$$\mathrm{HVI}(\mathbf{y}) := \mathrm{HVI}_{\mathbf{r}}^M(\mathbf{y} | \mathbf{y}_{[N]}) := \mathrm{HV}_{\mathbf{r}}^M(\mathbf{y}, \mathbf{y}_1, \dots, \mathbf{y}_N) - \mathrm{HV}_{\mathbf{r}}^M(\mathbf{y}_1, \dots, \mathbf{y}_N). \tag{7}$$

With help of the *inclusion-exclusion formula* we can explicitely rewrite the hypervolume as:

$$\mathrm{HV}_{\mathbf{r}}^M(\mathbf{y}_J) = \sum_{\emptyset \neq S \subseteq J} (-1)^{|S|+1} \cdot \prod_{m=1}^{M} \min_{s \in S} (y_{m,s} - r_m)_+. \tag{8}$$

The Symmetry Properties of the Hypervolume Function To state the symmetry properties of the hypervolume function, we will fix our reference point $\mathbf{r} := \mathbf{0}$, which is w.l.o.g. always possible by subtracting $\mathbf{r}$ from each $\mathbf{y}_n$.

The hypervolume then scales with each of its components. So for $\mathbf{c} = [c_1, \dots, c_M]^\top \in \mathbb{R}_{>0}^M$:

$$\mathrm{HV}^M(\mathbf{c} \odot \mathbf{y}_{[N]}) = c_1 \cdots c_M \cdot \mathrm{HV}^M(\mathbf{y}_{[N]}), \tag{9}$$

where $\mathbf{c} \odot \mathbf{y}_{[N]}$ means that the m-th entry of each point $\mathbf{y}_n \in \mathbb{R}^M_{\geq 0}$ is multiplied by c_m for all $m \in [M]$. Furthermore, the order of the points does not change the hypervolume:

$$\mathrm{HV}^M(\mathbf{y}_{\sigma(1)}, \dots, \mathbf{y}_{\sigma(N)}) = \mathrm{HV}^M(\mathbf{y}_1, \dots, \mathbf{y}_N), \tag{10}$$

for every permutation $\sigma \in \mathbb{S}_N$, where the latter denotes the *symmetric group*, the group of all permutations of N elements. But also the order of the M-components does not change the hypervolume:

$$\mathrm{HV}^M(\tau \odot \mathbf{y}_{[N]}) = \mathrm{HV}^M(\mathbf{y}_{[N]}), \tag{11}$$

for every permutation $\tau \in \mathbb{S}_M$, where $\tau \odot \mathbf{y}_{[N]}$ means that the row indices are permuted by τ.

To jointly formalize these equivariance properties of HV^M together, we define the following joint group as the semi-direct product of scalings and permutations:

$$G := \mathbb{R}^M_{>0} \rtimes \mathbb{S}_M \times \mathbb{S}_N, \tag{12}$$

where the group operation is defined as follows on elements $(\mathbf{c}_1, \tau_1, \sigma_1)$ and $(\mathbf{c}_2, \tau_2, \sigma_2) \in G$ via:

$$(\mathbf{c}_2, \tau_2, \sigma_2) \cdot (\mathbf{c}_1, \tau_1, \sigma_1) := (\mathbf{c}_2 \cdot \mathbf{c}_1^{\tau_2}, \tau_2 \circ \tau_1, \sigma_2 \circ \sigma_1) \in G, \tag{13}$$

where with $\mathbf{c}_1 = [c_{1,1}, \dots, c_{M,1}]^\top \in \mathbb{R}^M_{>0}$ we use the abbreviation:

$$\mathbf{c}_1^{\tau_2} := [c_{\tau_2(1),1}, \dots, c_{\tau_2(M),1}]^\top \in \mathbb{R}^M_{>0}. \tag{14}$$

This group then acts on the space of non-negative $(M \times N)$-matrices as follows. For $\mathbf{Y} = (y_{m,n})_{m,n} \in \mathbb{R}^{M \times N}_{\geq 0}$ and $(\mathbf{c}, \tau, \sigma) \in G$ we put:

$$(\mathbf{c}, \tau, \sigma) \odot \mathbf{Y} := \begin{bmatrix} c_1 \cdot y_{\tau(1),\sigma(1)} & \cdots & c_1 \cdot y_{\tau(1),\sigma(N)} \\ \vdots & \ddots & \vdots \\ c_M \cdot y_{\tau(M),\sigma(1)} & \cdots & c_M \cdot y_{\tau(M),\sigma(N)} \end{bmatrix}. \tag{15}$$

The mentioned symmetries of the hypervolume can then be jointly summarized by the following G-*equivariance* property:

$$\mathrm{HV}^M((\mathbf{c}, \tau, \sigma) \odot \mathbf{Y}) = c_1 \cdots c_M \cdot \mathrm{HV}^M(\mathbf{Y}). \tag{16}$$

This is the property that we will exploit for an efficient approximation of the hypervolume with a G-equivariant deep neural network.

The hypervolume for different dimensions M are related as follows:

$$\mathrm{HV}^M(\mathbf{y}_J) = \mathrm{HV}^{M+K}\left(\begin{bmatrix} \mathbf{y}_J \\ \mathbf{1}_{K \times |J|} \end{bmatrix}\right), \tag{17}$$

where we padded the $(K \times |J|)$-matrix with ones to $\mathbf{y}_J$, or, equivalently, K ones to each point $\mathbf{y}_j$, $j \in J$. We leverage this property to combine datasets with different dimensions M and to train a model that can generalize to multiple dimensions.

3 DeepHV - Equivariant Deep Hypervolume Approximation

In this section, we first present the G-equivariant neural network layers used in DeepHV that incorporates the symmetry properties defined in Sec. 2. Following this, we discuss the general architecture of DeepHV. We provide mathematical proofs in App. A.1.

3.1 The Equivariant Layers

Before we introduce our G-equivariant neural network layers, we first recall how $(\mathbb{S}_M \times \mathbb{S}_N)$-equivariant layers are constructed.

Theorem 1 (See Hartford et al. (2018) Thm. 2.1)**.** *A fully connected layer of a neural network* $\mathrm{vec}(\mathbf{Z}) = \sigma(\mathbf{W}\,\mathrm{vec}(\mathbf{Y}))$ *with input matrix* $\mathbf{Y} \in \mathbb{R}^{M \times N}$*, output matrix* $\mathbf{Z} \in \mathbb{R}^{M \times N}$*, weight matrix* $\mathbf{W} \in \mathbb{R}^{(MN) \times (MN)}$ *and strictly monotonic activation function* σ *is* $(\mathbb{S}_M \times \mathbb{S}_N)$*-equivariant if and only if it can be written as:*

$$\mathbf{Z} = \sigma\left(w_1 \cdot \mathbf{Y} + w_2 \cdot \mathbf{1}_M \cdot \mathbf{Y}_{\mathbb{M}:} + w_3 \cdot \mathbf{Y}_{:\mathbb{M}} \cdot \mathbf{1}_N^\top + w_4 \cdot \mathbf{Y}_{\mathbb{M}:\mathbb{M}} \cdot \mathbf{1}_{M \times N} + w_5 \cdot \mathbf{1}_{M \times N}\right), \tag{18}$$

where $\mathbf{1}$ *denotes the matrix of ones of the indicated shape,* $w_1, \dots, w_5 \in \mathbb{R}$ *and where* $\mathbb{M}$ *is the mean operation, applied to rows (*$\mathbf{Y}_{\mathbb{M}:}$*), columns (*$\mathbf{Y}_{:\mathbb{M}}$*) or both (*$\mathbf{Y}_{\mathbb{M}:\mathbb{M}}$*).*

