Forecasting Asset Dependencies to Reduce Portfolio Risk

Haoren Zhu,*[1] Shih-Yang Liu,*[1] Pengfei Zhao,†[2] Yingying Chen,[3] Dik Lun Lee [1]

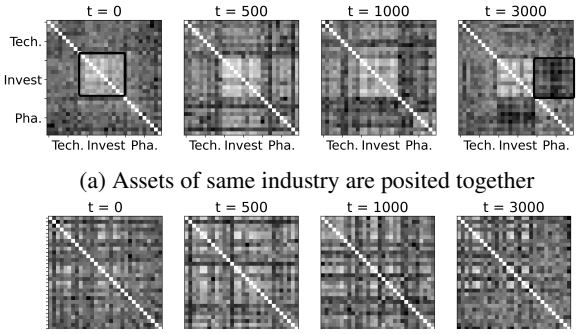
[1]Hong Kong University of Science and Technology,
[2]Beijing Normal University-Hong Kong Baptist University United International College,
[3]London School of Economics and Political Science,
{hzhual, sliuau, dlee}@cse.ust.hk, ericpfzhao@uic.edu.cn, y.chen233@lse.ac.uk

Abstract

Financial assets exhibit dependence structures, i.e., movements of their prices or returns show various correlations. Knowledge of assets' price dependencies can help investors create a diversified portfolio, which reduces portfolio risk due to the high volatility of the financial market. Since asset dependency changes with time in complex patterns, asset dependency forecast is an essential problem in finance. In this paper, we organize pairwise assets dependencies in an *Asset Dependency Matrix* (*ADM*) and formulate the problem of assets dependencies forecast to predict the future *ADM* given a sequence of past *ADM*s. We propose a novel idea viewing a sequence of *ADM*s as a sequence of images to capture the spatial and temporal dependencies among the assets. Inspired by video prediction tasks, we develop a novel *Asset Dependency Neural Network* (*ADNN*) to tackle the *ADM* prediction problem. Experiments show that our proposed framework consistently outperforms the baselines on both future *ADM* prediction and portfolio risk reduction tasks.

Introduction

In financial market, it is well known that asset returns are dependent on each other [1] , forming a complex dependence structure among the assets (Elton and Gruber 1973; Ane and Kharoubi 2003). For instance, stocks of the same industry may move up or down together in response to market news, and the price fluctuation in one industry at a particular time may result in price fluctuation of its downstream industries later. The dependency structure of a financial market is very complex, and the ability to predict it will result in significant financial advantages. For example, we can reduce portfolio risk by investing in different classes of assets that are independent or negatively correlated to each other. Asset dependency can be measured in many ways, including the correlation and covariance coefficients between two assets. Given a group of assets, the pairwise coefficients between the assets can be represented with a matrix, termed *Asset Dependency Matrix* (*ADM*). For example, if the correlation coefficient is used to measure the dependency of two assets, *ADM* is equivalent to the well-known correlation matrix.



(a) Assets of same industry are posited together

(b) Assets are randomly posited

Figure 1: Correlation matrix of a number of assets.

Various statistical methods have been proposed for predicting future *ADM*s. A simple prediction model uses the correlation matrix computed from the time window ending at t as the predicted *ADM* for $t' > t$. Later, improvements that consider asset groups, financial indexes, and beta estimations have been proposed (Elton, Gruber, and Padberg 1977; Baesel 1974; Blume 1975; Vasicek 1973). Unfortunately, existing financial solutions suffer from many restrictive statistical assumptions and limitations on modeling time-varying dependence structures.

Asset dependencies exhibit both spatial and temporal properties. *Spatial dependency* refers to the dependence between two assets at a particular time. It reflects the complex relationships between a group of assets. For example, high-tech stocks are inherently correlated and may respond to an event in the same way, e.g., increase of interest rates. *Temporal Dependency* refers to the dependency between two assets along the timeline. That is, a fluctuation of an asset's return due to some events will impact the future price movements of itself and other assets. A typical financial phenomenon known as "sector rotation" (Conover et al. 2008) is caused by the intertwined influences of both types of dependencies.

Figure 1(a) shows an example of spatiotemporal patterns in *ADM*s. Ten assets are selected from each of the technology, investment, pharmaceuticals sectors. Assets from

*These authors contributed equally.
†Corresponding author.

