Black-Litterman End-to-End*

Jetlir Duraj

University of Pittsburgh

Chenyu Yu

Princeton University

August 13, 2023

Abstract

We offer a version of the Black-Litterman model where the view generation process and the portfolio allocation process are *jointly* optimized in an end-to-end fashion (BLEnd2End). The views in our framework are *market-wide* statements on the performance of classical investment strategies and are learned through deep learning methods from a set of aggregate features. For a dataset of 14 ETFs covering 9 industries, the SPY and the treasury yield curve, we use views based on momentum, reversal and volatility, and we generate them using past macroeconomic data from FRED. BLEnd2End significantly outperforms the mean-variance benchmarks based on PCA factor models on test set. Moreover, BLEnd2End performs better than several other classical strategies like fix-mix or risk-parity. The BLEnd2End sub-strategy focused on industry sectors alone, performs better than sector-momentum and sector-reversal. While the included empirical exercise is on a balanced dataset of asset returns, the framework allows for unbalanced datasets as well, without introducing look-ahead bias.

Keywords: Black-Litterman model, portfolio optimization, objective views, deep learning, Sharpe ratio

1 Introduction

The Black-Litterman model is a long-standing asset allocation technique used to combine the views of investors about the market with a market benchmark to generate an optimal portfolio. It was introduced in 1992 by Fischer Black and Robert Litterman in Black and Litterman [1992], as an enhancement to the traditional mean-variance optimization approach pioneered by Harry Markowitz Markowitz [1952]. The original motivation for the Black-Litterman model (henceforth BL model) was the extreme portfolio weights often produced by the traditional mean-variance analysis and the resulting deterioration of the performance in practice. Its advantages lie beyond a regularization of the mean-variance approach though, because the BL model offers a systematic way to incorporate subjective beliefs of an investor in portfolio construction. Other advantages are of a computational nature: its parametric standing assumption is that market returns are normally distributed. This allows the formulation of the whole procedure as a Bayesian optimization problem with ready-to-use formulas, as long as the investor provides their views as expected portfolio returns, together with a confidence level on the same views. In fact, as Kolm et al. [2020] describes, the BL model can be cast as a special case of a more general BL-Bayes approach where a Bayesian statistical model of returns is updated based on investor views, and the posterior return distribution is used for portfolio construction. Another practical computational advantage of

*Duraj: jed169@pitt.edu, corresponding author. Yu: chenyu@princeton.edu. This research was supported in part by the University of Pittsburgh Center for Research Computing, RRID:SCR022735, through the resources provided. Specifically, this work used the H2P cluster, which is supported by NSF award number OAC-2117681. Any opinions, findings and conclusion contained in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation.
Link to code for the project.

BL is that it acts as a regularizer to the sometimes ill-posed correlation matrix inversion problem faced in the usual mean-variance approach.

A drawback of the traditional BL model is that the origin of the views is exogenous/subjective. The model is silent as to where these should come from. More importantly, the model does not contain a good theoretical approach for picking the confidence level for the views, which in turn will influence the final portfolio weights. A deeper drawback in our view, that is not tied to the BL model alone, is the fact that the process of producing the view estimates (signals) for portfolio construction is performed separately from the optimization of the portfolio weights. More generally, classical optimization-based investment techniques usually comprise two separate steps. The first step is estimation/return forecast, and the second one is portfolio optimization. The statistical model used in the estimation/forecast step is typically independent of the optimization step, other than producing the parameters for it. But the two steps are supposed to serve one final goal: that of maximizing portfolio performance. Given the inherent uncertainty in statistical estimation, it is unclear whether solving the estimation problem first, without taking into account its interplay with the final goal of portfolio performance, cannot be improved upon by a carefully designed joint optimization procedure. The hope is that a joint optimization approach, where view generation/estimation and portfolio construction are optimized as parts of a single model would deliver an increased performance of the final portfolio, at lower computational costs. Recent papers indeed have showcased that better investment performance can be achieved by combining and jointly optimizing the estimation/forecast and the portfolio allocation step. See for example Uysal et al. [2021] which focuses on portfolio construction based on risk parity principles.

This paper offers an approach to portfolio construction based on the traditional BL model, with two major innovations that overcome the drawbacks mentioned above. First, the views are modeled as statements on expected returns of typical market-wide quantitative strategies like momentum, reversal or volatility. They are not subjective, but rather learned through a machine-learning algorithm using as features other relevant and readily available financial data. Second, the view generation process is jointly optimized in an end-to-end fashion together with the portfolio allocation process, within a single optimization pipeline guided by a classical mean-variance objective criterion. We show in an empirical investigation with ETF data where the views are learned through macroeconomic time series, that our approach is viable and performs significantly better than the benchmark model the BL uses, as well as other classical portfolio construction techniques.

In more detail, we keep the normality assumption of the BL model on returns, but use a neural network approach within the PyTorch optimization framework to calculate the views, perform the update of the benchmark based on the views, and solve for the optimal portfolio weights. Neural networks act here as semi-parametric estimators for the variables at stake: the views as well as the optimal portfolio weights. Within the PyTorch framework we build an end-to-end framework, which we call BLEnd2End, where the following steps are performed sequentially:

> Step 1: The views (statements on returns of certain 'classical' strategies) are generated through a trainable neural network from a set of features.
>
> Step 2: The views are used to produce a BL update of the benchmark. Our model architecture allows for both the traditional BL update based on the Bayes' rule, as well as other ways to take a projection of a multi-dimensional normal probability distribution (the views) on another one (the prior given by the benchmark). The alternatives to Bayes' rule we consider are minimizing one of the following probability distances: the Kullback-Leibler distance, the Jensen-Shannon divergence, or the Wasserstein-2-metric.
>
> Step 3: The BL update (a vector of expected returns of the asset universe, and a covariance) is used to produce portfolio weights. Our model architecture allows for the case where the weights are generated by a trainable neural network that takes the BL update as a feature, as well as the case where the weights are determined through a separate optimization problem (weights solver) for a *fixed* BL update.
>
> Step 4: Given the weights produced in Step 3, portfolio performance is calculated based on a mean-variance objective inclusive of trading costs. This performance is used to update the model of view generation in Step 1 through the backpropagation algorithm. Furthermore, for the case where the portfolio weights are a trainable neural network of the BL update, portfolio performance is again used to update the model of weights in Step 3.

The steps above are repeated until a model of view generation and/or portfolio allocation is learned from the data. This model is then tested on unseen data. The approach described in Steps 1-4 contains two sub-cases

that are fairly distinct from one another. In the first sub-case, there is one **joint** model of view generation and portfolio allocation that is updated during training based on its portfolio performance. The portfolio allocation here uses a neural network, and not a solver. In the second sub-case, the **disjoint** case, only the view model is updated via backpropagation of the portfolio performance, because the weights are calculated using a solver for a *fixed* BL update. Because of implementation details, it is a priori unclear which of the two sub-cases should perform better. Nevertheless, intuition suggests that the joint optimization approach should be more in-line with the final goal of portfolio performance, and hence outperform the disjoint optimization approach.

In our empirical exercise, we consider a universe of 14 ETFs comprising 9 industry ETFs, SPY that invests in S&P500, as well as 4 U.S. treasury ETFs that span the yield curve: ETFs investing in respectively 1-3 months T-bills, 1-3 year T-bills, 7-10 year T-bills, and 20+ year T-bills. We work with monthly data, whose predictability is notoriously low, so that the financial performance of any portfolio construction approach will mostly come from diversification effects. The BL benchmark model we consider is rolling PCA factor regressions with various numbers of factors. We pick as views for this dataset 7 long-short, zero cost portfolios comprising one volatility strategy calculated from daily returns, one momentum strategy, three reversal strategies, as well as exposure and alpha with respect to Fama-French 3-factor model Fama and French [1993]. Our ETF universe allows for investment to be industry-specific, it allows for hedging with the S&P500, as well as investing in separate components of the yield curve, all at the same time. Given this extensive coverage, we pick as natural view predictors the macroeconomic time series from FRED.[1] The BL approach, where a standard benchmark is infused with views that are learned from macroeconomic data seems suitable for such a universe of assets, and the set of views under consideration.

We show that the joint optimization approach is indeed optimal for the dataset we consider. In fact, both the second and third-best BLEnd2End architectures picked during validation are of the joint-optimization type, confirming thus the initial intuition. All else equal, the joint optimization approach is also less costly than the disjoint approach in terms of computation time. The final portfolio outperforms the benchmark based on PCA factors significantly in terms of standard financial ratios like Sharpe, Sortino and Calmar ratios. The main 'mechanical' reason for this is that the joint optimization delivers simultaneously less extreme expected estimates of returns and less extreme portfolio weights than the benchmark models. Furthermore, we also compare the BLEnd2End portfolio with classical strategies like fix-mix on equities and bonds, risk parity, as well as sector momentum and reversal. We show that it outperforms all of the above alternatives significantly.

The rest of the paper is organized as follows. Section 1.1 concludes the introduction and comments on related literature. Section 2 introduces the BL approach and defines the mean-variance objective criterion. Section 3 explains our approach in detail. Section 4 contains details of our empirical exercise, together with details of the machine learning implementation. Section 5 concludes. The appendix contains a few technical details related to the probability distances we use as alternatives to Bayes' rule, together with a few summary statistics of financial variables we use for the empirical exercise.

1.1 Relation to existing literature

There is a vast literature, both academic and industry-focused, that discusses different interpretations and variations of the traditional BL approach. Here we focus only on a small part of it, giving more weight to literature we drew inspiration from, or that follows an approach related to ours. The reader should consider the recent survey Kolm et al. [2020] for a more comprehensive literature overview of the BL model.

We are not aware of another paper that follows the single-pipeline philosophy from view generation to portfolio allocation for the BL model. Within the classical portfolio optimization framework, Brandt et al. [2009] is the first paper we know that considers an end-to-end approach, where the estimation step for returns and covariance is avoided and instead the weights of the portfolio are modeled directly as a function of asset features. Ao et al. [2019] give an unconstrained regression formulation for the traditional mean-variance problem that avoids the covariance estimation step, and instead focuses on estimating the maximal achievable Sharpe ratio. In our work, we do not avoid the estimation step for either covariance or expected returns, but rather embed their estimation into a larger optimization framework that is guided by a classical portfolio optimization objective.

More recently, a few papers have conducted analysis based on the end-to-end philosophy on various investment optimization set ups. We mention here Uysal et al. [2021] and Li et al. [2022] that focus on risk parity strategies, and Chen et al. [2023], Guijarro-Ordonez et al. [2021] that focus respectively on asset pricing and statistical

[1] See https://research.stlouisfed.org/econ/mccracken/fred-databases/ for more.

arbitrage for equities. Ideas of integrating statistical estimation and portfolio construction into one framework using advancements in machine learning are also discussed in recent research textbooks in the investment literature, see e.g. section 4 of chapter 12 in Dixon et al. [2020].

Kolm et al. [2020] discusses the Bayesian interpretation of the BL model and the potential of using machine learning methods to generate objective investment views. We draw on this paper for the exposition of the Bayesian approach to BL. Meucci et al. [2014] considers a generalization of the BL framework where the views may be non-linear, even non-parametric, and where they are projected to a benchmark using minimization of the Kullback-Leibler distance. Bertsimas et al. [2012] departs from the statistical interpretation of the traditional BL setup and considers instead an inverse-optimization interpretation of the general BL philosophy of merging investor views with a market benchmark. This enables a more systematic definition of views confidence by allowing extensions to more general risk notions such as coherent risk measures. While we remain in this paper within the statistical framework and keep the normality assumption, these two papers inspired us to look at alternatives to Bayes' rule that are reasonably implementable within our framework. Interpreting the Bayes' rule as the projection of one probability distribution into another, we consider in our paper alternative projection methods based on minimizing standard probability distances, e.g. the Jensen-Shannon divergence or the Wasserstein-2 metric.

