A APPENDIX

This appendix is organized as follows. Section A.1 demonstrates a claim made in section 2 regarding the fact that the solution of $\nu_t = f(\nu_t, \boldsymbol{w}'_t, \boldsymbol{w}_t)$ can be obtained using a bisection method. Section A.2 presents proofs to the two propositions in section 3. Section A.3 presents further details on the correlation layer in Zhang et al. (2020) and its two deficiencies. Section A.4 presents further details on the augmented policy network architecture used to accelerate training. Section A.5 presents our hyper-parameter ranges and final selection. Finally, section A.6 presents a set of additional results.

A.1 SOLVING $\nu = f(\nu)$

In order to apply the bisection method to solve $\nu = f(\nu)$, we will make use of the following proposition.

Proposition A.1. *For any $0 < c_s < 1$ and $0 < c_p < 1$, the function $g(\nu) := \nu - f(\nu)$ is strictly increasing on $[0, 1]$ with $g(0) < 0$ and $g(1) > 0$.*

Proof. Recalling that $f(\nu, \boldsymbol{w}', \boldsymbol{w}) := 1 - c_s \sum_{i=1}^{m} (w'^i - \nu w^i)^+ - c_p \sum_{i=1}^{m} (\nu w^i - w'^i)^+$, we first obtain the two bounds at $g(0)$ and $g(1)$ as follows:

$$g(0) = 0 - \left(1 - c_s \sum_{i=1}^{m} (w'^i)^+ - c_p \sum_{i=1}^{m} (-w'^i)^+\right) = 0 - 1 + c_s < 0\,,$$

since $c_s < 1$, and

$$g(1) = 1 - \left(1 - c_s \sum_{i=1}^{m} (w'^i - w^i)^+ - c_p \sum_{i=1}^{m} (w^i - w'^i)^+\right) \geq \min(c_s, c_p) \|\boldsymbol{w}'^i - \boldsymbol{w}^i\|_1 > 0\,,$$

since $\min(c_s, c_p) > 0$. We can further establish the convexity of $g(\nu)$, given that it is the sum of convex functions. A careful analysis reveals that $g(\nu)$ is supported at 0 by the plane

$$g(\nu) \geq g(0) + \nu \left(1 - c_s + c_p \sum_{i=1}^{m} \mathbf{1}\{w'^i = 0\} w^i\right)\,,$$

where $\mathbf{1}\{A\}$ is the indicator function that returns 1 if A is true, and 0 otherwise. Hence, by convexity of $g(\nu)$, the fact that this supporting plane is strictly increasing implies that $g(\nu)$ is strictly increasing for all $\nu \geq 0$. □

Given Proposition A.1, we can conclude that a bisection method can be used to find the root of $g(\nu)$, which effectively solves $\nu = f(\nu)$.

A.2 PROOFS OF SECTION 3

We start this section with a lemma that will simplify some of our later derivations.

Lemma A.1. *A neural network architecture capturing a set of functions $\mathcal{B} \subseteq \{B : \mathbb{R}^{m \times h \times d} \rightarrow \mathbb{R}^{m \times h' \times d'}\}$ is permutation invariant in the first coordinate if and only if given any permutation operator σ, we have that $\{\sigma^{-1} \circ B \circ \sigma : B \in \mathcal{B}\} \supseteq \mathcal{B}$.*

Proof. The "only if" follows straightforwardly from the fact that equality between two sets implies that each set is a subset of the other.

