End-to-End Portfolio Optimization Project Report

Distributionally Robust, Risk Budgeting

Guangyu Zhou, Zheng Li, Chia-Yi Chien
Cornell CFEM

May 2025

Abstract

We study end-to-end portfolio construction using a differentiable optimization layer built with `CvxpyLayer`, jointly trained with a return prediction network. Starting from a distributionally robust (DR) optimization framework, we replicate prior results on a 20-stock S&P 500 universe (2000–2021), confirming its advantages in Sharpe ratio and drawdown control over sequential and equal-weight baselines. We then replace the DR core with a family of convex objectives, including (i) mean–variance formulations (standard and Sharpe-maximizing), and (ii) risk-only strategies such as minimum-variance, risk budgeting, and risk parity. Each optimization problem is reformulated to satisfy Disciplined Parametrized Programming (DPP), enabling efficient gradient propagation through a single cached KKT system. We deployed the models on an ETF-based portfolio selection due to the implementation of risk budgeting. Empirical results highlight that while equal weighting remains a strong baseline, direct Sharpe optimization offers the best risk-adjusted returns, and that model performance is often constrained more by risk estimation than return prediction.

Contents

1 Introduction

1.1 End-to-End Learning

End-to-end decision-focused learning directly *optimizes decisions and predictions*: unlike the classical predict then optimize two steps model, a prediction model $\widehat{\theta}(x)$ is trained jointly with an optimization layer that maps $\widehat{\theta}(x) \mapsto w^\star(\widehat{\theta}(x))$. The loss is computed on the final decision quality (utility, risk, cost), so the predictor learns only what is useful for optimal portfolio construction in financial setting.

1.2 Distributionally Robust [Costa and Iyengar, 2023]

Combines End-to-End learning with *distributionally robust optimization* (DRO) to account for model risk. The decision layer chooses portfolios by solving a minimax problem where the distribution of asset returns assumed live in an ambiguity set centered around a nominal distribution. Details on the construction of optimization layer in decision process will be covered in section 4.2.

1.3 End-to-End Risk Budgeting [Uysal et al., 2021]

We employ risk-only portfolio decision rules to avoid the instability caused by return estimation in traditional optimization. These portfolio selection approaches are inherently limited by the exclusion of expected returns. Therefore, we focus on three distinct yet representative risk-only methods, which are detailed in Section 4.5.

1.4 Motivation and Contributions of This Project

- Explore the performance of End-to-end model on portfolio optimization
- Reproduce baseline results.
- Explore alternative optimization objectives (Markowitz variants, CVaR, Risk budgeting).
- Benchmark different methods.

2 Literature Review

End-to-end learning has gained increasing attention not only in portfolio optimization but also in broader data-driven decision-making problems. These systems are typically categorized into model-free and model-based approaches. Model-free methods learn a direct mapping from inputs to decisions by minimizing task-specific loss functions ([Donti et al., 2017]; [Zhang et al., 2020]), while model-based methods incorporate a differentiable optimization layer into the prediction model, improving interpretability and alignment with decision objectives. This paper focuses on the model-based approach, aiming to incorporate and compare different optimization theories within an end-to-end learning framework.

A key challenge in model-based end-to-end learning is linking predictive models with differentiable optimization problems. [Agrawal et al., 2019] address this by introducing Differentiable Convex Optimization Layers, enabling end-to-end analytical differentiation through disciplined convex programs. They develop `CvxpyLayer`, a Python package based on disciplined parametrized programming, which converts `CVXPY` problems into differentiable layers compatible with `PyTorch` and `TensorFlow`.

Our work builds primarily on the methodologies proposed in [Costa and Iyengar, 2023] and [Uysal et al., 2021].

[Uysal et al., 2021] propose an end-to-end framework using a fully connected neural network to generate portfolio decisions. They test both model-free and model-based approaches, optimizing task-specific loss functions such as the Sharpe ratio and cumulative return. In

the model-based strategy, they embed risk-budgeting portfolio optimization within the decision layer. They also derive the KKT conditions of the optimization problem to clarify the differentiation process during backpropagation. Furthermore, a stochastic gating mechanism is introduced for asset selection, enhancing risk-based portfolio performance by filtering out low-volatility assets with limited return potential.

[Costa and Iyengar, 2023] proposes a end-to-end portfolio construction system with the decision layer is a distributionally robust optimization problem which requires both the point prediction and prediction errors as inputs to quantify and control model risk. The DR is formulated as a minimax optimization problem where the objective function is a combination of the mean loss and the worst-case risk. The worst case risk is taken over a set of probability measures within a "distance" δ of the empirical measure. The parameters that controls the risk appetite (γ) and model robustness (δ) can be learned directly during the training process from data, skipping possible hyperparameter selection tuning steps which could be difficult and computationally expensive in practical settings.

3 End-to-End Portfolio Construction

In this section, we present a general model-based end-to-end framework for portfolio optimization. The system consists of two main components: a prediction layer, which maps input data to intermediate variables, and an optimization (or decision) layer, which outputs portfolio weights by solving a differentiable and convex optimization problem. These layers are seamlessly connected, allowing gradients to flow through the entire architecture. Input data is passed sequentially through the prediction and decision layers, ultimately contributing to a task-specific loss function based on portfolio performance metrics. Model parameters are updated via backpropagation.

As in standard deep learning settings, our objective is to find a set of model parameters θ to minimize a loss function f. In portfolio optimization, traditional loss functions like mean squared error or cross-entropy are not applicable due to the absence of labeled data. Instead, we define the loss function based on portfolio performance metrics that promote desirable outcomes, such as higher returns or lower volatility. These functions depend on realized asset returns r (which are observable) and optimal asset weights (or asset allocation) z^* (which are latent). The asset weights are derived by solving a convex portfolio optimization problem c, which parameters are the outputs y from the prediction model given selected features x. By integrating prediction and optimization into a unified end-to-end learning framework, the portfolio optimization process can be expressed as follows:

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \mathbb{E}_{\mathcal{D}}[f(r, z^*(y(x;\theta)))] \\ \text{s.t.} \quad & z^*(y;\theta) = \arg\min_{z \in \mathcal{Z}} \mathbb{E}_{\mathcal{D}}[c(y,z)] \end{aligned} \tag{1}$$

3.1 Prediction Layer

In the prediction layer, we employ a simple fully connected neural network with single hidden layer. Given the noisy nature of financial data, overly complex models often harm out-of-sample performance ([Rasekhschaffe and Jones, 2019]).

The role of the neural network is to learn informative representations of the input features; each hidden node can be interpreted as a learned feature, which is then mapped to task-specific outputs. By selecting an appropriate loss function aligned with our optimization objective, we can give meaningful interpretation to these outputs.