Van Der Schans and Steehouwer [2017] considers a dynamical version of the BL model within the traditional setup, where views can be unconditional and conditional. In our case, the views are also conditional, as they are updated through time series features in a machine learning algorithm. Our focus is more on the benefits of the joint optimization approach to risk factor estimation and portfolio construction in a machine learning setup.

Creamer [2015] considers a BL setup where the views are generated through machine learning algorithms from various sets of features including news sentiment, analyst forecasts, and corporate social network indicators. Oprisor and Kwon [2020] uses the BL model to introduce views on regime-switching in the economy, into a multi-period portfolio optimization problem. They extract and predict regime-switching using a hidden Markov model. In this paper, we have not pursued a multi-period optimization extension, and we follow an end-to-end/single-pipeline approach focusing on multi-dimensional signals for view generation that we process within a neural network framework.

2 Preliminaries

Consider an asset universe comprised of n assets. For the exposition we assume that the asset universe does not change over time, i.e. to streamline the mathematical exposition we depict formulas that are based on the balanced dataset case. Our framework and coding platform allows for the case of unbalanced datasets though, and we discuss this in selected parts of the paper. In the conclusions, we give more details about the unbalanced dataset case without any additional work.

2.1 Mean-variance portfolio construction

Throughout this paper, we use the convex mean-variance objective function. Given an estimate of returns $\hat{\mu}_\tau$ at time τ and of the covariance $\hat{\Sigma}_\tau$, the portfolio optimization problem faced at time τ is given by

$$\max_w w'\hat{\mu_\tau} - \frac{\gamma^{ra}}{2} w'\hat{\Sigma}_\tau w - \gamma^{tr}\|w - w_{-1}\|_1, \tag{1}$$

where $\gamma^{ra} > 0$ is the risk-aversion parameter, $\gamma^{tr} > 0$ is the trade-aversion parameter, and w_{-1} are the current portfolio weights. The trading costs are modeled through the L^1-norm, hence we do not consider non-linear impact costs. This assumption is in-line with an agent that trades once a month, say at the beginning of the month, and whose trades are too small in size to affect the final monthly price of the assets it trades.

This objective is widely used in different forms in the portfolio optimization literature, see e.g. Boyd et al. [2017]. We use this objective criterion twofold: (1) for a given estimate $(\hat{\mu}, \hat{\Sigma})$ to find the optimal portfolio by solving (6) for w, and (2) given weights w, the *realized* returns μ and the covariance matrix Σ, to calculate portfolio performance during the training of the algorithms that we present below. We do not impose any short-selling constraints.

Leverage constraint. Financial returns data typically exhibit a low signal-to-noise ratio which makes generalization of models, especially those trained through machine learning models, particularly challenging. It is appropriate, in such context, to employ additional regularization steps that are based on domain knowledge.

One important economic metric in our context is the amount of leverage that is embedded in the portfolio. Without constraints, the portfolio weights can assume large positive and negative values. As leverage is directly tied to portfolio weights, we require a bound on a Euclidean norm for the weights of the portfolio for a fixed date.

There are two Euclidean norms one can possibly use: L^1-norm $\|\cdot\|_1$ and the L^2-norm $\|\cdot\|_2$. The L^1-norm relates directly to the amount of leverage of a portfolio. L^1-norm typically implies much more concentrated portfolios, with some larger weights and lots of smaller weights, L^2-norm typically implies more balanced portfolios with no assets dominating the portfolio. On the other hand, L^1-norm typically implies smaller trading activity focused on a small number of assets, than the L^2-norm. Hence, there is a trade-off between diversification and trading costs when choosing the Euclidean norm to bound the portfolio weights. Because of the norm inequalities, for n assets, there is the following bound between the two norms for the same set of weights w:

$$\|w\|_1 \leq \sqrt{n}\|w\|_2.$$

Hence, a bound of C on the L^2-norm of the portfolio weights, can translate to a varying, albeit bounded, amount of leverage of the portfolio on the date level. To allow for this additional flexibility in leverage choice we have focused in this paper on the L^2-norm. Hence, for all algorithms we consider below we require the portfolio weights constraint

$$\|w\|_2 \leq C.$$

From an implementation perspective, we ensure that the L^2-norm constraint is satisfied, by normalizing the produced weights from an optimal weights calculator as described This leads to a varying and effective soft bound on leverage for the portfolio of $\sqrt{n}C$.

2.2 The Bayesian approach and the original Black-Litterman model

Kolm et al. [2020] describes the Bayesian interpretation of the long-standing BL approach in detail. We use their generalized approach as a starting point and follow their description of the Bayesian BL model. The Bayesian BL approach consists of the following four parts.

1. **Parametric model for returns**. A parametric model for returns is given as a basis: $p(r|\theta)$.

   In the classical BL setup, we assume that $r \sim \mathcal{N}(\mu(\theta), \Sigma(\theta))$, i.e. we assume returns follow normal distributions.

2. **Prior/benchmark.** A prior distribution $\pi(\theta)$ for the parameter θ is given. This will in turn define through the conditional probability $p(r|\theta)$ a prior distribution for the returns.

   In the classical BL setup, we assume in the absence of investor views,

   $$\mu(\theta) \sim \mathcal{N}(\mu_b, C_b),$$

   with $C_b = \tau_b \Sigma_b$ a rescaling of the sample covariance matrix of the assets Σ_b. (μ_b, Σ_b) is a pair that satisfies a benchmark model $\mathcal{M}_b$.[2] $\tau_b > 0$ is an uncertainty parameter that scales with the amount of confidence we have in the estimation of the benchmark.

3. **Views.** A likelihood function is given $p(q \mid \theta)$ where $q \in \mathbb{R}^{V \times 1}$ represents the estimate of the view returns. Moreover, P is a matrix of dimension $V \times n$, called the view matrix, that can be used to calculate the view portfolio returns $P\mu$ for a given estimate μ of the asset returns.

   In the classical BL setup, we assume that the $p(q \mid \theta)$ satisfies

   $$P\mu(\theta) \sim \mathcal{N}(q, \Omega),$$

   i.e. $P\mu(\theta)$ is a normal distribution with mean given by the views q and covariance Ω that represents the confidence the investor has in the views.

[2] In the classical work Black and Litterman [1992] this is the CAPM model where expected returns satisfy $\mu_b = \gamma^{ra} \Sigma_b w^{mkt} + r_f$ with r_f the risk-free rate and w^{mkt} market shares of the assets.

It is assumed that the investor provides all three (P, q, Ω). A possible, agnostic choice for Ω is

$$\Omega = \tau_v P \Sigma_b P', \tag{2}$$

where $\tau_v > 0$ is a confidence scaling parameter, ' is the transpose operator for a matrix, and Σ_b is the same estimate for the covariance matrix, as the one used for the prior/benchmark. Note that if q is the product of a statistical machine learning model using objective predictors (hence the views are not subjective), then it is reasonable to use the scaled variance (2) formula with Σ_b the sample covariance matrix estimate, for the confidence matrix of views Ω.

4. **Bayesian update and portfolio optimization.** Given the observable q, use views in part 3., the prior in part 2. and Bayes-rule to get a posterior distribution.

$$p(r|q).$$

Then use this posterior distribution to maximize a chosen utility function for the investor, for instance equation (6), over portfolio weights w.

In the classical BL model, classical filtering relations for normal distributions deliver the following exact formulas for the update.

$$\mu_{BL-Bayes} = (P'\Omega^{-1}P' + C_b^{-1})^{-1}(P'\Omega^{-1}q + \mu_b), \quad \Sigma_{BL-Bayes} = (P'\Omega^{-1}P + C_b^{-1})^{-1}. \tag{3}$$

See e.g. Kolm et al. [2020], or the original paper Black and Litterman [1992] for the derivation of these formulas. Note that the formulas (3) require the sample covariance matrix Σ_b to be invertible to avoid ill-posedness, especially if the specification (2) is used to model uncertainty of the views. We call these two relations the *Bayes-BL update* to distinguish it from the other approaches at projecting the views to benchmark that we consider in this paper. In both the case of a Bayes-BL update, as well as in the case of minimizing another probability distance, the resulting BL update is a function of the views, making the final portfolio allocation weights an indirect function of the views as well.

3 Black-Litterman end-to-end model

In this section we explain our end-to-end approach for the BL model, together with the corresponding mathematical details. The goal of BL-end-to-end is to have a trainable system which functions as one pipeline, from inputing view prediction features, producing the BL update, to the calculation of the portfolio performance. This performance is used in turn to update the trainable model parts of the system.

We consider an investor who invests for a certain number of periods T in a certain asset universe, which does not need to be balanced over time. Each period, for a given vector of realized returns and realized covariance (as a measure of risk), the investor will rank different portfolio allocations $w \in \mathbb{R}^n$ via the mean-variance objective function (6). In the empirical implementation exercise, we ultimately consider an extensive set of portfolio return metrics to evaluate the performance of portfolio allocations. Throughout, we will keep the assumption of normal returns that underlies the classical BL model.[3] Our views will come from typical investment strategies, and we also assume that the set of views does not change over time, i.e. as time progresses, the investor is not allowed to delete certain views from consideration and add new ones.

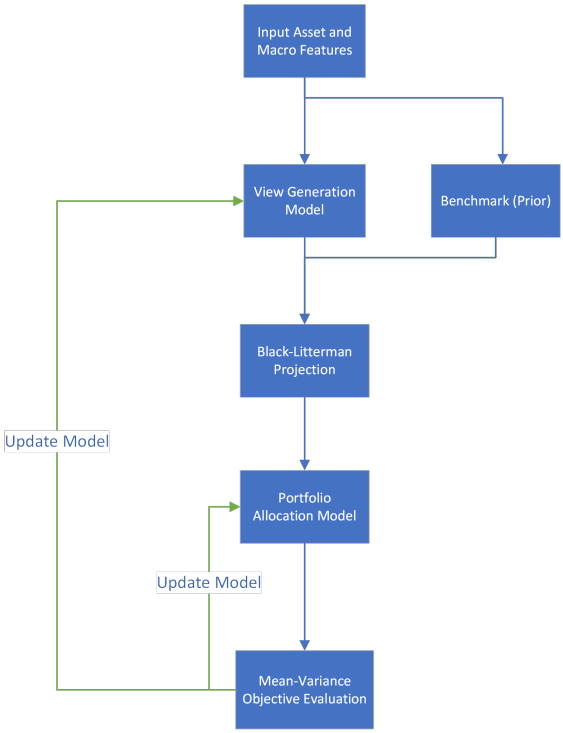
We consider two cases of a BL end-to-end system depending on which part of the system is a trainable model:

- **Joint optimization of view generation and portfolio allocation**: in this case there is one model, comprised of two parts that are modeled as neural networks (the view network and the weights network). The view neural network uses asset features and other aggregate features e.g. macroeconomic time series to generate the views. The weights neural network produces the portfolio allocation by using as input features the BL update: a n-dimensional vector of expected returns and a covariance matrix of dimensions $n \times n$. The resulting portfolio performance is used to update parameters of both networks at the same time.

