LEARNING THE PARETO FRONT WITH HYPERNETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-objective optimization (MOO) problems are prevalent in machine learning. These problems have a set of optimal solutions, called the Pareto front, where each point on the front represents a different trade-off between possibly conflicting objectives. Recent MOO methods can target a specific desired ray in loss space, however most approaches still face two grave limitations: (i) A separate model has to be trained for each point on the front; and (ii) The exact trade-off must be known prior to the optimization process. Here, we tackle the problem of learning the entire Pareto front, with the capability of selecting a desired operating point on the front after training. We call this new setup *Pareto-Front Learning* (PFL).

We describe an approach to PFL implemented using HyperNetworks, which we term *Pareto HyperNetworks* (PHNs). PHN learns the entire Pareto front simultaneously using a single hypernetwork, which receives as input a desired preference vector and returns a Pareto-optimal model whose loss vector is in the desired ray. The unified model is *runtime efficient* compared to training multiple models, and generalizes to new operating points not used during training. We evaluate our method on a wide set of problems, from multi-task regression and classification to fairness. PHNs learns the entire Pareto front in roughly the same time as learning a single point on the front, and also reaches a better solution set. PFL opens the door to new applications where models are selected based on preferences that are only available at run time.

1 INTRODUCTION

Multi-objective optimization (MOO) aims to optimize several, possibly conflicting objectives. MOO is abundant in machine learning problems, from multi-task learning (MTL), where the goal is to learn several tasks simultaneously, to constrained problems. In such problems, one aims to learn a single task while finding solutions that satisfy properties like fairness or privacy. It is common to optimize the main task while adding loss terms to encourage the learned model to obtain these properties.

MOO problems have a set of optimal solutions, the *Pareto front*, each reflecting a different trade-off between objectives. Points on the Pareto front can be viewed as an intersection of the front with a specific direction in loss space (a ray, Figure 1). We refer to this direction as *preference vector*, as it represents a single trade-off between objectives. When a direction is known in advance, it is possible to obtain the corresponding solution on the front (Mahapatra & Rajan, 2020).

However, in many cases, we are interested in more than one predefined direction, either because the trade-off is not known prior to training, or because there are many possible trade-offs of interest. For example, network routing optimization aims to maximize bandwidth, minimizing latency, and obey fairness. However, the cost and trade-off vary from one application running in the network to another, or even continuously change in time. The challenge remains to design a model that can be applied at inference time to any given preference direction, even ones not seen during training. We call this problem *Pareto front learning* (PFL).

Although several recent studies (Yang et al., 2019; Parisi et al., 2016) suggested to learn the trade-off curve in MOO problems, there is no existing scalable and general purpose MOO approach that provides Pareto-optimal solutions for numerous preferences in objective space. Classical approaches,

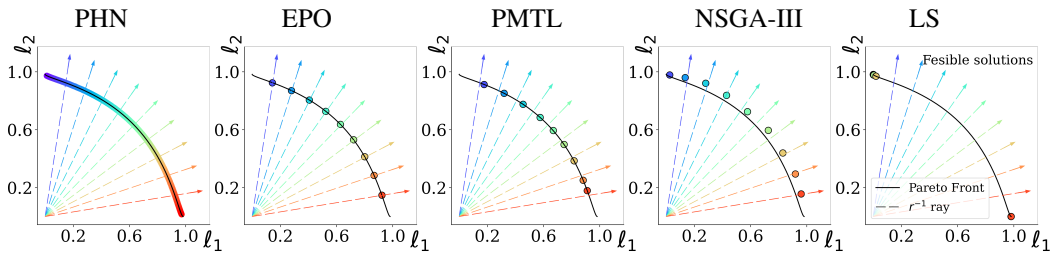


Figure 1: *Illustrative example using the popular task of Fonseca (1995)*: demonstrating the relation between Pareto front, preference rays and solutions. Pareto front (black solid line) for a 2D loss space and several rays (colored dashed lines) which represent various possible preferences. **Left:** A single PHN-EPO model learns the entire Pareto front, mapping any given preference ray to its corresponding solution on the front. The data and task are detailed in Section 5.1.

like genetic algorithms, do not scale to modern high dimensional problems. It is possible in principle to run a single-direction optimization multiple times, each for different preference, but this approach faces two major drawbacks: (i) *Scalability* – the number of models to be trained in order to cover the objective space grows exponentially with the number of objectives; and (ii) *Flexibility* – the decision maker cannot switch freely between preferences, unless all models are trained and stored in advance.

Here we put forward a new view of PFL as a problem of learning a conditional model, where the conditioning is over the preference direction. During training, a single unified model is trained to produce Pareto optimal solutions while satisfying the given preferences. During inference, the model covers the Pareto front by varying the input preference vector. We further describe an architecture that implements this idea using HyperNetworks. Specifically, we train a hypernetwork, termed *Pareto Hypernetwork* (PHN), that given a preference vector as an input, produces a deep network model tuned for that objective preference. Training is applied to preferences sampled from the $m-$dimensional simplex where m represents the number of objectives.

We evaluate PHN on a wide set of problems, from multi-class classification, through fairness and image segmentation to multi-task regression. We find that PHN can achieve superior overall solutions while being $10 \sim 50$ times faster (see Figure 5). PHN addresses both scalability and flexibility. Training a unified model allows using any objective preference at inference time. Finally, as PHN generates a continuous parametrization of the entire Pareto front, it could open new possibilities to analyze Pareto optimal solutions in large scale neural networks.

Our paper has the following contributions:

- We define the *Pareto-front learning* problem – learn a model that at inference time can operate on any given preference vector, providing a Pareto-optimal solution for that specified objective trade-off.
- We describe Pareto hypernetworks (PHN), a unified architecture based on hypernetworks that addresses PFL and show it can be effectively trained.
- Empirical evaluations on various tasks and datasets demonstrate the capability of PHNs to generate better objective space coverage compared to multiple baseline models, with significant improvement in training time.

2 MULTI-OBJECTIVE OPTIMIZATION

We start by formally defining the MOO problem. An MOO is defined by m losses $\ell_i : \mathbb{R}^d \to \mathbb{R}_+$, $i = 1, \dots, m$, or in vector form $\boldsymbol{\ell} : \mathbb{R}^d \to \mathbb{R}^m_+$. We define a partial ordering on the loss space $\mathbb{R}^m_+$ by setting $\boldsymbol{\ell}(\theta_1) \preceq \boldsymbol{\ell}(\theta_2)$ if for all $i \in [m]$, $\ell_i(\theta_1) \leq \ell_i(\theta_2)$. We define $\boldsymbol{\ell}(\theta_1) \prec \boldsymbol{\ell}(\theta_2)$ if $\boldsymbol{\ell}(\theta_1) \preceq \boldsymbol{\ell}(\theta_2)$ and for some $i \in [m], \ell_i(\theta_1) < \ell_i(\theta_2)$. We say that a point $\theta_1 \in \mathbb{R}^d$ *dominates* $\theta_2 \in \mathbb{R}^d$ if $\boldsymbol{\ell}(\theta_1) \prec \boldsymbol{\ell}(\theta_2)$.

If a point θ_1 dominates θ_2, then θ_1 is clearly preferable, as it improves some objectives and is not worse on any other objective. Otherwise, the solutions present a certain trade-off and selecting

between them requires additional information on the user preferences. A point that is not dominated by any other point is called *Pareto optimal*. The set of all Pareto optimal points is called the *Pareto front*. Since many modern machine learning models, e.g. deep neural networks, use non-convex optimization, one cannot expect global optimality. We call a point *local Pareto optimal* if it is Pareto optimal in some open neighborhood of the point.