Because the prediction layer is followed by an optimization layer, its output dimension must align with the parameters required by the optimization problem. A key contribution of our

work is to explore how differently portfolio optimization formulations perform in the model-based end-to-end learning system. Assuming a portfolio of n assets, we consider three types of outputs corresponding to different optimization frameworks:

1. Expected Return: In mean-variance optimization, the model predicts the expected return vector of dimension n, where each element corresponds to one asset.

2. Risk Budget: In risk-budgeting approaches, expected returns are not required. Instead, the neural network outputs a risk budget allocation of dimension n, assigning a target risk contribution to each asset.

3. Covariance Matrix: In cases where neither returns nor risk budgets are used directly, the model predicts the covariance matrix. To ensure symmetry and positive semi-definiteness, the network predicts the lower triangular elements of a matrix L (with positive diagonal entries), from which the full covariance matrix is reconstructed as $\Sigma = LL^T$. The output dimension is therefore $n(n+1)/2$, the number of elements in a lower triangle matrix.

We use a single hidden layer with a leaky ReLU activation function (negative slope = 0.1) to introduce nonlinearity and enhance feature representation. When predicting risk budgets, a softmax function is applied to the output layer to ensure normalized allocations.

3.2 Decision Layer

The decision layer is formulated as a convex optimization problem that outputs an asset allocation vector of dimension n. In this paper, we focus on long-only portfolios; therefore, the basic constraints in the optimization problem are $\sum_{i=1}^{n} z_i = 1$ and $z \geq 1$. By selecting different portfolio theories, we introduce useful inductive biases into the end-to-end learning system. Portfolio optimization is a well-established field with extensive literature, most of which is based on Markowitz's foundational theory of maximizing expected return for a given level of risk. Subsequent work has extended this framework by adding constraints, regularization terms, different risk measures, or practical considerations such as transaction costs.

To embed the optimization problem within the end-to-end framework, we formulate it using `CVXPY` in Python and implement it as a differentiable optimization layer using `Cvxpylayer`. This allows outputs from the prediction layer to be passed forward and gradients to flow backward from the loss function.

In this paper, we explore two groups of portfolio models: return-based portfolios, which take expected returns as input, and risk-only portfolios, which rely solely on risk. A key requirement for integration into the end-to-end system is that the optimization problem must be convex. Section 4 provides details on each selected model and the techniques used to ensure convexity.

4 Decision, Optimization-Layer Design and Construction

In this section, we will introduce the original optimization layer in Distributionally Robust structure, then talk about the extensions we incorporated into the end-to-end optimization layer. We have experimented on Markowitz variates, CVaR, Risk-only methods.

4.1 CVXPY Layers, DPP convex formulation

One thing needs additional attention is: the training of end-to-end requires the gradient of the optimization layer which are fed into prediction layer through backpropagation. This could be done by **CVXPY Layers** [Agrawal et al., 2019]. It allows a convex optimization problem to be embedded as an *implicit, differentiable layer* inside a neural network: the layer maps data

parameters $\theta \in \mathbb{R}^d$ to the optimal decision $w^\star(\theta)$, and automatically exposes Jacobian–vector products $\partial w^\star / \partial \theta$ for backpropagation.

For the layer to compile once and remain differentiable throughout training, the problem must satisfy *Disciplined Parametrized Programming (DPP)*, an extension of Boyd's *Disciplined Convex Programming (DCP)* rules:

DPP guarantees two key properties:

- Every composite expression in the programme is **affine, convex, or concave**.
- The entire problem can be reduced to *affine–saturated affine (ASA) form*[1].

Formally, a DPP optimization problem has the structure

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \;& f_0(x, \theta) \\ \text{s.t.} \;& f_i(x, \theta) \;\le\; \tilde{f}_i(x, \theta), \quad i = 1, \dots, m_1, \\ & g_i(x, \theta) \;=\; \tilde{g}_i(x, \theta), \quad i = 1, \dots, m_2, \end{aligned} \tag{3}$$

where x is the *variable* and $\theta \in \mathbb{R}^p$ collects *parameters*. The functions satisfy: f_i convex, $\tilde{f}_i$ concave, $g_i, \tilde{g}_i$ affine, and every expression tree is constructed using DPP rules.

Parameter–affine, parameter-free, variable-free. An expression is *parameter-affine* if it contains no variables and depends affinely on θ; it is *parameter-free* if it contains no parameters, and *variable-free* if it contains no variables.

Relation to DCP. Every DPP programme is also DCP, but the converse is not true. DPP imposes two extra restrictions:

(1) *Curvature tagging*: in DCP all parameters are treated as constants; under DPP, parameters are tagged as *affine*, just like variables.

(2) *Product rule*: the product atom $\phi_{\text{prod}}(x, y) = xy$ is affine in DCP *only* when one argument is *constant*; under DPP the product is affine when *either* argument is parameter-affine and the other is parameter-free.

These additional rules ensure that the symbolic structure of the cone programme does not change with θ, enabling a single KKT factorisation to be cached and reused during backpropagation in `CvxpyLayer`.

4.2 Original DR optimization

Let p denote any probability–mass function (PMF) in the probability simplex

$$\mathcal{Q} \;\triangleq\; \big\{\, p \in \mathbb{R}^T \;:\; p \ge 0,\; \mathbf{1}^\top p = 1 \big\}.$$

Then, the *deviation risk measure* $f_\varepsilon(z, p)$ associated with the set of outcomes $\varepsilon = (\varepsilon_1, \dots, \varepsilon_T)$, the portfolio z and PMF p is defined by

$$f_\varepsilon(z, p) \;\triangleq\; \min_c \sum_{j=1}^{T} p_j \, R\big(\varepsilon_j^\top z - c\big).$$

$f_\varepsilon(z, p)$ enjoys the following properties:

1. For any fixed $p \in \mathcal{Q}$, $f_\varepsilon(\,\cdot\,; p) : \mathcal{Z} \to \mathbb{R}_+$ is convex.
2. $f_\varepsilon(z, p) \ge 0$ for all $z \in \mathcal{Z}$, $p \in \mathcal{Q}$.

[1] A cone-program canonical form in which parameters enter only as right-hand-side data and affine coefficients.

3. Shift-invariance with respect to ε.

4. Symmetry with respect to ε.

Next, define the *ambiguity set* for the distribution p as a ϕ-divergence ball centered at the nominal PMF q:

$$\mathcal{P}(\delta) \triangleq \{\, p \in \mathcal{Q} \,:\, I_\phi(p, q) \le \delta \}.$$

where the size parameter $\delta > 0$ bounds the permissible distance between p and q.