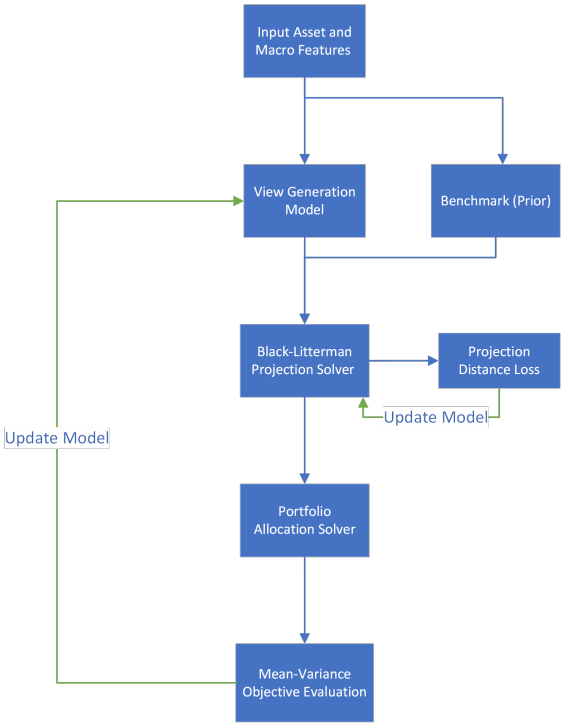
[3]This assumption is computationally convenient, not just for the case of the Bayes BL-update formulas in (3), but also in the case where the BL update is defined through minimizing a statistical distance, because there are closed-form solutions for many statistical distances provided the distributions are in the family of normal distributions.

- **Disjoint optimization of view generation and portfolio allocation**: in this case there is one neural network with trainable parameters, the view network, that performs the view generation from the view predictors. The portfolio weights are determined by solving for the weights in equation 6 through a separate solver that is run with *fixed* BL update. The resulting portfolio performance updates the view network by back-propagation, because the portfolio weights are a function of the BL update, which is in turn a function of the views produced by the view network.

Diagrams 1a and 1b describe the two cases we consider. Note that both cases are considered *end-to-end*, because in both cases there is one pipeline from feature-input to calculation of portfolio performance that is organized around the final goal of maximizing portfolio performance. In this sense, there is no split of the returns forecast exercise from the final goal of portfolio optimization, in contrast to more classical approaches where the researcher first performs the returns prediction exercise separately with a statistical loss (say minimizing mean-squared error) that is disjoint from the mean-variance objective, and then plugs the estimates into the mean-variance optimization problem.



(a) Diagram of jointly-optimized BLEnd2End-Bayesian model, where the BL update is produced through Bayes' rule and the weights through a trainable neural network.



(b) Diagram of disjointly-optimized BLEnd2End-Solver model, where the BL update is produced through minimizing a statistical distance like Kullback-Leibler, Jensen-Shannon, etc, and the weights are produced through a solver.

Figure 1: Diagrams to illustrate the differences between BLEnd2End-Bayesian and BLEnd2End-Solver.

The two cases differ mainly on whether the portfolio allocation is jointly determined with the view generation through a trainable model. In the second case the only model being updated is the view generation model, whereas both the BL update, and the portfolio allocation use solvers that take as input, respectively the views and the BL update. In the final implementation for our empirical exercise, we also take into account hybrid cases: e.g. the BL update uses the Bayes-BL update formulas (3), but the portfolio allocation is determined through a solver, or the separate case of diagram 1a where the BL update comes from minimizing a statistical distance, rather than from the Bayes-BL update formulas (3).

From an implementation point of view, in the joint case of diagram 1a, the joint model performing view generation and portfolio allocation must be defined *statically*, i.e. before deploying the end-to-end model, rather than during runtime. In the case of an unbalanced dataset, this presents a difficulty because the number of

assets that defines the dimension of the portfolio weights changes over time. Unless one is ready to accept imputing missing values due to an unbalanced cross section of assets, the joint optimization method we offer is only applicable to a cross section of assets that is balanced over time. In the second case of diagram 1b, the solver for the portfolio allocation is only deployed during runtime, and so this case can handle unbalanced cross sections of asset returns, without any need for imputation of values.

3.1 Strategies as views

In our model, we use a set of investment strategies suitable for the asset universe as investors' views $P \in \mathbb{R}^{V \times n}$. Here V is the number of views and n is the number of assets being traded. Then given the input features x, a set of predictors for the views, we use a neural network to predict the next period's view returns q. This neural network is called the `ViewNet`:

$$q = \texttt{ViewNet}(x).$$

In our empirical implementation this is either a feedforward neural network (FFN), or recurrent neural network (RNN).

The views in the original BL model can be thought as infusing investor's subjective statements about expected returns in the market with the market benchmark. The success of the BL end-to-end approach hinges on choosing a good set of strategies as views, that are a priori informative about the evolution of the whole market, as well as choosing a good set of view predictors to deploy in the `ViewNet` for prediction. One can expect the approach to be as good as the chosen views are.

Our approach to this is to consider a relatively sophisticated investor. Instead of using simple view statements of the form *"the Apple Inc. stock will outperform the Microsoft Inc. stock by 3% next month"*, which lead to a relatively sparse matrix of views P (as defined in section 2.2), we assume the investor can state claims about the future performance of *market-wide strategies*, e.g. *"the volatility strategy will outperform the momentum strategy by 3% next month"*. Such market-wide statements, especially in the presence of multiple views, typically lead to a non-sparse P as defined in section 2.2. The thinking is that modifying the original benchmark with such *market-wide* strategy insights should lead to better results.

Keeping with the assumption of normality of returns, we assume the q-vector output by the `ViewNet` is the mean of a normal distribution with dimension V, the number of views, and whose covariance matrix is given by (2), as described in section 2.2. Namely, for P the matrix producing the view portfolios, the view returns r_{BL} are distributed

$$r_{BL} \sim \mathcal{N}(q, \tau_v P \Sigma P'),$$

where τ_v is the confidence scaling vector, Σ is the estimator for the covariance. Σ can be the typical sample covariance estimator.

3.2 BL-update

For a given set of views q, under the normality assumption for asset returns, the traditional BL model produces an updated estimate of market-wide returns $\mu_{BL-Bayes}$ and of the covariance matrix of assets $\Sigma_{BL-Bayes}$ as described in equation (3). From a numerical standpoint, even if the sample covariance matrix Σ_b is invertible, the Bayes' update formulas in (3) involve multiple matrix inversions. Especially when the number of views is much smaller than the number of assets in the market, the matrix inversions, if at all possible, could still lead to a compounding of numerical errors. This issue becomes even more delicate in an end-to-end setup where the BL update is just one of the multiple steps of the model pipeline. For the case where the sample covariance estimate is not invertible, e.g. because there are many more assets than dates available, the traditional Bayes-BL formulas (3) simply do not work.[4] It could be estimated through e.g. a factor model then, and using equations (3) would involve bootstrapping with a matrix that is potentially not invertible, only compounding the difficulties. Therefore, we consider other ways to project the view distribution, described below, onto the benchmark based on minimizing well-known statistical distances.

Overall, we consider the following two distinct ways for the production of the `BL-update` for a fixed view distribution and benchmark distribution.

[4] In work in progress we are deploying a version of the BLEnd2End architecture to the S&P 500 universe. If the dataset consists of monthly returns, its time dimension is typically shorter than the size of the cross-section and hence the covariance matrix needs to be estimated through other means.

- **BLEnd2End-bayesian** If the sample covariance Σ_b is invertible, directly use Bayes rule and the formulas (3) to calculate the BL update analytically.
- **BLEnd2End-solver** Calculate the BL update by minimizing a distance measure *Dist* between two probability distributions. This approach is similar in that in Meucci et al. [2014]. Under the assumption of normality, the typical probability distances have exact formulas and can be calculated from the mean and covariances of the respective distributions.

  If (μ_b, Σ_b) are the mean and covariance of the benchmark, then the `BLSolver` solves the following constrained optimization problem.

$$(\mu_{BL}, \Sigma_{BL}) = \underset{(\mu,\Sigma)\ satisfies\ views}{\arg\min} Dist\left((\mu, \Sigma)|(\mu_b, \Sigma_b)\right). \tag{4}$$

  Note the restriction that μ, Σ has to satisfy the views. This means that the following are satisfied:

$$P\mu = q.$$

  Note that the matrix P of dimensions $V \times n$ is typically singular, as it will have rank at most equal to the number of strategies V. Hence typically, there is a whole subspace of permissible pairs (μ, Σ) in the optimization problem. One can incorporate this constraint via a Lagrangian.

  Finally, to cover cases where Σ_b is not invertible, one can use a traditional factor model for the covariance estimation. Namely, we use the decomposition

$$\Sigma = BB' + Diag(d \circ d), \tag{5}$$

  where $B \in \mathbb{R}^{V\times K}$, K is an integer smaller than V, and $d \in R^{V\times 1}$, is a strictly positive vector modeling the variance component of the views that is orthogonal to the K-factor variation.

  With this choice, the optimization problem (4) becomes the unconstrained optimization problem,

$$(\mu_{BL}, \Sigma_{BL}) = \underset{(\mu,B,d)}{\arg\min}\, Dist\left((\mu, B \cdot B' + Diag(d \circ d))|(\mu_b, \Sigma_b)\right) + \lambda\|P\mu - q\|^2.$$

  Here $\lambda > 0$ is a very large number penalizing deviations from the views. It is meant to approximate the indicator function that becomes $+\infty$ whenever $P\mu \neq q$, and is otherwise zero.

Statistical distance choices. In this paper, we consider three different choices for $Dist(\cdot, \cdot)$, all of which assume normal distribution. The first one is the Wasserstein-2 distance (henceforth WSD2), which for two n dimensional multivariate normal variables X, Y admits a closed-form expression, (see for instance Olkin and Pukelsheim [1982]).

$$W_2(X,Y)^2 = ||E[X] - E[Y]||^2 + trace\left(Cov(X,X) + Cov(Y,Y) - 2(Cov(X,X) \cdot Cov(Y,Y))^{\frac{1}{2}}\right).$$

Note that WSD2 does not require the covariance matrices to be invertible. This makes it computationally faster and more stable than the alternatives below.

The second probability distribution distance is the Kullback-Leibler (henceforth KL) divergence, that for two n dimensional multivariate normal distributions X, Y has the closed-form solution

$$\begin{aligned} KL(Y,X) = &\frac{1}{2}\left[\ln(det(Cov(X))) - \ln(det(Cov(Y)))\right] - \frac{1}{2}n \\ &+ \frac{1}{2}trace(Cov(X)^{-1}Cov(Y)) \\ &+ \frac{1}{2}(E[Y] - E[X])^t Cov(X)^{-1}(E[Y] - E[X]). \end{aligned}$$

The final probability distribution distance we consider is the Jensen-Shannon divergence (henceforth JSD), which is a symmetrized and smoothed version of KL Divergence. For normal distributions, canonical calculations give the

$$JSD(X,Y) = \frac{1}{2}KL\left(X, \frac{1}{2}(X+Y)\right) + \frac{1}{2}\left(Y, \frac{1}{2}(X+Y)\right).$$

For the **Jensen-Shannon divergence**, we use a formulation based on the differential entropy. More details, as well as other implementation details for the distances are contained in section A of the appendix.

3.3 Portfolio allocation

After obtaining μ_{BL} and Σ_{BL} both approaches as depicted in figures 1a and 1b calculate the optimal portfolio allocation by optimizing the mean-variance criterion (6), which we add here for reader's convenience.

$$\max_{w} w'\hat{\mu}_\tau - \frac{\gamma^{ra}}{2} w^T \hat{\Sigma}_\tau w - \gamma^{tr}\|w - w_{-1}\|_1. \tag{6}$$

On a technical note, the pre-weights/current portfolio weights w_{-1} are an additional parameter for the optimization problem. So that the overall input to the portfolio allocation procedure, leaving out trading cost parameter and risk aversion, takes as an input a vector of dimension $2 \times n + n \times n = n \times (2+n)$.