The most straightforward approach to MOO is linear scalarization, where we define a new single loss $\ell_{\boldsymbol{r}}(\theta) = \sum_i r_i \ell_i(\theta)$ given a vector $\boldsymbol{r} \in \mathbb{R}_+^m$ of weights. One can then apply standard, single-objective optimization algorithms. Linear scalarization has two major concerns. First, it can only reach the convex part of the Pareto front (Boyd et al., 2004, Chapter 4.7), as shown empirically in Figure 1. Second, if one wishes to target a specific ray in loss space, specified by a preference vector, it is not clear which linear weights lead to that desired Pareto optimal point.

In Fliege & Svaiter (2000) the authors propose updating the parameter values along the direction $d(\theta)$ given by

$$(d(\theta), \alpha(\theta)) = \arg\min_{\alpha, d} \alpha + ||d||_2^2, \text{ s.t. } \quad \forall i \, \nabla \ell_i^T d \leq \alpha, \tag{1}$$

which they prove is either a descent direction for all losses or a local Pareto optimal point. They show that following this update directions leads to convergence on the Pareto front. In Désidéri (2012) the author shows similar results for the convex combination of gradients with minimal norm, i.e. $d(\theta) = \arg\min_{d \in conv(\{\nabla \ell_i(\theta)\})} ||d||_2$. In both methods one has no control over which point on the Pareto front the optimization converges to.

To ameliorate this problem, Lin et al. (2019) suggested the Pareto multi-task learning (PMTL) algorithm; they split the loss space into separate cones, based on selected rays, and return a solution in each cone by using a constrained version of Fliege & Svaiter (2000). While this allows the user to get several points on the Pareto front, it scales poorly with the number of cones. Not only does one need to run and store a separate model per cone, we empirically observe that the run-time for each optimization increases with the number of cones due to the increased number of constraints.

Mahapatra & Rajan (2020) presented *Exact Pareto Optimal* (EPO), an MOO optimization method that can converge to a desired ray in loss space. Given a preference ray $\boldsymbol{r}$, EPO search for an exact Pareto optimal solution, i.e. a solution that is (i) Pareto optimal, and; (ii) Lies on the intersect of the Pareto front and the preference vector $\boldsymbol{r}$. The EPO method balances two goals: Finding a descent direction and getting to the desired ray. EPO searches for a point in the convex hull of the gradients, known by Désidéri (2012) to include descent directions, that has maximal angle with a vector d_{bal} which pulls the point to the desired ray. EPO combines gradient descent and controlled ascent enabling it to reach an exact Pareto optimal solution if one exists, or the closest Pareto optimal solution.

3 Related Work

Multitask learning. In multitask learning (MTL) we simultaneously solve several learning problems, while sharing information among tasks (Zhang & Yang, 2017; Ruder, 2017). In some cases, MTL based models outperform their single task counterparts in terms of per task performance and computational efficiency (Standley et al., 2019). MTL approaches map the loss vector into a single loss term using fixed or dynamic weighting scheme. The most frequently used approach is Linear Scalarization, in which each loss term's weight is chosen apriori. A proper set of weight is commonly selected using grid search. Unfortunately, such approach scales poorly with the number of tasks. Recently, MTL methods propose dynamically balance the loss terms using gradient magnitude (Chen et al., 2018), the rate of change in losses (Liu et al., 2019), task uncertainty (Kendall et al., 2018), or learning non-linear loss combinations by implicit differentiation (Navon et al., 2020). However, those methods seek a balanced solution and are not suitable for modeling tasks trade-offs.

Multi-objective optimization. The goal of Multi-Objective Optimization (MOO) is to find Pareto optimal solutions corresponding to different trade-offs between objectives (Ehrgott, 2005). MOO has wide variety of applications in machine learning, spanning Reinforcement Learning (Van Moffaert & Nowé, 2014; Pirotta & Restelli, 2016; Parisi et al., 2014; Pirotta et al., 2015; Parisi et al., 2016; Yang et al., 2019), neural architecture search (Lu et al., 2019; Hsu et al., 2018), fairness (Martínez et al., 2020; Liu & Vicente, 2020), and Bayesian optimization (Shah & Ghahramani, 2016; Hernández-Lobato et al., 2016). Genetic algorithms (GA) are a popular approach for small-scale

multi-objective optimization problems. GAs are designed to maintain a set of solutions during optimization. Therefore, they can be extended in natural ways to maintain solutions for different rays. In this area, leading approaches include NSGA-III (Deb & Jain, 2013) and MOEA/D (Zhang & Li, 2007). However, these gradient free methods scale poorly with the number of parameters and are not suitable for training large scale neural networks. Sener & Koltun (2018) proposed a gradient based MOO algorithm for MTL, based on MDGA (Désidéri, 2012), suitable for training large scale neural networks. Other recent works include (Lin et al., 2019; Mahapatra & Rajan, 2020) detailed in Section 2. Another recent approach that tries to get a more complete view of the Pareto front is Ma et al. (2020). This method extents a given Pareto optimal solution in its local neighborhood. In a concurrent work, Lin et al. (2020) propose extending Lin et al. (2019) for approximating the entire Pareto front, by utilizing hypernetworks. The proposed method is conceptually similar to our approach. Similarly, Dosovitskiy & Djolonga (2019) proposed learning a single model conditioning on the objective weight vector. The method uses feature-wise linear modulation Perez et al. (2017), and dynamically weighted LS loss criterion.

Hypernetworks. Ha et al. (2017) introduced the idea of hypernetworks (HN) inspired by genotype-phenotype relation in the human nature. HN presents an approach of using one network (hypernetwork) to generate weights for second network (target network). In recent years, HN are widely used in various domains such as computer vision (Klocek et al., 2019; Ha et al., 2017), language modeling (Suarez, 2017), sequence decoding (Nachmani & Wolf, 2019), continual learning (Oswald et al., 2020) and hyperparameter optimization (Mackay et al., 2019; Lorraine & Duvenaud, 2018). HN dynamically generates models conditioned on a given input, thus obtains a set of customized models using a single learnable network.

4 PARETO HYPERNETWORKS

Algorithm 1 PHN

while not converged **do**
  $\boldsymbol{r} \sim Dir(\boldsymbol{\alpha})$
  $\phi(\theta, \boldsymbol{r}) = h(\theta, \boldsymbol{r})$
  Sample mini-batch $(x_1, y_1), .., (x_B, y_B)$
  if LS **then**
    $g_\theta \leftarrow \frac{1}{B} \sum_{i,j} r_i \nabla_\theta \ell_i(x_j, y_j, \phi(\theta, \boldsymbol{r}))$
  if EPO **then**
    $\beta = EPO(\phi(\theta, \boldsymbol{r}), \boldsymbol{\ell}, \boldsymbol{r})$
    $g_\theta \leftarrow \frac{1}{B} \sum_{i,j} \beta_i \nabla_\theta \ell_i(x_j, y_j, \phi(\theta, \boldsymbol{r}))$
  $\theta \leftarrow \theta - \eta g_\theta$
return θ

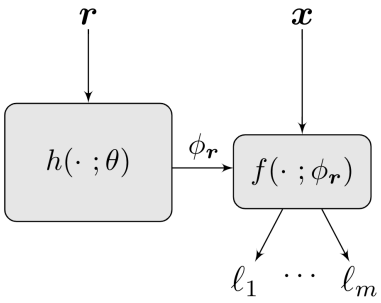


Figure 2: PHN receives an input preference and outputs the corresponding model weights.

We start with describing the basic hypernetworks architecture. Hypernetworks are deep models that emit the weights of another deep model as their output, named the target network. The weights emitted by the hypernetwork depend on its input, and one can view the training of a hypernetwork as training a family of target networks simultaneously, conditioned on the input. Let $h(\cdot\,;\theta)$ denote the hypernetwork with parameters $\theta \in \mathbb{R}^n$, and let f denote the target network whose parameters we denote by ϕ. In our implementation, we parametrize h using a feed-forward network with multiple heads. Each head outputs a different weight tensor of the target network. More precisely, the input is first mapped to higher dimensional space using an MLP, to construct shared features. These features are passed through fully connected layers to create weight matrix per layer in the target network.