The distributionally-robust decision layer then chooses the optimal portfolio $z_t^\star$ by solving the minimax problem

$$z_t^\star \;=\; \arg\min_{z \in \mathcal{Z}} \; \max_{p \in \mathcal{P}(\delta)} f_\varepsilon(z, p) \;-\; \gamma\, \hat{y}_t^\top z.$$

Using convex duality (see Appendix C of the paper), the inner max can be re-expressed, yielding the equivalent *convex* minimization problem

$$z_t^\star \;=\; \arg\min_{z \in \mathcal{Z},\, \lambda \ge 0,\, \xi,\, c} \; f_\varepsilon^\delta(z, c, \lambda, \xi) \;-\; \gamma\, \hat{y}_t^\top z,$$

where

$$f_\varepsilon^\delta(z, c, \lambda, \xi) \;\triangleq\; \xi \;+\; \delta\, \lambda \;+\; \frac{\lambda}{T} \sum_{j=1}^{T} \phi^\star\Big(\frac{R\big(\varepsilon_j^\top z - c\big) - \xi}{\lambda} \Big),$$

and $\phi^\star(\cdot)$ is the convex conjugate of the ϕ-divergence generating the ambiguity set.

Finally, the task loss guiding end-to-end training is defined as the out-of-sample Sharpe ratio over the next v periods:

$$\mathcal{L}_{\mathrm{SR}}\big(z_t^\star,\, \{y_j\}_{j=t}^{t+v}\big) \;\triangleq\; -\,\frac{\mathrm{mean}\big(\{y_j^\top z_t^\star\}_{j=t}^{t+v}\big)}{\mathrm{std}\big(\{y_j^\top z_t^\star\}_{j=t}^{t+v}\big)}.$$

Any differentiable performance measure can substitute $\mathcal{L}_{\mathrm{SR}}$ in the training loop.

4.3 Markowitz-Family Objectives

Harry Markowitz introduced portfolio selection as the trade-off between *expected return* $\mu^\top w$ and *variance* $w^\top \Sigma w$ [Markowitz, 1952]. The efficient frontier is obtained by solving

$$\min_w \; -\mu^\top w + \tfrac{\lambda}{2}\, w^\top \Sigma w, \quad \mathbf{1}^\top w = 1,\; w \ge 0,$$

where $\lambda > 0$ controls risk aversion.

4.3.1 Classical Markowitz: Mean–Variance

The classical objective balances expected return against variance,

$$\max_z \big(\hat{y}^\top z - \tfrac{\lambda}{2} z^\top \Sigma z \big)$$

Algorithm 1 Original Mean–Variance Objective

Require: expected returns $\hat{y}$, covariance Σ, risk-aversion λ
Ensure: optimal weights $z^\star$ (solution of a single CVXPY problem)
1: risk $\leftarrow z^\top \Sigma z$ ▷ implemented as `cp.quad_form`
2: objective $\leftarrow -\hat{y}^\top z \;+\; \lambda \times$ risk
3: $z^\star \leftarrow$ CVXPY_Solve(objective, $\sum_i z_i = 1,\; z \ge 0$)
4: **return** $z^\star$

the quadratic term $z^\top \Sigma z$ violates DPP because the covariance matrix Σ is treated as a *parameter* and appears inside a non-affine expression. We restore DPP compliance by factorizing $\Sigma = LL^\top$ (Cholesky or eigen-decomposition) once at the *numeric* level and exposing L as a *constant*. The risk term becomes

$$z^\top \Sigma z \;=\; \|L^\top z\|_2^2 \;=\; \text{sum\_squares}\big(L^\top z\big),$$

which is convex-quadratic with constant coefficients and therefore DPP-valid. In CVXPY:

Algorithm 2 DPP-Compliant Mean–Variance Layer

Require: expected returns $\hat{y}$, covariance Σ, risk-aversion λ
Ensure: optimal weights $z^\star$
$L \leftarrow \text{Cholesky}(\Sigma)$
$risk \leftarrow \|L^\top z\|_2^2$ ▷ equivalent to $z^\top \Sigma z$
$objective \leftarrow -\hat{y}^\top z \;+\; \lambda \times risk$
$z^\star \leftarrow \textsc{CvxpyLayerSolve}\Big(objective,\ \sum_i z_i = 1,\ z \geq 0\Big)$
return $z^\star$

4.3.2 Max Sharpe

Maximum Sharpe Ratio (Tangency). The "market portfolio" maximizes $\big(\mu^\top w - r_f\big)/\sqrt{w^\top \Sigma w}$.

Directly maximizing the Sharpe ratio

$$\mathcal{S}(z) \;=\; \frac{(\hat{y} - r_f \mathbf{1})^\top z}{\sqrt{z^\top \Sigma z}}$$

is *non-convex* and therefore violates DCP/DPP. Instead, we approximate the tangency portfolio by searching along the efficient frontier:

1. Solve the **mean–variance problem** $\min_z \; -\hat{y}^\top z + \lambda\, z^\top \Sigma z$ for a grid of risk-aversion parameters $\lambda \in \Lambda = \{10^{-2}, 10^{-1}, 1, 10, 10^2\}$.
2. Compute the out-of-sample Sharpe ratio $\mathcal{S}_\lambda$ of each solution.
3. Select the weight vector with the highest $\mathcal{S}_\lambda$.[2]

The same method tackling the quadratic term is applied as well.

Because each mean–variance sub-problem is DPP-compliant (see previous section), the entire procedure stays within the CVXPY Layer framework.

Algorithm 3 MaxSharpeRatio via Efficient-Frontier Search

Require: expected returns $\hat{y}$, covariance Σ, risk-free rate r_f
Require: grid of risk aversion values $\Lambda = \{0.01, 0.1, 1, 10, 100\}$
Ensure: portfolio weights $w^\star$ with highest Sharpe ratio
$bestS \leftarrow -\infty, \quad w^\star \leftarrow \varnothing$
for each $\lambda \in \Lambda$ **do**
  $w \leftarrow \textsc{SolveMarkowitz}(\hat{y}, \Sigma, \lambda)$
  $return \leftarrow \hat{y}^\top w$ ▷ expected portfolio return
  $risk \leftarrow \sqrt{w^\top \Sigma\, w}$
  $sharpe \leftarrow (return - r_f)/risk$
  if $sharpe > bestS$ **then**
    $bestS \leftarrow sharpe; \quad w^\star \leftarrow w$
return $w^\star$

[2]This step is technically a discrete arg max and thus non-differentiable; in practice the grid is coarse and changed only at validation time, so it does not interrupt gradient flow during training.

4.4 CVaR Variates Objectives

Conditional Value at Risk (CVaR), also known as Expected Shortfall, represents a significant advancement in portfolio risk management by focusing on the severity of potential losses in extreme scenarios. Unlike traditional variance-based risk measures which treat upside and downside deviations symmetrically, CVaR specifically targets the left tail of the return distribution where the most significant losses occur.