Furthermore, note that the objective function in (6) is invariant under permutations of the assets. To avoid deterioration of performance, this permutation invariance needs to be taken into account during the design of the system. See Zaheer et al. [2017] and Ravanbhakhsh et al. [2017] for examples of performance deterioration, and solutions to this type of problems in neural network algorithms using a special class of neural networks called DeepSets. Here we follow a simpler approach, that does not increase the number of trainable parameters: before inputting the BL-update (μ_{BL}, Σ_{BL}) to the portfolio allocation procedure, we find a canonical permutation of the covariance matrix Σ_{BL}, namely we order the rows of the covariance matrix in increasing order of their sum. Then we permute the excess returns estimate μ_{BL} and the pre-weights w_{-1} with the same permutation and feed them to the portfolio allocation procedure. In the end, we permute back the output weights w, as well as the pre-weights w_{-1} for the downstream task of portfolio performance evaluation.

In line with either the joint or disjoint optimization of view generation and portfolio allocation, we offer two different approaches to the portfolio allocation step.[5]

3.3.1 WeightNet and WeightDiffNet

In the joint-optimization philosophy, the weights are modeled as a feed-forward-network of the triplet $(\mu_{BL}, \Sigma_{BL}, w_{-1})$.

$$w = \texttt{FFN}(\mu_{BL}, \Sigma_{BL}, w_{-1}).$$

We call this variant of the weights model a `WeightNet`. The respective FFN takes as input a flattened vector of size $n \times (2+n)$.

We also consider another version of the weights model that uses further optimization deliberations which we call `WeightDiffNet`. Namely, consider first that the mean-variance objective function in (6) is strictly concave. This means that any solution to the first-order conditions of the optimization problem is a solution to the optimization problem. Furthermore, note that the objective function is only dependent on $\Delta = w - w_{-1}$, as it can be rewritten as follows, up to a constant that is independent of Δ, but still depends on w_{-1}.

$$\max_{\Delta} \quad \Delta'\hat{\mu}_\tau - \frac{\gamma^{ra}}{2}(\Delta + w_{-1})'\hat{\Sigma}_\tau(\Delta + w_{-1}) - \gamma^{tr}\|\Delta\|_1. \tag{7}$$

This formulation is emphasizing that the allocation decision problem can be expressed as finding the optimal *trades* from the old to the new portfolio allocation, rather than the optimal new portfolio allocation. Taking the first order condition with respect to Δ in (7) leads after some algebra to the first-order condition for trades Δ.

[5] A final technical note: because the infrastructure uses a stochastic-gradient descent algorithm for the global optimization of the architecture, we want to use batch-size as a hyperparameter for training. To allow for this, and to respect the flow of time, the code must carefully update the pre-weights after each date contained within the same batch.

$$(\gamma^{ra}\Sigma)\,\Delta + \gamma^{tr} sign(\Delta) = \mu - (\gamma^{ra}\Sigma)\,w_{-1}. \tag{8}$$

We see that Δ depends only on $\mu - \gamma^{ra}\Sigma w_{-1}$, and Σ. Recalling that γ^{tr}, γ^{ra} are fixed parameters, and 'hiding' them in the other variables, we can model the trading function as depending on the adjusted expected mean-vector $\mu_{BL,adj} = \mu - (\gamma^{ra}\Sigma)\,w_{-1}$, and the usual covariance $\Sigma_{BL,adj} = \Sigma_B L$. Overall, the `WeightDiffNet` models the trades Δ as

$$\Delta = \texttt{FFN}(\mu_{BL,adj}, \Sigma_{BL}).$$

The input size for the FFN now is $n \times (1+n)$. While the reduction by n dimensions may seem small for large n given the overall quadratic input size in n, it is not trivial for small n, because for the same inner architecture of hidden layers of the FFN, it has to process a smaller input in `WeightDiffNet` than in `WeightNet`. Our empirical results indeed show that often `WeightDiffNet` performs better than `WeightNet` in the overall end-to-end architecture.

3.3.2 Weight solver

This is the case of the disjoint optimization of the view generation and portfolio allocation: for a fixed pair of the BL update (μ_{BL}, Σ_{BL}), as well as pre-weights w_{-1} we maximize the objective function (6) with a separate optimization problem

$$w = \texttt{WeightSolver}(\mu_{BL}, \Sigma_{BL}, w_{-1}).$$

Once the performance using realized returns and covariance based on the weights w is calculated, it is back-propagated to the view generating network through the intermediate variables (μ_{BL}, Σ_{BL}), which are in turn functions of the views q produced by the view generating network `ViewNet`.

Finally, as an alternative to determining the weights by maximization of (6), we also experiment with a version of the `WeightSolver` which instead minimizes the deviation from the first order conditions (8).

$$\ell_{foc}(\Delta; \mu_{BL,adj}, \Sigma_{BL}) = \left\| (\gamma^{ra}\Sigma_{BL})\Delta + \gamma^{tr} sign(\Delta) - \mu_{BL,adj} \right\|^2. \tag{9}$$

4 Empirical study on an ETF portfolio

We evaluate our methodology with a dataset of 14 highly liquid ETFs that encompass a wide variety of industry sectors in the U.S. economy, as well as the S&P 500, together with 4 ETFs spanning the yield curve for U.S. treasuries. For such a broad investment coverage we use macroeconomic time series as view predictors. The next sections contains more details about the dataset and our empirical implementation.

4.1 Data description

Asset universe Our asset universe includes the highly liquid 14 ETFs described in Table 1. We pick ETFs as an asset class for our empirical exercise, since they are already diversified, and hence their returns are more likely to satisfy the normality assumption of the traditional BL model.

The monthly dataset we use is downloaded from the NYSE Trade and Quote (TAQ), accessed from the WRDS database. The time horizon of this study is from 2008-02 to 2021-01-01 and it uses monthly data.

The 9 sector ETFs cover a broad section of the U.S. economy. SPY is an ETF that tracks the S&P500 index, and its performance is representative of the total equity market performance in the U.S. One reason we included SPY in the asset universe, despite having the 9 sector ETFs, is for its potential hedging purposes.[6] The four different treasury ETFs track treasury yields of different maturities spanning the entire treasury yield curve. We have included these ETFs to allow for investment in both U.S. equity and fixed income markets.

[6] Even though we do not model hedging in detail in our model, we do not impose short-selling constraints. Hence, we want to allow for the possibility of endogenous hedging by the optimal portfolio, e.g. the portfolio might go long on several sector-ETFs and at the same time go short on the SPY.

Asset Symbol	Asset Name
BIL	SPDR® Bloomberg 1-3 Month T-Bill ETF
IEF	iShares 7-10 Year Treasury Bond ETF
SHY	iShares 1-3 Year Treasury Bond ETF
TLT	iShares 20+ Year Treasury Bond ETF
SPY	SPDR S&P 500 ETF Trust
XLB	SSgA Active Trust - Materials Select Sector SPDR ETF
XLE	SSgA Active Trust - The Energy Select Sector SPDR ETF
XLF	SSgA Active Trust - Financial Select Sector SPDR ETF
XLI	SSgA Active Trust - Industrial Select Sector SPDR ETF
XLK	SSgA Active Trust - Technology Select Sector SPDR ETF
XLP	SSgA Active Trust - Consumer Staples Select Sector SPDR ETF
XLU	SSgA Active Trust - Utilities Select Sector SPDR ETF
XLV	SSgA Active Trust - Health Care Select Sector SPDR
XLY	SSgA Active Trust - Consumer Discretionary Select Sector SPDR

Table 1: ETF assets.

Macroeconomic time series as view predictors Given the broad scope of investment encompassed by the 14 ETFs, and because we work with monthly data, we choose macroeconomic time series as the main view predictors. The majority of the monthly time series data is obtained from the FRED-MD database as constructed and described in McCracken and Ng [2016] and available at `https://research.stlouisfed.org/econ/mccracken/fred-databases/`. Namely, we employ 129 time series from FRED. All FRED variables are transformed to obtain stationary series according to the recommendations of McCracken and Ng [2016]. To this set, we add the cross sectional means and medians of the signals we use for the view construction explained below. In our empirical exercise, we consider 7 views adding thus 14 time series to the FRED ones. Finally, we add the cross-sectional mean and median of the effective percentage bid-ask spread for the ETFs, which is also downloaded from TAQ. We do not process the aggregated view signals and the additional signal further, because the signals are scaled to always be in the interval $(-1, 1)$ (see next section). Combined, this leads to a set of 145 time series predictors for the views.

4.2 ETF views description

We consider views based on classical signals for investment management like momentum, reversal, volatility, together with exposure as well as the alpha to a traditional model for equity risk factors like the Fama-French 3-factor model [Fama and French, 1993]. We use 7 long-short zero-cost portfolio views, defined through uni-sort from the following signals. The ETF raw signals are as follows.

- `vol_month` $= -std(r_{t-1}^{daily})$ is the volatility signal over the previous month, calculated from daily returns.
- `mom12` $= \sum_{l=1}^{12} r_{t-l}$ is the past year momentum, based on the returns from the previous 12 months.
- `momrev` $= \sum_{l=14}^{19} r_{t-l}$ is the momentum reversal signal, based on the six-months return one-year in the past.
- `strev` $= -r_{t-1}$ is the short term reversal signal, based on last months return.
- `lrrev` $= \sum_{l=13}^{60} r_{t-l}$ is the long-term reversal calculated as the cumulative returns from $t-60$ to $t-13$.
- `ff3_alpha` is the alpha with respect to the Fama-French 3-factor model [Fama and French, 1993] calculated from the previous month return.
- `ff3_beta` is the asset beta with respect to the Fama-French 3-factor model [Fama and French, 1993] calculated from the previous month return.

For each raw signal we construct a view portfolio by ranking the ETFs according to the signal and forming a long-short portfolio going long on the higher ranked ones and short on the lower-ranked ones. In more detail, we followed section 4.2 in Kozak et al. [2020] for the construction of long-short portfolios. If $rank_t(i,c)$ is the rank of ETF i in the raw signal c, one uses the normalized rank and its cross-sectional average

$$r_t(i,c) = \frac{rank_t(i,c)}{n+1}, \qquad \bar{r}_t(c) = \frac{1}{n}\sum_i r_t(i,c),$$

where n is the number of assets available. Then the signal is

$$z_t(i,c) = \frac{r_t(i,c) - \bar{r}_t(c)}{\sum_i |r_t(i,c) - \bar{r}_t(c)|}. \tag{10}$$

Note that $\sum_i z_t(i,c) = 0$, leading to a long-short zero-cost portfolio.[7]

Note that we use long-short portfolios that invest in the whole cross-section of 14 ETFs, rather than picking, say the first and bottom x-ranked ETFs. This ensures that the view matrix P, as described in section 2.2 is typically never sparse. In this sense, the views are *market-wide*, rather than focused on a few ETFs at a time.

4.3 Benchmark model for ETFs

We consider as a benchmark for our version of the BL model a statistical factor model that is estimated using rolling Principal Component Analysis (PCA) with the top K eigenvectors of the sample covariance estimator Σ_b. The factor regression producing the estimates $\hat{r}_t$ is

$$r_t = \alpha_t + \beta_t' F_t + \epsilon_t, \tag{11}$$

where F_t is a $K \times n$-dimensional matrix of factor returns based on the PCA factors from Σ_b at time t, β_t is a matrix of factor exposures of dimension $n \times K$ whereas α_t is a coefficient of alphas with respect to the PCA factors. We use a rolling estimation window of 36 months, and possible values $K = 3, 5, 7$ and 10. We also try the rolling PCA version without intercept $\alpha_t \equiv 0$ for robustness, its derived benchmark portfolio performs worse, and so we stick to the version with estimated intercepts.