[1]Financial assets may include stocks, securities, and derivatives, etc. Further, for brevity, we use "asset dependence" to refer to the dependence between the returns of two assets.

the same industry are placed next to each other in the matrix (i.e., having sequential indexes). Each entry in the matrix represents the correlation coefficient between two assets, where white indicates a coefficient of 1 and black a coefficient of -1. At $t = 0$, we can observe the central region (enclosed by the black box), representing dependencies between assets in the "invest" sector, shows positive correlations. On the other hand, the region to the right of the center, representing dependencies between assets in the "invest" sector and assets in the "pharmaceuticals" sector, shows weakly negative correlations. Both examples illustrate various strengths of spatial dependencies among assets. As time elapses, we can see that groups of assets show different correlations. At $t = 3000$, we can see that assets in "invest" remain generally positively correlated, while assets in "invest" and "pharmaceuticals" (the central right region enclosed by the black box) become more negative.

We formulate the problem of sequential *ADM* prediction as a future *ADM* prediction problem given a sequence of past *ADM*s. We propose solutions inspired by the video prediction problem (Oprea et al. 2020), which aims at predicting future image frames based on a sequence of past frames. Each frame corresponds to an *ADM* matrix, with each frame pixel representing the correlation between two assets. Video data are characterized with spatiotemporal properties since the value of a pixel is related to its neighbor pixels as well as pixels in previous frames. Spatiotemporal models with deep learning framework, e.g., "*Convolutional Recurrent Neural Networks*" *CRNN*, are successful (Shi et al. 2015; Mathieu, Couprie, and LeCun 2015; Hsieh et al. 2018; Babaeizadeh et al. 2017; Wang et al. 2018) due to their ability to extract features in both spatial and temporal dimensions.

A major challenge of directly applying *CRNN* to *ADM* prediction is that pixels of the same object are naturally grouped together both spatially and temporally in the images, while asset dependency values have no predefined placement in the *ADM*s. Two related values can be placed in the matrix far away from each other; likewise, two unrelated values can be placed close in the matrix. In Figure 1(b), assets are placed randomly, and we cannot see any clear spatial or temporal clustering of assets in the matrices. Thus, *CRNN* will not be effective if it directly takes *ADM*s with randomly placed assets as input. Comparing Figures 1(a) and 1(b), we can see that spatial and temporal patterns in *ADM*s can be better revealed with proper asset positioning. We refer to the determination of asset placements in *ADM*s for enhancing future *ADM*s prediction as the *ADM representation problem*.

A poorly placed *ADM* hampers the effectiveness of *CRNN*. To tackle this problem, a transformation function that is capable of obtaining better representations of the original *ADM*s is required. Motivated by representation learning (Bengio, Courville, and Vincent 2014), which automatically discovers optimal representations from the raw data, and *Mixture of Experts* (Jacobs et al. 1991) (*MoE*), which responds to different circumstances with greater specialization, we design *Asset Dependency Neural Network* (*ADNN*). The contributions are listed as follows:

- We propose the novel idea of viewing assets dependencies as a sequence of images and apply video prediction techniques to capture the spatiotemporal dependencies for forecasting assets dependencies.
- We observe that assets in *ADM* do not have a priori order like pixels in images, and we propose methods to transform *ADM*s so that spatial dependencies can be effectively discovered and exploited to enhance *ADM* prediction accuracy.
- We propose a novel model, *ADNN*, to learn an optimal transformation function that maps an *ADM* into an optimal representation before feeding it to *CRNN*. *ADNN* employs Mixture of Experts (*MoE*) to learn from the large variety of factors affecting asset returns. We achieve an integrated, end-to-end framework that can effectively improve the prediction accuracy.
- We evaluate the performance using not only *ADM* prediction accuracy but also the risks of portfolios constructed from the predicted *ADM*s. Experiment results show that our proposed framework significantly outperforms baselines in both aspects.

Related Work

Financial Applications of Dependencies

Dependency of assets is a crucial input to multiple financial applications. In portfolio diversification, an optimal portfolio invests in an optimal combination of assets (Markowitz 1952). Asset dependencies influence the diversification level of a portfolio, where higher dependency of assets incurs lower diversification of a portfolio. The dynamic feature of asset dependencies is also important to consider when constructing a portfolio. For example, when market conditions are deteriorating, correlations of assets increase and may even approach one. During the evolvement of asset correlations, a diversified portfolio at t may be no longer diversified at t' and produce unexpected losses (van Binsbergen 2011). Therefore, we need to model the dynamics of the *ADM* to achieve a consistent diversification level of a portfolio.