For a portfolio with random return X and a confidence level α (typically 95% or 99%), CVaR is defined as the expected loss given that the loss exceeds the Value at Risk (VaR) threshold:

$$\text{CVaR}_\alpha(X) = \mathbb{E}[X | X \leq \text{VaR}_\alpha(X)]$$

where $\text{VaR}_\alpha(X)$ is the α-quantile of the distribution of X.

4.4.1 CVaR's Natural Compatibility with DPP

Unlike quadratic risk measures that require complex transformations to achieve DPP compliance, CVaR optimization enjoys a natural advantage. This compatibility stems from the groundbreaking work of Rockafellar and Uryasev [Rockafellar and Uryasev, 2000], who reformulated CVaR calculation as a convex optimization problem.

The Rockafellar-Uryasev formulation introduces:

1. A scalar variable ν representing Value at Risk (VaR)
2. A set of auxiliary variables s representing excess losses beyond VaR
3. Linear constraints linking these variables to portfolio returns

This formulation expresses CVaR as:

$$\text{CVaR}_\alpha(X) = \nu + \frac{1}{(1-\alpha)n} \sum_{i=1}^{n} s_i$$

where n is the number of scenarios, and s_i are the auxiliary variables representing excess losses.

The critical insight that makes CVaR naturally DPP-compliant is that all parameters (specifically the returns matrix) appear linearly in the constraints:

$$s_i \geq -R_i z - \nu$$

Where R_i is the i-th row of the returns matrix R, and z is the vector of portfolio weights.

This linear relationship between parameters and variables satisfies the core requirement of DPP: that parameters must appear linearly in expressions.

4.4.2 Minimum CVaR Portfolio

While minimum variance portfolios may still expose investors to significant tail risk due to non-normal return distributions, Minimum CVaR portfolio focuses exclusively on minimizing tail risk, disregarding expected returns entirely. This approach is particularly valuable for risk-averse investors or during periods of market turbulence when capital preservation becomes the primary objective. By minimizing the expected loss in the worst scenarios, the Minimum CVaR portfolio provides robust downside protection against extreme market events.

Algorithm 4 DPP-Compliant Minimum CVaR Portfolio

Require: returns matrix $R \in \mathbb{R}^{n_{obs} \times n_y}$, confidence level α
Ensure: optimal weights $z^\star$
1: **Variables:** $z \in \mathbb{R}^{n_y}$ (weights), $\nu \in \mathbb{R}$ (VaR), $s \in \mathbb{R}^{n_{obs}}$ (auxiliary)
2: **Constraints:** $\sum_{i=1}^{n_y} z_i = 1$, $z \geq 0$, $s \geq 0$, $s \geq -Rz - \nu \mathbf{1}$
3: $cvar \leftarrow \nu + \frac{1}{(1-\alpha)n_{obs}} \sum_{i=1}^{n_{obs}} s_i$ ▷ Rockafellar-Uryasev formulation
4: $objective \leftarrow cvar$ ▷ Minimize CVaR
5: $z^\star \leftarrow \textsc{CvxpyLayerSolve}\big(objective, constraints, parameters = [R]\big)$
6: **return** $z^\star$

4.4.3 Mean-CVaR Portfolio

The Mean-CVaR portfolio represents a balanced approach that considers both expected returns and tail risk. Analogous to the classical Markowitz mean-variance framework, it replaces variance with CVaR as the risk measure, offering a more nuanced approach to risk management that specifically targets downside risk.

This formulation includes a risk aversion parameter λ that controls the trade-off between pursuing higher expected returns and limiting potential losses in adverse scenarios. Higher values of λ produce more conservative portfolios with stronger downside protection, while lower values emphasize expected return maximization at the cost of potentially larger tail risk.

Mean-CVaR optimization is particularly valuable in asymmetric markets where downside risk does not mirror upside potential, and where returns exhibit significant skewness or kurtosis. By focusing on the distribution tail rather than symmetric deviations around the mean, Mean-CVaR portfolios can provide more appropriate risk management for real-world financial markets.

Algorithm 5 DPP-Compliant Mean-CVaR Portfolio

Require: returns matrix $R \in \mathbb{R}^{n_{obs} \times n_y}$, expected returns $\hat{y} \in \mathbb{R}^{n_y}$, risk aversion λ, confidence level α
Ensure: optimal weights $z^\star$
1: **Variables:** $z \in \mathbb{R}^{n_y}$ (weights), $\nu \in \mathbb{R}$ (VaR), $s \in \mathbb{R}^{n_{obs}}$ (auxiliary)
2: **Constraints:** $\sum_{i=1}^{n_y} z_i = 1$, $z \geq 0$, $s \geq 0$, $s \geq -Rz - \nu \mathbf{1}$
3: $cvar \leftarrow \nu + \frac{1}{(1-\alpha)n_{obs}} \sum_{i=1}^{n_{obs}} s_i$ ▷ Rockafellar-Uryasev formulation
4: $objective \leftarrow -\hat{y}^\top z + \lambda \times cvar$ ▷ Return-risk trade-off
5: $z^\star \leftarrow \textsc{CvxpyLayerSolve}\big(objective, constraints, parameters = [R, \hat{y}]\big)$
6: **return** $z^\star$

4.5 Risk-Only

We focus on a class of risk-based portfolio optimization strategies that do not rely on return forecasts. Traditional mean-variance models, such as Markowitz's framework, are highly sensitive to expected return estimates, which often results in poor out-of-sample performance. Although end-to-end learning can improve predictive models by aligning them with optimization objectives, the inherent instability of return predictions still undermines decision quality. Recent research suggests that risk prediction tends to be more robust than return prediction. Therefore, we incorporate risk-only portfolio optimization methods that avoid return estimation entirely. Specifically, we consider three approaches: minimum variance, risk budgeting, and risk parity.

4.5.1 Minimum Variance

The minimum variance portfolio aims to achieve the lowest possible portfolio risk given a set of assets, without incorporating expected returns. The optimization problem is defined as:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & x^T \Sigma x \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_i = 1 \\ & x \geq 0. \end{aligned} \tag{2}$$

where Σ is the asset return covariance matrix. Although this formulation does not require a predictive model by nature, we integrate a learned covariance matrix from the neural network to replace the sample estimate, which is often noisy and unstable. This integration improves performance and compatibility with end-to-end training frameworks.