We define the *row scale* of a matrix $\mathbf{Y} = (y_{m,n})_{m,n} \in \mathbb{R}^{M \times N}$ via:

$$\mathbf{s}(\mathbf{Y}) := \left[\max_{n \in [N]} |y_{m,n}| \,\middle|\, m \in [M] \right]^\top \in \mathbb{R}^M, \tag{19}$$

and its row-wise inverse $\mathbf{s}^{-1}(\mathbf{Y}) \in \mathbb{R}^M$. We then abbreviate the *row-rescaled* matrix as:

$$\mathbf{Y}_\oslash := \mathbf{s}^{-1}(\mathbf{Y}) \odot \mathbf{Y} \in \mathbb{R}^{M \times N}. \tag{20}$$

With these notations, we are now able to construct a G-equivariant neural network layer with input channel indices $i \in I$ and output channel indices $o \in O$:

$$\begin{aligned} \mathbf{Z}^{(o)} = \sigma_\alpha \Bigg(\frac{1}{|I|} \sum_{i \in I} \mathbf{s}\left(\mathbf{Y}^{(i)}\right) \odot \Bigg(& w_1^{(o,i)} \cdot \left(\mathbf{Y}_\oslash^{(i)}\right) + w_2^{(o,i)} \cdot \mathbf{1}_M \cdot \left(\mathbf{Y}_\oslash^{(i)}\right)_{\mathbb{M}:} \\ & + w_3^{(o,i)} \cdot \left(\mathbf{Y}_\oslash^{(i)}\right)_{:\mathbb{M}} \cdot \mathbf{1}_N^\top + w_4^{(o,i)} \cdot \left(\mathbf{Y}_\oslash^{(i)}\right)_{\mathbb{M}:\mathbb{M}} \cdot \mathbf{1}_{M \times N} + w_5^{(o,i)} \cdot \mathbf{1}_{M \times N} \Bigg) \Bigg), \end{aligned} \tag{21}$$

where $\mathbb{M}$ is applied to the row-rescaled $\mathbf{Y}$ row-wise ($\mathbf{Y}_{\mathbb{M}:}$), column-wise ($\mathbf{Y}_{:\mathbb{M}}$) or both ($\mathbf{Y}_{\mathbb{M}:\mathbb{M}}$), and where, again, $\mathbb{M}$ denotes the mean operation (but could also be max or min), and weights $w_k^{(o,i)} \in \mathbb{R}$ for $k \in [5]$, $o \in O$, $i \in I$. σ_α denotes any homogeneous activation function, i.e. we require that for every $y \in \mathbb{R}$, $c \in \mathbb{R}_{>0}$:

$$\sigma_\alpha(c \cdot y) = c \cdot \sigma_\alpha(y) \tag{22}$$

This is, for instance, met by the leaky-ReLU activation function. An alternative way to achieve equivariance of the whole network w.r.t. to the scale of the whole network is by extracting the scales at the input layer and then multiplying them all against the output layer, in contrast to propagating the scale through every layer individually.

3.2 Architecture

All the DeepHV models presented in this work are comprised of the same architecture, with the exception that we change the number of input and output channels in order to trade off the number of model parameters and the expressivity of the model. We denote models as DeepHV-c, where c denotes the number of channels. Before the $\mathbb{R}^{N \times M}$ input is passed through the layers, we first extract and store the scale using eq. 19, and scale the inputs using eq. 20. We use five equivariant layers as in eq. 21, where the first layer has 1 input channel and c output channels. The three intermediate layers have the same number c of input and output channels. The last layer then maps back to 1 output channel, again resulting into $\mathbb{R}^{M \times N}$ output. Consequently, the mean is taken of the resulting output to bring it back to a permutation invariant quantity ($\mathbb{R}^1$). This is then passed through a sigmoid function to map values between 0 and 1, as this is the value which the hypervolume can have in the scaled setting. Finally, we rescale the output by multiplying it with the product of the scaling factors. This step ensures that the model output is scale equivariant.

4 Training

4.1 Data Generation

To generate training data, we adopt a similar strategy as proposed in Shang et al. (2022), which is defined as follows: 1. Uniformly sample an integer $n \in [1, 100]$; 2. Uniformly sample 1000 solutions in $[0, 1]^M$; 3. Obtain different fronts $\{P_1, P_2, ...\}$ using non-dominated sorting, where P_1 is the first Pareto front and P_2 is the following front after all solutions of P_1 are removed; 4. Identify all fronts P_i with $|P_i| \geq n$. If no front satisfies this conditions, go back to Step 2; 5. Randomly select one front P_i with $|P_i| \geq n$ and randomly select n solutions from the front to construct one solution set. Using this procedure, we generate datasets consisting of 1 million solution sets for each of the objective cases $3 \leq M \leq 10$. We use this procedure as it generates a wide variety of non-dominated solution sets (see Shang et al. (2022)) and so that we can compare our model performance with HV-Net. We split our datasets into 800K training points and 100K validation and test points, respectively. We note that as we only sample solution sets with at maximum a hundred solutions, our model is likely restricted to this setting. In addition, using the relation of eq. 17, we also pad all datasets of $M < 10$ to $M = 10$, by padding it with ones. As this does not change the hypervolume values of each solution set, we create a universal dataset that incorporates all generated objective cases.

4.2 PERFORMANCE

We train models with 64, 90, 128, and 256 channels (See Sec. 3.2 for architecture) on each objective case separately. In addition, we also train models on the combined dataset (See Sec. 4.1, denoted with -all) using 128 and 256 channels. All models have been trained with a learning rate of 10^{-5}, using Adam and the Mean Absolute Percentage Error (MAPE) loss function (de Myttenaere et al., 2016). As the (scaled) hypervolume tends to become small in high-dimension cases, more common loss functions such as the RMSE become rather small rather quickly, despite having large relative errors. The MAPE is better suited for this setting. For the separate models, we use a batch size of 64 and train for 200 epochs. For the models trained on all objective cases simultaneously, we train for 100 epochs with a batch size of 128. We pick the best model within these epochs based on the lowest loss on the validation partition.

We report MAPE test errors in Table 1 and compare with HV-Net (Shang et al., 2022). Note that HV-Net uses 1 million training datapoints, whereas we use 800K. When using 90 channels (DeepHV-90), our model has a comparable number of parameters (98.3K) w.r.t HV-Net (99.7K), however we obtain roughly a twofold increase of performance on most objective cases. In fact, using 64 channels (49K parameters), we already outperform HV-Net on most objective cases. This boost in performance is likely due to the incorporation of scale equivariance and an additional permutation invariance over the order of the objectives, which are not incorporated into HV-Net. For larger models, we obtain even better results.

Table 1: Test Mean Absolute Percentage Error (lower is better) for models trained on each objective M separately. Bold numbers indicate the best-performing model. Blue numbers indicate the best-performing model between HV-Net (Shang et al., 2022) and our models with a comparable number of model parameters.

M	DeepHV-64	DeepHV-90	DeepHV-128	DeepHV-256	HV-Net
3	0.00800	**0.00744**	0.00526	**0.00483**	0.015878
5	0.02065	**0.01608**	0.01689	**0.01209**	0.032067
8	0.04710	**0.03381**	0.01506	**0.01093**	0.042566
10	0.03203	**0.02783**	0.02395	**0.01148**	0.050867

We report results on DeepHV trained on all objective cases simultaneously in Table 2. Here it is shown that the general models have a significant increase in performance compared to the separate models. Interestingly, this shows that the model can generalize to multiple objective and can even gain additional performance in each objective case separately by seeing instances of others. This potentially allows for the training of a universal model that can be trained on even higher objective cases.

Table 2: Test Mean Absolute Percentage Error (lower is better) for models trained on each objective case separately, and for models trained on all objective cases simultaneously (denoted with additional -all). Bold numbers indicate the best-performing model in their respective architecture.