Regarding the "if" part, we start with the assumption that

$$\forall \sigma, \{\sigma^{-1} \circ B \circ \sigma : B \in \mathcal{B}\} \supseteq \mathcal{B}\,.$$

Next, we follow with the fact that for all permutation operator σ:

$$\begin{aligned}\{\sigma^{-1} \circ B \circ \sigma : B \in \mathcal{B}\} &\subseteq \{\sigma^{-1} \circ B \circ \sigma : B \in \{\sigma \circ B' \circ \sigma^{-1} : B' \in \mathcal{B}\}\} \\ &= \{\sigma^{-1} \circ \sigma \circ B' \circ \sigma^{-1} \circ \sigma : B' \in \mathcal{B}\} = \mathcal{B}\,,\end{aligned}$$

where we assumed for simplicity of exposition that $h = h'$ and $d = d'$, and exploited the fact that σ^{-1} is also a permutation operator. □

A.2.1 PROOF OF PROPOSITION 3.1

We first clarify that the correlation layer is associated with the following set of functions (see Procedure 1):

$$\mathcal{B} := \{B_{w,b} : w \in \mathbb{R}^{(m+1)\times d}, b \in \mathbb{R}\}$$

where

$$B_{w,b}(\mathcal{T})[i,:,1] := \left(\mathcal{T}[i,:,:]\bullet(\vec{1}w_0^\top) + \sum_{j=1}^{m}\mathcal{T}[j,:,:]\bullet(\vec{1}w_j^\top)\right)\vec{1} + b, \quad \forall\, i = 1, ..., m,$$

with $\bullet$ denoting the Hadamard (element-wise) product.

Let σ (associated with the bijection π) be an first coordinate (i.e. asset) permutation operator. For any correlation layer function $B_{w,b} \in \mathcal{B}$, one can construct a new set of parameters $w'_0 := w_0$, $w'_j := w_{\pi(j)}$, for all $j = 1, \ldots, m$, and $b' := b$ such that for all input tensor $\mathcal{T}$, we have that for all i:

$$B_{w',b'}(\sigma(\mathcal{T}))[i,:,1] = \left(\sigma(\mathcal{T})[i,:,:]\bullet(\vec{1}w_0^\top) + \sum_{j=1}^{m}\sigma(\mathcal{T})[j,:,:]\bullet(\vec{1}w_{\pi(j)}^\top)\right)\vec{1} + b \tag{A.4}$$

$$= \left(\mathcal{T}[\pi(i),:,:]\bullet(\vec{1}w_0^\top) + \sum_{j=1}^{m}\mathcal{T}[\pi(j),:,:]\bullet(\vec{1}w_{\pi(j)}^\top)\right)\vec{1} + b \tag{A.5}$$

$$= \left(\mathcal{T}[\pi(i),:,:]\bullet(\vec{1}w_0^\top) + \sum_{j'=1}^{m}\mathcal{T}[j',:,:]\bullet(\vec{1}w_{j'}^\top)\right)\vec{1} + b. \tag{A.6}$$

Hence,

$$\sigma^{-1}(B_{w',b'}(\sigma(\mathcal{T})))[i,:,1] = \left(\mathcal{T}[i,:,:]\bullet(\vec{1}w_0^\top) + \sum_{j=1}^{m}\mathcal{T}[j,:,:]\bullet(\vec{1}w_j^\top)\right)\vec{1} + b = B_{w,b}(\mathcal{T})[i,:,1]\,.$$

We can therefore conclude that $\{\sigma^{-1}\circ B\circ\sigma : B \in \mathcal{B}\} \supseteq \mathcal{B}$. Based on Lemma A.1, we conclude that $\mathcal{B}$ is permutation invariant in the first coordinate.

A.2.2 PROOF OF PROPOSITION 3.2

To prove Proposition 3.2, we demonstrate that all blocks used in the WaveCorr architecture are permutation invariant in the first coordinate (Steps 1 to 3). We then show that permutation invariance in the first coordinate is preserved under composition (Step 4). Finally, we can conclude in Step 5 that WaveCorr is asset permutation invariant.