While for training purposes of the BLEnd2End architecture we use only $3, 5$ PCAs due to limited computing power, we compare our results with the performance of the PCA benchmark for all of $K = 3, 5, 7, 10$. To calculate the performance of the PCA benchmarks, we take their implied prediction of the returns μ_{bench} from the rolling PCA exercise, and the (invertible) sample covariance $\Sigma_{bench} = \Sigma_b$ and use them as inputs in the mean-variance with trading costs objective given by equation (6). Here, we use the same values for risk aversion γ^{ra} and trade aversion γ^{tr} as for the BLEnd2End model estimation. Furthermore, when calculating the optimal portfolio weights, we introduce the same constraint on the L^2-norm of the portfolio weights, as for the BLEnd2End estimation (see discussion in 2.1). We solve the portfolio optimization problem for each date separately using the python `CVPXY` package [Diamond and Boyd, 2016].

4.4 Portfolio performance evaluation

We compare the performance of the BLEnd2End model with the performance of the benchmarks described above. Furthermore, we also compare with two classical strategies such as fix-mix on SPY and the treasury ETFs, as well as the risk parity optimal allocation. Finally, we compare the performance of BLEnd2End on the subset of the dataset consisting of the sector ETFs only, with the sector momentum and sector reversal strategy.

For all strategies considered, we evaluate portfolio performance using the typical portfolio ratios: Sharpe, Calmar and Sortino, but calculate maximum-drawdown and the realized leverage of the portfolios as well.

[7] An alternative would be to consider the normalization

$$z_t(i,c) = \frac{r_t(i,c) - \bar{r}_t(c)}{\sqrt{\frac{1}{n-1}\sum_i (r_t(i,c) - \bar{r}_t(c))^2}}.$$

This has the additional advantage of having standard deviation equal to one, so that the correlation of the signal z with the returns of the asset can be interpreted as the information coefficient. See chapter 9 of Zhou and Jain [2014] for more. Based on limited experimentation with both methods, we find that the construction (10) performs better for our dataset.

4.5 Machine learning implementation details

This section contains details about the architecture and hyperparameter space we search over in the training of the BLEnd2End architecture.

Data splitting The test phase constitutes the last 30% of dates, whereas the training and validation phase is the first 70% of dates. Validation takes up 30% of dates in the training-validation phase, or equivalently 21% of the overall dates.

Hyperparameter and architecture space We train a total of 6,912 models based on the combination of different neural network architectures and hyperparameters that are described below.

- **Number of PCA factors for the benchmark** are selected out of [3, 5].
- **Optimizer** We use the Adam optimizer (Kingma and Ba [2015]) for the global optimization, as well as for `BLSolver` case and the `WeightSolver` case. We always fix the momentum parameter of Adam at 0.9. For the case of the **joint** optimization of view generation and portfolio construction, see diagram 1a, the global Adam optimizer takes as trainable parameters the parameters of the `ViewNet` as well as of the `WeightNet`/`WeightDiffNet`, depending on the choice of the latter. For the case of **disjoint** optimization, see diagram 1b, it takes only the parameters of the `ViewNet`.
- **Batch sizes for the global optimizer** are selected out of [28, 56, 112]. Due to coding technicalities related to the appearance of the pre-weights w_{-1} in the mean-variance objective (6), the batch sizes have to be multiples of the number of assets. This is to ensure that each date is contained within a batch completely.
- **Learning rate of the global optimizer** comes from [0.001, 0.0001].
- **Global training criterion** is based on the mean-variance objective with trading costs (6). For each batch, we split it according to dates, and the training loss of the batch is minus the sum of (6), summed across dates contained in the batch.
- **L^1-regularization of the global training criterion** is selected from [0, 0.01, 0.1].
- **L^2-regularization for the global training criterion (weight decay in the Adam optimizer)** is also selected from [0, 0.01, 0.1].
- **Number of epochs for global training** is 40 epochs for the whole end-to-end system.
- **Validation phase criterion and early stopping** We use Sharpe ratio to evaluate the performance in the global optimization in each epoch. As an additional regularization procedure, we use early stopping if the Sharpe ratio has not improved after 8 consecutive epochs.
- `ViewNet` is selected from either of the following 4 neural network definitions:
  - `FFN` with hidden units of [64, 32, 16] or [64, 64, 32, 32, 16, 16] (note: 2 possible choices here), using `SiLU` (sigmoid linear unit) as activation function. The dropout rate is 0.05 and we also use batch normalization.
  - `RNN` in the form of GRUs (Gated Recurrent Units) with either 4 or 8 hidden states (note: 2 possible choices here). The sequence length of features fed to the GRU is 6 months and it has one layer. The dropout rate is again 0.05.
- `BL-updater` is selected from one of the following 4 possibilities:
  - **BLEnd2End-Bayesian** with view confidence scaler $\tau_v = 1/7$.
  - **BLEnd2End-Solver** We use Adam optimizer with 40 iterations and learning rate 0.0001. The probability distribution distance measure is chosen out of WSD2, JSD and KL divergence (note: 3 possible choices here). We use covariance factorization with $K = 10$ factors (see equation (5)).
- **Portfolio allocation** is determined by choosing one out of the 4 following possibilities:

- `WeightNet`: this is for the **joint** optimization case as explained in section 3. We use a `FFN` with hidden units [128,64,64,32,32] that takes as input a flattened vector of $(\mu_{BL}, \Sigma_{BL}, w_{-1})$ with length $n \times (n+2)$ ($n = 14$ in our dataset). Here w_{-1} are the current portfolio weights that are updated continuously during the training, so that for each date, the optimization problem uses the correct pre-weights. The activation function for the neural network is `SiLU` and the dropout rate is 0.05.
- `WeightDiffNet`: this is for the **joint** optimization case as explained in section 3. We use the same neural network structure as in `WeightNet`, with the same activation function and dropout rate. Now the input is a flattened vector of $(\mu_{BL,adj}, \Sigma_{BL})$ of length $n \times (n+1)$, as explained in section 3.3.
- `WeightSolver` uses Adam as an optimizer with a learning rate of 0.0001 and 40 iterations to either maximize the mean-variance criterion (6), or to minimize the loss based on the first-order-condition (8) (note: these are 2 possible choices).

- **Leverage constraint** The portfolio weights from any of the above portfolio calculation models are normalized with a bound of 0.5 on L^2-norm. This leads to an effective leverage bound of 1.87. This is a soft bound, i.e. the system chooses the leverage at the date level endogenously, as long as it is below the upper bound.

- **Choice criterion for the best model and ensemble procedure** We choose the best model architecture and hyperparameters using the last registered Sharpe ratio in the validation phase. The best model and architecture are then retrained with 5 additional distinct seeds to the seed used for the training. We ensemble the 6 thus obtained models and evaluate final performance based on the return time series on the test set.

4.6 Results

The best performing BLEnd2End architecture based on the validation phase results is of a **joint** optimization type (diagram 1a). It uses as a `ViewNet` an RNN with hidden state dimension of 4, and as a portfolio allocation model a `WeightDiffNet` with hidden layer units $[128, 64, 64, 32, 32]$. The total batch size for the global optimizer for the winning model is 28, i.e. two dates within each batch, the learning rate is 0.001, and it does not use L^1- and L^2 regularization. The `BLUpdater` chosen uses Bayes' rule. We conjecture, that the alternatives using other statistical distances do not beat the traditional Bayes' formula due to our underlying normality assumption: for normal distributions Bayes' rule gives an exact formula for the projection, whereas the probability distances we consider need to undergo a minimization procedure with a numerical solver that introduces additional numerical inaccuracies. Finally, the number of benchmark PCA factors of the best validated model is 5.

Before turning to the analysis of the portfolio performance of the BLEnd2End, we note here that, leaving out variation in terms of the parameters of the global optimizer (learning rate, L^1-, and L^2-regularization), the second and third best performing architectures are again of the **joint** type, but now with different choices for the `ViewNet` and of the model of the portfolio allocation (second- and third-best architecture use `WeightNet`). Furthermore, the second-best architecture uses a `BLSolver` with the WSD2 distance metric instead of Bayes' rule, whereas the third-best a `BLSolver` with the KL distance. With the caveat that we have not performed extensive hyperparameter search for the `WeightSolver`, our results emphasize the benefit of jointly optimizing the view generation and portfolio allocation with a single global objective criterion given by the mean-variance objective 6.

Portfolio performance In the following discussion, we refer to the resulting portfolio of the winning model from the validation phase as the BLEnd2End portfolio. A detailed breakdown of the portfolio performance in different phases can be found in Table 2. The BLEnd2End portfolio achieves an annualized Sharpe ratio of 0.92 on test set. What is more impressive, it exhibits a very low amount of downside risk in the test phase, as described by the maximum drawdown and the Sortino ratio.

Figure 2 below illustrates BLEnd2End portfolio performance through train, validation and test phase performance in terms of the cumulative returns.

Comparison with rolling PCA benchmarks BLEnd2End performs much better than the rolling PCA benchmarks. We focus in the presentation on the results vs. 5 PCA factors, because that is the optimal number of benchmark PCA factors chosen during the validation phase of the BLEnd2End training. The comparison is

	mean return (%)	volatility (%)	sharpe	sortino	calmar	mdd
train	2.378	3.116	0.769	8909.853	4.930	0.005
val	0.458	1.057	0.441	53.306	0.145	0.032
test	0.256	0.971	0.266	38.685	0.109	0.023

Table 2: BLEnd2End portfolio performance through the train, validation and test phase. Values are monthly and expressed in decimals, except for mean return and volatility.

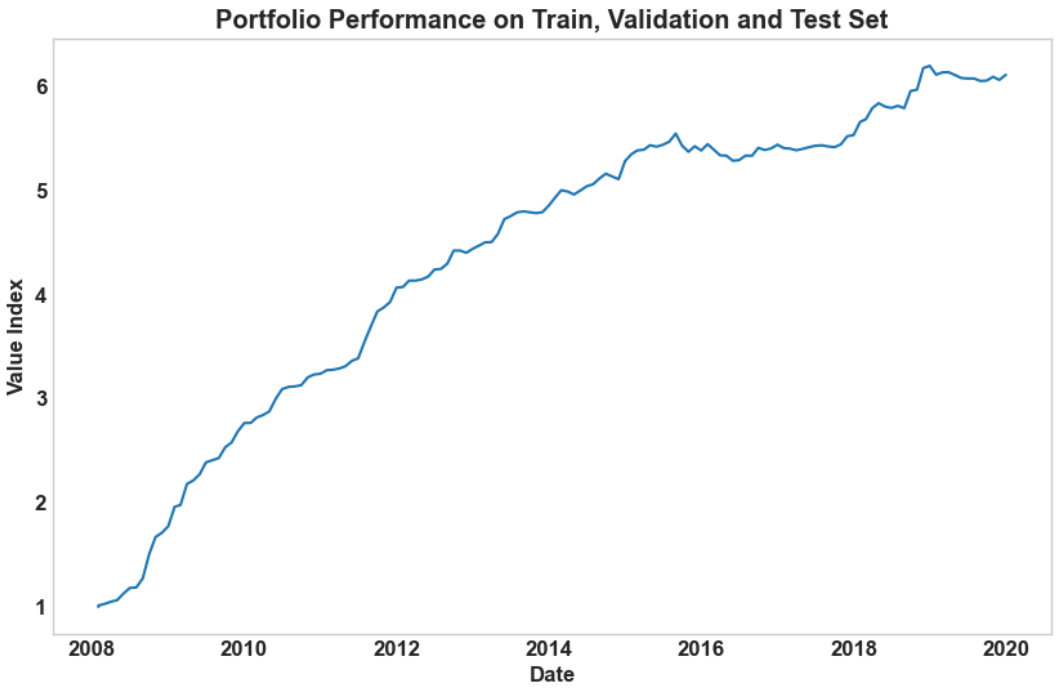


Figure 2: BLEnd2End portfolio value index, including train, validation and test set.

