1 Introduction

This is a short report about the results of the implemented experiments. Since the main interest of the thesis is the performance of end-to-end neural networks (see later) I will only mention them.

2 End-to-end neural networks

End-to-end NNs (neural networks) in my thesis are NNs that can both formulate beliefs about future asset performance (expected returns and covariance matrix) and use them as input into a convex optimization problem. The novice of this approach is that everything is done 'under the hood' of a neural network. In contrast to the old approach of separately estimating expected returns and covariance matrix.

The general architecture of an end-to-end NN is shown in the figure below. Please note that as input we are using a tensor of dimension $(n, 1, 50, 5)$. n represents sample size, 1 is the number of channels used (in our case we are using only information about asset daily returns), 50 is a lookback period (amount of days of daily returns in the past we are using) and 5 is the number of used assets.

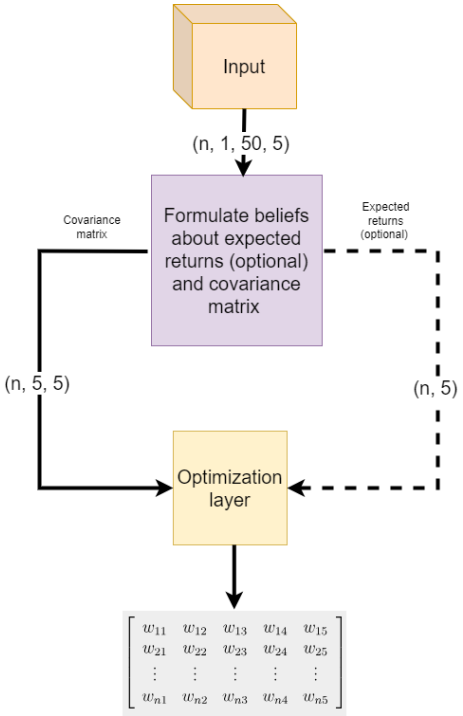

Figure 1: E2E NN general architecture.

To formulate beliefs about assets' expected returns and covariance matrix I am using different NN architectures: fully connected, convolutional, recurrent, and long-short-term memory models.

The optimization layer is the cvxpylayers (https://github.com/cvxgrp/cvxpylayers) class, which solves an optimization problem in an NN-compatible way.

As for the optimization problems originally I used the following

$$\begin{aligned} \underset{w}{\text{maximize}} \quad & r^T w - \gamma w^T C^{\frac{1}{2}} w - \alpha \left\| w \right\|_2^2 \\ \text{subject to} \quad & \sum_{i=1}^{n} w_i = 1, \end{aligned} \tag{1}$$

where r is the vector of NN beliefs about asset returns, C is the matrix of NN predictions for asset covariance, γ is a learnable (by the NN) risk aversion parameter, and α is a learnable regularization parameter.

But I noticed that the regularization term $\alpha \left\| w \right\|_2^2$ is not strong enough to prevent big allocations in a single asset, so I simplified the optimization problem to the following.

$$\begin{aligned} \underset{w}{\text{maximize}} \quad & r^T w - w^T C^{\frac{1}{2}} w \\ \text{subject to} \quad & \sum_{i=1}^{n} w_i = 1, \\ & -1 \leq w_i \leq 1, \ \forall i \in \{1, \ldots, n\}. \end{aligned} \tag{2}$$

A concrete example of an NN with this optimization problem is shown on the image below.

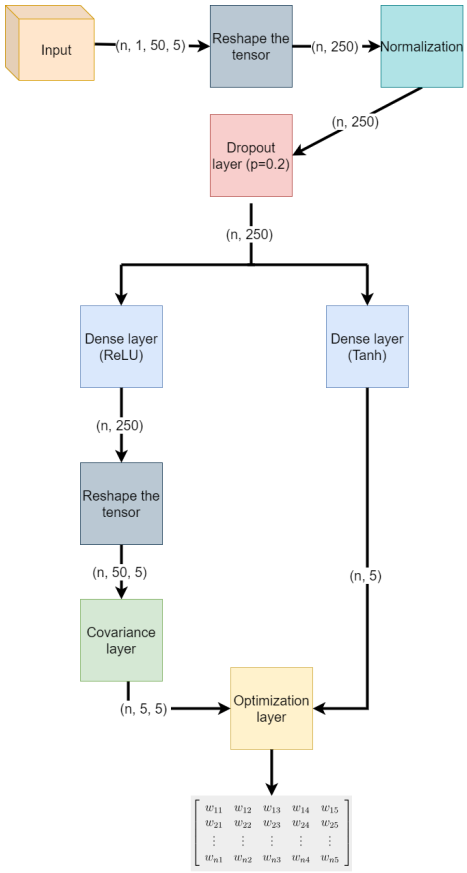

Figure 2: An NN with Optimization Problem 2.

I wanted to test the NNs performance in the context when they are not supposed to estimate expected returns, so also tested the following optimization problem.

