Journal of Science and Technology – Data Science and Artificial Intelligence
Volume 36, Issue 1, October 2026, 060–071

Real-Time Risk-Aversion-Adaptive Portfolio Selection by Learning a Hypernetwork

Khang Truong Tuan[1], Thang Tran Ngoc[1], Anh Dao Tuan[2,*]

[1]*Faculty of Mathematics and Informatics, Hanoi University of Science and Technology, Ha Noi, Vietnam*
[2]*KTH Royal Institute of Technology, Brinellvägen 8, SE-114 28 Stockholm, Sweden*
[*]*Corresponding author email: tadao@kth.se*

Abstract

Solving the Markowitz portfolio problem independently at each risk-aversion level requires one quadratic program per query. We study a learned alternative in which a hypernetwork maps a normalized preference vector, interpreted as a risk-aversion level, to a long-only portfolio in one forward pass. A softmax output layer enforces the budget and no-short-selling constraints, and training minimizes a Chebyshev scalarization without using a pre-solved frontier as a training target. The architecture, scalarizations and guarantees for the scalarized problem follow controllable Pareto front learning. Our main contribution is an evaluation against an exact oracle. For the long-only mean–variance problem, the critical-line algorithm yields the exact Chebyshev solution for every preference, enabling direct measurement and localization of the approximation error. On a ten-industry synthetic instance and a fifteen-asset VN30 instance, the mean Euclidean weight errors are 0.0083 and 0.0088 over ten training seeds. A query takes 44 microseconds on a single central processing unit (CPU) core for a fixed instance. As the number of assets increases from five to two hundred, the mean error grows from 0.0041 to 0.0734. The largest errors occur where the highest expected returns are nearly tied, and the error concentrates at active-set changes, with a median per-seed ratio of 2.45 on VN30. Supervised training on exact targets reduces the error four- to six-fold. Changing the market estimates requires retraining or fine-tuning. The exact solver computes the entire frontier in 0.22 to 3.35 ms, so the learned map offers no computational advantage on this problem.

Keywords: Chebyshev scalarization, efficient frontier, hypernetwork, mean–variance portfolio selection, pareto front learning.

1. Introduction

Mean–variance portfolio selection gives a family of efficient portfolios indexed by risk aversion [1, 2]. Serving a preference generally requires solving an optimization problem [3]. A robo-advisor must update its portfolio when the client's estimated risk aversion changes [4], and each movement of an interactive risk slider issues a query. These queries motivate a map from preferences to portfolios for fixed market estimates.

Amortized optimization [5] learns a map from problem parameters to solutions for subsequent evaluation by a forward pass. For multiple objectives, Pareto front learning conditions a hypernetwork [6] on a preference vector [7, 8]. COSMOS uses linear scalarization and a cosine-similarity penalty [9]. Controllable Pareto front learning (CPFL) with completed scalarization functions [10] and its transformer variant [11] provide the framework instantiated here. Their guarantees concern scalarized problems, not non-convex network training.

Other learned-frontier methods include neural approximations with error bounds for convex vector optimization [12], a sequence-to-sequence surrogate trained on solved instances [13], and deep solvers for mean–variance and mean–CVaR frontiers [14]. Our construction combines preference indexing, exact feasibility and training without solved-frontier targets. None of these properties is new in isolation. The preference-conditioned architecture, scalarizations, training loss and scalarized-problem theory are inherited [7, 10, 11]. The known softmax simplex layer [14] is adapted with exact normalization anchors and a risk-aversion interpretation that accounts for their scales.

Our contribution is an evaluation against Markowitz's exact critical line [2, 15, 16]. It yields the exact Chebyshev solution for any preference (Section 3.5), enabling exact error measurement and comparison with the strongest deterministic method, against which the learned map has no computational advantage (Section 5.10). Convexity and completeness justify the scalarized-problem guarantees (Lemma 2), while Remark 3 states the limitations for network training. Our contributions are computational and empirical:

1) An instantiation of CPFL for long-only mean–variance selection in which the output layer enforces the budget and no-short-selling constraints and the two normalization anchors are exact, one of them in closed form (Sections 2–3).

p-ISSN 3093-3331
e-ISSN 3093-3323
https://doi.org/10.51316/jst.194.dsai.2026.36.1.7
Received: Aug 21, 2026; Revised: Sep 25, 2026;
Accepted: Oct 3, 2026; Online: Oct 15, 2026

2) An exact oracle for the Chebyshev-scalarized problem, read off the critical line (Lemma 2, Section 3.5).
3) An evaluation on a scaling family of five to two hundred assets redrawn at every run, a synthetic ten-industry instance and a real fifteen-asset VN30 instance, with ten independent training runs each, against four non-learning and four learning-based alternatives (Section 5).
4) A diagnosis of where the learned map fails: at near-ties among the highest expected returns, which make the return-maximizing end numerically sensitive, and at changes of the active set, where the error concentrates irrespective of the training budget (Sections 5.4 and 5.8).
5) A cost comparison with the exact parametric solver, which the learned map does not outperform on this problem, and a measurement of the cost of updating the map when (μ,Σ) is re-estimated (Sections 5.10 and 5.11).

2. Problem Formulation

2.1. Feasible Set and Objectives

Let n be the number of risky assets, $\mu \in \mathbb{R}^n$ the vector of expected returns and $\Sigma \in \mathbb{R}^{n\times n}$ the covariance matrix of returns. Portfolios are long-only and fully invested:

$$X \;=\; \Delta_n \;=\; \big\{\mathbf{x}\in\mathbb{R}^n : \mathbf{1}^\top\mathbf{x}=1,\; \mathbf{x}\geq\mathbf{0}\big\} \tag{1}$$

Throughout we assume **(A1)** $\Sigma = \Sigma^\top \succ 0$, and **(A2)** $\arg\max_i \mu_i = \{i^*\}$ is a singleton. The bi-objective problem is:

$$\min_{\mathbf{x}\in X} F(\mathbf{x}) = \big(f_1(\mathbf{x}), f_2(\mathbf{x})\big) = \big(-\mu^\top\mathbf{x},\; \mathbf{x}^\top\Sigma\mathbf{x}\big) \tag{2}$$

Here f_1 is linear and, by (A1), f_2 is strictly convex. A point $\mathbf{x}^a \in X$ *dominates* $\mathbf{x}^b$ if $F(\mathbf{x}^a)\leq F(\mathbf{x}^b)$ componentwise and $F(\mathbf{x}^a)\neq F(\mathbf{x}^b)$; $\mathbf{x}$ is *Pareto optimal* if no point of X dominates it. We denote the Pareto set by $\mathscr{P}$; its image $F(\mathscr{P})$ is the efficient frontier.

2.2. Ideal and Nadir Points

Let $\mathbf{x}^{\max\mu}$ and $\mathbf{x}^{\min\sigma}$ minimize f_1 and f_2 over X. Since f_1 is linear and X is the convex hull of the unit vectors, (A2) gives $\mathbf{x}^{\max\mu} = \mathbf{e}_{i^*}$: the return-maximizing portfolio is the simplex vertex $\mathbf{e}_{i^*}$ and is therefore available in closed form. By (A1), $\mathbf{x}^{\min\sigma}$ is the unique solution of a strictly convex quadratic program. We assume $\mathbf{x}^{\max\mu} \neq \mathbf{x}^{\min\sigma}$; otherwise $\mathscr{P} = \{\mathbf{e}_{i^*}\}$ and the problem is trivial. The ideal and nadir points are:

$$\begin{aligned}\mathbf{z}^{\mathrm{id}} &= \big(f_1(\mathbf{x}^{\max\mu}),\, f_2(\mathbf{x}^{\min\sigma})\big)\\ \mathbf{z}^{\mathrm{nad}} &= \big(f_1(\mathbf{x}^{\min\sigma}),\, f_2(\mathbf{x}^{\max\mu})\big)\end{aligned} \tag{3}$$

These points satisfy $\mathbf{z}^{\mathrm{id}} < \mathbf{z}^{\mathrm{nad}}$ componentwise because both minimizers are unique. For two objectives $\mathbf{z}^{\mathrm{nad}}$ is the exact nadir point, $z_i^{\mathrm{nad}} = \max_{\mathbf{x}\in\mathscr{P}} f_i(\mathbf{x})$ (Section 2.3); for three or more objectives the analogous estimate can fail [17]. Both anchors are computed exactly throughout; Section 5.4 shows that a generic numerical solver is not a safe substitute.

2.3. Normalization and Scalarization

The objectives have different scales. On the instances below, f_1 is of order 10^{-1} and f_2 of order 10^{-2}. All scalarizations use the following normalized objectives:

$$\hat{f}_i = \frac{f_i - z_i^{\mathrm{id}}}{\mathrm{sc}_i}, \quad \mathrm{sc}_i = \max\big(z_i^{\mathrm{nad}} - z_i^{\mathrm{id}},\, 10^{-8}\big) \tag{4}$$

The bound 10^{-8} is an implementation safeguard against numerical degeneracy.