Step 1 - Dilated convolution, Causal convolution, Sum, and 1×1 convolution are permutation invariant in the first coordinate: The functional class of a dilated convolution, a causal convolution, a sum, and a 1×1 convolution block all have the form:

$$\mathcal{B} := \{B_g : g \in \mathcal{G}\},$$

where

$$B_g(\mathcal{T})[i,:,:] := g(\mathcal{T}(i,:,:)), \quad \forall\, i = 1, ..., m,$$

for some set of functions $\mathcal{G} \subseteq \{G : \mathbb{R}^{1\times h\times d} \to \mathbb{R}^{1\times h\times d'}\}$. In particular, in the case of dilated, causal, and 1×1 convolutions, this property follows from the use of 1×3, $1 \times [h - 28]$, and 1×1 kernels respectively. Hence, for any $g \in \mathcal{G}$, we have that:

$$\sigma^{-1}(B_g(\sigma(\mathcal{T}))) = \sigma^{-1}(\sigma(B_g(\mathcal{T}))) = B_g(\mathcal{T})\,,$$

which implies that $\{\sigma^{-1}\circ B\circ\sigma : B \in \mathcal{B}\} = \mathcal{B}$.

Step 2 - Relu and dropout are permutation invariant in all coordinates: We first clarify that Relu and dropout on a tensor in $\mathbb{R}^{m\times h\times d}$ are singleton sets of functions:

$$\mathcal{B} := \{B_g\}$$

where $g : \mathbb{R} \to \mathbb{R}$ and $B_g(\mathcal{T})[i,j,k] := g(\mathcal{T}[i,j,k])$. In particular, in the case of Relu, we have:

$$B_g(\mathcal{T})[i,j,k] := \max(0, \mathcal{T}[i,j,k])\,,$$

while, for dropout we have:

$$B_g(\mathcal{T})[i,j,k] := \mathcal{T}[i,j,k]\,,$$

since a dropout block acts as a feed through operator. Hence, we naturally have that:

$$\sigma^{-1}(B_g(\sigma(\mathcal{T}))) = \sigma^{-1}(\sigma(B_g(\mathcal{T}))) = B_g(\mathcal{T})\,,$$

which again implies that $\{\sigma^{-1}\circ B\circ\sigma : B \in \mathcal{B}\} = \mathcal{B}$.

Step 3 - Softmax is permutation invariant in the first coordinate: We first clarify that softmax on a vector in $\mathbb{R}^{m\times h\times 1}$ is a singleton set of functions:

$$\mathcal{B} := \{B\}$$

where

$$B(\mathcal{T})[i,j,1] := \frac{\exp(\mathcal{T}[i,j,1])}{\sum_{i'=1}^{m}\exp(\mathcal{T}[i',j,1])}\,.$$

Hence, we have that:

$$B(\sigma(\mathcal{T}))[i,j,1] := \frac{\exp(\mathcal{T}[\pi(i),j,1])}{\sum_{i'=1}^{m}\exp(\mathcal{T}[\pi(i'),j,1])} = \frac{\exp(\mathcal{T}[\pi(i),j,1])}{\sum_{i'=1}^{m}\exp(\mathcal{T}[i',j,1])}\,.$$

This allows us to conclude that:

$$\sigma^{-1}(B(\sigma(\mathcal{T})))[i,j,1] = B(\mathcal{T})[i,j,1]\,.$$

Hence, we conclude that $\{\sigma^{-1}\circ B\circ\sigma : B \in \mathcal{B}\} = \mathcal{B}$.

Step 4 - Permutation invariance in the first coordinate is preserved under composition: Given two asset permutation invariant blocks representing the set of functions $\mathcal{B}_1$ and $\mathcal{B}_2$, one can define the composition block as:

$$\mathcal{B} := \{B_1\circ B_2 : B_1 \in \mathcal{B}_1, B_2 \in \mathcal{B}_2\}\,.$$