Consider an $m-$dimensional MOO problem with an objective vector $\boldsymbol{\ell}$. Let $\boldsymbol{r} = (r_1, ..., r_m) \in S_m$ denote a preference vector representing a desired trade-off over the objectives. Here $S_m = \{\boldsymbol{r} \in \mathbb{R}^m \mid \sum_j r_j = 1, r_j \geq 0\}$ is the $m-$dimensional simplex. Our PHN model acts on the preference vector $\boldsymbol{r}$ to produce the weights $\phi(\theta, \boldsymbol{r}) = h(\boldsymbol{r}, \theta)$ of the target network. The target network is then applied to the training data in a usual manner. Throughout training, we randomly sample $\boldsymbol{r}$ at each iteration from a Dirichlet distribution with parameter $\alpha \in \mathbb{R}^m$. At inference time, given a preference direction $\boldsymbol{r}$, it is provided as input to the HN h, which outputs the corresponding Pareto

optimal solution, $\phi_{\boldsymbol{r}}$, namely, a set of model weights for that given $\boldsymbol{r}$. We train ϕ to be Pareto optimal and corresponds to the preference vector. For that purpose, we explore two alternatives: The first, named PHN-LS, uses *linear scalarization* with the preference vector $\boldsymbol{r}$ as loss weights, i.e the loss for input $\boldsymbol{r}$ is $\sum_i r_i \ell_i$. The second approach, named PHN-EPO, treats the preference $\boldsymbol{r}$ as a ray in the loss space and trains $\phi(\theta, \boldsymbol{r})$ to reach a Pareto optimal point on the inverse ray $\boldsymbol{r}^{-1}$, namely, $r_1 \cdot \ell_1 = \ldots = r_m \cdot \ell_m$. This is done by using the EPO update direction for each output of the Pareto hypernetwork. As the EPO descent direction is a convex combination of the gradients, we use the EPO algorithm to compute this adaptive weighting and backpropagate the reweighted losses. See pseudocode in Alg. 1.

PHN-LS: While linear scalarization has some theoretical limitations, it is a simple and fast approach that tends to works well in practice. One can think of the PHN-LS approach as optimizing the loss $\ell_{LS}(\theta) = \mathbb{E}_{\boldsymbol{r},x,y}[\sum_i r_i \ell_i(x, y, \phi(\theta, \boldsymbol{r}))]$ where we sample a mini-batch and a weighting $\boldsymbol{r}$. We note, as a direct result from the implicit function theorem, the following:

Proposition 1. *Let $\phi(\hat{\boldsymbol{r}})$ be a local Pareto optimal point corresponding to weights $\hat{\boldsymbol{r}}$, i.e. $\sum_{ij} \hat{r}_i \nabla \ell_i(x_j, y_j, \phi) = 0$ and assume the Hessian of $\sum_i \hat{r}_i \ell_i$ has full rank at $\phi(\hat{\boldsymbol{r}})$. There exists a neighborhood U of $\hat{\boldsymbol{r}}$ and a smooth mapping $\phi(\boldsymbol{r})$ such that $\forall \boldsymbol{r} \in U : \sum_{ij} r_i \nabla \ell_i(x_j, y_j, \phi(\boldsymbol{r})) = 0$.*

This shows that, under the full rank assumption, the connection between LS term $\boldsymbol{r}$ and the optimal model weights $\phi(\boldsymbol{r})$ is smooth and can be learnable by universal approximation (Cybenko, 1989).

We also observe that if for all $\boldsymbol{r}$, $\phi(\theta, \boldsymbol{r})$ is a local Pareto optimal point corresponding to $\boldsymbol{r}$, i.e., $\sum_{ij} r_i \nabla \ell_i(x_j, y_j, \phi(\boldsymbol{r})) = 0$, then $\nabla_\theta \ell_{LS}(\theta) = 0$ and this is a stationary point for ℓ_{LS}. However, there could exists stationary points and even local minima such that $\nabla_\theta \ell_{LS}(\theta) = 0$ but in general $\sum_{ij} r_i \nabla \ell_i(x_j, y_j, \phi(\boldsymbol{r})) \neq 0$ and the output of our hypernetwork is not a local Pareto optimal point. While these bad local minimas exists, similarly to standard deep neural networks, we empirically show that PHNs avoid these solutions to achieve good results.

PHN-EPO: The recent EPO algorithm can guarantee convergence to a local Pareto optimal point on a specified ray $\boldsymbol{r}$, addressing the theoretical limitations of LS. This is done by choosing a convex combination of the loss gradients which is a descent direction for all losses, or moves the losses closer to the desired ray. The EPO however, unlike LS, cannot be seen as optimizing a certain objective. Therefore, we cannot consider the PHN-EPO optimization, which uses the EPO descent direction, as optimizing a specific objective. We also note that since the EPO needs to solve a linear programming step at each iteration, the runtime of EPO and PHN-EPO are longer than the LS and PHN-LS counterparts (see Figure 5).

5 EXPERIMENTS

We evaluate Pareto HyperNetworks (PHNs) on a set of diverse multi-objective problems. The experiments show the superiority of PHN over other MOO methods. We will make our code publicly available in order to facilitate further research.

Compared methods: We compare the following approaches: **(1)** Our proposed Pareto HyperNetworks (**PHN-LS** and **PHN-EPO**) algorithm described in Section 4; **(2)** Linear scalarization **(LS)**; **(3)** Pareto MTL (**PMTL**) Lin et al. (2019); **(4)** Exact Pareto optimal (**EPO**) Mahapatra & Rajan (2020); **(5)** Continuous Pareto MTL (**CPMTL**) Ma et al. (2020). All competing methods train one model per preference vector to generate Pareto solutions. As these models optimize on each ray separately, we do not treat them as a baseline but as "*gold standard*" in term of performance. Their runtime, on the other hand, scales linearly with the number of rays.

Genetic algorithms are perhaps the most popular approaches for MOO. Two leading approaches in that area which also use preference information are NSGA-III (Deb et al., 2002) and MOEA/D (Zhang & Li, 2007). Unfortunately, these methods only work on small sized networks (like the toy example in Figure 1, which gave sub-par results even after 100,000 generations), but failed to converge to meaningful solutions in our larger-scale experiments. For that reason, the experiments below do not compare with GAs.

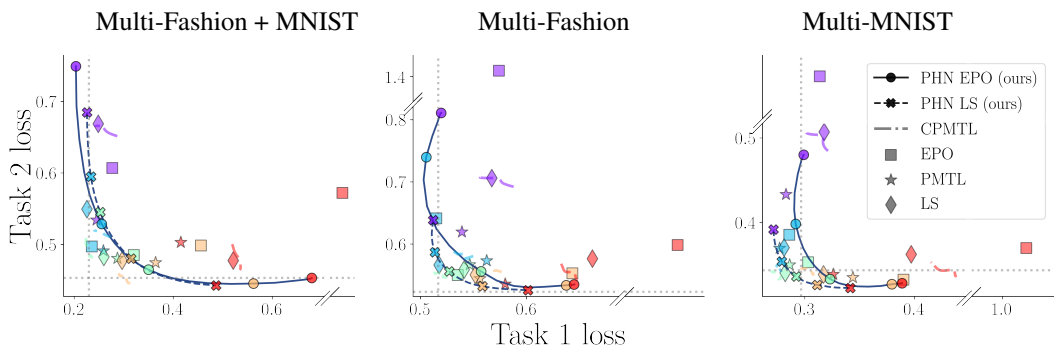


Figure 3: *Multi-MNIST*: Test losses produced by the competing method on three Multi-MNIST datasets. Colors corresponds to different preference vectors, using the same color map as in Figure 1. PHN generates a continuous Pareto front (solid line), while mapping any given preference to its corresponding solution (circles for PHN-EPO and x's for PHN-LS).

