Operations Research Letters 51 (2023) 680–686

Contents lists available at ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

WaveCorr: Deep reinforcement learning with permutation invariant convolutional policy networks for portfolio management

Saeed Marzban[a], Erick Delage[a,*,1], Jonathan Yu-Meng Li[b], Jeremie Desgagne-Bouchard[c], Carl Dussault[c]

[a] GERAD & Department of Decision Sciences, HEC Montréal, Montreal, Canada
[b] Telfer School of Management, University of Ottawa, Ottawa, Canada
[c] Evovest, Montreal, Canada

A R T I C L E   I N F O

Article history:
Available online 4 November 2023

Keywords:
Reinforcement learning
Deep learning
Portfolio optimization
Permutation invariance

A B S T R A C T

We present a new portfolio policy convolutional neural network architecture, WaveCorr, for deep reinforcement learning applied to portfolio optimization. WaveCorr is the first to treat asset correlation while preserving "asset invariance property", a new permutation invariance property that significantly increases the stability of performance in problems where input indexing is done arbitrarily. A general theory is also derived for verifying this property in other fields of application. Our experiments show that WaveCorr consistently outperforms other state-of-the-art convolutional architectures.

1. Introduction

In recent years, there has been a growing interest in applying Deep Reinforcement Learning (DRL) to solve dynamic decision problems that are complex in nature. One representative class of problems is that of portfolio management, whose formulation typically requires a large number of continuous state/action variables and a sophisticated form of risk function for capturing the intrinsic complexity of financial markets, trading environment, and investors' preferences. In this paper, we propose a new architecture of DRL for solving portfolio management problems that optimize a Sharpe ratio criteria. While there are several works in the literature that apply DRL for portfolio management problems such as [5,9,11] among others, little has been done to investigate how to improve the design of a Neural Network (NN) in DRL so that it can capture the nature of dependency exhibited in financial data more effectively. In particular, it is known that extracting and exploiting cross-asset dependencies over time is crucial to the performance of portfolio management. The neural network architectures adopted in most existing works, such as Long-Short-Term-Memory (LSTM) or Convolutional Neutral Network (CNN), however, only process input data on an asset-by-asset basis and thus lack a mechanism to capture cross-asset dependency information. The architecture presented in this paper, named as WaveCorr, offers a mechanism to extract the information of both time-series dependency and cross-asset dependency. It is built upon the WaveNet structure [13], which uses dilated causal convolutions at its core, and a new design of correlation block that can process and extract cross-asset information.

In particular, throughout our development, we identify and define an important property that can be used to guide the design of a neural network architecture that takes indexed data as input in order to ensure the stability of its performance. In the context of portfolio optimization, this property is referred to as *asset permutation invariance* and is motivated by the observation that the dependency across assets has a very different nature from the dependency across time. Namely, while the dependency across time is sensitive to the sequential relationship of data, the dependency across assets is not. To put it another way, given a multivariate time series data, the data would not be considered the same if the time indices are permuted, but the data should remain the same if the asset indices are permuted. While this property may appear more than reasonable, as discussed in Section 3, a naive extension of CNN that accounts for both time and asset dependencies can easily fail to satisfy this property. To the best of our knowledge, the only other works that have also considered extracting cross-asset dependency information in DRL for portfolio management are the recent works of [16] and [14]. While Zhang et al.'s work is closer to ours in that it is also built upon the idea of adding a correlation layer to a CNN-like module, its overall architecture is different from ours and, most notably, their design does not follow the property of asset permutation invariance and thus its performance can vary significantly when the ordering of assets changes. As further

[*] Corresponding author.
E-mail address: erick.delage@hec.ca (E. Delage).
[1] 3000 Chemin de la Côte-Sainte-Catherine, Montréal, Québec, H3T 2A7, Canada.

https://doi.org/10.1016/j.orl.2023.10.011
0167-6377/© 2023 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

shown in the numerical section, our architecture, which has a simpler yet permutation invariant structure, outperforms Zhang et al.'s architecture in many aspects. The work of [14] takes a very different direction from ours as it employs a transformer-based, instead of convolution-based, architecture. It also appears that its layer responsible for sharing information among assets is non-parametric (see section 3.3), which implies that it cannot adapt to the specific correlations present in the training data.

Overall, the contribution of this paper is three fold. First, we introduce a new permutation invariance property for policy network architectures, referred to as asset permutation invariance in the case of a portfolio policy network, and derive general theory for verifying its applicability in any relevant DRL domain. Second, we design the first CNN based portfolio policy network, named WaveCorr, that accounts for asset dependencies in a way that preserves this type of invariance. This achievement relies on the design of an innovative permutation invariant correlation processing layer. Third, and most importantly, we present evidence that WaveCorr significantly outperforms two other state-of-the-art policy network architectures using data from both Canadian (TSX) and American (S&P 500) stock markets. Specifically, our new architecture leads to an impressive 5%-25% absolute improvement in terms of average annual return, up to more than 200% relative improvement in average Sharpe ratio, and reduces, during the period of 2019-2020 (i.e. the Covid-19 pandemic), by 16% the maximum daily portfolio loss compared to the best competing method. Using the same set of hyper-parameters, we also measured an improvement of up to a factor of 5 in the stability of performance under random choices of initial asset ordering and weights, and observe that WaveCorr consistently outperforms our benchmarks under a number of variations of the model: including the number of available assets, the size of transaction costs, etc. Overall, we interpret this empirical evidence as strong support regarding the potential impact of the WaveCorr architecture on automated portfolio management practices, and, more generally, as supporting the claim that asset permutation invariance is an important property to take into account in the design of NN architectures when stability of performance is a concern.