We have that for all $B_1 \in \mathcal{B}_1$ and $B_2 \in \mathcal{B}_2$:

$$\begin{aligned} B &= B_1\circ B_2 \\ &= (\sigma^{-1}\circ B_1'\circ\sigma)\circ(\sigma^{-1}\circ B_2'\circ\sigma) \\ &= \sigma^{-1}\circ B_1'\circ B_2'\circ\sigma \\ &= \sigma^{-1}\circ B'\circ\sigma\,, \end{aligned}$$

where $B_1' \in \mathcal{B}_1$ and $B_2' \in \mathcal{B}_2$ come from the definition of asset permutation invariance, and where $B' := B_1'\circ B_2' \in \mathcal{B}$. We therefore have that $\{\sigma^{-1}\circ B\circ\sigma : B \in \mathcal{B}\} \supseteq \mathcal{B}$. Finally, Lemma A.1 allows us to conclude that $\mathcal{B}$ is asset permutation invariant.

Step 5 - WaveCorr is asset permutation invariant: Combing Step 1 to 4 with Proposition 3.1, we arrive at the conclusion that the architecture presented in Figure 2 is asset permutation invariant since it is composed of a sequence of blocks that are permutation invariant in the first coordinate. □

A.3 Correlation Layer in Zhang et al. (2020) Violates Asset Permutation Invariance

Assuming for simplicity that m is odd, the "correlational convolution layer" proposed in Zhang et al. (2020) takes the form of the following set of functions:

$$\mathcal{B} := \{B_{w,b} : \mathcal{W} \in \mathbb{R}^{m\times d\times d}, b \in \mathbb{R}\}$$

where

$$B_{w,b}(\mathcal{T})[i,j,k] := \sum_{\ell=1}^{m}\sum_{k'=1}^{d} \mathcal{T}[i-(m+1)/2+\ell, j, k']\mathcal{W}[\ell,k,k'] + b, \quad \begin{matrix} \forall\, i = 1,\dots,m \\ \forall j = 1,\dots,h \\ \forall k = 1,\dots,d \end{matrix},$$

where $\mathcal{T}[i',:,:] := 0$ for all $i' \notin \{1,\dots,m\}$ to represent a zero padding. Figure A.6 presents an example of this layer when $m = 5$, $h = 1$, and $d = 1$. One can already observe in this figure that correlation information is only partially extracted for some of the assets, e.g. the convolution associated to asset one (cf. first row in the figure) disregards the influence of the fifth asset. While this could perhaps be addressed by using a larger kernel, a more important issue arises with this architecture, namely that the block does not satisfy asset permutation invariance.

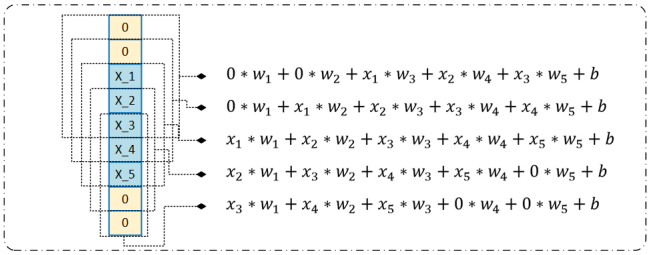


Figure A.6: An example of the correlation layer in Zhang et al. (2020)'s work over 5 assets

Proposition A.2. *The architecture of the correlational convolution layer block used in Zhang et al. (2020) violates permutation invariance in the first coordinate already when $m = 5$, $h = 1$, and $d = 1$.*

Proof. When $m = 5$, $h = 1$, and $d = 1$, we first clarify that the correlational convolution layer from Zhang et al. (2020) is associated with the following set of functions:

$$\mathcal{B} := \{B_{w,b} : w \in \mathbb{R}^5, b \in \mathbb{R}\}$$

where

$$B_{w,b}(\mathcal{T})[i] := \begin{cases} w_3\mathcal{T}[1] + w_4\mathcal{T}[2] + w_5\mathcal{T}[3] + b & \text{if } i = 1 \\ w_2\mathcal{T}[1] + w_3\mathcal{T}[2] + w_4\mathcal{T}[3] + w_5\mathcal{T}[4] + b & \text{if } i = 2 \\ w_1\mathcal{T}[1] + w_2\mathcal{T}[2] + w_3\mathcal{T}[3] + w_4\mathcal{T}[4] + w_5\mathcal{T}[5] + b & \text{if } i = 3 \\ w_1\mathcal{T}[2] + w_2\mathcal{T}[3] + w_3\mathcal{T}[4] + w_4\mathcal{T}[5] + b & \text{if } i = 4 \\ w_1\mathcal{T}[3] + w_2\mathcal{T}[4] + w_3\mathcal{T}[5] + b & \text{if } i = 5 \end{cases},$$

where we shortened the notation $\mathcal{T}[i,1,1]$ to $\mathcal{T}[i]$. Let's consider the asset permutation operator that inverts the order of the first two assets: $\pi(1) = 2$, $\pi(2) = 1$, and $\pi(i) = i$ for all $i \geq 3$. We will prove our claim by contradiction. Assuming that $\mathcal{B}$ is permutation invariant in the first coordinate, it must be that for any fixed values $\bar{w}$ such that $\bar{w}_4 \neq \bar{w}_1$, there exists an associated pair of values (w', b') that makes $B_{w',b'} \equiv \sigma^{-1} \circ B_{\bar{w},0} \circ \sigma$. In particular, the two functions should return the same values for the following three "tensors": $\mathcal{T}_0[i] := 0$, $\mathcal{T}_1[i] := \mathbf{1}\{i = 1\}$, and at $\mathcal{T}_2[i] := \mathbf{1}\{i = 2\}$. The first implies that $b' = 0$ since

$$b' = B_{w',b'}(\mathcal{T}_0)[1] = \sigma^{-1}(B_{\bar{w},0}(\sigma(\mathcal{T}_0)))[1] = 0\,.$$

However, it also implies that:

$$w'_2 = B_{w',0}(\mathcal{T}_1)[2] = \sigma^{-1}(B_{\bar{w},0}(\sigma(\mathcal{T}_1)))[2] = B_{\bar{w},0}(\mathcal{T}_2)[1] = \bar{w}_4$$

and that

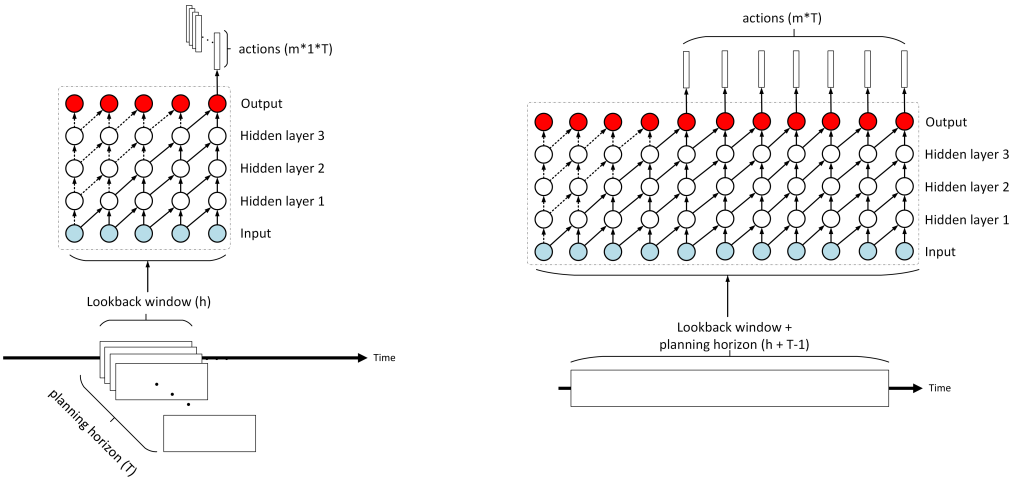
$$w'_2 = B_{w',0}(\mathcal{T}_2)[3] = \sigma^{-1}(B_{\bar{w},0}(\sigma(\mathcal{T}_2)))[3] = B_{\bar{w},0}(\mathcal{T}_1)[3] = \bar{w}_1\,.$$