Evaluation metrics: A common metric for evaluating sets of MOO solutions is the **Hypervolume** indicator (HV) (Zitzler & Thiele, 1999). It measures the volume in loss space of points dominated by a solution in the evaluated set. As this volume is unbounded, it is restrict to the volume in a rectangle defined by the solutions and a selected reference point. All solutions needs to be bounded by the reference point, therefore we select different reference point for each experiment. see further details in Appendix Section D. In addition, we evaluate **Uniformity**, proposed by Mahapatra & Rajan (2020). This metric is designed to quantify how well the loss vector $\boldsymbol{\ell}(\theta)$ is align with the input ray $\boldsymbol{r}$. Formally, for any point θ in the solution space and reference $\boldsymbol{r}$ we define the non-uniformity $\mu_{\boldsymbol{r}}(\boldsymbol{\ell}(\theta)) = D_{KL}(\hat{\boldsymbol{\ell}} \| \mathbf{1}/m)$, where $\hat{\ell}_j = \frac{r_j \ell_j}{\sum_i r_i \ell_i}$. We define the uniformity as $1 - \mu_{\boldsymbol{r}}$. The main metric, HV, measures the quality of our solutions, while the uniformity measures how well the output model fits the desired preference ray. We stress that the baselines are trained and evaluated for uniformity using the same rays.

Training Strategies: In all experiments, our target network shares the same architecture as the baseline models. For our hypernetwork we use an MLP with 2 hidden layers and linear heads. We set the hidden dimension to 100 for the Multi-MNIST and NYU experiment, and 25 for the Fairness and SARCOS. The Dirichlet parameter α in Alg. 1 is set to 0.2 for all experiments, as we empirically found this value to perform well across datasets and tasks. We provide a more detailed analysis on the selection of the hidden dimension and hyperparameter α in Appendix C.5. Extensive experimental details are provided in Appendix A.

Hyperparameter tuning: For our PHN method we select hyperparameters based on the HV computed on a validation set. Selecting hyperparameters for the baselines is non-trivial as there is no clear criteria that is reasonable in terms of runtime; In order to select hyperparameters based on HV, each approach needs to be trained multiple times on all rays. We therefore select hyperparameters based on a single ray, and apply those for all rays. Our selection criterion is as follow: we collect all models trained using all hyperparameter configurations, and filter out the dominated solutions. Finally, we select the combination of hyperparametrs with the highest uniformity.

5.1 Illustrative Example

We evaluate PHN on an illustrative MOO problem whose Pareto front is known (Fonseca, 1995). Here,

$$\ell_1(\theta) = 1 - \exp\big\{ - \|\theta - \mathbf{1}/\sqrt{d}\|_2^2 \big\}, \quad \ell_2(\theta) = 1 - \exp\big\{ - \|\theta + \mathbf{1}/\sqrt{d}\|_2^2 \big\},$$

with parameters $\theta \in \mathbb{R}^d$ and $d = 100$. This problem has a non-convex Pareto front in $\mathbb{R}^2$. We compared PHN with EPO, PMTL, MGDA and two genetic algorithms (GAs) NSGA-III and MOEA/D. The results are presented in Figure 1. Linear scalarization (LS) fails to generate solutions on the concave part of the Pareto front, as theoretically shown in Boyd et al. (2004). PMTL and EPO

Table 1: Method comparison on three Multi-MNIST datasets.

	Multi-Fashion+MNIST		Multi-Fashion		Multi-MNIST		
	HV ⇑	Unif. ⇑	HV ⇑	Unif. ⇑	HV ⇑	Unif. ⇑	Run-time (min., Tesla V100)
LS	2.70	0.849	2.14	0.835	2.85	0.846	9.0×5=45
CPMTL	2.76	-	2.16	-	2.88	-	10.2×5=51
PMTL	2.67	0.776	2.13	0.192	2.86	0.793	17.0×5=85
EPO	2.67	0.892	2.15	0.906	2.85	0.918	23.6×5 =118
PHN-LS (ours)	2.75	0.894	**2.19**	0.900	**2.90**	0.901	**12**
PHN-EPO (ours)	**2.78**	**0.952**	**2.19**	**0.921**	2.78	**0.920**	27

generate solutions along the entire PF, but the solutions generated by PMTL are only optimized to be in the vicinity of the preference vector $\boldsymbol{r}$. Results for MGDA and MOEA/D algorithms are shown in Appendix C.1. Our approach PHN-EPO generates a continuous cover of the PF in a single run. In addition, PHN returns exact Pareto optimal solutions, mapping an input preference into its corresponding point on the front. We note that similar to the EPO method, PHN-EPO can learn non-convex Pareto fronts. Additional synthetic experiments with non-convex, known Pareto fronts are presented in Appendix C.1.

5.2 Image Classification

We start by evaluating all methods on three MOO benchmark datasets: (1) Multi-MNIST (Sabour et al., 2017); (2) Multi-Fashion, and (3) Multi-Fashion + MNIST. In each dataset, two instances are sampled uniformly in random from the MNIST (LeCun et al., 1998) or Fashion-MNIST (Xiao et al., 2017) datasets. The two images are merged into a new image by placing one on the top-left corner and the other on the bottom-right corner. The sampled images can be shifted up to four pixels on each direction. Each dataset consists of 120,000 training examples and 20,000 test set examples. We allocate 10% of each training set for constructing validation sets. We train 5 models for each baseline, using evenly spaced rays. The results are presented in Figure 3 and Table 1. HV calculated with reference point $(2, 2)$. PHN outperforms all other methods in terms of HV and uniformity, with significant improvement in run time. To better understand the behaviour of PHN while varying the input preference vector, we visualize the per-pixel prediction contribution in Appendix C.3.

5.3 Fairness

Table 2: Comparison on three Fairness datasets. Run-time is evaluated on the Adult dataset.

	Adult		Default		Bank		
	HV ⇑	Unif. ⇑	HV ⇑	Unif. ⇑	HV ⇑	Unif. ⇑	Run-time (min., Tesla V100)
LS	0.628	0.109	0.522	0.069	0.704	0.215	0.9×10=9.0
PMTL	0.614	0.458	0.548	0.471	0.638	0.497	2.7×10=26.6
EPO	0.608	**0.734**	0.537	**0.533**	0.713	**0.881**	1.6×10=15.5
PHN-LS (ours)	**0.658**	0.289	**0.551**	0.108	0.730	0.615	**1.1**
PHN-EPO (ours)	0.648	0.701	0.548	0.359	**0.748**	0.821	1.8

Fairness has become a popular topic in machine learning in recent years (Mehrabi et al., 2019), and numerous approaches have been proposed for modeling and incorporating fairness into ML systems (Dwork et al., 2017; Kamishima et al., 2012; Zafar et al., 2019). Here, we adopt the approach of Bechavod & Ligett (2017). They proposed a 3-dimensional optimization problem, with classification objective, and two fairness objectives: False Positive (FP) fairness, and False Negative (FN) fairness. We compare PHN with EPO, PMTL and LS on three commonly used fairness datasets: Adult (Dua & Graff, 2017), Default (Yeh & Lien, 2009) and Bank (Moro et al., 2014).