For the mathematical analysis, $\mathbf{z}^{\mathrm{id}} < \mathbf{z}^{\mathrm{nad}}$, so $\mathrm{sc}_i = z_i^{\mathrm{nad}} - z_i^{\mathrm{id}}$. By construction $\hat{f}_i \geq 0$ on X. Moreover $\hat{f}_i \leq 1$ on $\mathscr{P}$: if $f_1(\mathbf{x}) > z_1^{\mathrm{nad}} = f_1(\mathbf{x}^{\min\sigma})$ for some $\mathbf{x}\in\mathscr{P}$, then $\mathbf{x}^{\min\sigma}$, which minimizes f_2, would dominate $\mathbf{x}$; the case $i=2$ is symmetric. Hence $\hat{F}(\mathscr{P}) \subset [0,1]^2$, with $\hat{F}(\mathbf{x}^{\max\mu}) = (0,1)$ and $\hat{F}(\mathbf{x}^{\min\sigma}) = (1,0)$. For a preference vector $\mathbf{r}$ in the open simplex $\mathscr{R} = \{\mathbf{r}\in\mathbb{R}^2_{>0} : r_1 + r_2 = 1\}$, the scalarized problem is:

$$\min_{\mathbf{x}\in X}\; s\big(\hat{F}(\mathbf{x}), \mathbf{r}\big), \qquad \hat{F} = (\hat{f}_1, \hat{f}_2) \tag{5}$$

We use the Chebyshev scalarization:

$$s_C(\mathbf{x},\mathbf{r}) = \max_{i=1,2}\; r_i\hat{f}_i(\mathbf{x}) \tag{6}$$

3. The Learned Solution Map and an Exact Oracle

Fig. 1 shows how the solution map is constructed and used. *Offline*: the return-maximizing anchor is obtained in closed form and the minimum-variance anchor by one convex quadratic program; together they fix $\mathbf{z}^{\mathrm{id}}$, $\mathbf{z}^{\mathrm{nad}}$ and sc in (4). *Training*: preference vectors are drawn from a Dirichlet law at every step (budgets in Section 5.1); the hypernetwork maps each of them through a softmax head to a portfolio that satisfies the budget and no-short-selling constraints exactly, and Adam minimizes the Chebyshev scalarization of the normalized objectives. No pre-solved frontier is used. *Deployment*: a query costs one forward pass (44 µs on one CPU core) for the instance (μ,Σ) used in training; Section 5.11 treats re-estimation.

3.1. Hypernetwork and Simplex Output Layer

The hypernetwork $h(\cdot,\phi) : \mathscr{R} \to X$ is a multilayer perceptron with input dimension 2, three rectified linear unit (ReLU) hidden layers of width 128 and a linear output layer of width n, followed by the softmax output layer:

$$\mathbf{x}_{\mathbf{r}} = \mathrm{softmax}(\mathbf{z}), \qquad (x_{\mathbf{r}})_i = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{7}$$

The image of (7) is the relative interior of Δ_n, so the budget and no-short-selling constraints hold for every parameter value, with no penalty and no projection.

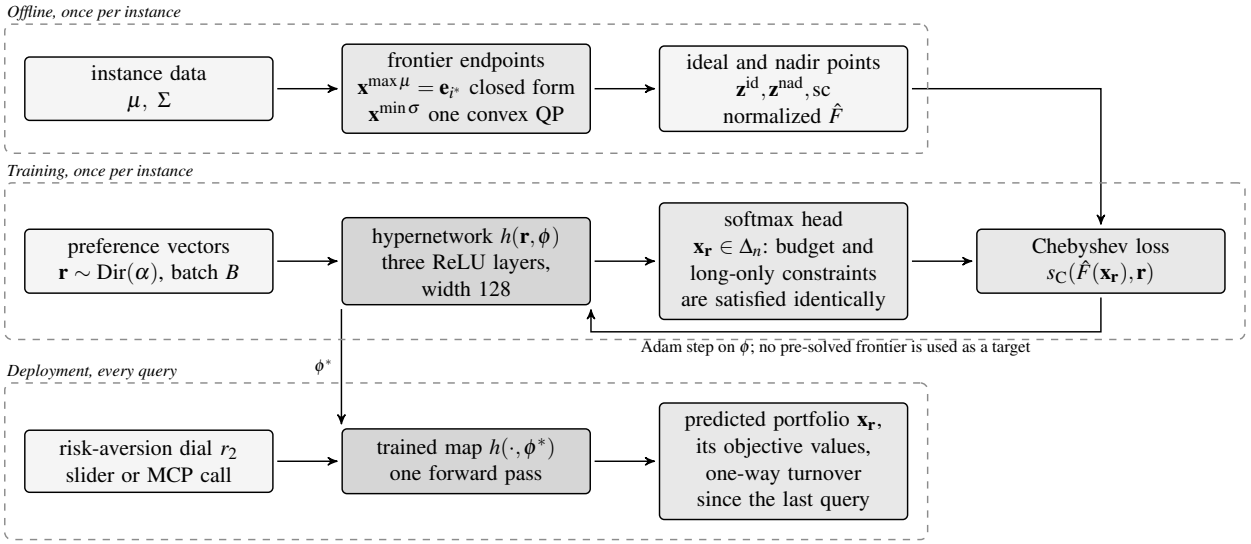


Fig. 1. Construction and use of the learned solution map

Keeping a network output in the simplex in this way is known [14]. The network is continuous and piecewise smooth: its trunk is piecewise linear and its head is C^∞. It has $q(n) = 33{,}408 + 129n$ parameters, 96% of them in the n-independent trunk at $n = 10$, so its evaluation cost depends only weakly on n.

The head cannot represent a zero weight, whereas long-only optima are typically sparse. A zero weight can be approached: $(x_\mathbf{r})_i \le e^{-(\max_j z_j - z_i)}$, so a logit gap of $\log(1/\varepsilon)$ suffices for a weight below ε. We call the residual positive weight the *softmax floor*. For assets that the exact solution never holds it is negligible: at most 4.7×10^{-9} on the ten-industry instance (Section 5.6) and 2.6×10^{-7} on VN30 (Section 5.7). For assets held over only part of the risk-aversion range, the softmax floor is not negligible: where their exact weight is zero, the network still assigns them up to about 1.2%, and these coordinates carry 10% to 15% of the allocation error (Section 5.2). The limitation lies in training, not in representation: across each change of the active set the logit gap must grow from order one to large values, and it is there that the error concentrates (Section 5.8). The predicted support is never exact, which matters for minimum lot sizes (Section 5.9).

3.2. Preference Vectors as Risk-Aversion Levels

The network input is $\mathbf{r} = (r_1, r_2)$ itself. For the linear scalarization of the normalized objectives, $r_1\hat{f}_1 + r_2\hat{f}_2 = (r_1/\mathrm{sc}_1)f_1 + (r_2/\mathrm{sc}_2)f_2 + \mathrm{const}$, which is the Markowitz objective $-\mu^\top\mathbf{x} + \lambda\,\mathbf{x}^\top\Sigma\mathbf{x}$. The corresponding coefficient is:

$$\lambda_{\mathrm{LS}}(\mathbf{r}) = \frac{r_2}{1 - r_2} \cdot \frac{\mathrm{sc}_1}{\mathrm{sc}_2} \tag{8}$$

This coefficient is an increasing bijection of $r_2 \in (0,1)$ onto $(0,\infty)$. For the Chebyshev scalarization, Lemma 2 gives $r_1\hat{f}_1 = r_2\hat{f}_2$ at $\mathbf{x}^*(\mathbf{r})$. At the solution, the return shortfall divided by the variance excess is:

$$\lambda_{\mathrm{C}}(\mathbf{r}) = \frac{r_2}{r_1} \cdot \frac{\mathrm{sc}_1}{\mathrm{sc}_2} = \frac{f_1(\mathbf{x}^*(\mathbf{r})) - z_1^{\mathrm{id}}}{f_2(\mathbf{x}^*(\mathbf{r})) - z_2^{\mathrm{id}}} \tag{9}$$

The shortfall ratio in (9) is evaluated at $\mathbf{x}^*(\mathbf{r})$ relative to the ideal point. It should be distinguished from a supporting Markowitz coefficient $\lambda_{\mathrm{M}}(\mathbf{r}) = 1/(2t^*(\mathbf{r}))$, with $t^*(\mathbf{r})$ from Section 3.5. At differentiable frontier points, this supporting coefficient equals the marginal rate $-\mathrm{d}f_1/\mathrm{d}f_2$. The coefficients λ_{LS} and λ_{C} increase strictly with r_2, while supporting Markowitz coefficients are nondecreasing along the frontier. Thus r_2 is a normalized, dimensionless preference coordinate that orders portfolios from the return-maximizing end to the minimum-variance end. It is not itself a conventional investor utility coefficient.