The rest of the paper unfolds as follows. Section 2 presents the portfolio management problem and risk averse reinforcement learning formulation. Section 3 introduces the new property of "asset permutation invariance" for portfolio policy network and presents a new network architecture based on convolution networks that satisfies this property. Finally, Section 4 presents the findings from our numerical experiments. We finally conclude in Section 5.

2. Problem statement

2.1. Portfolio management problem

The portfolio management problem consists of optimizing the reallocation of wealth among many available financial assets including stocks, commodities, equities, currencies, etc. at discrete points in time. In this paper, we assume that there are m risky assets in the market, hence the portfolio is controlled based on a set of weights $\boldsymbol{w}_t \in \mathbb{W} := \{\boldsymbol{w} \in \mathbb{R}_+^m | \sum_{i=1}^m w^i = 1\}$, which describes the proportion of wealth invested in each asset. Portfolios are rebalanced at the beginning of each period $t = 0, 1, ..., T-1$, which will incur proportional transaction costs for the investor, i.e. commission rates are of c_s and c_p, respectively. We follow [6] to model the evolution of the portfolio value and weights (see Fig. 1). Specifically, during period t the portfolio value and weights start at p_{t-1} and $\boldsymbol{w}_{t-1}$, and the changes in stock prices, captured by a random vector of asset returns $\boldsymbol{\xi}_t \in \mathbb{R}^m$, affect the

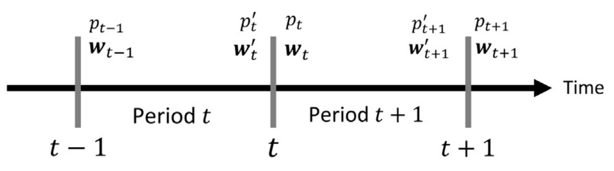
p_{t-1} $\boldsymbol{w}_{t-1}$
p'_t $\boldsymbol{w}'_t$
p_t $\boldsymbol{w}_t$
p'_{t+1} $\boldsymbol{w}'_{t+1}$
p_{t+1} $\boldsymbol{w}_{t+1}$
Time
Period t
Period $t+1$
$t-1$
t
$t+1$

Fig. 1. Portfolio evolution through time.

end of period portfolio value $p'_t := p_{t-1}\boldsymbol{\xi}_t^\top \boldsymbol{w}_{t-1}$, and weight vector $\boldsymbol{w}'_t := (p_{t-1}/p'_t)\boldsymbol{\xi}_t \bullet \boldsymbol{w}_{t-1}$, where $\bullet$ is a term-wise product. The investor then decides on a new distribution of his wealth $\boldsymbol{w}_t$, which triggers the following transaction cost:

$$c_s \sum_{i=1}^m (p'_t w'^i_t - p_t w^i_t)^+ + c_p \sum_{i=1}^m (p_t w^i_t - p'_t w'^i_t)^+ .$$

Denoting the net effect of transaction costs on portfolio value with $\nu_t := p_t/p'_t$, as reported in [7] one finds that ν_t is the solution of the following equations:

$$\nu_t = f(\nu_t, \boldsymbol{w}'_t, \boldsymbol{w}_t) := 1 - c_s \sum_{i=1}^m (w'^i_t - \nu_t w^i_t)^+ - c_p \sum_{i=1}^m (\nu_t w^i_t - w'^i_t)^+ .$$

This, in turn, allows us to express the portfolio's log return during the $t+1$-th period as:

$$\zeta_{t+1} := \ln(p'_{t+1}/p'_t) = \ln(\nu_t p'_{t+1}/p_t) = \ln(\nu_t(\boldsymbol{w}'_t, \boldsymbol{w}_t)) + \ln(\boldsymbol{\xi}_{t+1}^\top \boldsymbol{w}_t)$$

where we make explicit the influence of $\boldsymbol{w}'_t$ and $\boldsymbol{w}_t$ on ν_t.

We note that in [6], the authors suggest to approximate ν_t using an iterative procedure. However, we actually show in Appendix A.1 that ν_t can easily be identified with high precision using the bisection method.

2.2. Risk-averse reinforcement learning formulation

In this section, we formulate the portfolio management problem as a Markov Decision Process (MDP) denoted by $(\mathcal{S}, \mathcal{A}, r, P)$. In this regard, the agent (i.e. an investor) interacts with a stochastic environment by taking an action $a_t \equiv \boldsymbol{w}_t \in \mathbb{W}$ after observing the state $s_t \in \mathcal{S}$ composed of a window of historical market observations, which include the latest stock returns $\boldsymbol{\xi}_t$, along with the final portfolio composition of the previous period $\boldsymbol{w}'_t$. This action results in the immediate stochastic reward that takes the shape of an approximation of the realized log return, i.e. $r_t(s_t, a_t, s_{t+1}) := \ln(f(1, \boldsymbol{w}'_t, \boldsymbol{w}_t)) + \ln(\boldsymbol{\xi}_{t+1}^\top \boldsymbol{w}_t) \approx \ln(\nu(\boldsymbol{w}'_t, \boldsymbol{w}_t)) + \ln(\boldsymbol{\xi}_{t+1}^\top \boldsymbol{w}_t)$, for which a derivative is easily obtained. Finally, P captures the assumed Markovian transition dynamics of the stock market and its effect on portfolio weights: $P(s_{t+1}|s_0, a_0, s_1, a_1, ..., s_t, a_t) = P(s_{t+1}|s_t, a_t)$.