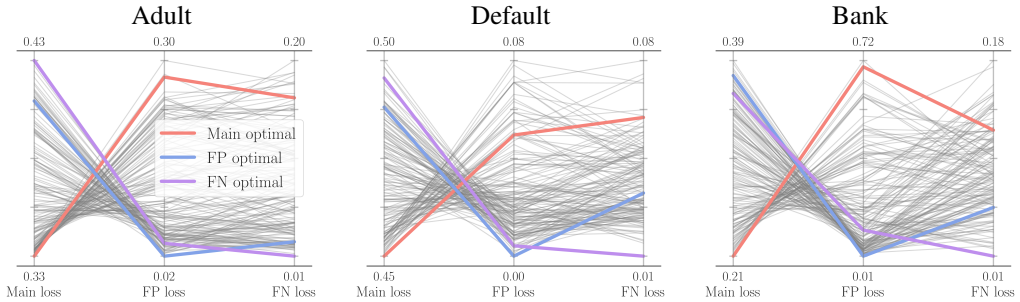


Figure 4: *Accuracy-Fairness trade-off*: PHN-EPO losses on three fairness dataset.

Each dataset is splitted into train/validation/test of sizes 70%/10%/20% respectively. All baselines are trained and evaluated on 10 preference rays. We present the results in Table 2. HV calculated with reference point $(1, 1, 1)$. PHN achieves best HV performance across all datasets, with reduced training time compared to the baselines. EPO performs best in terms of uniformity, however, recall it is evaluated on its training rays. In addition, we visualize the accuracy-fairness trade-off achieved by PHN in Figure 4. We also tested GAs on this task but the runtime was extremely high (two days, compared with a run time of minutes for gradient-based methods) with poor performance.

5.4 Pixel-wise classification and Regression

Image segmentation and depth estimation are key topics in image processing and computer vision with various applications (Minaee et al., 2020; Bhoi, 2019). We compare PHN to other methods on NYUv2 dataset (Silberman et al., 2012). NYUv2 is a challenging indoor scene dataset consists of 1449 RGBD images and dense per pixel labeling. We used this dataset as a multitask learning benchmark for semantic segmentation and depth estimation tasks. We split the dataset into train/validation/test of sizes 70%/10%/20% respectively, and use an ENet-based architecture for the target network (Paszke et al., 2016). Each baseline method is trained five times using evenly spaced rays. The results are presented in Table 3. HV is calculated with reference point $(3, 3)$. PHN-EPO achieves best HV and uniformity, while also improving upon the baselines in terms of runtime.

Table 3: Comparison on NYUv2.

	NYUv2		
	HV ⇑	Unif. ⇑	Run-time (hours, Tesla V100)
LS	3.550	0.666	$0.58 \times 5 = 2.92$
PMTL	3.554	0.679	$0.96 \times 5 = 4.79$
CPMTL	3.570	-	$0.71 \times 5 = 3.55$
EPO	3.266	0.728	$1.02 \times 5 = 5.11$
PHN-LS (ours)	3.546	0.798	**0.67**
PHN-EPO (ours)	**3.589**	**0.820**	1.04

Table 4: Comparison on SARCOS dataset.

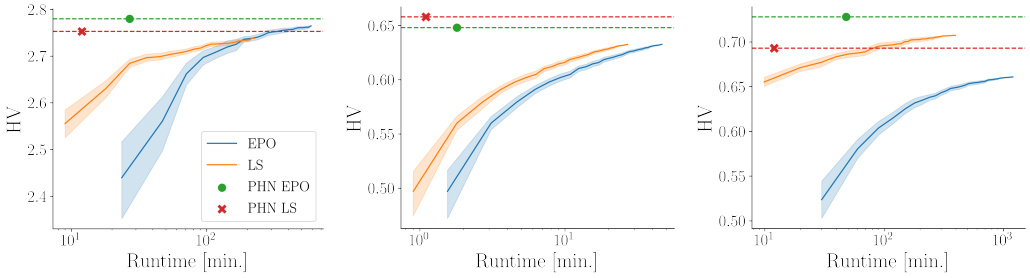
	SARCOS		
	HV ⇑	Unif. ⇑	Run-time (hours, Tesla V100)
LS	0.688	-0.008	$0.17 \times 20 = 3.3$
EPO	0.637	**0.548**	$0.5 \times 20 = 10$
PHN-LS (ours)	0.693	0.020	0.2
PHN-EPO (ours)	**0.728**	0.258	0.8

5.5 Multitask Regression

When learning with many objectives using preference-specific models, generating a good coverage of the solution space becomes too expensive in computation and storage space, as the number of rays needed grows with the number of dimensions. As a result, PHN provides a major improvement in training time and solution space coverage (measured by HV). To evaluate PHN in such settings, we use the SARCOS dataset (Vijayakumar), a commonly used dataset for multitask regression (Zhang & Yang, 2017). SARCOS presents an inverse dynamics problem with 7 regression tasks and 21 explanatory variables. We omit PMTL from this experiment because its training time is significantly longer compared to other baselines, with relatively low performance. For LS and EPO, we train 20 preference-specific models. The results are presented in Table 4. HV calculated with reference point

$(1, ..., 1)$. As before, PHN performs best in terms of HV, while significantly reducing the overall run-time.

6 THE QUALITY-RUNTIME TRADEOFF



(a) Multi Fashion + MNIST (b) Adult (c) SARCOS

Figure 5: Hypervolume vs runtime (min.) comparing PHN with preference-specific LS and EPO. For LS and EPO, each value was computed by evaluating with a subset of models. PHN achieves higher HV in significantly less run-time (x-axis in log scale). Shaded area marks the variance across selected subset of rays.

PHN learns the entire front in a single model, but the competing methods need to train multiple models to cover the pareto front. As a result, these methods have clear trade off between performance and runtime. To better understand this trade-off, Figure 5 shows the HV and runtime (in log-scale) of LS and EPO when training various number of models. For Fashion-MNIST, we trained 25 models, and at inference time, we selected subsets of various sizes (1 ray, 2 rays, etc) and computed their HV. The shaded area in Figure 5 reflects the variance over different selections of ray subsets. We similarly trained 30 EPO/LS models for Adult and 40 for SARCOS. PHN-EPO achieves superior or comparable hypervolume while being an order of magnitude faster. PHN-LS is even faster, but this may come at the expense of lower hypervolume.

To further understand the source for PHN performance gain, we evaluate the HV on the same rays as the competing methods. In many cases, PHN outperforms models trained on specific rays, when evaluated on those exact rays. We attribute the performance gain to the inductive bias effect that PHN induces on a model by sharing weights across rays. Results are in Section C.4 of the Appendix.

7 CONCLUSION

We proposed a novel MOO approach: Learning the Pareto front using a single unified model that can be applied to a specific objective preference at inference time. We name this learning setup Pareto front learning (PFL). We implement our idea using hypernetwork and evaluated on a large and diverse set of tasks and datasets. Our experiments show significant gains in both runtime and performance on all datasets. This novel approach provides the user with the flexibility of selecting the operating point at inference time, and open new possibilities where user preferences can be tuned and changed at deployment.

REFERENCES

Yahav Bechavod and Katrina Ligett. Penalizing unfairness in binary classification. *arXiv preprint arXiv:1707.00044*, 2017.

Amlaan Bhoi. Monocular depth estimation: A survey. *ArXiv*, abs/1901.09402, 2019.

Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*, pp. 794–803. PMLR, 2018.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Math. Control. Signals Syst.*, pp. 303–314, 1989.

Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. *IEEE transactions on evolutionary computation*, 18(4):577–601, 2013.

Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.*, pp. 182–197, 2002.

Alexey Dosovitskiy and Josip Djolonga. You only train once: Loss-conditional training of deep networks. In *International Conference on Learning Representations*, 2019.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Cynthia Dwork, Nicole Immorlica, Adam Tauman Kalai, and Max Leiserson. Decoupled classifiers for fair and efficient machine learning. *arXiv preprint arXiv:1707.06613*, 2017.