M	DeepHV-128	DeepHV-128-all	DeepHV-256	DeepHV-256-all
3	**0.00526**	0.00673	**0.00483**	0.00495
4	0.00964	**0.00887**	0.00782	**0.00714**
5	0.01689	**0.00956**	0.01209	**0.00773**
6	0.01511	**0.00954**	0.01354	**0.00761**
7	0.01164	**0.00900**	0.00967	**0.00707**
8	0.01506	**0.00849**	0.01093	**0.00657**
9	0.01506	**0.00796**	0.01216	**0.00601**
10	0.02395	**0.00772**	0.01148	**0.00598**

5 EXPERIMENTS

5.1 TIME COMPARISON

We compare the time performance of DeepHV with the exact hypervolume methods of Lacour et al. (2017) (complexity $O(N^{M/2+1})$) implemented in BoTorch (Balandat et al., 2020) and the method of Fonseca et al. (2006) (complexity ($O(N^{M-2} \log N)$) as implemented in Pymoo (Blank & Deb, 2020). In addition, we compare time performance with an approximate Monte Carlo (MC) hypervolume method implemented in Pymoo. We perform the tests solely on CPU to allow for fair comparison. Experiments shown in Fig. 2 comprised of the computation of 100 solution sets randomly drawn from the test sets. The MC method used 10K samples. In all cases, DeepHV is faster than the BoTorch and Pymoo MC implementation. Above $M = 5$, both variants of DeepHV are also faster than the exact Pymoo implementation. We also compare the Mean Absolute Percentage Error versus the computation time for DeepHV and the MC method with different number of samples in App. C.1.

5.2 BENCHMARKS

We empirically evaluate DeepHV on a range synthetic test problems from the DTLZ (Deb et al., 2002b) problem suite, which contains standard test problems from the MO optimization literature that are scalable in the number of input variables d and objectives M and contain vastly different types of Pareto fronts. In Sec. 5.2.1, we test DeepHV in the context of multi-objective evolutionary algorithms (MO EAs) and in Sec. 5.2.2 in the context of Bayesian optimization (BO). In the latter, we also evaluate on a real-world vehicle safety problem (Yang et al., 2005). See App. B.2 for more details regarding the problems and reference point specifications.

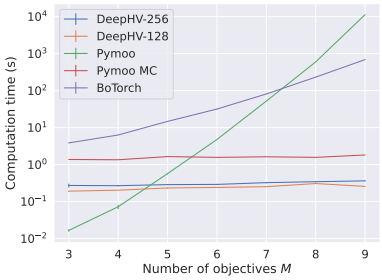

Figure 2: Comparison of computation time in seconds, shown on log-scale. We report the means and standard errors across 3 trials.

5.2.1 MULTI-OBJECTIVE EVOLUTIONARY ALGORITHMS

To use DeepHV in the context of MO EAs, we replace the exact computation of the hypervolume in SMS-EMOA (Beume et al., 2007) with that of DeepHV, and compare performance. In addition, we compare performance with NSGA-II (Deb et al., 2002a). SMS-EMOA (Beume et al., 2007) is an EA that uses the hypervolume to guide optimization. In its original form, it uses a steady-state $(\mu + 1)$ approach to evolve its population, where at each generation, one offspring is created and added to the population. Next, one solution is removed from the population by first ranking the population into separate fronts, followed by the removal of the solution that has the least contribution to the hypervolume of its respective front. This is done by computing the hypervolume contribution of each solution, by computing the hypervolume difference between the respective front, and the front without the individual solution. Instead of a steady-state approach, we use a $(\mu + \mu)$ approach to evolve the population, where we effectively double the population at each generation, and then need to select half of the population using the selection strategy described above. We stress that this is an extensive test of the performance of DeepHV, as the model needs to predict both the correct hypervolume of the front, but also the contribution of each solution in the front (i.e. the front and all of its sub-fronts need to be predicted with high accuracy). In the case of $M > 5$, we use the MC approximation of the hypervolume instead of the exact hypervolume as its use becomes computationally prohibitive.

As another baseline, we compare with the Non-dominated Sorting Genetic Algorithm (NSGA-II) which is a renowned MO EA, due to its low computational overhead. Although it generally is less sample-efficient than SMS-EMOA, it is often the method of choice due to a lack of scalable alternatives. In NSGA-II, the population is evolved using a $(\mu + \mu)$ approach, then individuals are first selected front wise using non-dominated sorting and then, remaining individuals are selected based on the Manhattan distance in the objective space, in order to maintain a diverse population. The task of the Manhattan distance corresponds to that of the hypervolume contribution in SMS-EMOA.

For each algorithm, we use a population size of 100 individuals and run for 10 generations, resulting into 1000 function evaluations, and we record the exact hypervolume of each generation. Further algorithmic details can be found in the App. B.1.

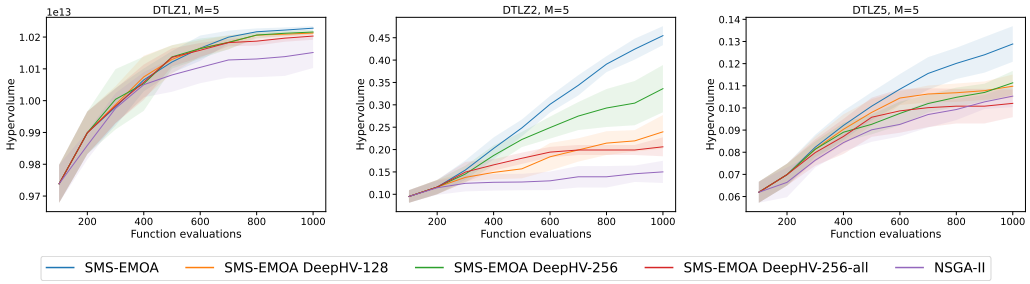

Figure 3: Optimization performance in terms of exact hypervolume (higher is better) versus function evaluations. We report the means and 2 standard errors across 5 trials.

Fig. 3 presents results on the DTLZ1, DTLZ2, and DTLZ5 problems for $M = 5$ and Fig. 4 reports the same problems for $M = 8$. We report results on additional synthetic test problems, also including the objective cases $M = 3$ and $M = 10$ in App. C.2. The specific DeepHV model using 256 channels (DeepHV-256) typically outperforms the smaller DeepHV-128 model. On most problems there is no clear performance difference between the general model DeepHV-256-all and DeepHV-256.

The closeness of the DeepHV variations to the exact SMS-EMOA provides an indication on the ability of the models to generalize to different Pareto front types. We observe that on all problems, exact SMS-EMOA (blue line) typically obtains higher hypervolumes than the DeepHV variations. For the $M = 3$ objective case, these difference are minuscule but for the $M = 5$ objective case this becomes more pronounced. This could be due to the fact that the generated training data (see Sec. 4.1) does not provide enough examples on these specific Pareto front shapes. In addition, with more objectives, the objective space also increases, requiring more training data to adequately learn it. This is also supported by the fact that in the objective cases $M = 8$ and $M = 10$, the SMS-EMOA equipped with the MC approximation method typically outperforms the DeepHV models. However, in all cases, the DeepHV models outperform NSGA-II, with performance differences becoming more pronounced at higher objective cases.

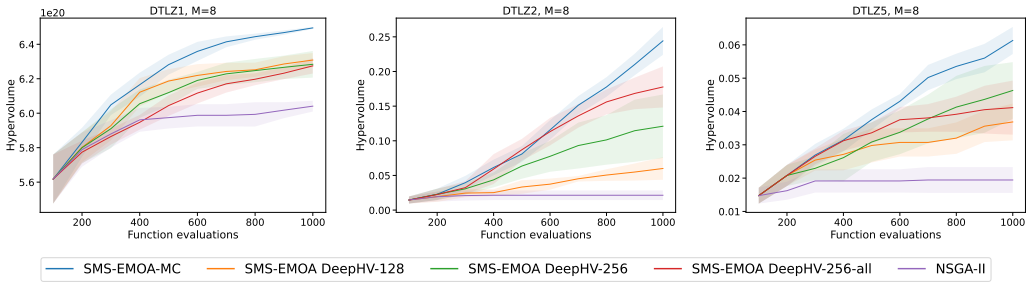

Figure 4: Optimization performance in terms of exact hypervolume (higher is better) versus function evaluations. We report the means and 2 standard errors across 5 trials.