Following the works of [11] and [1] on risk averse DRL, our objective is to identify a deterministic trading policy μ_θ (parameterized by θ) that maximizes the expected value of the Sharpe ratio measured on T-periods log return trajectories generated by μ_θ. Namely:

$$\max_\theta \; J_F(\mu_\theta) := \mathbb{E}_{\substack{s_0 \sim F \\ s_{t+1} \sim P(\cdot|s_t, \mu_\theta(s_t))}} \Big[SR\Big(r_0(s_0, \mu_\theta(s_0), s_1), r_1(s_1, \mu_\theta(s_1), s_2), ..., r_{T-1}(s_{T-1}, \mu_\theta(s_{T-1}), s_T) \Big) \Big] \quad (1)$$

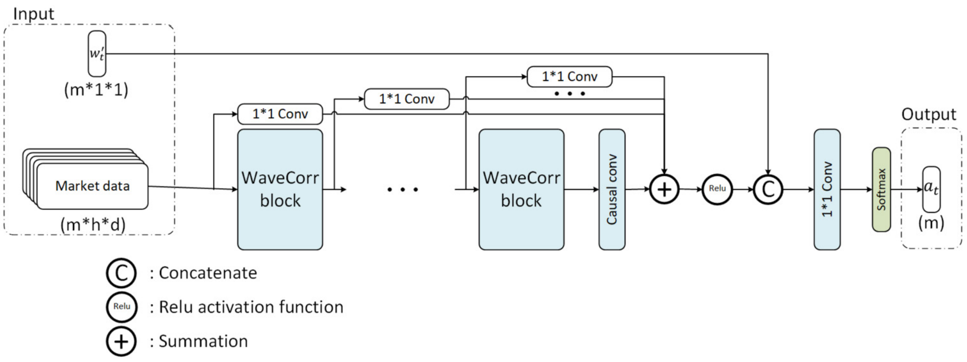

Fig. 2. The architecture of the WaveCorr policy network.

where F is some fixed distribution and

$$SR(r_{0:T-1}) := \frac{(1/T)\sum_{t=0}^{T-1} r_t}{\sqrt{(1/(T-1))\sum_{t=0}^{T-1}(r_t - (1/T)\sum_{t=0}^{T-1} r_t)^2}}.$$

The choice of using the Sharpe ratio of log returns is motivated by modern portfolio theory (see [10]), which advocates a balance between expected returns and exposure to risks, and where it plays the role of a canonical way of exercising this trade-off [12]. While it is inapt of characterizing downside risk, it is still considered a “gold standard of performance evaluation” by the financial community [3]. In [11], the trajectory-wise Sharpe ratio is used as an estimator of the instantaneous one in order to facilitate its use in RL. A side-benefit of this estimator is to offer some control on the variations in the evolution of the portfolio value which can be reassuring for the investor.

In the context of our portfolio management problem, since s_t is composed of an exogenous component s_t^{exo} which includes ξ_t and an endogenous state $\boldsymbol{w}'_t$ that becomes deterministic when a_t and s_{t+1}^{exo} are known, we have that:

$$\begin{aligned} &J_F(\mu_\theta) \\ &:= \mathbb{E}_{\substack{s_0\sim F\\ s_{t+1}\sim P(\cdot|s_t,\beta(s_t))}}\Big[SR\Big(r_0(\bar{s}_0^\theta, \mu_\theta(\bar{s}_0^\theta), \bar{s}_1^\theta), r_1(\bar{s}_1^\theta, \mu_\theta(\bar{s}_1^\theta), \bar{s}_2^\theta), \\ &\qquad \ldots, r_{T-1}(\bar{s}_{T-1}^\theta, \mu_\theta(\bar{s}_{T-1}^\theta), \bar{s}_T^\theta)\Big)\Big] \end{aligned}$$

where $\beta(s_t)$ is an arbitrary policy, and where the effect of μ_θ on the trajectory can be calculated using

$$\bar{s}_t^\theta := \left(s_t^{exo}, \frac{\xi_t \bullet \mu_\theta(\bar{s}_{t-1}^\theta)}{\xi_t^\top \mu_\theta(\bar{s}_{t-1}^\theta)}\right),$$

for $t \geq 1$, while $\bar{s}_0^\theta := s_0$. Hence,

$$\begin{aligned} \nabla_\theta J_F(\mu_\theta) :=& \mathbb{E}\Big[\nabla_\theta SR\Big(r_0(\bar{s}_0^\theta, \mu_\theta(\bar{s}_0^\theta), \bar{s}_1^\theta), \ldots, \\ & r_{T-1}(\bar{s}_{T-1}^\theta, \mu_\theta(\bar{s}_{T-1}^\theta), \bar{s}_T^\theta)\Big)\Big], \end{aligned} \tag{2}$$

where $\nabla_\theta SR$ can be obtained by backpropagation using the chain rule. This leads to the following stochastic gradient step:

$$\begin{aligned} \theta_{k+1} =& \theta_k + \alpha \nabla_\theta SR\Big(r_0(\bar{s}_0^\theta, \mu_\theta(\bar{s}_0^\theta), \bar{s}_1^\theta), \ldots, \\ & r_{T-1}(\bar{s}_{T-1}^\theta, \mu_\theta(\bar{s}_{T-1}^\theta), \bar{s}_T^\theta)\Big), \end{aligned}$$

with $\alpha > 0$ as the step size.