3.3. Training by Direct Scalarized-Loss Minimization

Training minimizes the following expected loss by stochastic gradient descent:

$$\min_{\phi}\ \mathbb{E}_{\mathbf{r}\sim\mathrm{Dir}(\alpha)}\ s_{\mathrm{C}}\big(\hat{F}(h(\mathbf{r},\phi)), \mathbf{r}\big) \tag{10}$$

The anchors are computed once beforehand (Section 2.2). Each iteration draws B preference vectors, evaluates h on them in one batched pass and takes an Adam step on the mean loss, differentiated through (7); no pre-solved frontier is used. As an implementation step ensuring $r_i > 0$ in (6), Dirichlet samples are clipped componentwise at 10^{-4} and renormalized. The loop runs for a fixed number of steps, without a convergence test, early stopping or checkpoint selection. Problem (10) is non-convex in ϕ, and no convergence guarantee is claimed for it; Section 4 concerns the problem in $\mathbf{x}$.

3.4. A Numerical Continuation Solver

A generic numerical alternative to Section 3.5 traces the frontier on the epigraph program (12) of Section 4 by sequential least-squares programming (SLSQP) with

analytic Jacobians, a numerical counterpart of the critical line [2]. Each grid point uses four initial portfolios, one uniform and three random, in two sweep directions, giving eight solves per grid point. The sweeps run in parallel, so no warm start from a neighbouring point is available. Validation covers 106 stored frontiers: 80 scaling references (10 runs, four sizes and two scalarizations), 20 large-size Chebyshev references (10 runs each at $n = 100$ and $n = 200$), and six additional checks (three each at $n = 20$ and $n = 50$). Using the same stored anchors, their weights agree with the exact critical-line solver within 1.2×10^{-6} for $n \leq 100$ and 5.3×10^{-5} at $n = 200$. This comparison uses common anchors and does not validate the truncated $n = 200$ anchors (Section 5.4). SLSQP supplies the numerical reference curves in Fig. 2 to Fig. 4 and is a computational baseline in Section 5.10.

3.5. An Exact Parametric Solver: the Critical Line on the Simplex

For this problem the whole solution map can be computed exactly. The following parametric quadratic program is the Markowitz problem with $\lambda = 1/(2t)$ for $t > 0$:

$$P(t): \quad \min_{\mathbf{x}\in\Delta_n} \tfrac{1}{2}\mathbf{x}^\top\Sigma\mathbf{x} - t\,\mu^\top\mathbf{x}, \qquad t \geq 0 \qquad (11)$$

By (A1) its solution $\mathbf{x}(t)$ is unique, and as t decreases from $+\infty$ to 0 it moves from the vertex $\mathbf{e}_{i^*}$ to $\mathbf{x}^{\min\sigma}$. On every interval of t on which the positive-weight set $\mathscr{F} = \{i : x_i > 0\}$ is constant, the Karush–Kuhn–Tucker (KKT) system is linear in t, so $\mathbf{x}_{\mathscr{F}}(t)$ and the multipliers of the assets outside $\mathscr{F}$ are affine in t, with coefficients given by $\Sigma^{-1}_{\mathscr{F}\mathscr{F}}$. The path is therefore piecewise linear, with finitely many breakpoints at which an asset enters or leaves $\mathscr{F}$: this is Markowitz's critical line [2, 15], here on the budget simplex [16]. A linear scalarization of (4) corresponds to $t = r_1\mathrm{sc}_2/(2r_2\mathrm{sc}_1)$. For the Chebyshev scalarization, Lemma 2 characterizes $\mathbf{x}^*(\mathbf{r})$ as the unique Pareto optimal point with $r_1\hat{f}_1 = r_2\hat{f}_2$. Along the path $\hat{f}_1$ is non-decreasing and $\hat{f}_2$ non-increasing, so $r_1\hat{f}_1 - r_2\hat{f}_2$ changes sign on exactly one segment; there it is a quadratic function of t, whose root gives $t^*(\mathbf{r})$ and $\mathbf{x}^*(\mathbf{r})$ in closed form. Here $\mathscr{F}$ is the free set of the critical-line algorithm and the held set of the portfolio. Numerical held-set thresholds used in the experiments are specified in Section 5.8. This solver is the exact oracle for the accuracy metrics below, the strongest non-learning baseline (Section 5.10) and the source of targets for the supervised baseline (Section 5.5).

4. Convexity and Optimality of the Scalarized Subproblem

This section concerns problem (5) in $\mathbf{x}$ for fixed $\mathbf{r}$; it contains no result on the training problem (10). The guarantees of CPFL for the scalarized problem [10, 11] rest on two hypotheses: the objectives are convex on a convex feasible set, and the scalarization is complete in the sense of [10], i.e. every solution of (5) is weakly Pareto optimal for (2), and every Pareto optimal point solves (5) for some $\mathbf{r} \in \mathscr{R}$ or is a limit of such solutions as $\mathbf{r}$ approaches a vertex of $\mathscr{R}$. Proposition 1 establishes the first and Lemma 2 the second for the Chebyshev scalarization.

Proposition 1. *Under* (A1)–(A2)*, (2) is a convex bi-objective problem on a non-empty compact convex set, with f_1 linear and f_2 strictly convex. The Pareto set $\mathscr{P}$ is non-empty and compact, each scalarized problem (5) attains its minimum, and $s_{\mathrm{C}}(\cdot,\mathbf{r})$ is convex on X for every $\mathbf{r} \in \mathscr{R}$.*

X is the intersection of a hyperplane with the closed non-negative orthant, hence compact and convex; since F is continuous, (5) attains its minimum by Weierstrass' theorem. For $w \in [0,1]$ let $\mathbf{x}(w)$ minimize $g_w = wf_1 + (1-w)f_2$ over X. The minimizer is unique, by (A1) for $w < 1$ and by (A2) for $w = 1$; it is Pareto optimal, and it depends continuously on w by Berge's maximum theorem. Conversely, since the problem is convex, every Pareto optimal point minimizes some g_w [17]. Hence $\mathscr{P} = \mathbf{x}([0,1])$ is non-empty and compact. Finally, each $\hat{f}_i$ is a positive affine transformation of f_i and hence convex, and $s_{\mathrm{C}}(\cdot,\mathbf{r})$ is a pointwise maximum of convex functions.

Lemma 2 is standard for the weighted Chebyshev scalarization of convex bi-objective problems [17]; we include a proof because the exact oracle of Section 3.5 rests on it.

Lemma 2. *Under* (A1)–(A2)*, with the exact anchors of (3), for every $\mathbf{r} \in \mathscr{R}$ the Chebyshev problem (5)–(6) has a unique solution $\mathbf{x}^*(\mathbf{r})$. It is Pareto optimal, and it is the unique Pareto optimal point with $r_1\hat{f}_1(\mathbf{x}) = r_2\hat{f}_2(\mathbf{x})$. The map $\mathbf{r} \mapsto \mathbf{x}^*(\mathbf{r})$ is a continuous bijection of $\mathscr{R}$ onto $\mathscr{P} \setminus \{\mathbf{x}^{\max\mu}, \mathbf{x}^{\min\sigma}\}$, and $\mathbf{x}^*(\mathbf{r}) \to \mathbf{x}^{\max\mu}$ as $r_2 \to 0$ and $\mathbf{x}^*(\mathbf{r}) \to \mathbf{x}^{\min\sigma}$ as $r_2 \to 1$.*

(i) If $\mathbf{p} \in \mathscr{P}$, $\mathbf{x} \in X$ and $F(\mathbf{x}) = F(\mathbf{p})$, then $\mathbf{x} = \mathbf{p}$: otherwise their midpoint would have the same f_1 and, by strict convexity, a smaller f_2, and would dominate $\mathbf{p}$. Moreover, the objective vectors of two points of $\mathscr{P}$ are never ordered componentwise. Hence $\hat{F}$ is injective on the compact set $\mathscr{P}$ and is a homeomorphism of $\mathscr{P}$ onto $\hat{F}(\mathscr{P})$. (ii) By the proof of Proposition 1, $\mathscr{P} = \mathbf{x}([0,1])$ is connected, so $\hat{F}(\mathscr{P}) \subset [0,1]^2$ is compact and connected; it contains $(0,1)$ and $(1,0)$ (Section 2.3), so its projection on the first axis is $[0,1]$. As its points are pairwise incomparable, $\hat{F}(\mathscr{P})$ is the graph of a strictly decreasing $\varphi : [0,1] \to [0,1]$, continuous because its graph is compact, with $\varphi(0) = 1$ and $\varphi(1) = 0$. (iii) Since X is compact and F continuous, every $\mathbf{x} \in X$ is weakly dominated by a point of $\mathscr{P}$; as s_{C} is non-decreasing in each $\hat{f}_i$, $\min_X s_{\mathrm{C}} = \min_{u\in[0,1]} \max(r_1 u, r_2\varphi(u))$. Because $r_1 u$ increases strictly from 0 and $r_2\varphi(u)$ decreases strictly to 0, this minimum is attained only at the unique

$u^* \in (0,1)$ with $r_1 u^* = r_2 \varphi(u^*)$. Let $\mathbf{p}^* \in \mathscr{P}$ be the point with $\hat{F}(\mathbf{p}^*) = (u^*, \varphi(u^*))$. A minimizer $\mathbf{x}$ satisfies $r_i \hat{f}_i(\mathbf{x}) \le r_1 u^* = r_2 \varphi(u^*)$, i.e. $\hat{F}(\mathbf{x}) \le \hat{F}(\mathbf{p}^*)$; since $\mathbf{p}^*$ is Pareto optimal, equality holds, and $\mathbf{x} = \mathbf{p}^*$ by (i). Thus $\mathbf{x}^*(\mathbf{r}) = \mathbf{p}^*$; and a point of $\mathscr{P}$ with $r_1 \hat{f}_1 = r_2 \hat{f}_2$ has first coordinate u^* and therefore equals $\mathbf{p}^*$. (iv) The crossing u^* solves $u/\varphi(u) = r_2/r_1$, and $u \mapsto u/\varphi(u)$ is a continuous, strictly increasing bijection of $(0,1)$ onto $(0,\infty)$. Hence $\mathbf{r} \mapsto u^*$ is a homeomorphism of $\mathscr{R}$ onto $(0,1)$, with $u^* \to 0$ as $r_2 \to 0$ and $u^* \to 1$ as $r_2 \to 1$. Since $\mathbf{x}^*(\mathbf{r}) = (\hat{F}|_{\mathscr{P}})^{-1}(u^*, \varphi(u^*))$ and $\hat{F}^{-1}(0,1) = \mathbf{x}^{\max \mu}$, $\hat{F}^{-1}(1,0) = \mathbf{x}^{\min \sigma}$ on $\mathscr{P}$, the last assertion follows.