Jean-Antoine Désidéri. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathematique*, 350:313–318, 2012.

Matthias Ehrgott. *Multicriteria optimization*, volume 491. Springer Science & Business Media, 2005.

Yu G Evtushenko and Mikhail Anatol'evich Posypkin. Nonuniform covering method as applied to multicriteria optimization problems with guaranteed accuracy. *Computational Mathematics and Mathematical Physics*, 53(2):144–157, 2013.

Jörg Fliege and Benar Fux Svaiter. Steepest descent methods for multicriteria optimization. *Math. Methods Oper. Res.*, 51(3):479–494, 2000.

Carlos Manuel Mira da Fonseca. *Multiobjective genetic algorithms with application to control engineering problems.* PhD thesis, University of Sheffield, 1995.

David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. *ArXiv*, abs/1609.09106, 2017.

Daniel Hernández-Lobato, Jose Hernandez-Lobato, Amar Shah, and Ryan Adams. Predictive entropy search for multi-objective bayesian optimization. In *International Conference on Machine Learning*, pp. 1492–1501, 2016.

Chi-Hung Hsu, Shu-Huan Chang, Jhao-Hong Liang, Hsin-Ping Chou, Chun-Hao Liu, Shih-Chieh Chang, Jia-Yu Pan, Yu-Ting Chen, Wei Wei, and Da-Cheng Juan. Monas: Multi-objective neural architecture search using reinforcement learning. *arXiv preprint arXiv:1806.10332*, 2018.

X. Huang and Serge J. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 1510–1519, 2017.

Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. Fairness-aware classifier with prejudice remover regularizer. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 35–50. Springer, 2012.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7482–7491, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

Sylwester Klocek, Lukasz Maziarka, Maciej Wolczyk, J. Tabor, J. Nowak, and M. Smieja. Hypernetwork functional image representation. In *ICANN*, 2019.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

X. Lin, Zhiyuan Yang, Q. Zhang, and S. Kwong. Controllable pareto multi-task learning. *ArXiv*, abs/2010.06313, 2020.

Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. Pareto multi-task learning. In *Advances in Neural Information Processing Systems*, pp. 12060–12070, 2019.

Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1871–1880, 2019.

Suyun Liu and Luis Nunes Vicente. Accuracy and fairness trade-offs in machine learning: A stochastic multi-objective approach. *arXiv preprint arXiv:2008.01132*, 2020.

Jonathan Lorraine and David Duvenaud. Stochastic hyperparameter optimization through hypernetworks. *arXiv preprint arXiv:1802.09419*, 2018.

Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, and Wolfgang Banzhaf. Nsga-net: neural architecture search using multi-objective genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 419–427, 2019.

Shangzhen Luan, Chen Chen, Baochang Zhang, Jungong Han, and Jianzhuang Liu. Gabor convolutional networks. *IEEE Transactions on Image Processing*, 27(9):4357–4366, 2018.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30*, pp. 4765–4774. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf`.

Pingchuan Ma, Tao Du, and Wojciech Matusik. Efficient continuous pareto exploration in multi-task learning. *arXiv preprint arXiv:2006.16434*, 2020.

Matthew Mackay, Paul Vicol, Jonathan Lorraine, David Duvenaud, and Roger Grosse. Self-tuning networks: Bilevel optimization of hyperparameters using structured best-response functions. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=r1eEG20qKQ`.

D. Mahapatra and V. Rajan. Multi-task learning with user preferences: Gradient descent with controlled ascent in pareto optimization. 2020.

N. Martínez, Martín Bertrán, and G. Sapiro. Minimax pareto fairness: A multi objective perspective. 2020.

Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and A. Galstyan. A survey on bias and fairness in machine learning. *ArXiv*, abs/1908.09635, 2019.

Shervin Minaee, Yuri Boykov, F. Porikli, Antonio J. Plaza, N. Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *ArXiv*, abs/2001.05566, 2020.

Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.

Eliya Nachmani and L. Wolf. Hyper-graph-network decoders for block codes. In *NeurIPS*, 2019.

Aviv Navon, Idan Achituve, Haggai Maron, Gal Chechik, and Ethan Fetaya. Auxiliary learning by implicit differentiation. *arXiv preprint arXiv:2007.02693*, 2020.

J. Oswald, C. Henning, J. Sacramento, and Benjamin F. Grewe. Continual learning with hypernetworks. *ArXiv*, abs/1906.00695, 2020.

Simone Parisi, Matteo Pirotta, Nicola Smacchia, Luca Bascetta, and Marcello Restelli. Policy gradient approaches for multi-objective sequential decision making. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pp. 2323–2330. IEEE, 2014.

Simone Parisi, Matteo Pirotta, and Marcello Restelli. Multi-objective reinforcement learning through continuous pareto manifold approximation. *Journal of Artificial Intelligence Research*, 57:187–227, 2016.

Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and E. Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *ArXiv*, abs/1606.02147, 2016.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. *arXiv preprint arXiv:1709.07871*, 2017.

Matteo Pirotta and Marcello Restelli. Inverse reinforcement learning through policy gradient minimization. In *AAAI Conference on Artificial Intelligence*, pp. 1993–1999. AAAI Press, 2016.

Matteo Pirotta, Simone Parisi, and Marcello Restelli. Multi-objective reinforcement learning with continuous pareto frontier approximation. In *29th AAAI Conference on Artificial Intelligence, AAAI 2015 and the 27th Innovative Applications of Artificial Intelligence Conference, IAAI 2015*, pp. 2928–2934. AAAI Press, 2015.

Sebastian Ruder. An overview of multi-task learning in deep neural networks, 2017.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in neural information processing systems*, pp. 3856–3866, 2017.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, pp. 527–538, 2018.

Amar Shah and Zoubin Ghahramani. Pareto frontier learning with expensive correlated objectives. In *International Conference on Machine Learning*, pp. 1919–1927, 2016.

Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European conference on computer vision*, pp. 746–760. Springer, 2012.

Trevor Standley, Amir R Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? *arXiv preprint arXiv:1905.07553*, 2019.

J. Suarez. Character-level language modeling with recurrent highway hypernetworks. 2017.

Kristof Van Moffaert and Ann Nowé. Multi-objective reinforcement learning using sets of pareto dominating policies. *The Journal of Machine Learning Research*, 15(1):3483–3512, 2014.

Sethu Vijayakumar. The sarcos dataset, 2000. *URL http://www. gaussianprocess. org/gpml/data*.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Runzhe Yang, Xingyuan Sun, and Karthik Narasimhan. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. In *Advances in Neural Information Processing Systems*, pp. 14636–14647, 2019.

I-Cheng Yeh and Che-hui Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 36(2): 2473–2480, 2009.

Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez-Rodriguez, and Krishna P Gummadi. Fairness constraints: A flexible approach for fair classification. *J. Mach. Learn. Res.*, 20(75):1–42, 2019.

Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731, 2007.

Yu Zhang and Qiang Yang. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*, 2017.

Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary Computation*, 3(4):257–271, 1999.

Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, 8(2):173–195, 2000.

Appendix: Learning the Pareto Front with Hypernetworks

A EXPERIMENTAL DETAILS

Pixelwise Classification and Regression: For NYUv2 we performed a hyperparameters search over the number of epochs and learning rates $\{1e-3, 5e-4, 1e-4\}$. We train each method for 150 epochs with Adam (Kingma & Ba, 2015) optimizer and select an optimal set of learning rate and number of epochs according to Sec. 5. We evaluate PHN over 25 rays.

Image classification: For the Multi-MNIST experiments we use a LeNet-based target network (LeCun et al., 1998) with two task-specific heads. We train all methods using Adam optimizer with learning rate $1e-4$ for 150 epochs and batch size 256. We then choose the best number of epochs for each method using the validation set. We evaluate PHN over 25 rays.