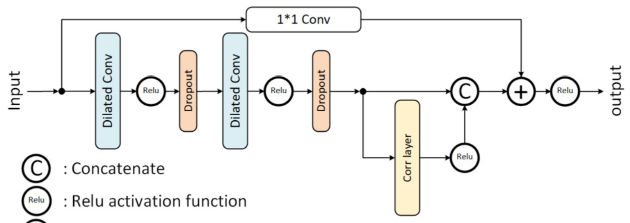

Fig. 3. WaveCorr residual block.

3. The new permutation invariant WaveCorr architecture

There are several considerations that go into the design of the network for the portfolio policy network μ_θ. First, the network should have the capacity to handle long historical time series data, which allows for extracting long-term dependencies across time. Second, the network should be flexible in its design for capturing dependencies across a large number of available assets. Third, the network should be parsimoniously parameterized to achieve these objectives without being prone to overfitting. To this end, the WaveNet structure [13] offers a good basis for developing our architecture and was employed in [16]. Unfortunately, a direct application of WaveNet in portfolio management struggles at processing the cross-asset correlation information. This is because the convolutions embedded in the WaveNet model are 1D and extending to 2D convolutions increases the number of parameters in the model, which makes it more prone to the issue of over-fitting, a notorious issue particularly in RL. Most importantly, naive attempts at adapting WaveNet to account for such dependencies (as done in [16]) can make the network become sensitive to the ordering of the assets in the input data, an issue that we will revisit below.

We first present the general architecture of WaveCorr in Fig. 2. Here, the network takes as input a tensor of dimension $m \times h \times d$, where m: the number of assets, h: the size of look-back time window, d: the number of channels (number of features for each asset), and generates as output an m-dimensional wealth allocation vector. The WaveCorr blocks, which play the key role for extracting cross time/asset dependencies, form the body of the architecture. In order to provide more flexibility for the choice of h, we define a causal convolution after the sequence of WaveCorr blocks to adjust the receptive field so that it includes the whole length of the input time series. Also, similar to the WaveNet structure, we use skip connections in our architecture.

The design of the WaveCorr residual block in WaveCorr extends a simplified variation [2] of the residual block in WaveNet by adding our new correlation layers (and a Relu, concatenation operations following right after). As shown in Fig. 3, the block includes two layers of dilated causal convolutions followed by Relu activation functions and dropout layers. Having an input of dimensions

Table 1
The structure of the network.

Layer	Input shape	Output shape	Kernel	Activation	Dilation rate
Dilated conv	$(m \times h \times d)$	$(m \times h \times 8)$	(1×3)	Relu	1
Dilated conv	$(m \times h \times 8)$	$(m \times h \times 8)$	(1×3)	Relu	1
Corr layer	$(m \times h \times 8)$	$(m \times h \times 1)$	$([m+1] \times 1)$	Relu	-
Dilated conv	$(m \times h \times 9)$	$(m \times h \times 16)$	(1×3)	Relu	2
Dilated conv	$(m \times h \times 16)$	$(m \times h \times 16)$	(1×3)	Relu	2
Corr layer	$(m \times h \times 16)$	$(m \times h \times 1)$	$([m+1] \times 1)$	Relu	-
Dilated conv	$(m \times h \times 17)$	$(m \times h \times 16)$	(1×3)	Relu	4
Dilated conv	$(m \times h \times 16)$	$(m \times h \times 16)$	(1×3)	Relu	4
Corr layer	$(m \times h \times 16)$	$(m \times h \times 1)$	$([m+1] \times 1)$	Relu	-
Causal conv	$(m \times h \times 17)$	$(m \times 1 \times 16)$	$(1 \times [h-28])$	Relu	-
1×1 conv	$(m \times 1 \times 17)$	$(m \times 1 \times 1)$	(1×1)	Softmax	-

$m \times h \times d$, the convolutions output tensors of dimension $m \times h \times d'$ where each slice of the output tensor, i.e. an $m \times 1 \times d$ matrix, contains the dependency information of each asset over time. By applying different dilation rates in each WaveCorr block, the model is able of extracting the dependency information for a longer time horizon. A dropout layer with a rate of 50% is considered to prevent over-fitting, whereas for the gradient explosion/vanishing prevention mechanism of residual connection we use a 1×1 convolution (presented on the top of Fig. 3), which inherently ensures that the summation operation is over tensors of the same shape. The *Corr* layer generates an output tensor of dimensions $m \times h \times 1$ from an $m \times h \times d$ input, where each slice of the output tensor, i.e. an $m \times 1 \times 1$ matrix, is meant to contain cross-asset dependency information. The concatenation operator combines the cross-asset dependency information obtained from the *Corr* layer with the cross-time dependency information obtained from the causal convolutions.