In particular, the Chebyshev scalarization is complete for (2). The function $s_C(\cdot, \mathbf{r})$ is not differentiable on the set $\{r_1 \hat{f}_1 = r_2 \hat{f}_2\}$, which contains its minimizer. Standard remedies are projected subgradient methods [18], smoothing of the maximum [19], and epigraph reformulation. The continuation solver of Section 3.4 solves the following epigraph program:

$$\min_{\mathbf{x} \in X,\, \theta \in \mathbb{R}} \theta \quad \text{s.t.} \quad \theta - r_i \hat{f}_i(\mathbf{x}) \ge 0,\ i = 1,2 \qquad (12)$$

By Proposition 1, (12) is a convex program, and it satisfies Slater's condition (take any $\mathbf{x} \in X$ and $\theta > \max_i r_i \hat{f}_i(\mathbf{x})$). The KKT conditions are therefore necessary and sufficient, and every local solution is global.

Remark 3. By Lemma 2, $\mathbf{x}^*$ is continuous on $\mathscr{R}$. On every compact $K \subset \mathscr{R}$ it can therefore be approximated uniformly, to any accuracy, by sufficiently wide ReLU networks with a softmax head [20–22]: approximate $\mathbf{x}^*$ uniformly by maps into the relative interior of Δ_n, and their logarithms by ReLU networks. The target is, however, only piecewise smooth. By Section 3.5, $\mathbf{x}^*$ is smooth between the finitely many preferences at which the held set $\mathscr{H}(\mathbf{r}) = \{i : x_i^*(\mathbf{r}) > 0\}$ changes, and in general not differentiable there. A network of fixed size need not place its own kinks at these points, and across each of them its softmax head must drive a logit from order one towards $-\infty$; Section 5.8 measures the resulting error. The guarantees of [10, 11] concern (5) in $\mathbf{x}$; the training problem (10) is non-convex in ϕ for every scalarization, and no convergence guarantee is claimed for it.

5. Results and Discussion

5.1. Experimental Protocol and Reported Statistics

Two types of instances are used. The *scaling* family (Section 5.4) is a factor model dominated by one market factor, with $n \in \{5, 10, 20, 50, 100, 200\}$ assets and a new instance at every run, so its spread reflects both instance heterogeneity and optimization noise. The two *case studies* are fixed instances, on which only the training seed varies: a ten-industry synthetic problem with annualized returns 0.068–0.121, volatilities 0.130–0.290 and mean pairwise correlation 0.297 (Section 5.6), and the VN30 instance of Section 5.7. Every block uses $S = 10$ runs; run s uses instance seed s and training seed $1000 + s$, so variants are compared in pairs. We report means $\pm$ one sample standard deviation and, for skewed distributions, medians and 95% bootstrap percentile intervals.

Training uses Adam with a cosine schedule from 10^{-3} to 10^{-5}, $\alpha = 1$ and batches of $B = 128$ preference vectors, for $T = 30{,}000$ steps on the scaling family and its ablations and $T = 40{,}000$ steps on the case studies. The test grid consists of K evenly spaced values of r_2 in $[10^{-3}, 1 - 10^{-3}]$ ($K = 51$ for the scaling family, 101 for the case studies). It excludes $r_2 \in \{0, 1\}$, in accordance with the open set $\mathscr{R}$ and the training clip; the endpoints are known in closed form. All errors are measured against the exact solver of Section 3.5. The case-study networks of Fig. 2 to Fig. 3, the scalarization ablation and the $n = 10$ block of Table 6 were trained on a computing cluster with anchors from a numerical solver, exact to 2×10^{-13} for $n \le 100$; all other runs and all timings use one workstation (Intel Core i5-12500H, one single-threaded process per core).

5.2. Evaluation Metrics

Let $\hat{\mathbf{x}}_k = h(\mathbf{r}_k, \phi)$ and $\mathbf{x}_k^* = \mathbf{x}^*(\mathbf{r}_k)$ be the predicted and exact portfolios at the K test preferences. Accuracy in decision space is measured by the mean Euclidean distance (MED) of [10], defined as:

$$\mathrm{MED} = \frac{1}{K} \sum_{k=1}^{K} \left\| \hat{\mathbf{x}}_k - \mathbf{x}_k^* \right\|_2 \qquad (13)$$

The *allocation error* is $\mathrm{L1} = \frac{1}{K} \sum_k \|\hat{\mathbf{x}}_k - \mathbf{x}_k^*\|_1$. The quantity $\mathrm{L1}/2$ is the fraction of wealth to be reallocated to move from the predicted to the exact portfolio, and $\mathrm{MED} \le \mathrm{L1} \le \sqrt{n}\,\mathrm{MED}$. The *scalarization gap* measures suboptimality in the training objective and is defined by:

$$\mathrm{Gap} = \frac{1}{K} \sum_{k=1}^{K} \left[s_C\big(\hat{F}(\hat{\mathbf{x}}_k), \mathbf{r}_k\big) - s_C\big(\hat{F}(\mathbf{x}_k^*), \mathbf{r}_k\big) \right] \ge 0 \qquad (14)$$

This gap is expressed in the units of (4), in which the frontier spans $[0, 1]$ on each axis. The *hypervolume (HV) ratio* [23] is $\mathrm{HV}(\{F(\hat{\mathbf{x}}_k)\})/\mathrm{HV}(\{F(\mathbf{x}_k^*)\})$, computed exactly in two dimensions for the raw objectives (f_1, f_2) in the minimization convention, with the reference point $\rho = \max_k F(\mathbf{x}_k^*) + 0.1\,(\max_k F(\mathbf{x}_k^*) - \min_k F(\mathbf{x}_k^*))$ (componentwise), approximately $(1.1, 1.1)$ in normalized coordinates. The ratio is invariant under positive affine rescaling of either objective and is therefore comparable across instances and sizes, whereas absolute hypervolumes are not. Lower MED, L1 and gap indicate better approximation, while higher HV indicates better frontier coverage.

HV measures coverage independently of preference assignments. MED measures whether each preference receives its target portfolio. A map on the frontier with

incorrect assignments can have high HV and high MED, which the gap also detects. Finally, the *path length* is the arc length $\sum_k \|\mathbf{x}_{k+1} - \mathbf{x}_k\|_2$ of the weight path over the grid, and *turnover* is one-way, $\frac{1}{2}\|\mathbf{x}_{k+1} - \mathbf{x}_k\|_1$ per grid step.

5.3. Baselines

Section 5.10 compares four non-learning methods: the continuation solver (Section 3.4); a per-query convex solve, i.e. single-start SLSQP on (12) from the uniform portfolio; the exact solver (Section 3.5); and linear interpolation in r_2 between exact solutions at m evenly spaced preferences, which is feasible because Δ_n is convex. Four learning-based variants share the architecture, the budget and the seeds (Section 5.5): the linear scalarization, i.e. the Pareto hypernetwork PHN-LS [7]; the Cobb–Douglas scalarization of [10]; supervised regression on the exact solution, an optimistic form of learning from pre-solved frontiers [13]; and a sparsemax head [24] with the Chebyshev loss. Three further methods are not benchmarked: PHN-EPO [7] targets the same exact Pareto point as Lemma 2 but requires a linear program per sample and step; the minimizers of COSMOS [9] need not be Pareto optimal, so it has no exact target; and [12, 14] approximate the frontier as a set, not as a preference-indexed map.