qualitatively the same for all of the PCA benchmarks we have calculated ($3, 5, 7$ and 10 PCA factors). Table 3 depicts the performance of the 5 PCA factor benchmark across train, validation and test phases. Table 4 focuses on the comparison on test set.

	mean return (%)	volatility (%)	sharpe	sortino	calmar	mdd
train	0.399	4.800	0.084	0.709	0.013	0.314
val	-0.296	2.022	-0.149	-4.393	-0.022	0.135
test	0.321	2.619	0.124	2.147	0.023	0.137

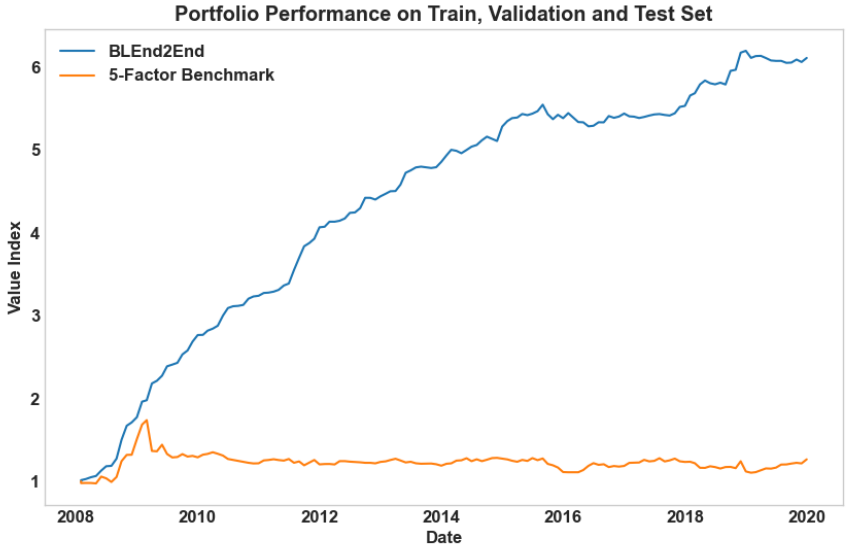
Table 3: 5 PCA benchmark portfolio performance through the train, validation and test phase. Values are monthly and expressed in decimals, except for mean return and volatility.

	sharpe	sortino	calmar	mdd	mean leverage
BLEnd2End	0.266	38.685	0.109	0.023	0.653
5-Factor Benchmark	0.124	2.147	0.023	0.137	1.383

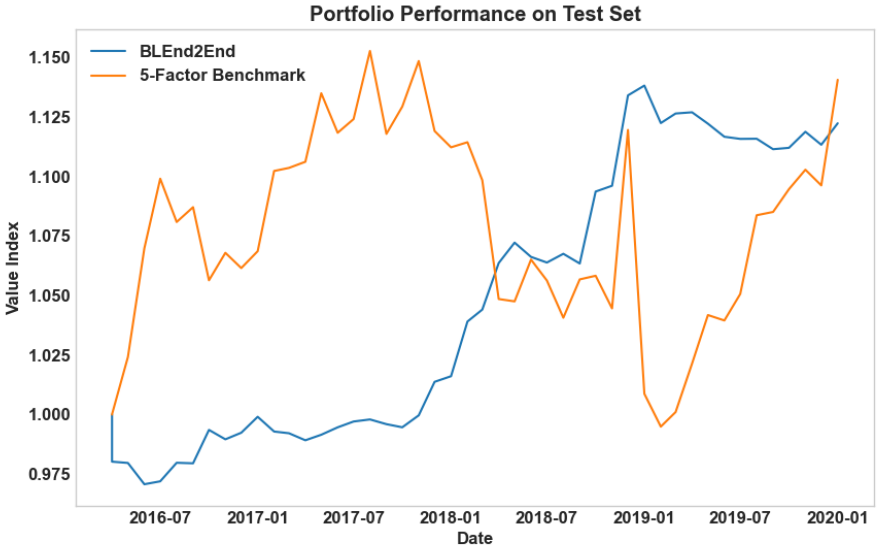
Table 4: BLEnd2End portfolio performance compared with benchmark portfolio on test set. All values are monthly.

Note that, even though the return of the benchmark portfolio is on the same order of magnitude as BLEnd2End on the test set, it suffers from greater volatility and greater drawdowns. Furthermore, the BLEnd2End approach infused with the views coming from the set of strategies we consider takes on a far smaller amount of leverage than the benchmark, even though we impose the same leverage constraint on the benchmark portfolio as for BLEnd2End: 0.5 upper bound on the L^2-norm on the final portfolio weights. Due to an endogenously chosen smaller amount of leverage, the BLEnd2End portfolio is able to achieve far better Sharpe, Sortino and Calmar ratio.

Figure 3a illustrates the cumulative returns of the 5 PCA factors benchmark and the BLEnd2End approach for the whole time period, whereas Figure 3b depicts the comparison in the test set.



(a) Cumulative returns of BLEnd2End portfolio compared with the 5 PCA factors benchmark. All values are monthly.



(b) Cumulative returns of BLEnd2End portfolio compared with the 5 PCA factors benchmark, comparison on test set only. All values are monthly.

Figure 3

Portfolio performance compared with common ETF investment strategies We compare BLEnd2End portfolio performance with other common ETF investment strategies. We first consider the optimal risk parity portfolio with expanding window sample covariance estimation. Second, we consider the fix-mix strategy with 60% stocks (SPY) and 40% bonds (TLT). Finally, we also consider the equal-weights portfolio on the whole cross-section. Table 5 shows that BLEnd2End portfolio is superior to all the other common investment strategies for its high risk-adjusted returns, especially high Sortino ratio due to minimum drawdown.

	sharpe	sortino	calmar	mdd
BLEnd2End	0.266	38.685	0.109	0.023
Risk Parity	-0.028	-3.620	-0.003	0.035
60-40	0.087	2.005	0.011	0.155
Equal Weights	0.076	1.687	0.010	0.147

Table 5: BLEnd2End portfolio statistics compared to other common ETF investment strategies on test set. All values are monthly.

Note that only the risk-parity portfolio's drawdown is on the same order as that of BLEnd2End. This is not surprising as risk parity is a conservative strategy in terms of risk exposure. However, BLEnd2End portfolio's average return is far higher than risk-parity. Figure 4 depicts the comparative evolution of portfolio cumulative returns.

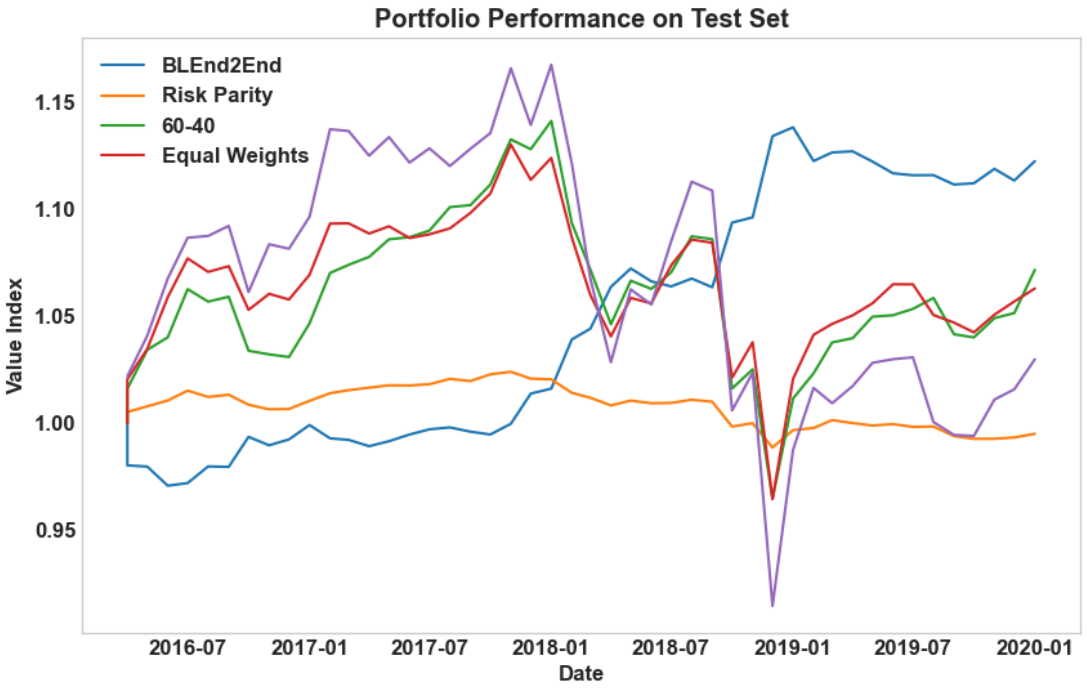


Figure 4: BLEnd2End cumulative returns on test set compared with risk parity, 60/40 fix-mix and the equal weights portfolios.

We also compare the return of the subset portfolio of BLEnd2End consisting of sector ETFs only with two standard long-only ETF strategies: sector momentum rotation, and sector momentum reversal. We define sector momentum rotation as the strategy that uses the cumulative return of the respective sector over the past year as the signal. The sectors are ranked according to the momentum cross-sectionally: the higher the return in the last year, the higher the relative investment into the sector. Sector momentum reversal uses the cumulative return of the sector over the previous quarter as the signal. The sectors are here ranked according

to the negative momentum cross-sectionally: the higher the return in the past quarter, the lower the share of the sector in the strategy portfolio.

The results in Table 6 confirm that the BLEnd2End sub-strategy which achieves an annualized Sharpe ratio of 0.86, performs better than the two sector ETF strategies.

	sharpe	sortino	calmar	mdd
BLEnd2End Sectors	0.248	39.345	0.078	0.028
Sector Momentum Rotation	0.063	1.034	0.009	0.207
Sector Momentum Reversal	0.106	1.611	0.018	0.171

Table 6: BLEnd2End portfolio restricted to sector ETFs, compared to sector momentum rotation strategy and sector momentum reversal strategy. All values are monthly.

Figure 5 depicts the relative evolution of cumulative returns on the test set for this comparison.

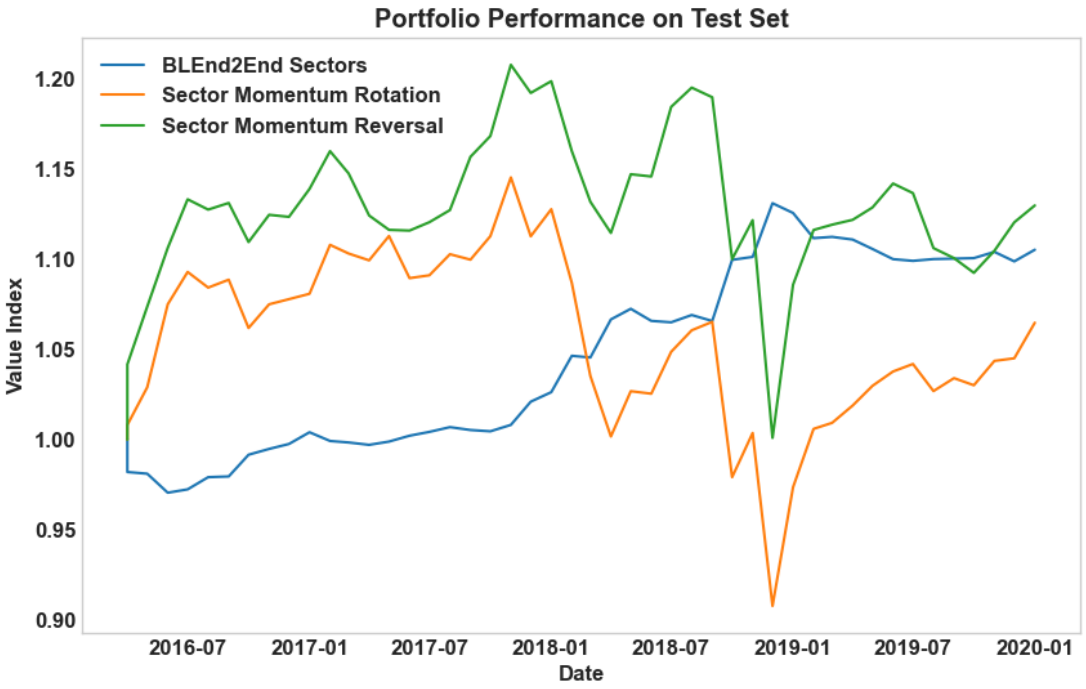


Figure 5: BLEnd2End portfolio sub-strategy restricted to sector assets compared to sector momentum rotation and reversal.