Before defining the *Corr* layer, we take a pause to introduce a property that will be used to further guide its design, namely the property of asset permutation invariance (API). This property is motivated by the idea that the set of possible investment policies that can be modeled by the portfolio policy network should not be affected by the way the assets are indexed in the problem. On a block per block level, we will therefore impose that, when the asset indexing of the input tensor is reordered, the set of possible mappings obtained should also only differ in its asset indexing. More specifically, we let $\sigma : \mathbb{R}^{m\times h\times d} \to \mathbb{R}^{m\times h\times d}$ denote a first coordinate permutation operator over a tensor $\mathcal{T}$ such that $\sigma(\mathcal{T})[i, :, :] = \mathcal{T}[\pi(i), :, :]$, where $\pi : \{1, ..., m\} \to \{1, ..., m\}$ is a bijective function. Furthermore, we consider $\sigma^{-1} : \mathbb{R}^{m\times h\times d'} \to \mathbb{R}^{m\times h\times d'}$ denote its "inverse" such that $\sigma^{-1}(\mathcal{O})[i, :, :] := \mathcal{O}[\pi^{-1}(i), :, :]$, with $\mathcal{O} \in \mathbb{R}^{m\times h\times d'}$.

Definition 3.1. (Permutation Invariance) A neural network architecture capturing a set of functions $\mathcal{B} \subseteq \{B : \mathbb{R}^{m\times h\times d} \to \mathbb{R}^{m\times h'\times d'}\}$ is **permutation invariant** in the first coordinate if given any permutation operator σ, we have that $\{\sigma^{-1} \circ B \circ \sigma : B \in \mathcal{B}\} = \mathcal{B}$, where $\circ$ stands for function composition. In the context of portfolio policy networks, we refer to the property of permutation invariance in the first coordinate as asset permutation invariance (API).

One can verify, for instances, that the architectures of all the neural network blocks described so far in WaveCorr are permutation invariant in the first coordinate, and furthermore that this property is preserved under composition (see Appendix A.2.2).

With this property in mind, we now detail the design of a permutation invariant *Corr* layer via Procedure 1, where we denote as $CC : \mathbb{R}^{(m+1)\times h\times d} \to \mathbb{R}^{1\times h\times 1}$ the operator that applies an $(m+1) \times 1$ convolution, and as $Concat_1$ the operator that concatenates two tensors along the first dimension. In Procedure 1,

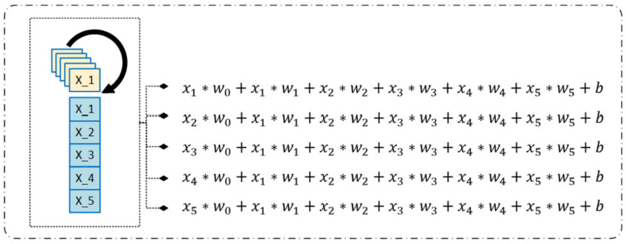
X_1
X_1
X_2
X_3
X_4
X_5
$x_1 * w_0 + x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + x_4 * w_4 + x_5 * w_5 + b$
$x_2 * w_0 + x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + x_4 * w_4 + x_5 * w_5 + b$
$x_3 * w_0 + x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + x_4 * w_4 + x_5 * w_5 + b$
$x_4 * w_0 + x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + x_4 * w_4 + x_5 * w_5 + b$
$x_5 * w_0 + x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + x_4 * w_4 + x_5 * w_5 + b$

Fig. 4. An example of the *Corr* layer over 5 assets.

the kernel is applied to a tensor in $\mathbb{R}^{(m+1)\times h\times d}$ constructed from adding the i-th row of the input tensor to the top of the tensor. Concatenating the output tensors from each run gives the final output tensor. Fig. 4 gives an example for the case with $m = 5$, and $h = d = 1$. Effectively, one can show that the *Corr* layer satisfies permutation invariance (see Appendix for a proof).

Procedure 1 *Corr* layer.

```
Result: Tensor that contains correlation information, $\mathcal{O}_{out}$ of dimension $m \times h \times 1$
Inputs: Tensor $\mathcal{O}_{in}$ of dimension $m \times h \times d$
Define an empty tensor $\mathcal{O}_{out}$ of dimension $0 \times h \times 1$
for $i = 1 : m$ do
    Set $O_{out} := Concat_1(\mathcal{O}_{out}, CC(Concat_1(\mathcal{O}_{in}[i, :, :], \mathcal{O}_{in})))$
end for
```

Proposition 3.1. *The Corr layer block satisfies permutation invariance in the first coordinate.*

Table 1 summarizes the details of each layer involved in the WaveCorr architecture: including kernel sizes, internal numbers of channels, dilation rates, and types of activation functions. Overall, the following proposition confirms that this WaveCorr portfolio policy network satisfies asset permutation invariance (API) (see Appendix for its proof).

Proposition 3.2. *The WaveCorr portfolio policy network architecture satisfies the API property.*

Finally, it is necessary to discuss some connections with the recent work of [16], where the authors propose an architecture that also takes both sequential and cross-asset dependency into consideration. Their proposed architecture, from a high level perspective, is more complex than ours in that theirs involves two sub-networks, one LSTM and one CNN, whereas ours is built solely on CNN. Our architecture is thus simpler to implement, less susceptible to overfitting, and allows for more efficient computation. The most noticeable difference between their design and ours is at the level of the *Corr* layer block, where they use a convolution with a $m \times 1$ kernel to extract dependency across assets and apply a standard padding trick to keep the output tensor invariant in size. Their approach suffers from two issues (see Appendix A.3 for details): first, the kernel employed in their design may only capture partial dependency information, and second, the more problematic issue, their design is not asset permutation invariant and thus the performance of their network can be highly sensitive to the ordering of assets. This second issue is further confirmed empirically in Section 4.3.