5.2.2 Multi-objective Bayesian optimization

We use DeepHV in the context of MO BO by using DeepHV to compute the hypervolume improvement (HVI, eq. 7). We use the HVI in a Upper Confidence Bound (UCB) acquisition function, which we name UCB-DeepHV, which is defined as follows:

$$\alpha_{\text{UCB}-\text{DeepHV}}(\mathbf{x}, \beta) = \text{HVI}(\mathbf{f}_{\text{UCB}}(\mathbf{x}, \beta)), \text{ where } \mathbf{f}_{\text{UCB}}(\mathbf{x}, \beta) = \mu(\mathbf{x}) + \beta\sigma(\mathbf{x}). \tag{23}$$

Here, $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ are the mean and standard deviation of the GP posterior at input $\mathbf{x}$, whereas β is a trade-off parameter that controls exploration and exploration. In this work we use $\beta = 0.3$.

We compare with the state-of-the-art methods qParEgo and qEHVI (Daulton et al., 2020). ParEgo (Knowles, 2006) randomly scalarizes the objectives and uses the Expected Improvement acquisition function (Jones et al., 1998). qParego uses an MC-based Expected Improvement acquisition function and uses exact gradients via auto-differentiation for acquisition optimization. Expected Hypervolume Improvement (EHVI) is the expectation of the HVI (Yang et al., 2019), thus integrating the HVI over the Gaussian process posterior. qEHVI is a parallel extension of EHVI and allows for acquistion optimization using auto-differentation and exact gradients, generally making it more computationally tractable than EHVI. qEHVI was empirically shown to outperform many other MO BO methods (including qParEgo) (Daulton et al., 2020; 2022).

We evaluate optimization performance in terms of the exact hypervolume on four test problems. Note that above the five objectives case, qEHVI becomes computationally prohibitive. We provide additional experiments on other objective cases in Sec. C.3. Fig. 5 shows results for a range of DTLZ synthetic test problems. In general, it is expected that qEHVI is superior to UCB-DeepHV, as it uses the exact hypervolume and integrates out the uncertainty of the GP posterior in a more elaborate way. So if DeepHV performs on par with qEHVI, it would mean that it approximated the hypervolume of the test problems well. qEHVI ranks best on DTLZ2 and DTLZ7. On DTLZ1 and DTLZ5, qEHVI performs worse than qParego and UCB-DeepHV. UCB-DeepHV ranks higher than qParEgo on DTLZ5 and DTLZ7, and on par on the other problems, which is a strong result. On DTLZ5 the UCB-DeepHV is the highest ranked algorithm. The DeepHV models generally share similar performance, where some models perform slightly better than others on different problems.

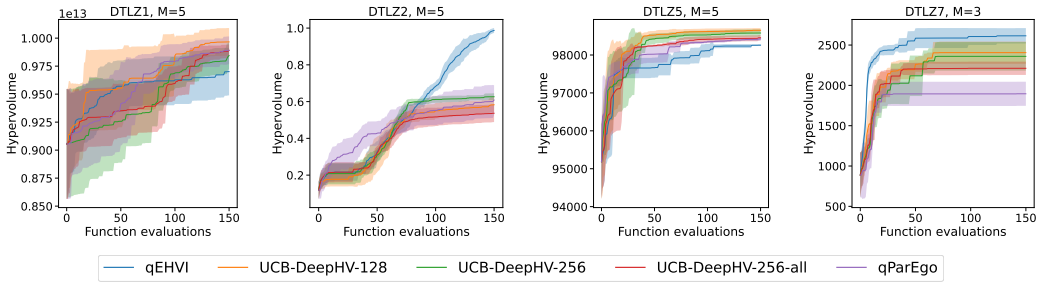

Figure 5: Sequential optimization performance in terms of hypervolume (higher is better) versus function evaluation on 5-objective DTLZ1, DTLZ2 and DTLZ5 problems, and on the 3-objective DTLZ7 problem. We report the means and 2 standard errors across 5 trials.

6 Discussion

We presented DeepHV, a novel hypervolume approximation based on deep learning. DeepHV is scale-equivariant in each of the objectives as well as permutation invariant w.r.t. both the objectives and the samples, thus incorporating important symmetry properties of the hypervolume. We showed that our method obtains lower approximation error compared to HV-Net that does not incorporate all of these symmetries. We showed that DeepHV achieved lower approximation error than an MC based approximation and is considerably faster than exact methods, becoming more pronounced at high objective cases $M > 4$. Therefore, we believe DeepHV is promising as a replacement for computationally demanding exact hypervolume computations in state-of-the-art (high-throughput) methods. A limitation is that the DeepHV models are currently only trained on solution sets with at maximum 100 non-dominated solutions, restricting their use to this setting. However it can handle arbitrary amount of objectives and samples by design. We tested DeepHV in context of multi-objective Bayesian optimization and multi-objective evolutionary algorithms and we showed competitive performance compared with state-of-the-art methods. We believe these results are promising, and with better training data generation methods, DeepHV could potentially be even more competitive with exact hypervolume methods.

7 ETHICS STATEMENT

The paper presents a method that can be used in multi-objective optimization problems. These problems find widespread use throughout society. For example in material design (catalysts, medicines, etc.), car design or finance. Although many such applications are for the general good, there are also applications which can have ethically debatable motivations, for instance, when used in the design of toxins, narcotics, or weaponry.

8 REPRODUCIBILITY STATEMENT

We are in the process of releasing code, models and datasets for the current work. Where applicable, we have run experiments multiple times with different random seeds and have stated mean and standard error of the results. In addition, we have described important implementation and experimental details in either the main body of the text or the Appendix.

REFERENCES

Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BOTORCH: A framework for efficient Monte-Carlo Bayesian optimization. In *Advances in Neural Information Processing Systems*, 2020.

Nicola Beume, Boris Naujoks, and Michael Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 9 2007. doi: 10.1016/j.ejor.2006.08.008.

Julian Blank and Kalyanmoy Deb. Pymoo: Multi-Objective Optimization in Python. *IEEE Access*, 8: 89497–89509, 2020. doi: 10.1109/ACCESS.2020.2990567.

Jim Boelrijk, Bob Pirok, Bernd Ensing, and Patrick Forré. Bayesian optimization of comprehensive two-dimensional liquid chromatography separations. *Journal of Chromatography A*, 1659:462628, 12 2021. doi: 10.1016/j.chroma.2021.462628.

Karl Bringmann and Tobias Friedrich. Approximating the volume of unions and intersections of high-dimensional geometric objects. *Computational Geometry: Theory and Applications*, 43(6-7): 601–610, 8 2010. doi: 10.1016/j.comgeo.2010.03.004.

Settles Burr. Active Learning Literature Survey. Computer Sciences Technical Report. Technical Report January, Wisconsin-Madison Department of Computer Sciences, 2009.

Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. Differentiable expected hypervolume improvement for parallel multi-objective Bayesian optimization. *Advances in Neural Information Processing Systems*, 2020-Decem, 6 2020.

Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Multi-Objective Bayesian Optimization over High-Dimensional Search Spaces. In *The 38th Conference on Uncertainty in Artificial Intelligence*, 2022.