5 Conclusions and future research

In this paper we have implemented an end-to-end version of the Black-Litterman model, a leading and highly-interpretable approach to portfolio construction that regularizes the standard mean-variance approach. Rather than splitting the estimation and portfolio allocation steps as in the classical approach, we consider a framework where the generation of views, as well as the portfolio allocation are performed in a single pipeline with one overarching financial objective: optimizing a mean-variance criterion with trading costs.

Our framework keeps the normality assumption of returns from the classical model, but introduces two new ideas to the traditional framework. First, the views are defined as a set of market-wide strategies, that are predicted through machine learning methods from relevant financial time series. Second, our framework allows for the joint optimization, in a single model, of the view generation mechanism and of the portfolio allocation.

We achieve this joint optimization leveraging neural networks in the Pytorch optimization framework. While our focus is the joint optimization approach, we consider a wide range of model variations within the single-pipeline philosophy. We allow for the disjoint optimization case where only the view generation model is updated during training, while the weights are solved for a fixed BL update through a separate optimization problem. We also allow for alternatives to the Bayes' update rule of projecting the views into the benchmark, considering instead BL update solvers that minimize typical probability distribution distances like the Kullback-Leibler divergence, the Wasserstein-2 distance, or the Jensen-Shannon divergence.

We showcase the performance of our approach in an ETF data universe, consisting of 14 ETFs where 9 are sector ETFs, 4 ETFs span the yield curve of U.S. treasuries, and the SPY ETF tracking S&P 500. For this exercise, we pick 7 views focused on market-wide strategies consisting of a traditional volatility strategy, 3 reversal strategies, 1 momentum strategy, and two strategies capturing exposure and alpha to the traditional equity risk factors from the classical Fama and French [1993] model. As view predictors, we choose a large cross section of aggregate macroeconomic time series from FRED. As a benchmark model within the BL approach, we consider rolling regressions with varying numbers of PCA factors extracted from the sample covariance.

We confirm that the joint optimization approach, where a single model is learned for both view generation and portfolio allocation constitutes the best architecture within the single-pipeline philosophy for our empirical exercise. The BLEnd2End approach delivers a huge improvement on the PCA benchmark, confirmed by significantly higher Sharpe ratios. The main reason for the superior performance is that the BLEnd2End portfolio exhibits a much smaller downside, as evidenced by very small maximum drawdowns and large Sortino ratios. BLEnd2End is able to outperform other ETF strategies like risk parity, the 60% stocks - 40% bonds fix-mix, and equal-weights portfolio. Furthermore, we show that the BLEnd2End sub-portfolio, restricted to sector ETFs, is able to outperform other long-standing ETF strategies like sector momentum rotation and sector reversal.

The empirical exercise in the paper has focused on the case of a balanced cross-section of assets: the same 14 ETFs are present throughout the time period we consider. In practical investment management the case of unbalanced cross-sections is more relevant. Designing methods to deal with unbalanced cross-sections is important because it decreases the presence of survivorship bias in financial studies. Moreover, dealing with such datasets is sometimes outright unavoidable even in cases where the number of assets does not change over time. E.g. the S&P500 index contains (roughly) the same number of stocks over time, but the set of constituents changes continuously, as new stocks fulfill the criteria for membership while others are replaced. Our BLEnd2End framework allows dealing with unbalanced datasets seamlessly, without the need of introducing balancedness in an indirect way, e.g. through imputations. In ongoing work, we are extending our empirical applications of the framework to the set of constituents of S&P500.

References

M. Ao, Y. Li, and X. Zheng. Approaching mean-variance efficiency for large portfolios. *The Review of Financial Studies*, 32(7):2890–2919, 2019.

D. Bertsimas, V. Gupta, and I. C. Paschalidis. Inverse optimization: A new perspective on the black-litterman model. *Operations Research*, 60(6):1389–1403, 2012.

F. Black and R. Litterman. Global portfolio optimization. *Financial Analysts Journal*, 48:28–43, 1992.

S. P. Boyd, E. Busseti, S. Diamond, R. N. Kahn, K. Koh, P. Nystrup, and J. Speth. Multi-period trading via convex optimization. *Found. Trends Optim.*, 3:1–76, 2017.

M. W. Brandt, P. Santa-Clara, and R. Valkanov. Parametric portfolio policies: Exploiting characteristics in the cross-section of equity returns. *The Review of Financial Studies*, 22(9):3412–3447, 2009.

L. Chen, M. Pelger, and J. Zhu. Deep learning in asset pricing. *Management Science*, 2023.

T. C. Cover and J. A. Thomas. Wiley and Sons Inc., 2 edition.

G. Creamer. Can a corporate network and news sentiment improve portfolio optimization using the black-litterman model? *Quantitative Finance*, 15:1405–1416, 2015.

S. Diamond and S. P. Boyd. Cvxpy: A python-embedded modeling language for convex optimization. *Journal of machine learning research : JMLR*, 17, 2016.

M. F. Dixon, I. Halperin, and P. A. Bilokon. *Machine Learning In Finance: From Theory to Practice*. Springer Nature Switzerland AG, Cham, Switzerland, 2020.

E. F. Fama and K. R. French. Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, 33:3–56, 1993. URL https://api.semanticscholar.org/CorpusID:153834826.

J. Guijarro-Ordonez, M. Pelger, and G. Zanotti. Deep learning statistical arbitrage. *arXiv preprint arXiv:2106.04028*, 2021.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. *ICLR 2015*, 2015.

P. N. Kolm, G. Ritter, and J. Simonian. Black–litterman and beyond: The bayesian paradigm in investment management. *Journal of Portfolio Management*, 47(1-2):91–113, 2020. doi: 10.3905/jpm.2021.1.222.

S. Kozak, S. Nagel, and S. Santosh. Shrinking the cross-section. *Journal of Financial Economics*, 135:271–292, 2020.

X. Li, A. S. Uysal, and J. M. Mulvey. Multi-period portfolio optimization using model predictive control with mean-variance and risk parity frameworks. *European Journal of Operational Research*, 299(3):1158–1176, 2022.

H. Markowitz. Portfolio selection. *Journal of Finance*, 7:77–91, 1952.

M. W. McCracken and S. Ng. Fred-md: A monthly database for macroeconomic research. *Journal of Business & Economic Statistics*, 34(4):574–589, 2016.

A. Meucci, D. Ardia, and M. Colasante. Portfolio construction and systematic trading with factor entropy pooling. *SSRN working paper*, 2014.

I. Olkin and F. Pukelsheim. The distance between two random vectors with given dispersion matrices. *Linear Algebra and its Applications*, 48:257–263, 1982.

R. Oprisor and R. H. Kwon. Multi-period portfolio optimization with investor views under regime switching. *Journal of Risk and Financial Management*, 2020. URL https://api.semanticscholar.org/CorpusID:234420438.

S. Ravanbhakhsh, J. Schneider, and B. Poczos. Deep learning with sets and point clouds. *arXiv: https://arxiv.org/pdf/1611.04500.pdf*, 2017.

S. Uysal, X. Li, and J. M. Mulvey. End-to-end risk budgeting portfolio optimization with neural networks. *ERN: Optimization Techniques; Programming Models; Dynamic Analysis (Topic)*, 2021. URL https://api.semanticscholar.org/CorpusID:235795388.

M. Van Der Schans and H. Steehouwer. Time-dependent black-litterman. *Journal of Asset Management*, 18: 371–387, 2017.

M. Zaheer, S. Kottur, S. Ravanbhakhsh, B. Poczos, R. Salakhutdinov, and A. J. Smola. Deep sets. *31st Conference of Neural Information Processing Systems (NIPS 2017)*, pages 271–292, 2017.

X. Zhou and S. Jain. *Active Equity Management*. Xinfeng Zhou, 2014.

Appendix

A On statistical distances

On the Jensen-Shannon divergence For a random variable X, let h be its differential entropy (see section 8.1 of Cover and Thomas)

$$h(X) = -\int f(x)\ln(f(x))dx, \quad f \text{ density of } X.$$

Then the Jensen-Shannon divergence can be defined as

$$JSD(X,Y) = \frac{1}{2}\left(KL(X|\frac{1}{2}(X+Y)) + KL(Y|\frac{1}{2}(X+Y))\right). \tag{12}$$

In terms of differential entropy, this can be expressed as

$$JSD(X,Y) = h(\frac{1}{2}(X+Y)) - \frac{1}{2}(h(X)+h(Y)). \tag{13}$$

Canceling constants leads to the following for X, Y multivariate normal variables.

$$JSD(X,Y) = ln(det(Cov(\frac{1}{2}(X+Y)))) - \frac{1}{2}(ln(det(Cov(X))) + ln(det(Cov(Y)))). \tag{14}$$

We need an assumption to calculate $Cov(\frac{1}{2}(X+Y))$. We assume here ad hoc that views are independent from benchmark, i.e. the subcase X is independent to Y. This leads to

$$Cov(\frac{1}{2}(X+Y)) = \frac{1}{4}(Cov(X) + Cov(\gamma)).$$

The independence assumption can be supported by imagining that we are doing an underlying large simulation using independent draws to get the BL update, for a fixed (μ_b, Σ_b) for the benchmark, and pair (μ, Σ) as candidate for the BL update. But since for normal distributions the JSD has explicit parametric forms in the parameters we want to estimate, we do not actually need to implement the simulation.

Implementation details. To avoid numerical problems due to gradients, we take following steps in the code:

- We calculate determinants of covariance matrices as the product of eigenvalues.
- We calculate the trace of a multiplication of matrices where an inverse is involved, by first diagonalizing the matrices and using their respective eigenvalues whenever we can.
- Whenever we use eigenvalues of a matrix for calculations, we replace negative eigenvalues with a small $\epsilon > 0$. Negative values should not happen, because we are dealing with covariance matrices, but they may occur numerically.
- Once the eigenvalues are ensured to be bounded away from zero, we calculate the log of the determinant, whenever needed, as the sum of logs of the eigenvalues.

B Summary statistics for the variables

B.1 ETF returns and view signals

Variable	mean	std	median	25th	75th	counts
ret_e	-0.001	0.043	-0.000	-0.021	0.019	2,562
ret	-0.001	0.043	0.000	-0.019	0.019	2,562
ff3_alpha	-0.003	0.043	-0.000	-0.025	0.018	2,548
ff3_beta	0.021	0.053	0.016	-0.009	0.055	2,548
mom12	-0.011	0.102	-0.018	-0.061	0.025	2,536
vol_month_signal	-0.041	0.026	-0.040	-0.058	-0.025	2,548
momrev_signal	0.005	0.080	0.008	-0.021	0.043	2,529
ltrev_signal	-0.014	0.512	0.126	-0.093	0.244	2,280
strev_signal	0.001	0.042	-0.000	-0.019	0.021	2,548

Table 7: ETF signals and returns. All are monthly values.