4.5.2 Risk Budgeting

In risk budgeting, total portfolio risk is decomposed into the risk contributions of individual assets. The risk contribution of asset i is defined as by the product of each asset's exposure to the portfolio by its marginal risk:

$$\mathrm{RC}_i(x_1, \ldots, x_n) = x_i \cdot \frac{\partial \mathcal{R}(x_1, \ldots, x_n)}{\partial x_i}$$

where $\mathcal{R}(x)$ is the portfolio risk measure:

$$\mathcal{R}(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i \cdot \frac{\partial \mathcal{R}(x_1, \ldots, x_n)}{\partial x_i}$$

Given a set of target risk budgets $\{b_1, \ldots, b_n\}$ with $\sum b_i = 1$, the objective is to align risk contributions with the target budgets. The standard optimization problem:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \left(\mathrm{RC}_i(x_1, \ldots, x_n) - b_i\right)^2 \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_i = 1, \\ & \sum_{i=1}^{n} b_i = 1, \\ & x \geq 0, \\ & b \geq 0. \end{aligned} \tag{3}$$

In our end-to-end learning framework, the neural network predicts the risk budget, which is then embedded into the optimization problem to derive the optimal asset allocation. We use portfolio volatility as the risk measure throughout this paper.

$$\begin{aligned} \mathcal{R}(x_1, \ldots, x_n) &= \sigma(x) \\ &= \sqrt{x^T \Sigma x} \end{aligned}$$

It comes that the marginal risk and the risk contribution of the i-th asset are given by:

$$\frac{\partial \mathcal{R}(x)}{\partial x} = \frac{\Sigma x}{\sqrt{x^T \Sigma x}}$$
$$\frac{\partial \mathcal{R}(x)}{\partial x_i} = \frac{(\Sigma x)_i}{\sqrt{x^T \Sigma x}}$$
$$\mathrm{RC}_i = x_i \frac{(\Sigma x)_i}{\sqrt{x^T \Sigma x}}$$

The risk budgeting portfolio optimization problem rewritten from problem 3:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \left(x_i \frac{(\Sigma x)_i}{\sqrt{x^T \Sigma x}} - b_i \right)^2 \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_i = 1, \\ & \sum_{i=1}^{n} b_i = 1, \\ & x \geq 0 \\ & b \geq 0. \end{aligned}$$

is non-convex and not compatible with differentiable convex programming layers `CvxpyLayer`. Instead, we adopt the convex approximation proposed by [Bruder and Roncalli, 2012]:

$$\begin{aligned} \underset{y}{\text{minimize}} \quad & \sqrt{y^T \Sigma y} \\ \text{s.t.} \quad & \sum_{i=1}^{n} b_i \ln(y_i) \geq c, \\ & y \geq 0. \end{aligned} \tag{4}$$

where c is a positive constant. To ensure the predicted risk budget sums to one, we apply a softmax function to the neural network output. The resulting allocation is normalized as $x_i^* = \frac{y_i^*}{\sum_{i=1}^{n} y_i^*}$.

A common concern with risk-budgeting portfolios is the tendency to assign excessively large weights to low-volatility assets, which can negatively impact performance. [Uysal et al., 2021] address this issue by introducing an asset selection mechanism. We incorporate a regularization term into the objective. The modified optimization problem is:

$$\begin{aligned} \underset{y}{\text{minimize}} \quad & \sqrt{y^T \Sigma y} - \gamma \sum_{i=1}^{n} y_i \log y_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} b_i \ln(y_i) \geq c, \\ & y \geq 0. \end{aligned} \tag{5}$$

4.5.3 Risk Parity

The risk parity portfolio, also known as the equal risk contribution portfolio, is a special case of risk budgeting where $\mathrm{RC}_i = \mathrm{RC}_j$ for all i, j. Since no return or explicit budget prediction is needed, we again use a learned covariance matrix Σ predicted by the neural network. To ensure that the risk parity optimization problem is convex, we inherit problem (5).

We implement an iterative procedure to approximate risk parity ([Maillard et al., 2008]):

Algorithm 6 Risk Parity Portfolio

initialization: Set a uniform risk budget $b = \frac{1}{n}\mathbf{1}$
for each iteration **do**
    Use Σ predicted by neural network and the latest risk budget to find optimal asset allocation y^* by solving Problem (5)
    Compute and normalize risk contributions RC
    Check how even the RC is: $\texttt{std}(RC)$
    Update risk budget: $b \leftarrow RC$
    if $\texttt{std}(RC) <$ threshold **then**
        break
With even RC and predicted Σ, find the optimal asset allocation for risk parity portfolio by solving Problem (5)

5 Experiments

5.1 Results for Original DR

Data Description

The researchers conducted several experiments in the original distribution robust paper to test different model setups. The experiment utilized historical U.S. stock market data spanning January 2000 to October 2021. The dataset contains weekly returns of 20 S&P 500 stocks and eight financial factors (e.g., market capitalization, momentum, value) sourced from Kenneth French's database. The training period (2000–2013) included 60% of the data, and the rest 40% (2013–2021) is the out-of-sample test set. The task loss function combined the Sharpe ratio (evaluating risk-adjusted returns) and mean squared error (MSE) to balance financial performance and prediction accuracy.

Models Compared

- **Equal Weight (EW)**: A baseline strategy assigning equal weights to all assets, promoting diversification without optimization.
- **Predict-then-Optimize (PO)**: A two-stage approach with a linear prediction layer and fixed risk parameter (γ), optimizing nominal portfolio variance.
- **Base**: An end-to-end system training a linear predictor but excluding risk measures, maximizing expected returns alone.
- **Nominal**: An end-to-end system incorporating a sample-based risk measure (variance) with learnable γ.
- **Distributionally Robust (DR)**: The proposed system, integrating a distributionally robust optimization (DRO) layer with learnable γ and robustness parameter (δ), penalizing worst-case risk over an ambiguity set.

Performance Analysis and Model Robustness

The results, as shown in Figures 1a and 1b, demonstrate advantages of the DR system. DR has the highest Sharpe ratio throughout the years. The Equal Weight strategy comes in second place during most of the years. In 2021, DR attained the highest Sharpe ratio at 1.30, compared

to 1.24 for the Nominal and only 0.64 for the Base model, proving its superior risk-adjusted returns. The PO system, constrained by a fixed γ, has a Sharpe ratio of 0.88, while the Equal Weights strategy has a Sharpe ratio of 1.05 due to its passive diversification.