Arnaud de Myttenaere, Boris Golden, Bénédicte Le Grand, and Fabrice Rossi. Mean Absolute Percentage Error for regression models. *Neurocomputing*, 192:38–48, 6 2016. doi: 10.1016/J.NEUCOM.2015.12.114.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 4 2002a. doi: 10.1109/4235.996017.

Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable multi-objective optimization test problems. *Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002*, 1:825–830, 2002b. doi: 10.1109/CEC.2002.1007032.

Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable Test Problems for Evolutionary Multiobjective Optimization. In *Evolutionary Multiobjective Optimization*, pp. 105–145. Springer, London, 9 2005. doi: 10.1007/1-84628-137-7{_}6.

Carlos M. Fonseca, Luís Paquete, and Manuel López-Ibáñez. An improved dimension-sweep algorithm for the hypervolume indicator. *2006 IEEE Congress on Evolutionary Computation, CEC 2006*, pp. 1157–1163, 2006. doi: 10.1109/CEC.2006.1688440.

Jason Hartford, Devon R Graham, Kevin Leyton-Brown, and Siamak Ravanbakhsh. Deep models of interactions across sets. In *35th International Conference on Machine Learning, ICML 2018*, volume 5, 2018.

Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization 1998 13:4*, 13(4):455–492, 1998. doi: 10.1023/A:1008306431147.

Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285, 5 1996. doi: 10.1613/JAIR.301.

Joshua Knowles. ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1): 50–66, 2006. doi: 10.1109/TEVC.2005.851274.

K. F. Koledina, S. N. Koledin, A. P. Karpenko, I. M. Gubaydullin, and M. K. Vovdenko. Multi-objective optimization of chemical reaction conditions based on a kinetic model. *Journal of Mathematical Chemistry*, 57(2):484–493, 2 2019. doi: 10.1007/s10910-018-0960-z.

Renaud Lacour, Kathrin Klamroth, and Carlos M. Fonseca. A box decomposition algorithm to compute the hypervolume indicator. *Computers & Operations Research*, 79:347–360, 3 2017. doi: 10.1016/J.COR.2016.06.021.

Xingtao Liao, Qing Li, Xujing Yang, Weigang Zhang, and Wei Li. Multiobjective optimization for crash safety design of vehicles using stepwise regression model. *Structural and Multidisciplinary Optimization 2007 35:6*, 35(6):561–569, 11 2007. doi: 10.1007/S00158-007-0163-X.

Benjamin P. MacLeod, Fraser G.L. Parlane, Connor C. Rupnow, Kevan E. Dettelbach, Michael S. Elliott, Thomas D. Morrissey, Ted H. Haley, Oleksii Proskurin, Michael B. Rooney, Nina Taherimakhsousi, David J. Dvorak, Hsi N. Chiu, Christopher E.B. Waizenegger, Karry Ocean, Mehrdad Mokhtari, and Curtis P. Berlinguette. A self-driving laboratory advances the Pareto front for material properties. *Nature Communications 2022 13:1*, 13(1):1–10, 2 2022. doi: 10.1038/s41467-022-28580-6.

Kaname Narukawa and Tobias Rodemann. Examining the performance of evolutionary many-objective optimization algorithms on a real-world application. *Proceedings - 2012 6th International Conference on Genetic and Evolutionary Computing, ICGEC 2012*, pp. 316–319, 2012. doi: 10.1109/ICGEC.2012.90.

Steve O'Hagan, Warwick B. Dunn, Marie Brown, Joshua D. Knowles, and Douglas B. Kell. Closed-loop, multiobjective optimization of analytical instrumentation: Gas chromatography/time-of-flight mass spectrometry of the metabolomes of human serum and of yeast fermentations. *Analytical Chemistry*, 77(1):290–303, 2005. doi: 10.1021/ac049146x.

Akira Oyama, Takehisa Kohira, Hiromasa Kemmotsu, Tomoaki Tatsukawa, and Takeshi Watanabe. Simultaneous structure design optimization of multiple car models using the K computer. *2017 IEEE Symposium Series on Computational Intelligence, SSCI 2017 - Proceedings*, 2018-January: 1–4, 2 2018. doi: 10.1109/SSCI.2017.8285350.

Alma A.M. Rahat, Richard M Everson, and Jonathan E Fieldsend. Alternative infill strategies for expensive multi-objective optimisation. In *GECCO 2017 - Proceedings of the 2017 Genetic and Evolutionary Computation Conference*, volume 8, pp. 873–880, 2017. doi: 10.1145/3071178.3071276.

Ke Shang, Weiyu Chen, Weiduo Liao, and Hisao Ishibuchi. HV-Net: Hypervolume Approximation based on DeepSets. *IEEE Transactions on Evolutionary Computation*, 2022. doi: 10.1109/TEVC.2022.3181306.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, 4:2951–2959, 2012.

Ryoji Tanabe and Hisao Ishibuchi. An easy-to-use real-world multi-objective optimization problem suite. *Applied Soft Computing*, 89:106078, 4 2020. doi: 10.1016/J.ASOC.2020.106078.

Weisen Tang, Hai Lin Liu, Lei Chen, Kay Chen Tan, and Yiu ming Cheung. Fast hypervolume approximation scheme based on a segmentation strategy. *Information Sciences*, 509:320–342, 1 2020. doi: 10.1016/J.INS.2019.02.054.

Lyndon While, Lucas Bradstreet, and Luigi Barone. A fast way of calculating exact hypervolumes. *IEEE Transactions on Evolutionary Computation*, 16(1):86–95, 2 2012. doi: 10.1109/TEVC.2010.2077298.

Jian Wu, Saul Toscano-Palmerin, Peter I. Frazier, and Andrew Gordon Wilson. Practical multi-fidelity Bayesian optimization for hyperparameter tuning. *35th Conference on Uncertainty in Artificial Intelligence, UAI 2019*, 2019.

Kaifeng Yang, Michael Emmerich, André Deutz, and Thomas Bäck. Multi-Objective Bayesian Global Optimization using expected hypervolume improvement gradient. *Swarm and Evolutionary Computation*, 44(October 2018):945–956, 2019. doi: 10.1016/j.swevo.2018.10.007.

R. J. Yang, N. Wang, C. H. Tho, J. P. Bobineau, and B. P. Wang. Metamodeling Development for Vehicle Frontal Impact Simulation. *Journal of Mechanical Design*, 127(5):1014–1020, 9 2005. doi: 10.1115/1.1906264.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbhakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola. Deep sets. In *Advances in Neural Information Processing Systems*, 2017.

A PROOFS

A.1 PROOF OF SECTION3.1

Proof To show that the layers in eq. 21 are really G-equivariant, let $\mathbf{Y}^I := (\mathbf{Y}^{(i)})_{i\in I}$ and $g(\mathbf{Y}^I)$ the left hand side of eq. 21, and $(\mathbf{c}, \tau, \sigma) \in G$. Then $(\mathbf{c}, \tau, \sigma) \odot \mathbf{Y}^I = ((\mathbf{c}, \tau, \sigma) \odot \mathbf{Y}^{(i)})_{i\in I}$. We can also see that:

$$\mathbf{s}((\mathbf{c}, \tau, \sigma) \odot \mathbf{Y}^{(i)}) = \left[\max_{n\in[N]} |c_m \cdot y_{\tau(m),\sigma(n)}| \,\middle|\, m \in [M]\right]^\top \tag{24}$$

$$= \left[c_m \cdot \max_{n\in[N]} |y_{\tau(m),n}| \,\middle|\, m \in [M]\right]^\top = \mathbf{c} \cdot \mathbf{s}(\mathbf{Y}^{(i)})^\tau, \tag{25}$$

where σ acts trivially on the $(M \times 1)$-vector $\mathbf{s}(\mathbf{Y}^{(i)})$. If we act on the defining equation for $\mathbf{Y}^{(i)}_\oslash$:

$$\mathbf{Y}^{(i)} = \mathbf{s}(\mathbf{Y}^{(i)}) \odot \mathbf{Y}^{(i)}_\oslash, \tag{26}$$

with $(\mathbf{c}, \tau, \sigma)$ we then get:

$$(\mathbf{c}, \tau, \sigma) \odot \mathbf{Y}^{(i)} = (\mathbf{c}, \tau, \sigma) \odot \left(\mathbf{s}(\mathbf{Y}^{(i)}) \odot \mathbf{Y}^{(i)}_\oslash\right) \tag{27}$$

$$= (\mathbf{c} \cdot \mathbf{s}(\mathbf{Y}^{(i)})^\tau, \tau, \sigma) \odot \mathbf{Y}^{(i)}_\oslash \tag{28}$$

$$= (\mathbf{s}((\mathbf{c}, \tau, \sigma) \odot \mathbf{Y}^{(i)}), \tau, \sigma) \odot \mathbf{Y}^{(i)}_\oslash \tag{29}$$

$$= \mathbf{s}((\mathbf{c}, \tau, \sigma) \odot \mathbf{Y}^{(i)}) \odot (\tau, \sigma) \odot \mathbf{Y}^{(i)}_\oslash. \tag{30}$$

This shows that:

$$\left((\mathbf{c}, \tau, \sigma) \odot \mathbf{Y}^{(i)}\right)_\oslash = (\tau, \sigma) \odot \mathbf{Y}^{(i)}_\oslash. \tag{31}$$

With this we get:

$$
\begin{aligned}
& h((\mathbf{c}, \tau, \sigma) \odot \mathbf{Y}^{(i)}) && (32)\\
& := w_1^{(o,i)} \cdot \left((\mathbf{c}, \tau, \sigma) \odot \mathbf{Y}^{(i)}\right)_{\oslash} + w_2^{(o,i)} \cdot \mathbf{1}_M \cdot \left(\left((\mathbf{c}, \tau, \sigma) \odot \mathbf{Y}^{(i)}\right)_{\oslash}\right)_{\mathbb{M}:} && (33)\\
& \quad + w_3^{(o,i)} \cdot \left(\left((\mathbf{c}, \tau, \sigma) \odot \mathbf{Y}^{(i)}\right)_{\oslash}\right)_{:\mathbb{M}} \cdot \mathbf{1}_N^\top && (34)\\
& \quad + w_4^{(o,i)} \cdot \left(\left((\mathbf{c}, \tau, \sigma) \odot \mathbf{Y}^{(i)}\right)_{\oslash}\right)_{\mathbb{M}:\mathbb{M}} \cdot \mathbf{1}_{M\times N} + w_5^{(o,i)} \cdot \mathbf{1}_{M\times N} && (35)\\
& = w_1^{(o,i)} \cdot (\tau, \sigma) \odot \mathbf{Y}_{\oslash}^{(i)} + w_2^{(o,i)} \cdot (\tau, \sigma) \odot \left(\mathbf{1}_M \cdot \left(\mathbf{Y}_{\oslash}^{(i)}\right)_{\mathbb{M}:}\right) && (36)\\
& \quad + w_3^{(o,i)} \cdot (\tau, \sigma) \odot \left(\left(\mathbf{Y}_{\oslash}^{(i)}\right)_{:\mathbb{M}} \cdot \mathbf{1}_N^\top\right) && (37)\\
& \quad + w_4^{(o,i)} \cdot (\tau, \sigma) \odot \left(\left(\mathbf{Y}_{\oslash}^{(i)}\right)_{\mathbb{M}:\mathbb{M}} \cdot \mathbf{1}_{M\times N}\right) + w_5^{(o,i)} \cdot (\tau, \sigma) \odot \mathbf{1}_{M\times N} && (38)\\
& = (\tau, \sigma) \odot h(\mathbf{Y}^{(i)}). && (39)
\end{aligned}
$$

With these equations we can finally get:

$$
\begin{aligned}
g((\mathbf{c}, \tau, \sigma) \odot \mathbf{Y}^I) &= \sigma_\alpha \left(\frac{1}{|I|} \sum_{i\in I} \mathbf{s}\left((\mathbf{c}, \tau, \sigma) \odot \mathbf{Y}^{(i)}\right) \odot h\left((\mathbf{c}, \tau, \sigma) \odot \mathbf{Y}^{(i)}\right)\right) && (40)\\
&= \sigma_\alpha \left(\frac{1}{|I|} \sum_{i\in I} \mathbf{c} \cdot \mathbf{s}(\mathbf{Y}^{(i)})^\tau \odot (\tau, \sigma) \odot h\left(\mathbf{Y}^{(i)}\right)\right) && (41)\\
&= \sigma_\alpha \left(\frac{1}{|I|} \sum_{i\in I} (\mathbf{c} \cdot \mathbf{s}(\mathbf{Y}^{(i)})^\tau, \tau, \sigma) \odot h\left(\mathbf{Y}^{(i)}\right)\right) && (42)\\
&= \sigma_\alpha \left(\frac{1}{|I|} \sum_{i\in I} (\mathbf{c}, \tau, \sigma) \odot \mathbf{s}(\mathbf{Y}^{(i)}) \odot h\left(\mathbf{Y}^{(i)}\right)\right) && (43)\\
&= \sigma_\alpha \left((\mathbf{c}, \tau, \sigma) \odot \frac{1}{|I|} \sum_{i\in I} \mathbf{s}(\mathbf{Y}^{(i)}) \odot h\left(\mathbf{Y}^{(i)}\right)\right) && (44)\\
&= (\mathbf{c}, \tau, \sigma) \odot \sigma_\alpha \left(\frac{1}{|I|} \sum_{i\in I} \mathbf{s}(\mathbf{Y}^{(i)}) \odot h\left(\mathbf{Y}^{(i)}\right)\right) && (45)\\
&= (\mathbf{c}, \tau, \sigma) \odot g(\mathbf{Y}^I). && (46)
\end{aligned}
$$

where we in the second to last step made use of the homogeneity of the activation function σ_α and that it is applied element-wise. This shows that g is G-equivariant.

B Details on Experiments

B.1 Algorithmic details

Multi-objective Evolutionary Algorithms: For both NSGA-II and SMS-EMOA we use the default settings in the open-source implementation of Pymoo [1], where only for SMS-EMOA we replace the hypervolume computation with DeepHV or the MC hypervolume approximation and all other settings are kept default. For the MC hypervolume approximation we use $10,000$ samples.

Multi-objective Bayesian Optimization: For all methods and experiments we use an independent Gaussian Process (GP) with a constant mean function and a Matérn-5/2 kernel with automatic relevance detection (ARD) and fit the GP hyperparameters by maximizing the marginal log-likelihood.

[1] (https://github.com/anyoptimization/pymoo)

All experiments are initialized with $2d + 1$ randomly drawn datapoints. We then allow for a optimization budget of 150 function evaluations. For qEHVI and qParEgo we use 128 quasi-MC samples. All methods are optimized using L-BFGS-B with 10 restarts and 64 raw samples (according to BoTorch naming convention).

Methods using DeepHV: We ensure that solution sets passed to DeepHV only contain non-dominated solutions and are shifted so that the reference point lies at $[0]^M$. This is done by subtracting the specified reference point from the solution set and applying non-dominated sorting to only keep the non-dominated solutions.

B.2 Synthetic problems

Information on the reference points used for hypervolume computation of each problem can be found in Table 3 and are based on suggested values in Pymoo (Blank & Deb, 2020) and BoTorch (Balandat et al., 2020). The DTLZ problems are minimization problems. As BoTorch assumes maximization, we multiply the objectives and reference points for all synthetic problems by -1 and maximize the resulting objectives.