Remark 3.1. To the best of our knowledge, all the previous literature on permutation invariance (PI) (see for e.g. [4,8,15]) of neural networks have considered a definition that requires (see Definition 1 in [4]) that all functions $B \in \mathcal{B}$ are PI, i.e. that for any permutation operator σ, we have that $\sigma^{-1} \circ B \circ \sigma = B$. Definition 3.1 is more flexible as it focuses on the representation power of the NN architecture rather than the resulting NN, i.e. it does not impose the PI property on every functions of the set $\mathcal{B}$ but rather on the set as a whole, i.e. if $B \in \mathcal{B}$ then $\sigma^{-1} \circ B \circ \sigma$ is also in $\mathcal{B}$ for all σ. This distinction is crucial given that [4] observe that correlations cannot be modeled using their definition of PI networks "as they are agnostic to identities of entities". In fact, the architecture proposed in Fig. 3 is not permutation invariant according to Cai et al.'s PI definition and ends up not suffering from this deficiency.

4. Experimental results

In this section, we present the results of a series of experiments evaluating the empirical performance of our WaveCorr DRL approach. We start by presenting the experimental set-up. We follow with our main study that evaluates WaveCorr against a number of popular benchmarks. We finally shed light on the superior performance of WaveCorr with comparative studies that evaluate the sensitivity of its performance to permutation of the assets, number of assets, and size of transaction costs (see Appendix A.6). All codes are available at https://github.com/saeedmarzban/waveCorr.

4.1. Experimental set-up

Data sets We employ three data sets. **Can-data** includes the daily closing prices of 50 Canadian assets from 01/01/2003 until 01/11/2019 randomly chosen among the 70 companies that were continuously part of the Canadian S&P/TSX Composite Index during this period. **US-data** contains 50 randomly picked US assets among the 250 that were part of S&P500 index during the same period. Finally, **Covid-data** considered 50 randomly drawn assets from S&P/TSX Composite Index for period 01/11/2011-01/01/2021 and included open, highest, lowest, and closing daily prices. The Can-data and US-data sets were partitioned into training, validation, and test sets according to the periods 2003-2009/2010-2013/2014-2019 and 2003-2009/2010-2012/2013-2019 respectively, while the Covid-data was only divided in a training (2012-2018) and testing (2019-2020) periods given that hyper-parameters were reused from the previous two studies. We assume with all datasets a constant commission rate of $c_s = c_p = 0.05\%$ in the comparative study, while the sensitivity analysis considers no transaction costs unless specified otherwise.

Benchmarks In our main study, we compare the performance of WaveCorr to CS-PPN [16], EIIE [6], and the equal weighted portfolio (EW). Note that both CS-PPN and EIIE were adapted to optimize the Sharpe-ratio objective described in Section 2.2 that exactly accounts for transaction costs.

Hyper-parameter selection Based on a preliminary unreported investigation, where we explored the influence of different optimizers (namely ADAM, SGD, RMSProp, and SGD with momentum), we concluded that ADAM had the fastest convergence. We also narrowed down a list of reasonable values (see Table A.5) for the following common hyper-parameters: initial learning rate, decay rate, minimum rate, look-back window size h, planning horizon T. For each method, the final choice of hyper-parameter settings was done based on the average annual return achieved on both a 4-fold cross-validation study using Can-data and a 3-fold study with the US-data. The final selection (see Table A.5) favored, for each method, a candidate that appeared in the top 5 best performing settings of both data-sets in order to encourage generalization power among similarly performing candidates. Note that in order to decide on the number of epochs, an early stopping criterion was systematically employed.

Important implementation details Exploiting the fact that our SGD step involves exercising the portfolio policy network for T consecutive steps (see equation (2)), a clever implementation was able to reduce WavCorr's training time by a factor of 4 (see Appendix A.4 for details). We also implemented our stochastic gradient descent approach by updating, after each episode k, the initial state distribution F to reflect the latest policy μ_{θ_k}. This was done in order for the final policy to be better adapted to the conditions encountered when the portfolio policy network is applied on a longer horizon than T.

Metrics We evaluated all approaches using out-of-sample data ("test data"). "Annual return" denotes the annualized rate of return for the accumulated portfolio value. "Annual vol" denotes the prorated standard deviation of daily returns. Trajectory-wise Sharpe ratio (SR) of the log returns, Maximum drawdown (MDD), i.e. biggest loss from a peak, and average Turnover, i.e. average of the trading volume, are also reported (see [16] for formal definitions). Finally, we report on the average "daily hit rate" which captures the proportion of days during which the log returns outperformed EW.

4.2. Comparative evaluation of WaveCorr

In this set of experiments the performances of WaveCorr, CS-PPN, EIIE, and EW are compared for a set of 10 experiments (with random reinitialization of NN parameters) on the three datasets. The average and standard deviations of each performance metric are presented in Table 2 while Fig. A.8 (in the Appendix) presents the average out-of-sample portfolio value trajectories. The main takeaway from the table is that WaveCorr significantly outperforms the three benchmarks on all data sets, achieving an absolute improvement in average yearly returns of 3% to 25% compared to the best alternative. It also dominates CS-PPN and EIIE in terms of Sharpe ratio, maximum drawdown, daily hit rate, and turnover. EW does appear to be causing less volatility in the US-data, which leads to a slightly improved SR. Another important observation consists in the variance of these metrics over the 10 experiments. Once again WaveCorr comes out as being generally more reliable than the two other DRL benchmarks in the Can-data, while EIIE appears to be more reliable in the US-data sacrificing average performance. Overall, the impressive performance of WaveCorr seems to support our claim that our new architecture allows for a better identification of the cross-asset dependencies. In conditions of market crisis (i.e. the Covid-data), we finally observe that WaveCorr exposes the investors to much lower short term losses, with an MDD of only 31% compared to more than twice as much for CS-PPN and EIIE, which reflects of a more effective hedging strategy.