B.2 Macroeconomic variables from FRED

Variable	mean	std	median	25th	75th	counts
macro_ts_RPI	0.002	0.024	0.002	-0.000	0.004	176
macro_ts_W875RX1	0.001	0.009	0.002	-0.000	0.004	176
macro_ts_DPCERA3M086SBEA	0.002	0.014	0.002	0.000	0.003	176
macro_ts_CMRMTSPLx	0.001	0.015	0.001	-0.002	0.006	176
macro_ts_RETAILx	0.003	0.024	0.003	-0.002	0.008	176
macro_ts_INDPRO	0.000	0.015	0.001	-0.003	0.005	176
macro_ts_IPFPNSS	-0.000	0.015	-0.000	-0.004	0.005	176
macro_ts_IPFINAL	-0.000	0.017	-0.001	-0.004	0.006	176
macro_ts_IPCONGD	-0.000	0.015	-0.001	-0.005	0.005	176
macro_ts_IPDCONGD	0.000	0.054	0.001	-0.011	0.013	176
macro_ts_IPNCONGD	-0.001	0.008	-0.001	-0.005	0.004	176
macro_ts_IPBUSEQ	-0.000	0.027	0.001	-0.006	0.009	176
macro_ts_IPMAT	0.001	0.015	0.002	-0.003	0.006	176
macro_ts_IPDMAT	0.000	0.021	0.002	-0.005	0.007	176
macro_ts_IPNMAT	-0.001	0.016	0.001	-0.005	0.006	176
macro_ts_IPMANSICS	-0.000	0.017	0.000	-0.003	0.005	176
macro_ts_IPB51222S	0.000	0.047	-0.001	-0.028	0.023	176
macro_ts_IPFUELS	0.000	0.027	0.001	-0.010	0.009	176
macro_ts_CUMFNS	-0.002	1.165	0.049	-0.243	0.317	176
macro_ts_HWI	33.864	287.158	36.500	-126.250	191.500	176
macro_ts_HWIURATIO	0.005	0.069	0.005	-0.013	0.030	176
macro_ts_CLF16OV	0.000	0.004	0.001	-0.001	0.002	176
macro_ts_CE16OV	0.000	0.013	0.001	-0.000	0.002	176
macro_ts_UNRATE	-0.001	0.845	0.000	-0.125	0.100	176
macro_ts_UEMPMEAN	0.064	1.385	0.100	-0.600	0.800	176
macro_ts_UEMPLT5	-0.001	0.172	0.001	-0.045	0.054	176
macro_ts_UEMP5TO14	-0.001	0.151	-0.008	-0.040	0.032	176
macro_ts_UEMP15OV	0.002	0.090	-0.006	-0.035	0.019	176

Variable	mean	std	median	25th	75th	counts
macro_ts_UEMP15T26	-0.001	0.144	-0.008	-0.058	0.052	176
macro_ts_UEMP27OV	0.003	0.073	-0.003	-0.034	0.033	176
macro_ts_CLAIMSx	-0.001	0.216	-0.010	-0.039	0.020	176
macro_ts_PAYEMS	0.000	0.012	0.001	0.001	0.002	176
macro_ts_USGOOD	-0.000	0.011	0.001	-0.001	0.003	176
macro_ts_CES1021000001	-0.001	0.013	0.002	-0.007	0.008	176
macro_ts_USCONS	-0.000	0.014	0.002	-0.002	0.004	176
macro_ts_MANEMP	-0.001	0.009	0.001	-0.001	0.002	176
macro_ts_DMANEMP	-0.001	0.011	0.001	-0.001	0.002	176
macro_ts_NDMANEMP	-0.000	0.007	0.000	-0.001	0.001	176
macro_ts_SRVPRD	0.001	0.012	0.001	0.001	0.002	176
macro_ts_USTPU	0.000	0.010	0.001	-0.000	0.002	176
macro_ts_USWTRADE	-0.000	0.006	0.001	-0.000	0.001	176
macro_ts_USTRADE	0.000	0.013	0.000	-0.001	0.002	176
macro_ts_USFIRE	0.000	0.003	0.001	-0.000	0.002	176
macro_ts_USGOVT	-0.000	0.005	0.000	-0.001	0.001	176
macro_ts_CES0600000007	40.828	0.647	41.000	40.600	41.200	176
macro_ts_AWOTMAN	-0.001	0.142	0.000	-0.100	0.100	176
macro_ts_AWHMAN	41.429	0.672	41.600	41.300	41.900	176
macro_ts_HOUST	6.892	0.335	6.986	6.625	7.133	176
macro_ts_HOUSTNE	4.596	0.340	4.620	4.357	4.838	176
macro_ts_HOUSTMW	4.992	0.313	5.063	4.764	5.227	176
macro_ts_HOUSTS	6.215	0.346	6.304	5.976	6.466	176
macro_ts_HOUSTW	5.439	0.418	5.499	5.121	5.787	176
macro_ts_PERMIT	6.940	0.342	7.033	6.681	7.186	176
macro_ts_PERMITNE	4.689	0.330	4.749	4.434	4.907	176
macro_ts_PERMITMW	5.052	0.282	5.133	4.828	5.254	176
macro_ts_PERMITS	6.257	0.347	6.342	6.013	6.496	176
macro_ts_PERMITW	5.487	0.421	5.585	5.204	5.829	176
macro_ts_ACOGNO	0.001	0.029	0.003	-0.007	0.013	176
macro_ts_AMDMNOx	0.001	0.052	0.004	-0.020	0.025	176
macro_ts_ANDENOx	-0.000	0.108	-0.004	-0.054	0.048	176
macro_ts_AMDMUOx	0.001	0.010	0.002	-0.004	0.007	176
macro_ts_BUSINVx	0.002	0.006	0.003	0.001	0.006	176
macro_ts_ISRATIOx	-0.000	0.030	0.000	-0.010	0.010	176
macro_ts_M1SL	0.000	0.125	0.000	-0.008	0.007	176
macro_ts_M2SL	0.000	0.006	-0.000	-0.002	0.002	176
macro_ts_M2REAL	0.004	0.008	0.003	0.001	0.006	176
macro_ts_BOGMBASE	0.000	0.038	-0.000	-0.014	0.013	176
macro_ts_TOTRESNS	0.000	0.101	0.000	-0.028	0.026	176
macro_ts_NONBORRES	0.000	2.059	-0.002	-0.034	0.027	176
macro_ts_BUSLOANS	-0.000	0.014	0.001	-0.004	0.004	176
macro_ts_REALLN	0.000	0.006	0.000	-0.001	0.002	176
macro_ts_NONREVSL	-0.000	0.009	-0.000	-0.001	0.001	176
macro_ts_CONSPI	0.000	0.004	0.000	-0.000	0.001	176
macro_ts_S&P 500	0.007	0.040	0.014	-0.006	0.029	176
macro_ts_S&P: indust	0.008	0.039	0.016	-0.004	0.030	176
macro_ts_S&P div yield	-0.003	0.097	-0.015	-0.043	0.024	176
macro_ts_S&P PE ratio	0.003	0.066	0.004	-0.016	0.026	176
macro_ts_FEDFUNDS	-0.029	0.157	0.000	-0.010	0.010	176
macro_ts_CP3Mx	-0.025	0.181	0.000	-0.030	0.020	176

Variable	mean	std	median	25th	75th	counts
macro_ts_TB3MS	-0.028	0.173	0.000	-0.020	0.023	176
macro_ts_TB6MS	-0.027	0.160	0.000	-0.030	0.020	176
macro_ts_GS1	-0.027	0.163	0.000	-0.030	0.030	176
macro_ts_GS5	-0.019	0.203	0.000	-0.130	0.093	176
macro_ts_GS10	-0.017	0.204	-0.005	-0.120	0.110	176
macro_ts_AAA	-0.015	0.179	-0.015	-0.110	0.070	176
macro_ts_BAA	-0.017	0.219	-0.015	-0.120	0.080	176
macro_ts_COMPAPFFx	0.174	0.262	0.090	0.040	0.210	176
macro_ts_TB3SMFFM	-0.114	0.253	-0.050	-0.090	-0.020	176
macro_ts_TB6SMFFM	-0.031	0.249	0.000	-0.030	0.053	176
macro_ts_T1YFFM	0.075	0.292	0.085	0.017	0.220	176
macro_ts_T5YFFM	0.933	0.751	0.930	0.550	1.462	176
macro_ts_T10YFFM	1.656	1.043	1.710	1.050	2.442	176
macro_ts_AAAFFM	3.274	1.292	3.470	2.375	4.197	176
macro_ts_BAAFFM	4.358	1.536	4.530	3.150	5.220	176
macro_ts_TWEXAFEGSMTHx	0.001	0.016	0.001	-0.009	0.009	176
macro_ts_EXSZUSx	-0.002	0.022	-0.002	-0.015	0.012	176
macro_ts_EXJPUSx	-0.000	0.022	-0.000	-0.013	0.013	176
macro_ts_EXUSUKx	-0.002	0.022	-0.000	-0.013	0.012	176
macro_ts_EXCAUSx	0.000	0.020	-0.001	-0.011	0.012	176
macro_ts_WPSFD49207	-0.000	0.009	-0.000	-0.005	0.005	176
macro_ts_WPSFD49502	-0.000	0.012	-0.000	-0.007	0.006	176
macro_ts_WPSID61	0.000	0.010	0.000	-0.005	0.005	176
macro_ts_WPSID62	0.000	0.042	0.001	-0.023	0.024	176
macro_ts_OILPRICEx	-0.000	0.137	-0.009	-0.072	0.059	176
macro_ts_PPICMM	-0.000	0.040	-0.001	-0.026	0.024	176
macro_ts_CPIAUCSL	0.000	0.003	-0.000	-0.001	0.001	176
macro_ts_CPIAPPSL	0.000	0.008	0.000	-0.004	0.004	176
macro_ts_CPITRNSL	0.000	0.017	-0.001	-0.009	0.009	176
macro_ts_CPIMEDSL	0.000	0.002	-0.000	-0.001	0.002	176
macro_ts_CUSR0000SAC	0.000	0.007	-0.000	-0.003	0.004	176
macro_ts_CUSR0000SAD	0.000	0.004	-0.000	-0.001	0.001	176
macro_ts_CUSR0000SAS	0.000	0.001	-0.000	-0.001	0.001	176
macro_ts_CPIULFSL	0.000	0.003	-0.000	-0.002	0.002	176
macro_ts_CUSR0000SA0L2	0.000	0.004	-0.000	-0.002	0.002	176
macro_ts_CUSR0000SA0L5	0.000	0.003	-0.000	-0.001	0.002	176
macro_ts_PCEPI	0.000	0.002	-0.000	-0.001	0.001	176
macro_ts_DDURRG3M086SBEA	0.000	0.004	-0.000	-0.002	0.002	176
macro_ts_DNDGRG3M086SBEA	-0.000	0.008	0.000	-0.004	0.004	176
macro_ts_DSERRG3M086SBEA	0.000	0.001	-0.000	-0.001	0.001	176
macro_ts_CES0600000008	-0.000	0.003	-0.000	-0.002	0.001	176
macro_ts_CES2000000008	-0.000	0.007	-0.000	-0.003	0.003	176
macro_ts_CES3000000008	-0.000	0.003	-0.000	-0.002	0.002	176
macro_ts_UMCSENTx	-0.119	4.216	0.200	-2.200	2.650	176
macro_ts_DTCOLNVHFNM	-0.000	0.029	0.000	-0.004	0.004	176
macro_ts_DTCTHFNM	-0.000	0.035	0.000	-0.003	0.003	176
macro_ts_INVEST	0.000	0.011	0.000	-0.005	0.004	176
macro_ts_VIXCLSx	-0.001	0.296	0.041	-0.141	0.162	176