We therefore have a contradiction since $\bar{w}_4 = w'_2 = \bar{w}_1 \neq \bar{w}_4$ is impossible. We must therefore conclude that $\mathcal{B}$ was not permutation invariant. □

We close this section by noting that this important issue cannot simply be fixed by using a different type of padding, or a larger kernel in the convolution. Regarding the former, our demonstration made no use of how padding is done. For the latter, our proof would still hold given that the fixed parameterization $(\bar{w}, 0)$ that we used would still identify a member of the set of functions obtained with a larger kernel.

A.4 AUGMENTED POLICY NETWORK TO ACCELERATE TRAINING

We detail in this section how the structure of the portfolio management problem (2) can be exploited for a more efficient implementation of a policy network, both in terms of computation time and hardware memory. This applies not only to the implementation of WaveCorr policy network but also policy networks in Jiang et al. (2017) and Zhang et al. (2020). In particular, given a a multiperiod objective as in (2), calculating the gradient $\nabla_\theta SR$ involves the step of generating a sequence of actions $a_0, a_1, ..., a_{T-1}$ from a sample trajectory of states $s_0, s_1, ..., s_{T-1} \in \mathbb{R}^{m\times h\times d}$ over a planning horizon T, where m : the number of assets, h : the size of a lookback window, d : the number of features. The common way of implementing this is to create a tensor $\mathcal{T}$ of dimension $m \times h \times d \times T$ from $s_0, ..., s_{T-1}$ and apply a policy network $\mu_\theta(s)$ to each state s_t in the tensor $\mathcal{T}$ so as to generate each action a_t. Assuming for simplicity of exposition that the state is entirely exogenous, this procedure is demonstrated in Figure A.7(a), where a standard causal convolution with $d = 1$ and kernel size of 2 is applied. In this procedure, the memory used to store the tensor $\mathcal{T}$ and the computation time taken to generate all actions $a_0, ..., a_{T-1}$ grow linearly in T, which become significant for large T. It is possible to apply the policy network $\mu_\theta(s)$ to generate all the actions $a_0, ..., a_{T-1}$ more efficiently than the procedure described in Figure A.7(a). Namely, in our implementation, we exploit the sequential and overlapping nature of sample states $s_0, ..., s_{T-1}$ used to generate the actions $a_0, ..., a_{T-1}$, which naturally arises in the consideration of a multiperiod objective. Recall firstly that each sample state $s_t \in \mathbb{R}^{m\times h\times d}$, $t \in \{0, ..., T-1\}$, is obtained from a sample trajectory, denoted by $\mathcal{S} \in \mathbb{R}^{m\times(h+T-1)\times d}$, where $s_t = \mathcal{S}[:, t+1 : t+h, :]$, $t = 0, ..., T-1$. Thus, between any s_t and s_{t+1}, the last $h-1$ columns in s_t overlap with the first $h-1$ columns in s_{t+1}. The fact that there is a significant overlap between any two consecutive states s_t, s_{t+1} hints already that processing each state s_{t+1} separately from s_t, as shown in Figure A.7(a), would invoke a large number of identical calculations in the network as those that were already done in processing s_t, which is wasteful and inefficient. To avoid such an issue, we take an augmented approach to apply the policy network. The idea is to use a sample trajectory $\mathcal{S}$ directly as input to an augmented policy network $\vec{\mu}_\theta : \mathbb{R}^{m\times(h+T-1)\times d} \to \mathbb{R}^{m\times T}$, which reduces to exactly the same architecture as the policy network $\mu_\theta(s_t)$ when generating only the t-th action. Figure A.7(b) presents this augmented policy network $\vec{\mu}_\theta(S)$ for our example, and how it can be applied to a trajectory $\mathcal{S}$ to generate all actions $a_0, ..., a_{T-1}$ at once. One can observe that the use of an augmented policy network allows the intermediate calculations done for each state s_t (for generating an action a_t) to be reused by the calculations needed for the other states (and generating other actions). With the exact same architecture as the policy network $\mu_\theta(s)$, the augmented policy network $\vec{\mu}_\theta(S)$, which takes a trajectory with width $h + T - 1$ (thus including T many states), would by design generate T output, each corresponds to an action a_t. This not only speeds up the generation of actions $a_0, ..., a_{T-1}$ significantly but also requires far less memory to store the input data, i.e. the use of a tensor with dimension $(m \times (h + T) \times d)$ instead of $m \times h \times d \times T$. The only sacrifice that is made with this approach is regarding the type of features that can be integrated. For instance, we cannot include features that are normalized with respect to the most recent history (as done in Jiang et al. (2017)) given that this breaks the data redundancy between two consecutive time period. Our numerical results however seemed to indicate that such restrictions did not come at a price in terms of performance.