Table 2
The average (and standard deviation) performances using three data sets.

Method	Annual return	Annual vol	SR	MDD	Daily hit rate	Turnover
Can-data						
WaveCorr	27% (3%)	16% (1%)	1.73 (0.25)	16% (2%)	52% (1%)	0.32 (0.01)
CS-PPN	21% (4%)	19% (2%)	1.14 (0.34)	17% (4%)	51% (1%)	0.38 (0.05)
EIIE	-1% (8%)	29% (4%)	-0.01 (0.28)	55% (9%)	47% (1%)	0.64 (0.08)
EW	4% (0%)	14% (0%)	0.31 (0.00)	36% (0%)	-	0.00 (0.00)
US-data						
WaveCorr	19% (2%)	16% (2%)	1.17 (0.20)	20% (4%)	50% (1%)	0.11 (0.02)
CS-PPN	14% (2%)	15% (2%)	0.94 (0.17)	22% (6%)	49% (1%)	0.15 (0.08)
EIIE	16% (1%)	15% (0%)	1.09 (0.06)	20% (1%)	50% (0%)	0.17 (0.02)
EW	15% (0%)	13% (0%)	1.18 (0.00)	18% (0%)	-	0.00 (0.00)
Covid-data						
WaveCorr	56% (13%)	26% (5%)	2.16 (0.50)	31% (9%)	51% (2%)	0.19 (0.05)
CS-PPN	31% (27%)	51% (6%)	0.60 (0.48)	67% (7%)	50% (2%)	0.3 (0.09)
EIIE	11% (30%)	76% (17%)	0.20 (0.43)	77% (13%)	46% (2%)	0.76 (0.27)
EW	27% (0%)	29% (0%)	0.93(0.00)	47% (0%)	-	0.01 (0.00)

Table 3
The average (and standard dev.) performances over random asset permutation in Can-data.

	Annual return	Annual vol	SR	MDD	Daily hit rate	Turnover
WaveCorr	48% (1%)	15% (1%)	3.15 (0.19)	14% (3%)	56% (0%)	0.48 (0.01)
CS-PPN	35% (5%)	18% (1%)	2.00 (0.37)	22% (4%)	54% (1%)	0.54 (0.03)

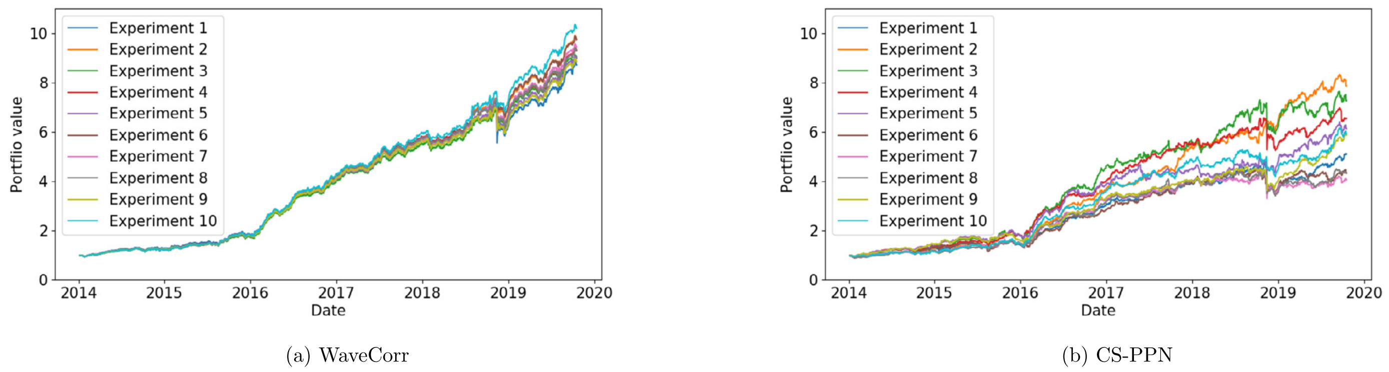

Fig. 5. Comparison of the wealth accumulated by WaveCorr and CS-PPN under random initial permutation of assets on Can-data's test set. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

4.3. Sensitivity analysis

Sensitivity to permutation of the assets In this set of experiment, we are interested in measuring the effect of asset permutation on the performance of WaveCorr and CS-PPN. Specifically, each experiment now consists in resampling a permutation of the 50 stocks instead of the initial parameters of the neural networks. The results are summarized in Table 3 and illustrated in Fig. 5. We observe that the learning curves and performance of CS-PPN are significantly affected by asset permutation compared to WaveCorr. In particular, one sees that the standard deviation of annual return is reduced by more than a factor of about 5 with WaveCorr. We believe this is entirely attributable to the new PI structure of the *Corr* layer.