In the multi-objective Bayesian optimization problems we use the implementations of the synthetic functions as provided in Botorch (Balandat et al., 2020). Whereas in the multi-objective evolutionary algorithm experiments we use the implementations of Pymoo (Blank & Deb, 2020).

DTLZ: The DTLZ test suite contains a range of test functions which are considered standard test problems in the multi-objective optimization literature. Mathematical formulas of each test function are provided in Deb et al. (2005). The problems are scalable in both input dimension d and in the number of objectives M. We set the input dimension to $d = 2M$ for each problem. The shape of the Pareto front of each of these problems can be viewed in the Pymoo documentation (`https://pymoo.org/problems/many/dtlz.html`).

Table 3: Specification of reference points used for all benchmark problems.

Problem	Reference Point Botorch	Reference Point Pymoo
DTLZ1	$[-400]^M$	$[400]^M$
DTLZ2	$[-1.1]^M$	$[1]^M$
DTLZ5	$[-10]^M$	$[1]^M$
DTLZ7	$[-15]^M$	$[15]^M$
Convex DTLZ2	Not performed	$[1]^M$
Vehicle Safety	$[-1864.72, -11.82, 0.20]$	Not performed

Vehicle Crash Safety: The vehicle crash safety problem is described in Tanabe & Ishibuchi (2020) and is a 3-objective problem with $d = 5$ parameters describing the size of different components of the vehicle's frame. The goal is to minimize mass, toe-box intrusion (i.e. vehicle damage), and acceleration in a frontal collision.

C Additional results

C.1 Timing results

We show the comparison of the Mean Absolute Percentage Error (MAPE) versus the computation for DeepHV and the approximate MC method (See Sec. 5.1 for details) for the cases of $M = 3$, $M = 5$, $M = 8$ and $M = 10$, in Fig. 6. It is shown that in all cases, DeepHV obtains a significantly lower MAPE at similar runtimes. The approximate MC method requires much more computation time to obtain similar a MAPE, which significantly increases at higher objective case M.

C.2 Multi-Objective Evolutionary Algorithms

To study the performance of DeepHV in the context of multi-objective algorithms on a broader range of problems, we ran additional experiments on the DTLZ7 and Convex DTLZ2 problems. These are

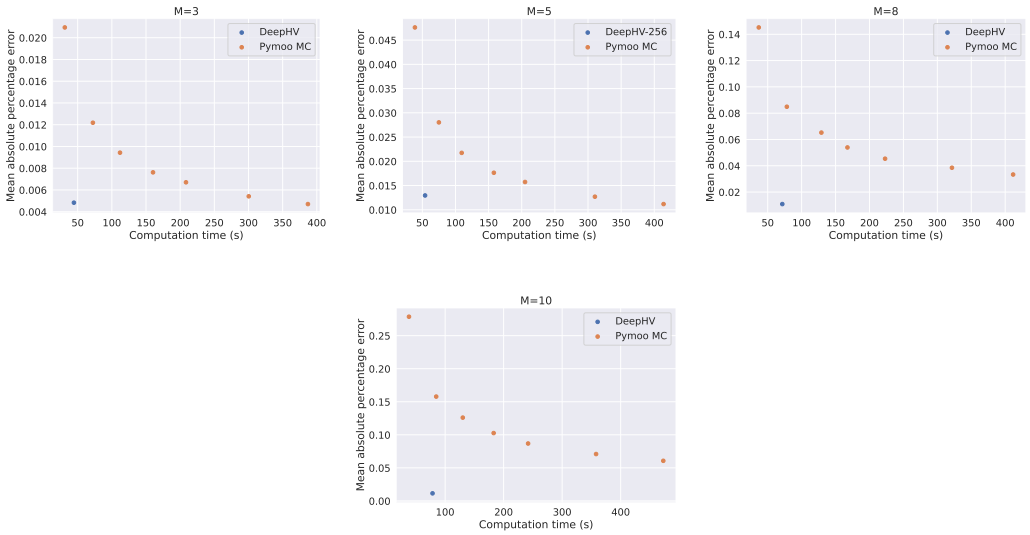

Figure 6: Comparison of computation time in seconds of the computation of 10000 solution sets randomly drawn for the objective cases $M = 3$, $M = 5$, $M = 8$ and $M = 10$. All computations performed on an Intel(R) Xeon(R) CPU E5-2640 CPU v4.

shown for the case $M = 5$ in Fig. C.2. On DTLZ7 ($M = 5$), all methods have relatively similar performance, with NSGA-II performing best on average, followed by DeepHV-256 and the exact SMS-EMOA. On Convex DTLZ2, SMS-EMOA using the exact hypervolume scores best, followed by the DeepHV models. NSGA-II performs significantly worse. We also report results for DTZL1,

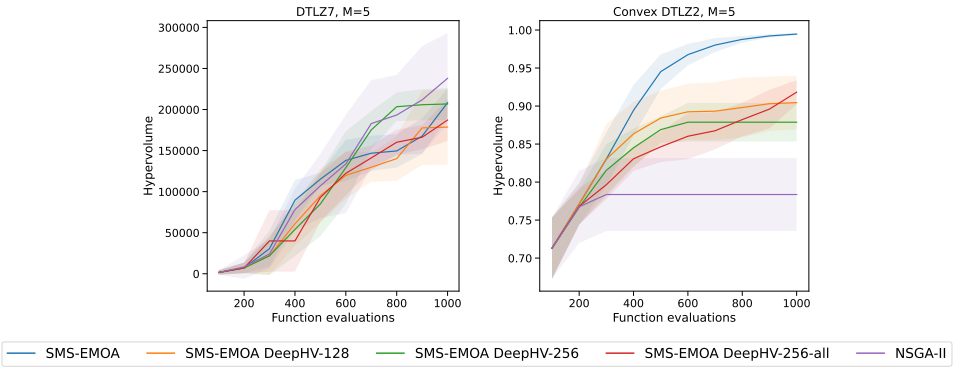

Figure 7: Optimization performance in terms of hypervolume (higher is better) versus function evaluations on 5-objective DTLZ7, and Convex DTLZ2 problems. We report the means and 2 standard errors across 5 trials.

DTLZ2, Convex DTLZ2, DLTZ5, and DTLZ7 for the cases $M = 3$ and $M = 10$ in Fig. 8 and Fig. 9, respectively. All methods perform relatively similar on all problems in the case $M = 3$, where only DTLZ2 and Convex DTLZ2 SMS-EMOA with exact hypervolume computation performs slightly better at higher function evaluations. Hence for the case $M = 3$, all DeepHV models seems to have learned a satisfactory hypervolume approximation. This picture changes for the $M = 10$ case, where the SMS-EMOA-MC is best on all problems. The DeepHV models are competitive for the DTLZ1 and Convex DTLZ2 problems, but are inferior for the DTLZ2 and DTLZ5 problems. Again, the DeepHV models always outperform NSGA-II. Note that for DTLZ7 ($M = 10$), all methods do not seem to find solutions better than the defined reference point.