(a) $\mu_\theta(s)$ applied to each state separately

(b) $\vec{\mu}_\theta(S)$ applied to the full trajectory

Figure A.7: Comparison between the use of policy network $\mu_\theta(s)$ and of the augmented policy network $\vec{\mu}_\theta(S)$

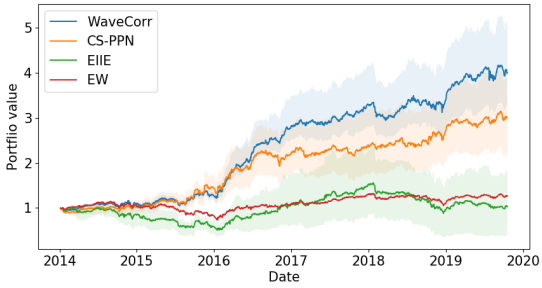
A.5 Hyper-parameters Selection

Table A.6: List of Selected Hyper-parameters.

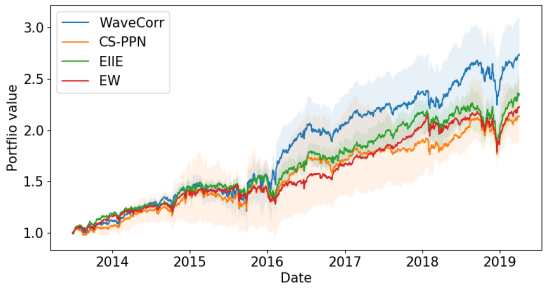
Hyper-parameter	Search range	WaveCorr	CS-PPN	EIIE
Learning rate	$\{5 \times 10^{-5},\ 10^{-4},\ 10^{-3},\ 5 \times 10^{-3}\}$	5×10^{-5}	5×10^{-5}	10^{-4}
Decay rate	$\{0.9999,\ 0.99999,\ 1\}$	0.99999	0.99999	1
Minimum rate	$\{\ 10^{-6},\ 10^{-5}\}$	10^{-5}	10^{-5}	10^{-5}
Planning horizon T	$\{32,\ 64\}$	32	32	32
Look back window size h	$\{32,\ 64\}$	32	32	32
Number of epochs	$[0,\ \infty)$	5000	5000	5000