5.4. Scaling with the Number of Assets

Table 1. Accuracy across problem sizes

n	MED	Median	95% CI	Gap (10^{-3})	HV ratio
5	0.0041 $\pm$ 0.0050	0.0007	[0.0013, 0.0073]	0.22 $\pm$ 0.46	0.9992
10	0.0083 $\pm$ 0.0073	0.0055	[0.0045, 0.0130]	0.53 $\pm$ 1.14	0.9973
20	0.0133 $\pm$ 0.0057	0.0118	[0.0101, 0.0167]	0.22 $\pm$ 0.24	0.9989
50	0.0309 $\pm$ 0.0409	0.0188	[0.0147, 0.0579]	0.76 $\pm$ 1.47	0.9956
100	0.0596 $\pm$ 0.0463	0.0453	[0.0343, 0.0892]	0.74 $\pm$ 0.60	0.9941
200	0.0734 $\pm$ 0.0588	0.0473	[0.0416, 0.1086]	1.30 $\pm$ 1.43	0.9898

Table 1 reports the accuracy of the Chebyshev hypernetwork across the number of assets n, with exact anchors and exact references: mean $\pm$ one standard deviation over $S = 10$ runs that redraw both the instance and the training seed ($B = 128$, $T = 30{,}000$, $K = 51$), median, and 95% bootstrap percentile interval for the mean (10^4 resamples). The gap in (14) is in units of 10^{-3}.

Exact anchors are necessary. A generic solver does not guarantee them: at $n = 200$, SLSQP applied to the linear program $\max_{\Delta_{200}} \mu^\top \mathbf{x}$ from three starting points stops at a diversified portfolio in nine of ten instances, short of $\max_i \mu_i$ by 12% to 28% of sc_1. Training and reference then share a truncated problem, from which the steepest part of the frontier is removed, so comparing the two cannot reveal the error; its symptom is a solution at the first test point that holds eight to twelve assets instead of two or three. All results below use the closed-form vertex.

Table 1 shows that the error grows with n: mean MED rises from 0.0041 at $n = 5$ to 0.0309 at $n = 50$ and 0.0734 at $n = 200$, while the HV ratio exceeds 0.989 at every size. The distribution is right-skewed because each run draws a new instance; at $n = 200$ the ten runs range from 0.0091 to 0.1744, with median 0.0473.

The largest errors are dominated by one identifiable mechanism. In each of the nine runs whose largest local error exceeds 0.5 (one at $n = 50$, three at $n = 100$, five at $n = 200$), this error occurs at the first test point, $r_2 = 10^{-3}$, where the network concentrates on a different asset from the exact solution, one whose expected return is only 0.04 to 0.41 annualized expected-return percentage points below the best. When the highest expected returns are nearly tied, portfolios with nearly equal objective values can differ entirely in composition, a known sensitivity of mean–variance portfolios to their inputs [25]; the median difference between the two highest expected returns decreases from 2.03 percentage points at $n = 5$ to 0.11 at $n = 200$. Near-ties explain the largest errors but not the whole growth with n: without the points whose local error exceeds 0.5, MED is still 0.0235 to 0.0422 for $n \geq 50$. In the affected runs the HV ratio is at least 0.965 and the largest scalarization gap is 0.008 to 0.045: the predicted portfolios are nearly optimal in value but different in composition, which HV alone would conceal and MED alone would overstate in objective terms.

5.5. Scalarization, Training Budget and Learning-Based Baselines

Table 2. Comparison of scalarizations (Linear denotes linear scalarization)

n		Chebyshev	Linear	Cobb–Douglas
5	mean $\pm$ sd	0.0035 $\pm$ 0.0045	0.0051 $\pm$ 0.0076	0.0544 $\pm$ 0.0340
	median	0.0006	0.0014	0.0502
10	mean $\pm$ sd	0.0081 $\pm$ 0.0070	0.0165 $\pm$ 0.0324	0.0586 $\pm$ 0.0297
	median	0.0053	0.0011	0.0567
20	mean $\pm$ sd	0.0132 $\pm$ 0.0053	0.0289 $\pm$ 0.0240	0.0710 $\pm$ 0.0140
	median	0.0118	0.0287	0.0688
50	mean $\pm$ sd	0.0213 $\pm$ 0.0137	0.0522 $\pm$ 0.0658	0.0786 $\pm$ 0.0305
	median	0.0178	0.0260	0.0737

Table 2 reports MED for the three scalarizations of [10]: mean $\pm$ one standard deviation and median over $S = 10$ paired runs, $K = 51$. Each network is scored against the exact solution map of its own scalarization. Thus MED compares approximation to each method's own target, rather than error against a common target.

Table 2 compares the three scalarizations. The linear scalarization is convex, as a positive combination of convex objectives; for Cobb–Douglas convexity is not claimed. Chebyshev has the lowest mean at every n and Cobb–Douglas the highest.

At $n = 10$, the linear scalarization is more accurate in seven of ten paired runs and has the lower median (0.0011 against 0.0053). Its mean and standard deviation (0.0165 and 0.032) are driven by one run with MED 0.102. The two targets are not equally hard: under the linear scalarization the vertex $\mathbf{e}_{i^*}$ is optimal on a whole interval of small r_2, so part of the target map is constant.

Table 3. Effect of the training budget

		$B = 64$		$B = 128$	
n		$T = 15$k	$T = 30$k	$T = 15$k	$T = 30$k
10	mean ± sd	0.0125 ± 0.0091	0.0143 ± 0.0206	0.0105 ± 0.0069	0.0083 ± 0.0073
	median	0.0104	0.0064	0.0087	0.0055
50	mean ± sd	0.0312 ± 0.0133	0.0216 ± 0.0149	0.0300 ± 0.0122	0.0309 ± 0.0409
	median	0.0292	0.0177	0.0279	0.0188

Table 3 treats the training budget as a 2×2 factorial design (batch size B, number of steps T) for the Chebyshev scalarization and reports MED mean $\pm$ one standard deviation and median over $S = 10$ paired runs using the same instance and training seeds in all four arms.

Table 3 varies B and T separately. The number of steps is the effective factor: doubling T lowers MED in 9 and 10 of ten paired runs at $n = 10$ ($B = 64$ and $B = 128$) and in 10 and 8 at $n = 50$, whereas doubling B at fixed T helps in 4 to 10 of ten runs. At equal total budget TB, more and smaller steps ($B = 64$, $T = 30$k) beat fewer and larger ones ($B = 128$, $T = 15$k) in nine of ten runs at both sizes. Since single runs dominate some means, the conclusions rest on medians and paired comparisons. The scalarization moves MED by a larger factor than the budget (Table 2).

Learning-based baselines. With the same architecture, budget and seeds, supervised regression on the exact solution is four to six times more accurate in decision space than the Chebyshev map (MED 0.0020 against 0.0083 at $n = 10$, 0.0055 against 0.0309 at $n = 50$, 0.0014 against 0.0088 on VN30): training without targets costs accuracy in the weights. Each method is best at what it optimizes: on VN30 the Chebyshev networks attain the lower scalarization gap (6.4×10^{-5} against 2.2×10^{-4}). Moreover, the supervised variant requires an exact solver, which here is itself the better way to serve the frontier (Section 5.10). PHN-LS, scored against its own exact solution map, has mean MED 0.0164 at $n = 10$ and 0.0526 at $n = 50$, with the heavy tail seen in Table 2. The sparsemax head is treated in Section 5.9.

5.6. Case Study on a Ten-Industry Synthetic Instance

Fig. 2 shows results on the ten-industry synthetic instance ($n = 10$, $K = 101$), averaged over ten training seeds. (a) Efficient frontier in the volatility/expected-return plane: learned map (dashed, markers; band: one standard deviation across seeds) and numerical SLSQP continuation reference (solid, Section 3.4). Errors are evaluated against the exact critical-line oracle. (b) Predicted composition against r_2; at small r_2 the portfolio approaches a single-asset vertex.

Table 4. Accuracy on the two case studies

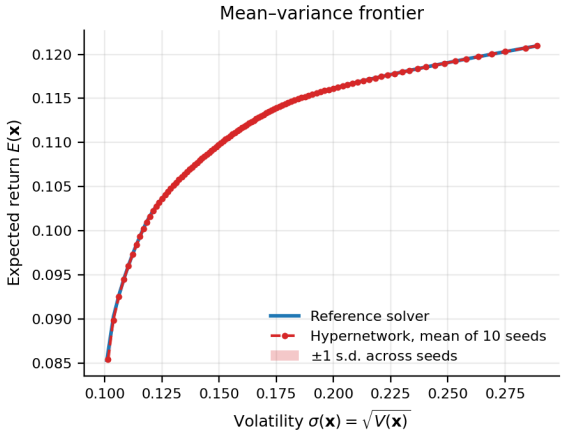
	Synthetic, $n = 10$	VN30, $n = 15$
MED	0.0083 ± 0.0017	0.0088 ± 0.0004
MED range (min, max)	[0.0065, 0.0122]	[0.0079, 0.0094]
Allocation error L1	0.0165 ± 0.0034	0.0169 ± 0.0008
Gap (10^{-3})	0.079 ± 0.028	0.064 ± 0.007
HV ratio	0.9998 ± 0.0002	0.9998 ± 0.0000
Exact path length	1.990	1.263
Predicted path length	1.978 ± 0.022	1.241 ± 0.003
MED / path	0.00415 ± 0.00086	0.00695 ± 0.00033
One-way turnover / step	0.0176 ± 0.0002	0.0128 ± 0.0001
Parameters	34,698	35,343

Table 4 reports accuracy on the two case studies against the exact solver as the mean $\pm$ one standard deviation over $S = 10$ training seeds, $K = 101$; the range is the smallest and largest MED over the seeds. "Path" is the arc length of the weight path; turnover is one-way, $\frac{1}{2}\|\mathbf{x}_{k+1} - \mathbf{x}_k\|_1$ per grid step.