Fairness: We use a $3-$layer feed-forward target network with hidden dimensions $[40, 20]$. We use learnable embeddings for all categorical variables. We train all methods for 35 epochs using Adam optimizer with learning rates $\{1e-3, 5e-4, 1e-4\}$. We choose the best hyperparameter configuration using the validation set. Further details on the datasets are provided in Table 5. We evaluate PHN over a sample of 150 rays.

Multitask regression: For the SARCOS experiment, we use a $4-$layer feed-forward target network with 256 hidden dimension. All categorical features are passed through embedding layers. We train all methods for 1000 epochs using Adam optimizer and learning rates $\{1e-3, 5e-4, 1e-4\}$. We choose the best hyperparameter configuration base on the validation set. We use 40,036 training examples, 4,448 validation examples, and 4,449 test examples. We evaluate PHN over a sample of 100 rays.

Table 5: Fairness datasets.

	Adult	Default	Bank
Train	32559	21600	29655
Validation	3618	2400	3295
Test	9045	6000	8238

B SCALABILITY

The size of the HN grows linearly with the target network size. This somewhat limits the applicability of PHN to large-sized target networks. However, our approach can be easily extended for allowing the training of large NNs. First, one can learn only part of the target network parameters using HN. For instance, lower layers in vision applications tend to be highly conserved across classes and often even across tasks (Luan et al., 2018). Alternative way is to learn preference specific normalization layers (Huang & Belongie, 2017; Perez et al., 2017) as in Dosovitskiy & Djolonga (2019). In our experiments, we pass the preference vector through a small MLP to generate shared hidden features, which are used for constructing all target weights. However, one can use different parts of the shared features to generate different weight tensors, hence reduce the number of HN parameters. Similarly, one can reduce the dimension of that shared preference representation to achieve a similar effect. We investigate the effect of the latter in Appendix C.5.

C ADDITIONAL EXPERIMENTS

C.1 KNOWN PARETO FRONT

We provide two additional experiments over popular MOO benchmarks whose Pareto front is known. In addition in Figure 6, we provide additional results for MGDA (Sener & Koltun, 2018) and MOEA/D (Zhang & Li, 2007) on the illustrative example from Section 5.1.

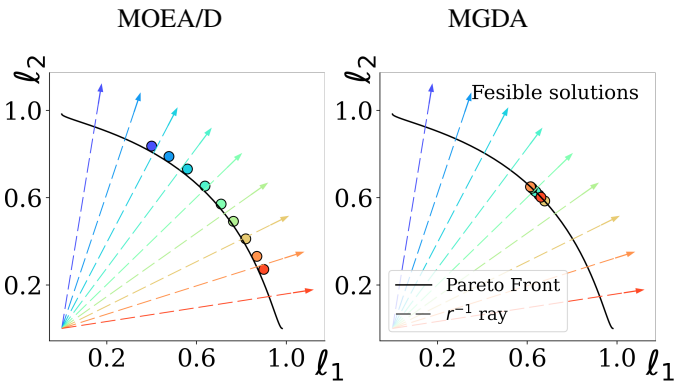


Figure 6: *Illustrative example using the popular task of Fonseca (1995)*: The genetic algorithm MOEA/D fails to reach the front after $100,000$ generations. MGDA obtain Pareto optimal solutions, but is not suitable for generating preference-specific solutions and provides a minimal coverage of the PF.

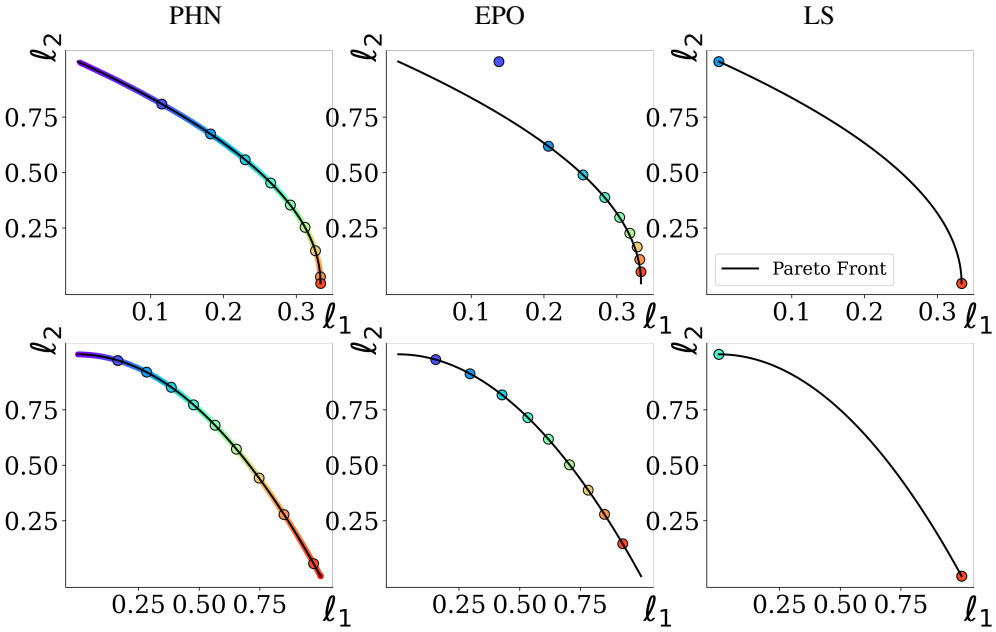


Figure 7: *Known Pareto front*: **Top:** evaluation on Problem 2 from Evtushenko & Posypkin (2013). **Bottom:** evaluation on ZDT2 (Zitzler et al., 2000). PHN converges to the entire PF, mapping a ray to its corresponding solution on the front. The circles in the PHN plots are solution corresponds to the baseline rays.

We evaluate PHN on two MOO problems with known, non-convex Pareto front: ZDT2 (Zitzler et al., 2000) and problem two from Evtushenko & Posypkin (2013), henceforth Problem2. In both problems, the density of solutions across the Pareto-optimal region is non-uniform.

For a solution $\theta = (\theta_1, \theta_2) \in [0, 1]^2$, Problem2 is given by,

$$\ell_1(\theta) = ((\theta_1 - 1)\theta_2^2 + 1)/3, \quad \ell_2(\theta) = \theta_1,$$

and ZDT2 is given by,

$$\ell_1(\theta) = \theta_1, \quad \ell_2(\theta) = 1 - \left(\frac{\theta_1}{1 + 9\theta_2}\right)^2.$$

The results are presented in Figure 7. PHN-EPO converges to the entire PF using a single HN model. Interestingly, EPO fails to converge to the PF for all rays in ZDT2, in contrast with PHN-EPO.

C.2 GENERALIZATION TO UNSEEN RAYS

When the number of objectives is large, it is likely that the rays encountered in test time are "far" from rays encountered during training, because of the curse of dimensionality. Although we evaluated PHN on unseen rays[1] through all experiments, here we would like to better examine PHN's generalization performance in that scenario. we conducted an additional, controlled experiment on SARCOS dataset with 7 tasks. During training, we sample rays from a predefined grid. During inference, we measure HV and Uniformity on two sets of rays: (i) Randomly sampled train rays (ii) Randomly sampled test rays from a shifted grid, so they are most distant from the train grid. In both cases, test samples are different from training samples. We find that testing with unseen test rays only causes a minor decrease in both HV and Uniformity compared with train rays. The insignificant difference in performance in terms of HV is 0.7337 for the train rays vs. 0.7336, and Uniformaty 0.163 vs. 0.158. This experiment indicates that PHN can generalize well to unseen rays, and can be applied to problems with many objectives. We note that the HV results presented here are higher than the ones reported in Section 5.5. We attribute this inconsistency to change of preference vectors distribution used during training. We provide analysis on the robustness to the choice of the Dirichlet distribution parameter α in Appendix C.5.