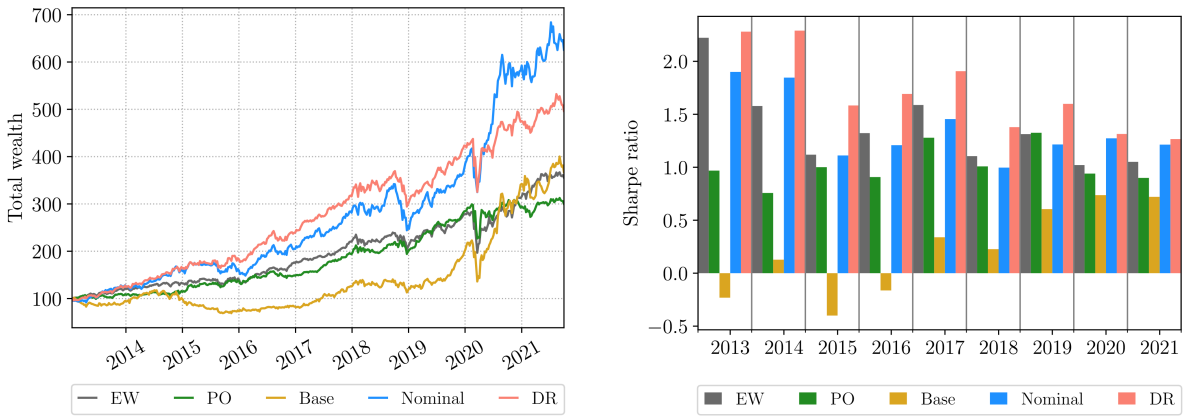
DR employed end-to-end learning by jointly optimizing the prediction and decision layers. This alignment enabled it to learn representations that directly served portfolio objectives rather than merely minimizing prediction error. The distributionally robust layer allowed DR to optimize performance over a set of probability measures, which reduces sensitivity to data noise.

Moreover, the learnable risk parameter γ and robustness parameter δ allowed DR to dynamically adjust the risk-return trade-off. The original paper presented a performance comparison between different setups of the DR models. And the result shows that the DR model trained with both learnable γ and δ outperformed the original DR model. This adaptive calibration ensured the model was neither overly conservative nor excessively aggressive.

In conclusion, the DR system's integration of end-to-end learning with distributional robustness provided an effective framework for handling financial uncertainty. This is especially important in volatile financial environments, where overlapping asset returns and model noise can make traditional approaches less effective.

From both Figures 1a and 1b, the Base model appears to perform the worst among all methods. We argue that, even without the distributional robust mechanism, improvements can be achieved by modifying the Base model—specifically through changes in the optimization layer and the prediction layer. The following sections present an analysis of our proposed models and their corresponding performance.

Figures



(a) Total Wealth Evolution

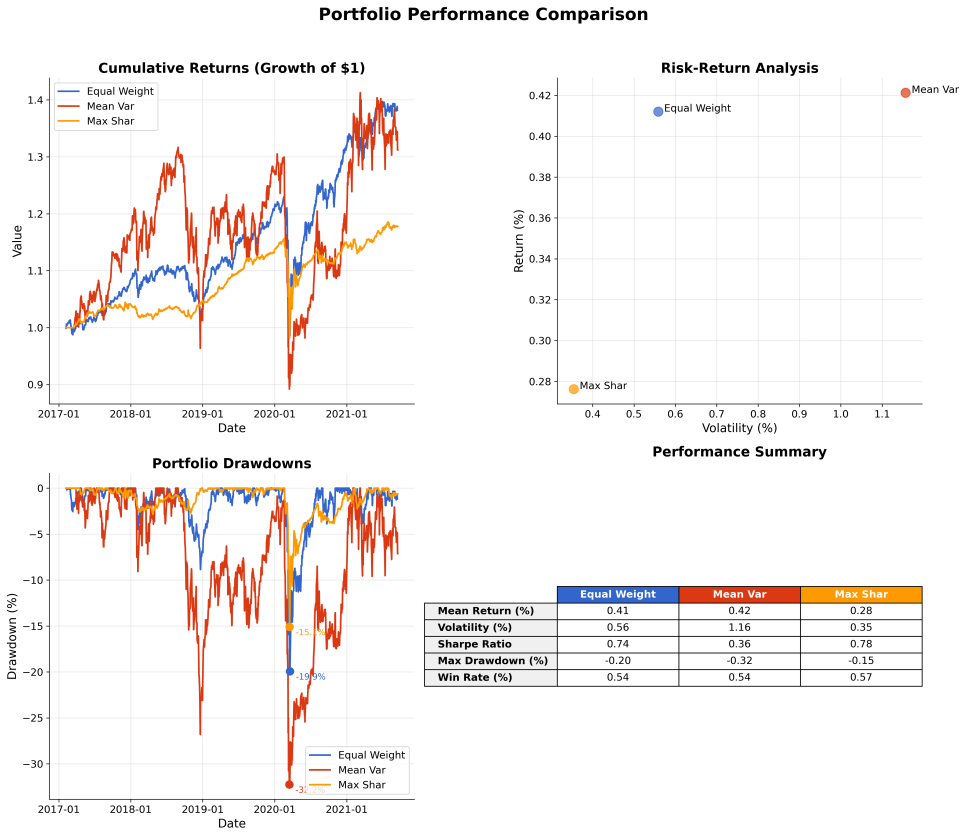
(b) Sharpe Ratios of Different Models

Figure 1: Performance comparison of the five portfolio construction systems

5.2 Results for Mean–Variance and Max Sharpe

Starting from this section, we use a different dataset consisting of several ETFs—such as VTI, IWM, and AGG—covering the period from 2010 to 2021. The change in tickers from the original study is motivated by the implementation of the risk budgeting model, which is theoretically more effective when applied to broad, well diversified instruments such as ETFs. We adopt a 60:40 split for training and testing. As input features, we include lagged returns up to 5 days, as well as rolling mean and variance calculated over the past 10, 20, and 30 days. Model selection and tuning are conducted using 4-fold cross-validation. During the testing phase, portfolio

weights are generated once every 14 trading days, on the first day of each evaluation window, and held constant for the following 14-day period.



	Equal Weight	Mean Var	Max Shar
Mean Return (%)	0.41	0.42	0.28
Volatility (%)	0.56	1.16	0.35
Sharpe Ratio	0.74	0.36	0.78
Max Drawdown (%)	-0.20	-0.32	-0.15
Win Rate (%)	0.54	0.54	0.57

Figure 2: Markowitz Portfolio Performance

In this experiment, we compare the baseline (Equal Weights, EW) with two optimized approaches: end-to-end Mean-Variance (E2E MV) and end-to-end Mean-Variance optimization with maximum Sharpe ratio, and the results are in Figure 2.

- **Equal Weights (EW)**: Allocates an identical proportion of capital to each asset, regardless of individual asset risk or expected return. Serves as a simple baseline.
- **End-to-End Mean-Variance (E2E MV)**: Learns portfolio weights by minimizing portfolio variance directly through a data-driven, model-integrated approach. The prediction layer outputs asset return forecasts and feed them into the mean variance optimization layer.
- **Mean-Variance with Max Sharpe Ratio (MV-MaxSharpe)**: Very similar to the above, but optimize the weights to obtain maximum Sharpe ratio.