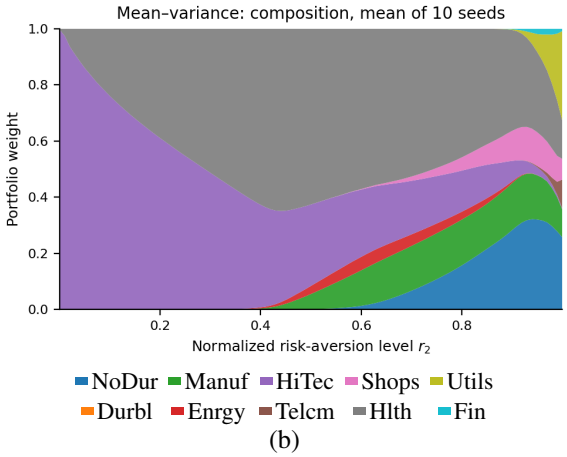
Table 4 compares the learned map with the exact solution. On the synthetic instance MED is 0.00825 ± 0.00171 over ten runs with exact anchors (0.0083 for the ten networks of Fig. 2). The allocation error L1 = 0.0165 means that 0.82% of wealth must be reallocated to reach the exact portfolio, and the predicted weight path reproduces the exact path length to 0.6%. The two frontiers in Fig. 2(a) are visually indistinguishable.

The map acts as a risk dial. As r_2 increases from 0.1 to 0.9, expected return decreases from 0.1184 to 0.1017 and volatility from 0.2367 to 0.1198, a 49% reduction in volatility for a 14% reduction in expected return, and the in-sample Sharpe ratio, with decimal annual risk-free rate $r_f = 0.02$ (2%), increases monotonically from 0.415 to 0.682. The composition shifts accordingly (Fig. 2(b)), from the two highest-return industries at $r_2 = 0.1$ (information technology 76.0%, health care 24.0%) to eight industries at $r_2 = 0.9$, led by health care (36.0%), with information technology at 7.3%. The consumer-durables industry, dominated in both objectives, is excluded at every level with predicted weights of at most 4.7×10^{-9}: the softmax floor.

Moving the dial has a cost in decision space: the mean one-way turnover between adjacent points of the 101-point grid is 0.0176, so traversing the whole range

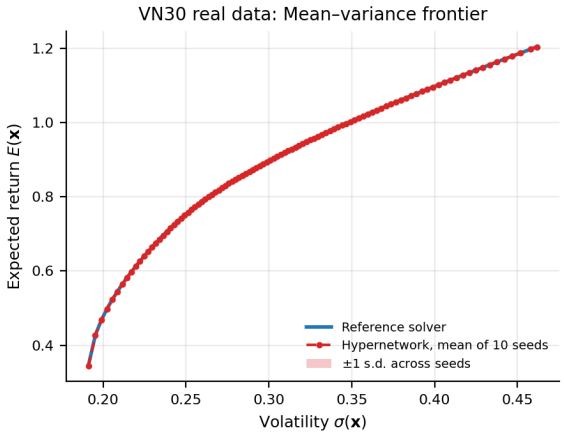


(a)

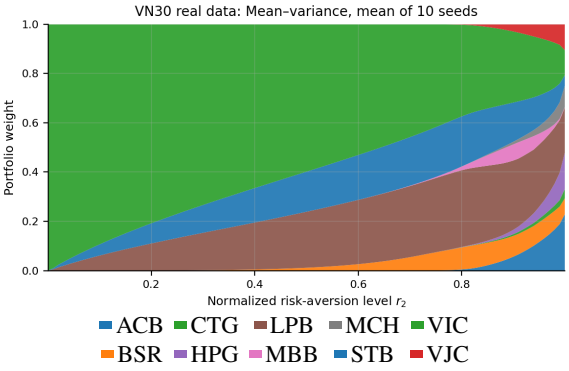


(b)

Fig. 2. Synthetic instance: learned frontier versus numerical SLSQP reference (a), and learned portfolio weights (b)



(a)



(b)

Fig. 3. VN30 instance: learned frontier versus numerical SLSQP reference (a), and learned portfolio weights (b)

reallocates about 1.76 times the portfolio value. The frontier is smooth in objective space but not static in decision space.

5.7. Real-Market Instance: VN30

Fig. 3 shows results on the VN30 instance ($n = 15$, $K = 101$), averaged over ten training seeds. (a) Efficient frontier with the learned map and numerical SLSQP continuation reference. Errors are evaluated against the exact critical-line oracle. (b) Predicted composition against r_2; only the ten tickers exceeding 0.5% are labelled.

Data. The universe is the VN30 index of the Ho Chi Minh Stock Exchange (composition effective August 3, 2026), with daily closing prices from the DNSE provider via the `vnstock` package for the window 2024-08-07 to 2026-08-07. One ticker with fewer than 200 trading days is dropped. Among the remaining 29, VPL (listed 2025-05-13) limits the common calendar to 307 days, so this 307-day common calendar is used only for preliminary screening: the nine tickers with $\mu_i \le r_f$ over it (FPT, GVR, SHB, SSB, TCB, VCB, VIB, VPL, VRE) are dropped, and the common calendar of the twenty survivors spans the full window, 486 trading days. Because the estimates change with the window, the same filter, at the deposit rate $r_f = 5.9\%$, is applied again on this window and drops BID, GAS, MSN, SAB and VNM. The filter is not required by the mean–variance problem; it is kept for comparability with Sharpe-based variants of this construction. The resulting universe has $n = 15$ tickers (ACB, BSR, CTG, HDB, HPG, LPB, MBB, MCH, MWG, SSI, STB, VHM, VIC, VJC, VPB), with annualized mean log-returns from 6.6% to 120.4%, annualized volatilities from 24.3% to 48.0% and mean pairwise correlation 0.357; these estimates describe one strong two-year window, not a stable market.

Scope. The screen uses returns estimated on the sample that defines the instance, which introduces a selection effect. The in-sample experiments below therefore validate the approximation of the exact solver by the learned map, not investment performance.

Accuracy. MED is 0.00878 ± 0.00041 over ten training seeds (Table 4); L1 $= 0.0169$, so 0.84% of wealth must be reallocated to reach the exact portfolio.

The predicted path length matches the exact one to 1.7%, and the two frontiers in Fig. 3(a) nearly coincide. Normalized by path length, the real instance is harder than the synthetic one (0.00695 against 0.00415), as expected of a frontier that holds up to ten assets at once. The five tickers never held by the exact solution receive predicted weights of at most 2.6×10^{-7}.

Risk dial. As r_2 increases from 0.1 to 0.9, expected return decreases from 112.7% to 62.6% and volatility from 41.7% to 22.2%, while the in-sample Sharpe ratio is *non-monotone*: it rises from 2.562 to a maximum of 2.813 at $r_2 = 0.7$ and falls to 2.554. The reason is that the return-maximizing endpoint is a vertex of the simplex, not a tangency portfolio: the least risk-averse portfolios are not the best in risk-adjusted terms, and a client-facing system exposing this dial should say so. The composition shifts (Fig. 3(b)) from near-concentration in one ticker at $r_2 = 0.1$ (VIC 89.3%) to ten tickers at $r_2 = 0.9$ (VIC 28.9%, LPB 28.1%, STB 15.9%), with mean one-way turnover 0.0128 per grid step.

Out of sample. Over eleven estimation windows with subsequent holding periods (250-day windows, 21-day holding periods, August 21, 2025 to August 3, 2026), networks retrained on each window stay within 0.026 of the exact weights, and their realized annual returns and Sharpe ratios differ from those of the exact portfolios by at most 0.7 percentage points and 0.025. The ex-ante expected return of the exact portfolios (124.5% on average) far exceeds the realized one (75.4%): the approximation error is small compared with the estimation error. The absolute levels inherit the screen and are not evidence of investment performance.

5.8. Localization of the Residual Approximation Error

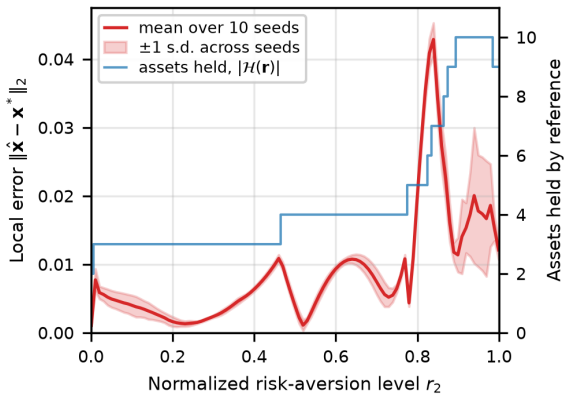


Fig. 4. Local error and held-set size on VN30

Fig. 4 shows local approximation error against r_2 on the VN30 instance (mean over ten training seeds; band: one standard deviation) and number of assets held by the reference solution (right axis, step). The held-set diagnostic uses the numerical reference curve, while local errors are evaluated against the exact critical-line oracle. The error grows with held-set size and concentrates near its changes.