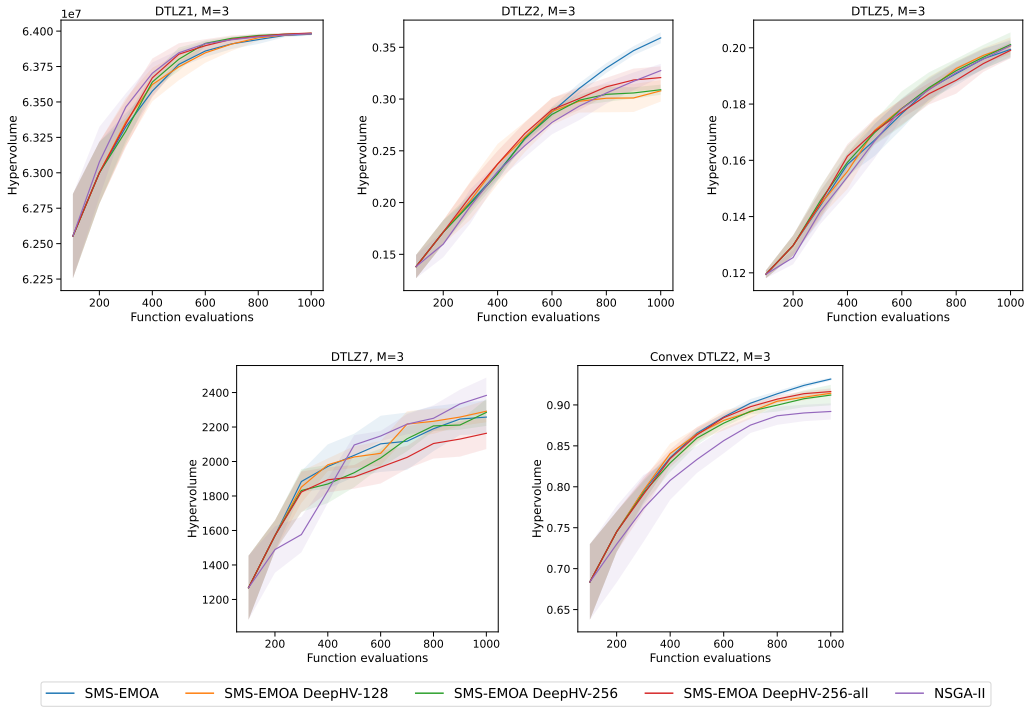

Figure 8: Optimization performance in terms of hypervolume (higher is better) versus function evaluations on 3-objective DTLZ1, DTLZ2 and DTLZ5, DTLZ7 and Convex DTLZ2 problems. We report the means and 2 standard errors across 5 trials.

C.3 Multi-Objective Bayesian Optimization

In order to further evaluate the performance of DeepHV in the context of multi-objective Bayesian optimization, we also report results for the DTLZ1, DTLZ2 and DTLZ5 synthetic test problems for the objective cases $M = 3$ and $M = 4$ in Fig. 10 and Fig. 11, respectively. In addition, we show results for the vehicle crash safety problem ($M = 3$). In general, DeepHV is competitive with qParEgo and qEHVI and even ranks best on DTLZ1 (M=3). Although qEHVI clearly performs best on DTLZ2 (both cases), DeepHV is either on par with qParego ($M = 3$) or slightly worse ($M = 4$).

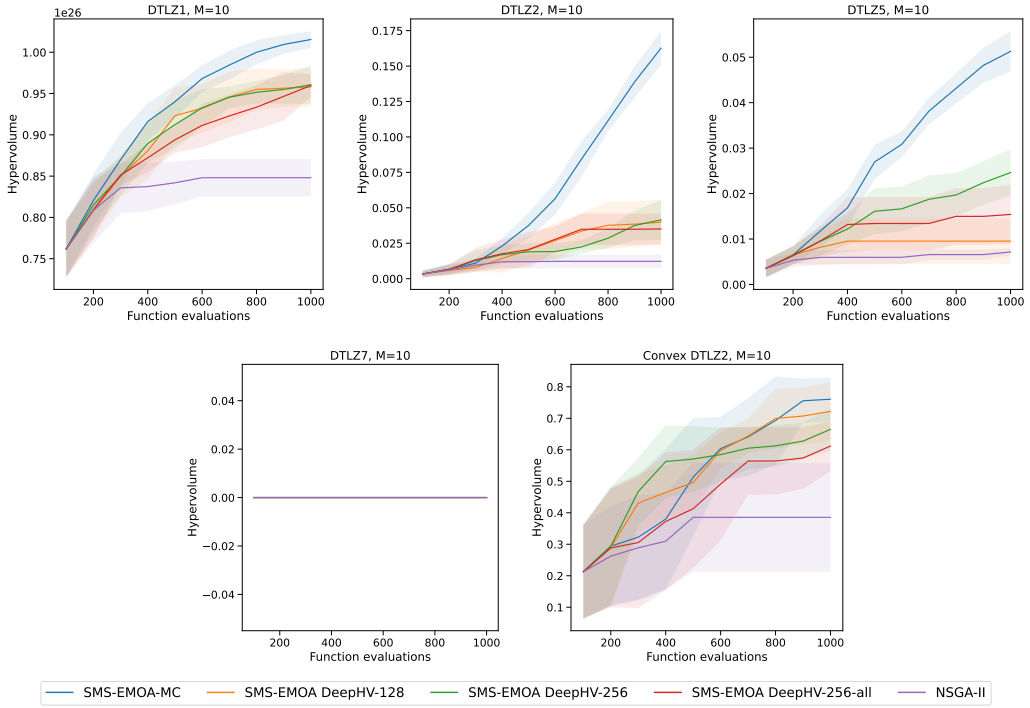

Figure 9: Optimization performance in terms of hypervolume (higher is better) versus function evaluations on 10-objective DTLZ1, DTLZ2 and DTLZ5, DTLZ7 and Convex DTLZ2 problems. We report the means and 2 standard errors across 5 trials.

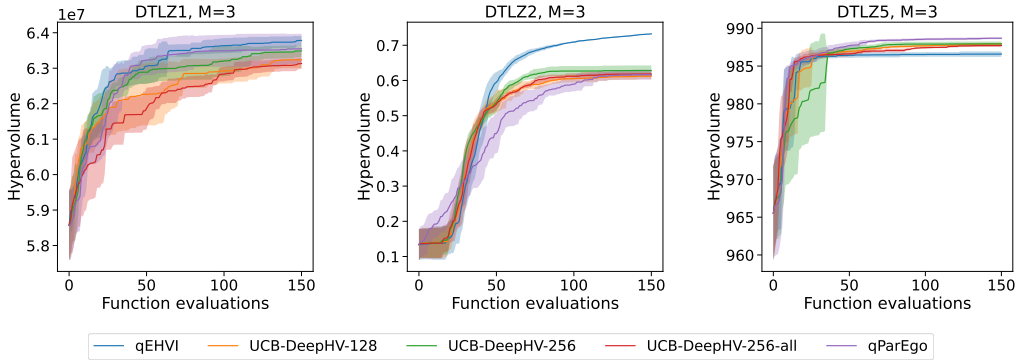

Figure 10: Sequential optimization performance in terms of hypervolume (higher is better) versus function evaluation on 3-objective DTLZ1, DTLZ2 and DTLZ5 problems.

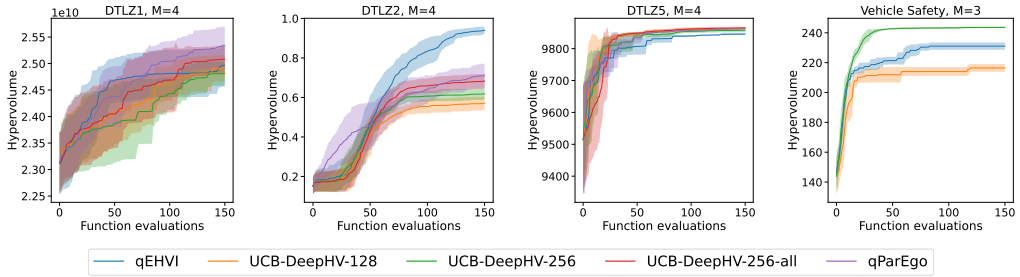

Figure 11: Sequential optimization performance in terms of hypervolume (higher is better) versus function evaluation on 4-objective DTLZ1, DTLZ2 and DTLZ5 problems and 3-objective Vehicle Safety problem. Note to reviewer: for the Vehicle Safety problem the 256 models are not yet finished. We show qEHVI in green, UCB-DeepHV-128 in blue and qParego in orange.