A.6 ADDITIONAL RESULTS

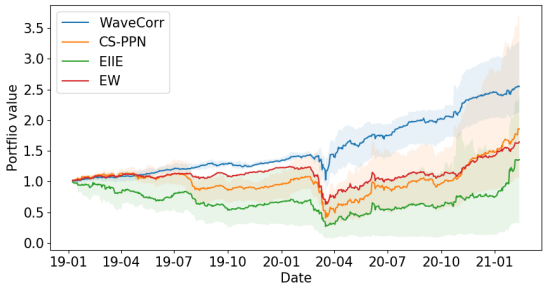
A.6.1 COMPARATIVE STUDY



(a) Can-data



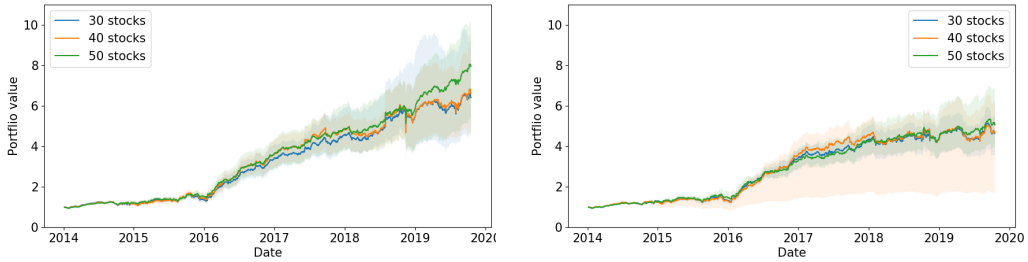
(b) US-data



(c) Covid-data

Figure A.8: Average (solid curve) and range (shaded region) of out-of-sample wealth accumulated by WaveCorr, CS-PPN, EIIE, and EW over 10 experiments using Can-data, US-data, and Covid-data.

A.6.2 SENSITIVITY TO NUMBER OF ASSETS



(a) WaveCorr (b) CS-PPN

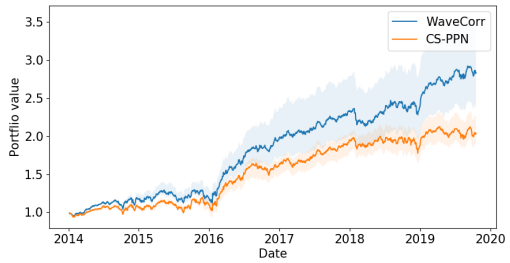
Figure A.9: Average (solid curve) and range (shaded region) of the out-of-sample wealth accumulated, on 10 experiments using Can-data, by WaveCorr and CS-PPN when increasing the number of assets.

A.6.3 PERFORMANCE COMPARISON UNDER MAXIMUM HOLDING CONSTRAINT

In practice, it is often required that the portfolio limits the amount of wealth invested in a single asset. This can be integrated to the risk-averse DRL formulation:

$$\bar{J}_F(\mu_\theta) := \mathbb{E}_{\substack{s_0 \sim F \\ s_{t+1} \sim P(\cdot|s_t, \mu_\theta(s_t))}} [SR(r_0(s_0, \mu_\theta(s_0), s_1), ...)] - \frac{M}{T} \sum_{t=0}^{T-1} \sum_{i=1}^{m} \max(0,\ w_t^i - w_{max})$$

where w_{max} is the maximum weight allowed in any asset, and M is a large constant. This new objective function penalizes any allocation that goes beyond w_{max}, which will encourage μ_θ to respects the maximum weight allocation condition. The commission rates are considered to be $c_s = c_p = 0.5\%$, and the experiments here are done over Can-data using the full set of 70 stocks, with a maximum holding of 20%. The results are summarized in Table A.7 and illustrated in Figure A.10. As noted before, we observe that WaveCorr outperforms CS-PPN with respect to all performance metrics.



(a) Out-of-sample cumulative returns

Figure A.10: Average (solid curve) and range (shaded region) of the out-of-sample wealth accumulated, on 10 experiments using Can-data, by WaveCorr and CS-PPN under maximum holding constraint.

Table A.7: The average (and standard dev.) performances when imposing a maximum holding constraints over 10 random initial NN weights in Can-data.

	Annual return	Annual vol	SR	MDD	Daily hit rate	Turnover
WaveCorr	20% (2%)	13% (0%)	1.55 (0.18)	14% (1%)	53% (1%)	0.17 (0.01)
CS-PPN	13% (1%)	13% (1%)	1.00 (0.15)	15% (2%)	50% (1%)	0.22 (0.03)