Table 5. Error along the risk-aversion range

Region of the dial	Mean $\lvert\mathscr{H}_{10^{-6}}\rvert$	Local error
$r_2 \in [0.0, 0.1)$	2.9	0.0048
$r_2 \in [0.1, 0.5)$	3.1	0.0042
$r_2 \in [0.5, 0.9)$	4.9	0.0123
$r_2 \in [0.9, 1.0]$	9.8	0.0164
At changes (9 grid points)	—	0.0190
Elsewhere (92 points)	—	0.0078

Table 5 reports approximation error along the risk-aversion range on the VN30 instance as the mean over $S = 10$ training seeds. $\lvert\mathscr{H}_{10^{-6}}\rvert$ counts assets whose exact weight exceeds 10^{-6} for this detailed VN30 profile. Table 6 instead uses 10^{-4} as a common numerical threshold for comparisons across instances, filtering smaller residual weights. These thresholded diagnostics differ from the exact positive-weight set. The displayed dial regions are intersected with the test grid $[10^{-3}, 1 - 10^{-3}]$.

MED averages a local error that varies strongly along the dial. Fig. 4 and Table 5 decompose the VN30 error and show two effects.

First, the error grows with the size of the held set. On $r_2 \in [0, 0.1)$, near the return-maximizing end, the exact solution holds 2.9 assets on average and the mean local error is 0.0048; on $r_2 \in [0.9, 1]$, near the minimum-variance end, it holds 9.8 assets and the error is 0.0164, 3.4 times larger. A near-vertex target is easy for a softmax head, which approaches a vertex as its logits separate. A target with many small positive weights is hard: there the loss is dominated by $\hat{f}_2$ and is nearly flat in the directions that distinguish these weights.

Second, the error concentrates at changes of the held set. At the nine grid points where $\lvert\mathscr{H}_{10^{-6}}\rvert$ differs from its value at the preceding point, the mean local error is 0.0190, against 0.0078 at the other ninety-two, a ratio of 2.44 (per-seed median 2.45, range 2.27–2.61). The concentration holds on average, not pointwise: the largest peak in Fig. 4 (0.043 at $r_2 = 0.84$) lies in a cluster of changes, but two local maxima are not near any change. This is the effect anticipated in Remark 3: the breakpoints of the piecewise-linear ReLU trunk need not align with the target's active-set changes. The complete network includes a smooth softmax head and is not piecewise linear. Separately, that head cannot attain exact zero weights on either side of a change.

Table 6 reports the concentration of approximation error at changes of the held set, with ten runs per block. Here an asset belongs to $\mathscr{H}_{10^{-4}}$ if its exact weight

Table 6. Error at active-set changes

		Mean local error		Per-run ratio		
Block	Mean number of change points	At changes	Elsewhere	Median	Range	> 1
VN30, $K = 101$	9.0	0.0190	0.0078	2.45	[2.27, 2.61]	10/10
Ten-industry, $K = 101$	7.0	0.0170	0.0076	1.96	[1.71, 3.54]	10/10
Scaling $n = 10$, $K = 101$	4.1	0.0241	0.0066	2.83	[1.32, 10.50]	10/10
Scaling $n = 50$, $K = 51$	10.0	0.0498	0.0276	1.70	[0.70, 2.86]	9/10

exceeds 10^{-4}. A change point is a grid point at which $|\mathscr{H}_{10^{-4}}|$ differs from that at the preceding point. The change-count column is the arithmetic mean over the ten runs.

The per-run ratio divides the mean error at change points by that elsewhere. For this skewed statistic, the median, range and count of runs above one are reported.

Both effects extend beyond the VN30 instance, to different degrees. With one definition for all blocks (Table 6), the median per-run ratio lies between 1.70 and 2.83 and exceeds one in nine or ten of the ten runs of every block. Varying the held-set threshold from 10^{-6} to 10^{-3} changes the VN30 ratio of mean errors only from 2.44 to 2.18, and longer training does not remove the concentration (median 2.14 to 2.65 over the four budget arms of Table 3 at $n = 10$), although it lowers the error everywhere. Error and held-set size are strongly associated on VN30 and the ten-industry instance, with rank correlations 0.73 and 0.68. The association is weaker on the scaling family (0.32 at $n = 10$).

5.9. Sparse Output Layers and Thresholding

A sparsemax output layer [24], the Euclidean projection of the logits onto the closed simplex, produces exact zeros. Trained with the same loss, budget and seeds, it removes the softmax floor but is not a better model class here. At $n = 10$ it sets 68% of predicted weights exactly to zero, averaged over assets, test preferences and ten runs. Its median MED is lower than softmax (0.0022 against 0.0055), but two runs have markedly larger MED (0.0185 and 0.0556), indicating poor approximation rather than a crashed training run; its MED on VN30 is 0.0202; and at $n = 50$ it fails in most runs (mean MED 0.299). The cause is structural: a weight that is exactly zero passes no gradient to its logit, so an asset excluded early in training cannot re-enter. For trading systems a simpler rule suffices. Thresholding at $\tau = 0.005$ (0.5%) followed by renormalization, applied to the ten VN30 networks, reduces the mean number of held assets from 15.0 to 4.6 (exact solution: 4.5) and makes the predicted support exact at 83% of the grid points, with MED unchanged and expected return changed by at most 0.51 percentage points; above about 1% the rule removes assets that the exact solution holds. Thresholding at the smallest tradable weight therefore handles minimum lot sizes; integer lot constraints require a mixed-integer model, outside the scope of this paper.

5.10. Computational Cost

Table 7 reports the cost of serving the frontier on one core of an Intel Core i5-12500H (laptop GPU where stated), as the median over repeated calls. Setup is paid once per instance (μ, Σ); a query returns the portfolio for one preference. MED is against the exact solution, mean over the ten scaling instances. Network training: $T = 30{,}000$ steps; interpolation setup: 17 cold SLSQP solves. †Computed with the truncated $n = 200$ anchors of Section 5.4.

Table 7 compares computational costs on the same processor core. The continuation-reference MED 0.36† at $n = 200$ uses truncated anchors, unlike the exact critical-line results and the exact-anchor hypernetwork results. This entry documents the anchor defect and is not an exact-reference comparison. *Against a generic solver the learned map is fast*: a cold SLSQP solve of (12) takes 1.32 ms at $n = 10$ and 3.8 s at $n = 200$, a forward pass takes approximately 44–45 μs across the tested sizes (the n-independent trunk holds 96% of the parameters at $n = 10$ and 56% at $n = 200$), and training (47 s at $n = 10$) is recovered after about 37,000 queries. This estimate divides training time by the per-query saving (1.32 ms − 44 μs). It includes network training and inference, excludes the one-time normalization-anchor computation, and involves no solved-frontier training targets. *Against the exact parametric solver it is not*: the critical-line path costs 224 μs at $n = 10$ and 3.35 ms at $n = 200$, and a query on it 38 and 96 μs, without approximation error. For $n \leq 50$ training never pays for itself, and at $n = 200$ only after about 1.4 million queries, provided the approximation error is acceptable. Linear interpolation between $m = 17$ exact solutions matches the network's MED from $n = 10$ on, at 11 μs per query, and a laptop GPU is slower than one CPU core at this network size (245 μs per pass). Thus the learned map has no computational advantage here. Objective pairs without a parametric solver remain to be tested.

5.11. Updating the Map when the Market Estimates Change

The network is trained for one instance (μ, Σ), and "real-time" refers to inference for that instance. Eleven rolling VN30 estimation windows of 250 trading days, advanced by 21 days, give ten adjacent window transitions. We evaluate the ten transitions with three seeds (30 comparisons).

Table 7. Cost of serving the efficient frontier

	Setup per instance			One query			MED		
Method	$n = 10$	$n = 50$	$n = 200$	$n = 10$	$n = 50$	$n = 200$	$n = 10$	$n = 50$	$n = 200$
SLSQP continuation reference (cluster, core-s)	none	none	none	64.7 ms	227.6 ms	91.6 s	$< 10^{-6}$	$< 10^{-6}$	$0.36^{\dagger}$
Per-query SLSQP, cold start	none	none	none	1.32 ms	8.52 ms	3.8 s	2×10^{-7}	6×10^{-7}	3×10^{-7}
Exact critical line (Section 3.5)	224 μs	808 μs	3.35 ms	38 μs	43 μs	96 μs	exact	exact	exact
Interpolation, $m = 17$ cold SLSQP solves	22.4 ms	144.9 ms	64.6 s	11 μs	11 μs	11 μs	0.0074	0.0135	0.0203
Hypernetwork, CPU core (this paper)	47.2 s	58.5 s	72.0 s	44 μs	44 μs	45 μs	0.0083	0.0309	0.0734
Hypernetwork, laptop GPU	as above	as above	as above	245 μs	252 μs	980 μs	as above	as above	as above

On the new window, the network of the previous window has MED 0.143 against the exact solution of the new window, compared with 0.0110 after retraining. Fine-tuning reduces MED on the new window to 0.048 and 0.022 after 1,000 and 5,000 steps (1.6 and 8 s on one core), against MED 0.051 and 0.028 from random initialization on that same window, and full retraining takes about 66 s. Retraining thus fits a daily or monthly batch schedule but is not a real-time operation, whereas the exact solver recomputes the frontier in milliseconds.