C.3 MNIST PHN INTERPRETATION

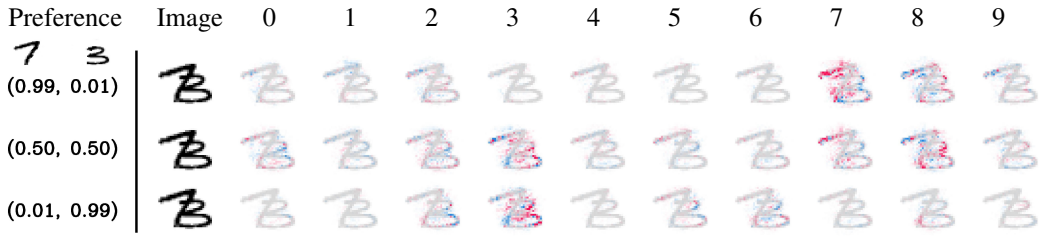


Figure 8: *PHN interpretation*: We observe a sample from Multi-MNIST dataset where two images of 3 and 7 were overlayed on top of each other. We show the effect that conditioning on different preference rays has on the decision making of the model. Red pixels denote increased contribution to the prediction. See text for details.

In order to better understand the behaviour of PHN while varying the input preference vector, we visualize the per-pixel prediction contribution in Figure 8. We use the SHAP framework (Lundberg & Lee, 2017) for the visualization. Red pixels increase the probability for a given class, while blue ones decrease this probability. The top left and bottom right digits are labeled as $\{7, 3\}$ respectively. In the first row (ray: $(0.99, 0.01)$), focused on task 1 (with label 7), most of the red pixels come from the top left digit in column 7, as expected. Similarly, in the third row (ray: $(0.01, 0.99)$) most of the red pixels are focused on the bottom right digit in column 3.

C.4 SAME-RAY EVALUATION

In Section 5 of the main text we evaluate PHN on multiple evenly spaced rays. For completeness, we report here the HV and Uniformity achieved by PHN, while evaluated on the same rays used for training and evaluating the baseline methods. The results are provided in Table 6. Surprisingly, in many cases PHN achieves better performance. We attribute this to the inductive bias effect of sharing weights across all preference rays.

Table 6: Evaluation of PHN on baselines' ray.

	Fashion+MNIST		Multi-Fashion		Multi-MNIST		Adult		Default		Bank		NYUv2		SARCOS	
	HV	Unif.	HV	Unif.	HV	Unif.	HV	Unif.	HV	Unif.	HV	Unif.	HV	Unif.	HV	Unif.
PHN-LS	2.740	0.852	2.190	0.838	2.900	0.840	0.624	0.282	0.538	0.106	0.695	0.603	3.494	0.683	0.693	-0.051
PHN-EPO	2.760	0.908	2.180	0.860	2.760	0.857	0.631	0.742	0.531	0.395	0.717	0.833	3.565	0.701	0.728	0.190

[1] With $\sim$one probability.

C.5 DESIGN CHOICE ANALYSIS

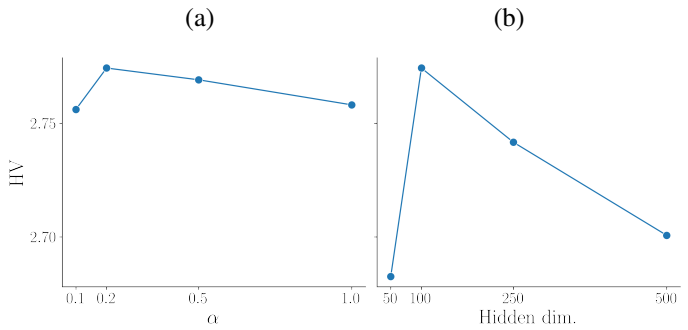


Figure 9: *Hyperparameters analysis*: Performance analysis on different architecture designs and sampling choices.

In Section 5 we have used fixed choices of hypernetworks' hidden dimension and the Dirichlet parameter α (see Alg. 1). Here we examine the effect of different choices for these hyperparameters. We use the Multi-Fashion + MNIST dataset, and the PHN-EPO variant of our method. We first fix the hidden dimension to 100, and vary α from 0.1 to 1 (uniform sampling). The results are presented in Figure 9(a). It appears that at least for this dataset, PHN is quite robust to changes in α. Next, we set $\alpha = 0.2$ and examine the effect of changing the dimension of the hidden layers in our hypernetwork. We present the results in Figure 9(b). Here, the effect on the HV is stronger, with best HV achieved using hidden dimension of 100.

C.6 ADDITIONAL RUNTIME ANALYSIS

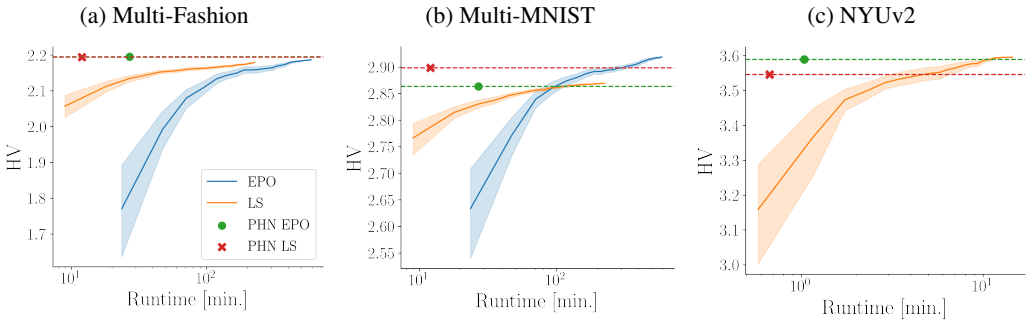


Figure 10: Runtime analysis for NYUv2, and two additional Multi-MNIST dataset.

In this section we extend the runtime analysis from Section 6. In Figure 10 we provide additional runtime analysis on the Multi-MNIST, Multi-Fashion, and NYUv2 datasets. For NYUv2 we omit the EPO method, since it was outperformed by LS in both HV and runtime.

D HYPERVOLUME

The hypervolume metric is defined as follows: Given a set of points $S \subset \mathbb{R}^n$ and a reference point $\boldsymbol{\rho} \in \mathbb{R}^n_+$, the hypervolume of S is measured by the region of non-dominated points bounded above by $\boldsymbol{\rho}$.

$$HV(S) = VOL\big(\{q \in \mathbb{R}^n_+ \mid \exists p \in S : (p \preceq q) \wedge (q \preceq \rho)\}\big),$$

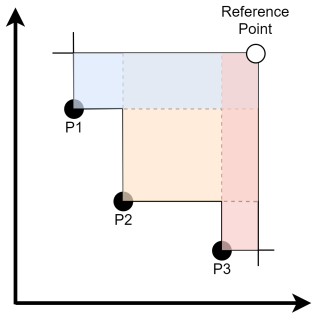


Figure 11: *Hypervolume illustration*: The colored area represents the hypervolume obtained by Pareto set $\{P_1, P_2, P_3\}$ with respect to the reference point.

where VOL is the euclidean volume. We can interpret hypervolume as the measure of the union of the boxes created by the non-dominated points.

$$HV(S) = VOL\left(\bigcup_{\substack{p \in S \\ p \preceq \boldsymbol{\rho}}} \prod_{i=1}^{n} [p_i, \rho_i]\right)$$

As a measure of quality for MOO solutions, it takes into account both the quality of individual solutions, which is the volume they dominate, and also the diversity, measured in the overlap between dominated regions. See example in Fig. 11.