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & w^T C^{\frac{1}{2}} w \\ \text{subject to} \quad & \sum_{i=1}^{n} w_i = 1, \\ & -1 \leq w_i \leq 1, \ \forall i \in \{1, \ldots, n\}. \end{aligned} \tag{3}$$

A concrete example of an NN with this optimization problem is shown on the image below.

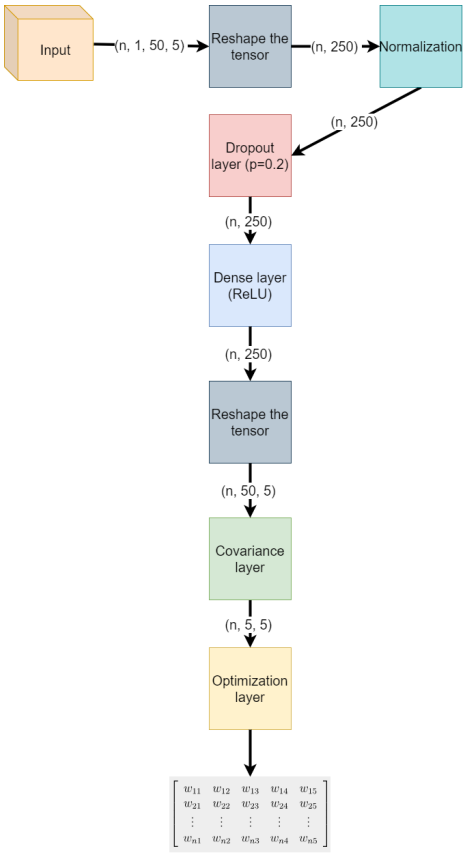

Figure 3: An NN with Optimization Problem 3.

3 Model training and backtesting

As data for the experiments, I used daily returns of VWO, SPY, VNQ, LQD, and DBC from December 2007 to December 2023.

To see the influence of the data timing on model training and backtesting performance, I subdivided the original experiment into 2 subparts. The one that uses the whole original period (2007, 2023), and the one that uses only the (2007, 2020) period. Let's name the sub-experiments ExperimentA and ExperimentB respectively.

For each of the sub-experiments the dataset was divided into training and 3 years of backtesting datasets. So we obtained the [2007, 2020) training period and [2020, 2023] backtesting period for ExperimentA. And the [2007, 2017) training period and [2017, 2020] backtesting period for ExperimentB.

3.1 Model training

The training subset of each experiment dataset is further subdivided into the actual training set, which will be used for NN training and validation set in 80 to 20 proportion. We measure NN performance on the validation subset to select the optimal architecture (optimal amount of layer, hyperparameters, location of layer, and so on).

Each model was trained to maximize the Sharpe ratio of the proposed portfolio and minimize a regularization parameter, which is individual for each NN architecture.

Since the regularization parameter depends on the architecture, the loss functions are incomparable. I am using the following methodology to select between optimal portfolios.

1. Compute the average Sharpe ratio of the equally weighted portfolio (EWP) on the validation set.
2. Compute the average Sharpe ratio of the portfolio formed by the trained network on the validation set.
3. Compute *Validation factor*. Validation factor $= \frac{\text{Sharpe ratio of NN portfolio}}{\text{Sharpe ratio of EWP}}$, computed using data from validation set.

In general Validation factor can be seen as a measure of the ability of the NN to construct a good portfolio for the near future. I expected that it would be strongly correlated to the results of the backtest.

An interesting point noticed during models training is that the Validation factor extremely depends on the training period. Below you can see the results of training for the same architecture using different periods.

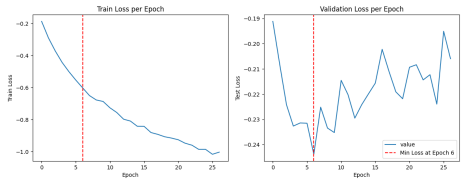

Figure 4: Result of DenseNetMinVar training for ExperimentA.

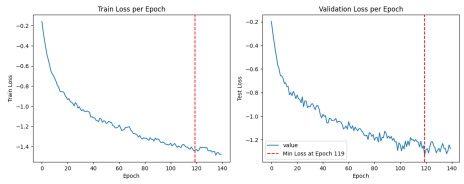

Figure 5: Result of DenseNetMinVar training for ExperimentB.

For the NN from Figure 3.1 Validation factor is 1.14. For the NN from Figure 3.1 Validation factor is 52.35.

4 Backtesting

For each experiment, the backtesting period lasts for 3 years. During backtesting the used model is retrained each year. For example, if backtest is run in the context of ExperimentA using LstmNetMinVar (will be described in the thesis report), from 2020 to 2021 the model trained in the 2007-2020 period is used, and from 2021 to 2022 the model trained on 2007-2021 period and for 2022-2023 the model trained on 2007-2022 period.

Overall the results of backtests are not very promising. From the 8 architectures trained only 1 was able to beat equally weighted portfolio on backtest. None of them was able to bite SPY. The validation factor seems to be not a good predictor of future network performance.

I am currently rechecking the results. But to this, I need to retrain all the models. As technically this means to retrain 48 models (8 architectures * 2 different training periods * 3 retraining dates). This will take some time. I've decided to report on the current findings.