5.12. Deployment: An Interactive Slider and an MCP Server

A query of constant, sub-millisecond cost makes the trained map a deployable component, and we built two interfaces around the same network. The first is a web page with a single slider for r_2: each movement triggers one forward pass and updates the portfolio, its position on the frontier and the one-way turnover relative to the previous setting. The second is a Model Context Protocol (MCP) server that exposes the map as a tool to language-model assistants: the user requests the portfolio at a given risk-aversion level and adjusts it in follow-up prompts, and each request returns weights without solving a quadratic program. On this problem the same interfaces could also expose the exact solver (Section 5.10); they demonstrate the interface, not an advantage of the learned map. Code, trained weights and reproduction scripts are available from the authors on request.

6. Conclusion

For the long-only mean–variance problem, the critical line provides an exact oracle against which a controllable Pareto front learner can be evaluated. A hypernetwork with a softmax output layer reproduces the exact frontier to a mean Euclidean weight error of 0.0083 on a ten-industry synthetic instance and 0.0088 on a fifteen-asset VN30 instance, but the exact solver computes the whole frontier in milliseconds, so on this problem the learned map offers no computational advantage. The error grows with the number of assets, peaks at near-ties among the highest expected returns and concentrates at changes of the active set, independently of the training budget; the main limitations are the model class and the training signal. The hypernetwork is an amortized approximation for a fixed instance, but it does not outperform the exact critical-line solver on the tested problem. The case for the learned map therefore rests on objective pairs without a parametric solver, which remain to be tested.

Acknowledgments

The authors thank Hanoi University of Science and Technology for access to its GPU cluster for the case-study training runs.

The authors also thank the anonymous reviewers for their careful reading of the manuscript and their constructive comments.

Declaration of Competing Interest

The authors declare no conflicts of interest.

Declaration on the Use of Generative AI

During the preparation of this work the authors used generative artificial-intelligence tools to assist with language editing, with drafting portions of the manuscript text and with implementing parts of the experimental code. All scientific content, including the problem formulation, the mathematical statements and their proofs, the experimental design, the analysis of the results and the conclusions, is the authors' own. The authors reviewed and verified the entire manuscript, including every numerical value reported, and take full responsibility for its content.

References

[1] H. Markowitz, Portfolio selection, The Journal of Finance, vol. 7, no. 1, pp. 77–91, Mar. 1952. https://doi.org/10.1111/j.1540-6261.1952.tb01525.x

[2] H. Markowitz, Portfolio selection: efficient diversification of investments, New York: John Wiley & Sons, 1959.

[3] A. Salo, M. Doumpos, J. Liesiö, and C. Zopounidis, Fifty years of portfolio optimization, European Journal of Operational Research, vol. 318, no. 1, pp. 1–18, Oct. 2024. https://doi.org/10.1016/j.ejor.2023.12.031

[4] H. Keffert, Robo-advising: optimal investment with mismeasured and unstable risk preferences, European Journal of Operational Research, vol. 315, no. 1, pp. 378–392, May 2024.
https://doi.org/10.1016/j.ejor.2023.12.002

[5] B. Amos, Tutorial on amortized optimization, Foundations and Trends in Machine Learning, vol. 16, no. 5, pp. 592–732, Jun. 2023.
https://doi.org/10.1561/2200000102

[6] D. Ha, A. M. Dai, and Q. V. Le, Hypernetworks, in Proceedings of the 5th International Conference on Learning Representations (ICLR), Feb. 2017.
https://openreview.net/forum?id=rkpACe1lx

[7] A. Navon, A. Shamsian, G. Chechik, and E. Fetaya, Learning the Pareto front with hypernetworks, in Proceedings of the 9th International Conference on Learning Representations (ICLR), Jan. 2021.
https://openreview.net/forum?id=NjF772F4ZZR

[8] L. P. Hoang, D. D. Le, T. A. Tuan, and T. N. Thang, Improving Pareto front learning via multi-sample hypernetworks, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, iss. 7, pp. 7875–7883, Jun. 2023.
https://doi.org/10.1609/aaai.v37i7.25953

[9] M. Ruchte and J. Grabocka, Scalable Pareto front approximation for deep multi-objective learning, in Proceedings of IEEE International Conference on Data Mining (ICDM), pp. 1306–1311, Dec. 2021.
https://doi.org/10.1109/ICDM51629.2021.00162

[10] T. A. Tuan, L. P. Hoang, D. D. Le, and T. N. Thang, A framework for controllable Pareto front learning with completed scalarization functions and its applications, Neural Networks, vol. 169, pp. 257–273, Jan. 2024.
https://doi.org/10.1016/j.neunet.2023.10.029

[11] T. A. Tuan, N. V. Dung, and T. N. Thang, A hyper-transformer model for controllable Pareto front learning with split feasibility constraints, Neural Networks, vol. 179, Nov. 2024, Art. no. 106571.
https://doi.org/10.1016/j.neunet.2024.106571

[12] Z. Feinstein and B. Rudloff, Deep learning the efficient frontier of convex vector optimization problems, Journal of Global Optimization, vol. 90, no. 2, pp. 429–458, Oct. 2024.
https://doi.org/10.1007/s10898-024-01408-x

[13] P. Chatigny, I. Sergienko, R. Ferguson, J. Weir, and M. Bergeron, Learning the efficient frontier, in Advances in Neural Information Processing Systems, vol. 36, pp. 22003–22020, Dec. 2023.
https://doi.org/10.52202/075280-0965

[14] X. Warin, Deep learning for efficient frontier calculation in finance, Journal of Computational Finance, vol. 26, no. 1, Jun. 2022.
https://doi.org/10.21314/JCF.2021.017

[15] H. Markowitz, The optimization of a quadratic function subject to linear constraints, Naval Research Logistics Quarterly, vol. 3, no. 1–2, pp. 111–133, Mar. 1956.
https://doi.org/10.1002/nav.3800030110

[16] A. Niedermayer and D. Niedermayer, Applying Markowitz's critical line algorithm, in Handbook of portfolio construction: contemporary applications of Markowitz techniques, J. B. Guerard Jr. (Ed.), Boston, MA: Springer, pp. 383–400, 2010.
https://doi.org/10.1007/978-0-387-77439-8_12

[17] K. Miettinen, Nonlinear multiobjective optimization, vol. 12, New York, NY: Springer, 1998.
https://doi.org/10.1007/978-1-4615-5563-6

[18] T. N. Thang and T. N. Hai, Self-adaptive algorithms for quasiconvex programming and applications to machine learning, Computational and Applied Mathematics, vol. 43, no. 4, Jun. 2024, Art. no. 249.
https://doi.org/10.1007/s40314-024-02764-w

[19] X. Lin, X. Zhang, Z. Yang, F. Liu, Z. Wang, and Q. Zhang, Smooth Tchebycheff scalarization for multi-objective optimization, in Proceedings of the 41st International Conference on Machine Learning (ICML), Proceedings of Machine Learning Research, vol. 235, pp. 30479–30509, 2024.
https://proceedings.mlr.press/v235/lin24y.html

[20] G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of Control, Signals, and Systems, vol. 2, no. 4, pp. 303–314, Dec. 1989.
https://doi.org/10.1007/BF02551274

[21] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, Multilayer feedforward networks with a nonpolynomial activation function can approximate any function, Neural Networks, vol. 6, no. 6, pp. 861–867, 1993.
https://doi.org/10.1016/S0893-6080(05)80131-5

[22] T. Galanti and L. Wolf, On the modularity of hypernetworks, in Advances in Neural Information Processing Systems, vol. 33, pp. 10409–10419, 2020.
https://proceedings.neurips.cc/paper/2020/hash/75c58d36157505a600e0695ed0b3a22d-Abstract.html

[23] E. Zitzler and L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach, IEEE Transactions on Evolutionary Computation, vol. 3, no. 4, pp. 257–271, Nov. 1999.
https://doi.org/10.1109/4235.797969

[24] A. F. T. Martins and R. F. Astudillo, From softmax to sparsemax: a sparse model of attention and multi-label classification, in Proceedings of The 33rd International Conference on Machine Learning (ICML), Proceedings of Machine Learning Research, vol. 48, pp. 1614–1623, 2016.
https://proceedings.mlr.press/v48/martins16.html

[25] M. J. Best and R. R. Grauer, On the sensitivity of mean-variance-efficient portfolios to changes in asset means: some analytical and computational results, The Review of Financial Studies, vol. 4, no. 2, pp. 315–342, Apr. 1991.
https://doi.org/10.1093/rfs/4.2.315