The experimental results reveal different characteristics of the three portfolio optimization approaches. The Equal Weight baseline demonstrates robust performance with a Sharpe ratio of 0.74, benefiting from its low volatility (0.56%) and moderate return (0.41%). This simple strategy outperforms both machine learning approaches in absolute risk-adjusted returns. The may suggest that in the tested market regime, the costs of estimation errors in predictive models may outweigh their theoretical advantages.

The Mean Var model, employing end-to-end return predictions for mean-variance optimization, achieves the highest raw returns (0.42%) but suffers from high volatility (1.16%), resulting in the poorest Sharpe ratio (0.36). This indicates that while the prediction layer successfully

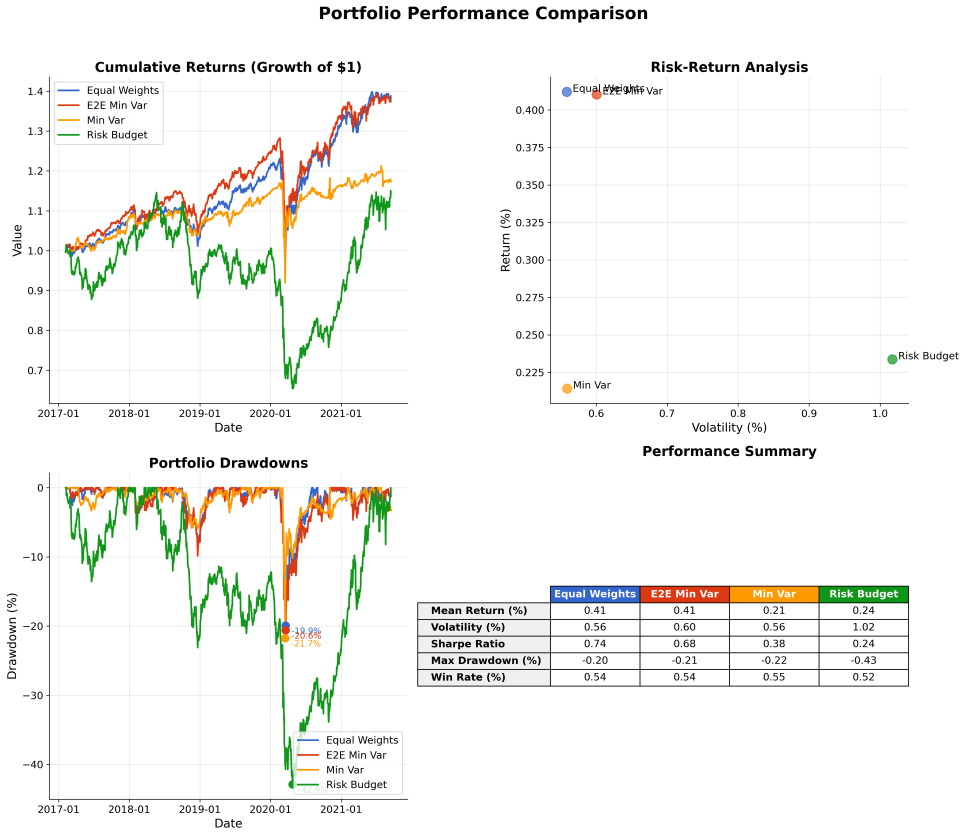
identifies higher-return assets, it fails to manage risk effectively, which is a critical limitation of classical mean-variance frameworks when applied to noisy return predictions.

In contrast, the Max Sharpe strategy demonstrates the effectiveness of direct Sharpe ratio maximization, achieving the highest risk-adjusted performance (Sharpe ratio 0.78) despite lower absolute returns (0.28%). By explicitly optimizing the risk-return trade-off through its optimization layer, the model maintains remarkably low volatility (0.35%) and minimal drawdown (-0.15%), outperforming both competitors in downside protection. This suggests that shaping the asset return prediction to align with the optimization target (Sharpe ratio) creates more effective portfolio constraints than standalone return prediction.

The nearly identical win rates across models (54-57%) imply comparable short-term prediction accuracy, emphasizing that performance differences are primarily from risk management rather than return prediction alone. The Max Sharpe model's superior drawdown characteristics (-0.15% vs -0.32% for Mean Var) highlight the value of integrated risk optimization frameworks when deploying machine learning predictions in portfolio construction.

The Mean Var model's poor risk-adjusted returns despite superior return prediction (0.42% vs 0.28%) demonstrate that unconstrained optimization of predicted returns can be counterproductive. Conversely, the Max Sharpe framework's performance validates the importance of co-optimizing prediction and risk management objectives in end-to-end architectures.

5.3 Results for Risk Models



	Equal Weights	E2E Min Var	Min Var	Risk Budget
Mean Return (%)	0.41	0.41	0.21	0.24
Volatility (%)	0.56	0.60	0.56	1.02
Sharpe Ratio	0.74	0.68	0.38	0.24
Max Drawdown (%)	-0.20	-0.21	-0.22	-0.43
Win Rate (%)	0.54	0.54	0.55	0.52

Figure 3: Risk Related Portfolio Performance

In this experiment, we compare the baseline(EW) with three risk related portfolio optimization strategies, and the results are in Figure 3.

- **E2E Min Var (End-to-End Minimum Variance)**: Minimum variance optimization layer takes the covariance matrix as input from the prediction layer.

- **Min Var (Historical Minimum Variance)**: Traditional minimum variance optimization layer that computes weights using the historical covariance matrix to minimize portfolio variance
- **Risk Budget**: Allocates capital so that each asset contributes a share, which is the output of the prediction layer) to the portfolio's total risk, balancing diversification with risk control.

The traditional Minimum Variance strategy underperforms significantly, with both the lowest mean return (0.21%) and second-worst Sharpe ratio (0.38). This poor performance underscores the limitations of static historical covariance estimates in dynamic markets. Surprisingly, the Risk Budget approach demonstrates severe shortcomings, achieving only 0.24% mean return while suffering from dramatically higher volatility (1.02%), resulting in the worst Sharpe ratio (0.24). This suggests either insufficient robustness in its neural network-based risk contribution predictions or a tendency to over-concentrate risk contribution during market stress periods, as evidenced by its substantial maximum drawdown (-0.43% vs -0.20% for Equal Weights).

The near-identical win rates (52-55%) across all strategies imply no consistent short-term predictive advantage. The results collectively advocate for cautious adoption of machine learning approaches in portfolio optimization. While the end-to-end framework shows potential through its matching of baseline returns, its failure to improve risk-adjusted returns suggests current implementations may not justify their computational advantages compared to simple theories. The severe under-performance of both covariance-based optimization strategies (Min Var and Risk Budget) relative to Equal Weights raises fundamental questions about the viability of traditional mean-variance frameworks in low-signal environments.