Sensitivity to number of assets In this set of experiments, we measure the effect of varying the number of assets on the performance of WaveCorr and CS-PPN. We therefore run 10 experiments with growing subsets of 30, 40, and 50 assets from Can-data. Results are summarized in Table 4 and illustrated in Fig. A.9 (in the Appendix). While having access to more assets should in theory be beneficial for the portfolio performance, we observe that it is not necessarily the case for CS-PPN. On the other hand, as the number of assets increases, a significant improvement, with respect to all metrics, is achieved by WaveCorr. This evidence points to a better use of the correlation information in the data by WaveCorr.

5. Conclusion

This paper presented a new architecture for portfolio management that is built upon WaveNet [13] and a new design of correlation block that can process and extract cross-asset information while preserving the property of asset permutation invariance, whereas a naive extension of CNN, such as in the recent works of [16], does not. The empirical evidence indicates that API leads to better and more stable performance. Finally, the results presented in Appendix A.2 lay important foundations for analyzing the API property of different architectures and applications of DRL: see, for e.g., Appendix A.7 which discusses the API property of the transformer architecture used in [14].

Data availability

All code and data is shared on a github repository, which reference is included in the main text of the paper.

Table 4
The average (and std. dev.) performances as a function of the number of assets in Can-data.

# of stocks	Annual return	Annual vol	SR	MDD	Daily hit rate	Turnover
WaveCorr						
30	37.7% (4%)	19% (1%)	2.02 (0.27)	22% (3%)	55% (1%)	0.39 (0.02)
40	38.5% (4%)	21% (1%)	1.81 (0.17)	23% (2%)	55% (1%)	0.44 (0.04)
50	43.0% (5%)	17% (2%)	2.57 (0.52)	20% (6%)	55% (1%)	0.43 (0.02)
CS-PPN						
30	30.3% (3%)	17% (1%)	1.80 (0.20)	21% (4%)	53% (1%)	0.42 (0.04)
40	29.8% (7%)	17% (2%)	1.70 (0.34)	22% (3%)	53% (1%)	0.41 (0.09)
50	32.2% (4%)	16% (1%)	2.07 (0.28)	18% (3%)	52% (1%)	0.43 (0.05)

Acknowledgement

The authors gratefully acknowledge the financial support from the Natural Sciences and Engineering Research Council of Canada [Grants RGPIN-2016-05208 and RGPIN-2014-05602], the Canada Research Chair Program [Grant CRC-2018-00105], Mitacs [Grant IT15577], the NSERC-CREATE Program on Machine Learning in Quantitative Finance and Business Analytics [Grant 512046], the Digital Research Alliance of Canada (alliancecan.ca), and Evovest (evovest.com).

Appendix. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.orl.2023.10.011.

References

[1] Saud Almahdi, Steve Y. Yang, An adaptive portfolio trading system: a risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown, Expert Syst. Appl. 87 (2017) 267–279.

[2] Shaojie Bai, J. Zico Kolter, Vladlen Koltun, An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, 2018.

[3] David H. Bailey, Marcos Lopez de Prado, The Sharpe ratio efficient frontier, J. Risk 15 (2) (2012) 13.

[4] Desmond Cai, Shiau Hong Lim, Laura Wynter, Efficient reinforcement learning in resource allocation problems through permutation invariant multi-task learning, in: 2021 60th IEEE Conference on Decision and Control (CDC), IEEE Press, 2021, pp. 2270–2275.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[6] Zhengyao Jiang, Dixing Xu, Jinjun Liang, A deep reinforcement learning framework for the financial portfolio management problem, 2017.

[7] Bin Li, Jialei Wang, Dingjiang Huang, Steven C.H. Hoi, Transaction cost optimization for online portfolio selection, Quant. Finance 18 (8) (2018) 1411–1424.

[8] Yan Li, Lingxiao Wang, Jiachen Yang, Ethan Wang, Zhaoran Wang, Tuo Zhao, Hongyuan Zha, Permutation invariant policy optimization for mean-field multi-agent reinforcement learning: a principled approach, CoRR, arXiv:2105.08268 [abs], 2021.

[9] Zhipeng Liang, Hao Chen, Junhao Zhu, Kangkang Jiang, Yanran Li, Adversarial deep reinforcement learning in portfolio management, 2018.

[10] Harry Markowitz, Portfolio selection, J. Finance 7 (1) (1952) 77–91.

[11] John Moody, Lizhong Wu, Yuansong Liao, Matthew Saffell, Performance functions and reinforcement learning for trading systems and portfolios, J. Forecast. 17 (5–6) (1998) 441–470.

[12] William F. Sharpe, Mutual fund performance, J. Bus. 39 (1) (1966) 119–138.

[13] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu, Wavenet: a generative model for raw audio, 2016.

[14] Ke Xu, Yifan Zhang, Deheng Ye, Peilin Zhao, Mingkui Tan, Relation-aware transformer for portfolio policy learning, in: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20), IJCAI Organization, 2020, pp. 4647–4653.

[15] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R. Salakhutdinov, Alexander J. Smola, Deep sets, in: Advances in Neural Information Processing Systems, vol. 30, Curran Associates Inc., 2017.

[16] Yifan Zhang, Peilin Zhao, Bin Li, Qingyao Wu, Junzhou Huang, Mingkui Tan, Cost-sensitive portfolio selection via deep reinforcement learning, IEEE Trans. Knowl. Data Eng. (2020).