In addition, during the training period we encountered warnings that the optimization problem was solved but potentially inaccurate. This could be another possible explanation for such under-performance.

5.4 Aggregated Analysis

	EW	e2e Min V	Min V	RB	Mean V	Max Shar
Mean Return (%)	0.41	0.41	0.21	0.24	0.42	0.28
Volatility (%)	0.56	0.60	0.56	1.02	1.16	0.35
Sharpe Ratio	0.74	0.68	0.38	0.24	0.36	0.78
Max Drawdown (%)	-0.20	-0.21	-0.22	-0.43	-0.32	-0.15
Win Rate (%)	0.54	0.54	0.55	0.52	0.54	0.57

Figure 4: Aggregated Portfolio Performance

In the aggregated table of six portfolio optimization strategies in Figure 4, the underperformance of our most models reflect potential flaws in our construction and implementation. Firstly, the lack of win rate advantages compared to the Equal Weight suggest our prediction layer's inability of capturing underlying behaviors. Secondly, due to the significant training cost, we stopped training our model in the test phase whereas the original paper continue trains the model in the test phase with trained hyper-parameters from the training period. The continue training allows the model to learn the most up-to-date market characteristics. Thirdly, some models like end-to-end minimum variance and end-to-end risk budget raised warnings about solving inaccurately after trying different solvers. This advocates for future modifications on the optimization layers. Despite the potential limitations, we still observe several performance highlights. The end-to-end Max Sharpe model yields the highest Sharpe ratio among all with impressive drawdown protection. The end-to-end Minimum Variance closely matches the Equal Weight result. Its improvement on the traditional Minimum Variance model suggests the importance of deploying such end-to-end construction with a machine learning prediction layer.

6 Conclusions and Future Work

We began by exploring the *Distributionally Robust End-to-End* (DR-E2E) layer. Among several models compared, the DR-E2E model—with learnable risk (γ) and robustness (δ) parameters—achieved the highest Sharpe ratios, outperforming alternatives like Equal Weight, Base, and Nominal. Its success is attributed to end-to-end learning and robustness to data uncertainty. The Base model performed the worst, but we argued that even without DR mechanisms, performance can improve by refining its architecture. We deployed six models: Equal Weights, E2E Min Variance, Min Variance, Risk Budget, Mean Variance, and Max Sharpe.

6.1 Key findings

- **Equal Weight as a Robust Baseline:** The Equal Weight (EW) strategy consistently performed well, achieving a Sharpe ratio of 0.74. Its simplicity and low volatility allowed it to outperform more complex models in risk-adjusted returns.
- **Max Sharpe Optimization is Most Effective:** The end-to-end Max Sharpe model delivered the best risk-adjusted performance with a Sharpe ratio of 0.78. Despite having lower absolute returns, it maintained the lowest volatility and drawdown, demonstrating the value of directly optimizing for the Sharpe ratio.
- **Prediction Alone is Insufficient:** The E2E Mean-Variance model yields higher returns but fails to manage risk, leading to the worst Sharpe ratio. This highlights the limitations of optimizing based solely on return predictions without integrating effective risk control.
- **Traditional and Risk-Based Models Underperformed:** Both the historical Minimum Variance and Risk Budgeting strategies produced weak Sharpe ratios and higher drawdowns. Their underperformance suggests limitations in using static covariance estimates and challenges in reliable risk attribution from machine learning models.

6.2 Limitations

- The discretized approach to the tangency portfolio is only *piece-wise* differentiable, potentially hindering gradient-based hyper-parameter.
- The model is yet to include transaction cost and turnover. optimization.
- We stopped training our model in the test phase, which can be detrimental to models' performance, especially with training periods covering years of daily data.
- Some models, like end-to-end minimum variance and end-to-end risk budget, raised warnings about solving inaccurately after trying different solvers.

6.3 Next Steps

1. More exploration on the distributionally robust setting:

   The original setting for the DR layer is to use nominal distribution as the "center" of the ambiguity set, however, nominal distribution may not be a very suitable assumption for prediction outcomes and error. Exploration on more distribution could be a promising lead.

2. More delicate design of Prediction Layer:

   The characteristic of end-to-end model comes from the fact that prediction layer is not only learning from the data, but also working for optimization of the decision process. Therefore, how do me design the prediction layer so that it could be more suitable and obedient for the final goal of portfolio optimization becomes another interesting point.

3. Deploy the models on different datasets:

   Due to the implementation of risk budget strategy and consistency through testing, our testing data contains several ETFs, which may make it difficult to construct a portfolio with a high Sharpe ratio. In the future, having stocks in our portfolio may encourage some aggressive allocations.

4. Higher fold cross validation with more parameter tunings:

   Due to the computational cost, we were unable to explore different combinations of hyperparameters for our models. Doing so will surely increase our models' performance.

5. Include Transaction cost and Turnover:

   Our models' implementations are currently too naive without realistic considerations such as transaction costs and turnovers. These realistic frictions should be the focus for future modifications

References

[Agrawal et al., 2019] Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., and Kolter, J. Z. (2019). Differentiable convex optimization layers. *Advances in Neural Information Processing Systems*, 32.

[Bruder and Roncalli, 2012] Bruder, B. and Roncalli, T. (2012). Managing risk exposures using the risk budgeting approach. *Available at SSRN 2009778*.

[Costa and Iyengar, 2023] Costa, G. and Iyengar, G. N. (2023). Distributionally robust end-to-end portfolio construction. *Quantitative Finance*, 23(10):1465–1482.

[Donti et al., 2017] Donti, P., Amos, B., and Kolter, J. Z. (2017). Task-based end-to-end model learning in stochastic optimization. *Advances in Neural Information Processing Systems*, 30.

[Maillard et al., 2008] Maillard, S., Roncalli, T., and Te
ïletche, J. (2008). The properties of equally weighted risk contribution portfolios. Working paper.

[Markowitz, 1952] Markowitz, H. (1952). Portfolio selection. *The Journal of Finance*, 7(1):77–91.

[Rasekhschaffe and Jones, 2019] Rasekhschaffe, K. C. and Jones, R. C. (2019). Machine learning for stock selection. *Financial Analysts Journal*, 75(3):70–88.

[Rockafellar and Uryasev, 2000] Rockafellar, R. T. and Uryasev, S. (2000). Optimization of conditional value-at-risk. *Journal of Risk*, 2(3):21–41.

[Uysal et al., 2021] Uysal, A. S., Li, X., and Mulvey, J. M. (2021). End-to-end risk budgeting portfolio optimization with neural networks.

[Zhang et al., 2020] Zhang, Z., Zohren, S., and Roberts, S. (2020). Deep learning for portfolio optimization. *arXiv preprint arXiv:2005.13665*.