Conventional *ADM*s Prediction Methods

Multivariate GARCH (*MGARCH*) models are extensions of univariate volatility models to estimate time-varying asset dependencies. Modification of *MGARCH* models includes the Dynamic Conditional Correlation (*DCC*) model (Engle 2002) and its variants (Rangel and Engle 2012; Engle, Ledoit, and Wolf 2019), which address the dynamic correlation structure of *ADM*s by parameterizing the conditional correlations. However, these statistics-oriented approaches (1) hold unrealistic statistical assumptions, which make them inapplicable to real data(Caporin and McAleer 2013) and (2) utilize linear regression in parameter estimation, which is incapable of modeling the high-dimensional dependencies embedded in *ADM*s. Compared with existing works, our framework is more flexible with no imposing structures required on *ADM*s, making it capable of capturing different spatiotemporal patterns.

Convolutional Recurrent Neural Network

CRNN represents a series of spatiotemporal models combining convolutional neural networks (*CNN*s) for extracting

spatial features and recurrent neural networks (*RNN*s) for temporal features. *ConvLSTM* (Shi et al. 2015) is one of the pioneer works with convolution structure in both the input-to-state and state-to-state transitions. After that, other advanced spatiotemporal modeling techniques have been proposed, e.g., PredRNN (Wang et al. 2017), and Eidetic 3D LSTM (E3D-LSTM) (Wang et al. 2018). Due to the *ADM* representation problem introduced before, the transformation of *ADM* needs to be done appropriately. According to the authors' best knowledge, our work is the first to use the spatiotemporal model to solve sequential *ADM* prediction considering the *ADM* representation problem.

Problem Definition

ADM Definition

In financial convention, given the historical daily price of an asset $\{p_0, p_1, ..., p_T\}$, where p_t denotes the price at time t, the returns of the asset at time t is calculated by:

$$r_t = log(p_t) - log(p_{t-1}) \tag{1}$$

where logarithm normalization is applied to the price data so that the derived returns are weak stationary. Denote the returns of asset i at time t as $r_t^{a_i}$ and let $f(a_i, a_j)_t$ denote the dependency measurement between asset i and asset j at time t. For correlation function f of two assets a_i and a_j in period $[t - n_{lag}, t]$, $f(a_i, a_j)_t$, is calculated as follows:

$$f(a_i, a_j)_t = \frac{\Sigma_{p=t-n_{lag}}^{t}(r_p^{a_i} - \overline{r^{a_i}})(r_p^{a_j} - \overline{r^{a_j}})}{\sqrt{\Sigma_{p=t-n_{lag}}^{t}(r_p^{a_i} - \overline{r^{a_i}})^2}\sqrt{\Sigma_{p=t-n_{lag}}^{t}(r_p^{a_j} - \overline{r^{a_j}})^2}} \tag{2}$$

It is natural to use matrix to group all pairwise dependencies. Then, *ADM* at time t can be formulated as:

$$\mathcal{M}_t = \begin{bmatrix} f(a_i,a_i)_t & f(a_i,a_j)_t & \cdots & f(a_i,a_n)_t \\ f(a_j,a_i)_t & f(a_j,a_j)_t & \cdots & f(a_j,a_n)_t \\ \vdots & \vdots & \ddots & \vdots \\ f(a_n,a_i)_t & f(a_n,a_j)_t & \cdots & f(a_n,a_n)_t \end{bmatrix} \tag{3}$$

ADM Prediction Model

Given a historical *ADM* sequence denoted as:

$$\mathcal{M}_t^{k,u} = \{\mathcal{M}_{t-(k-d)\cdot u} | 1 \le d \le k, t \ge (k-d)\cdot u, \text{ with } t, k, u, d \in Z^+\} \tag{4}$$

where k denotes the length of the input sequence, t denotes the current time and u denotes the rolling distance between each input matrix in the day unit. Figure 2 illustrates the *ADM* sequence construction process. $\mathcal{M}_t^{k,u} \in R^{k\times c\times w\times h}$ where c denotes the number of channels, w and h denotes the width and height of *ADM*. Our target is to predict $\mathcal{M}_{t+h} \in R^{1\times c\times w\times h}$, where h denotes the horizon, which is the duration that an investor expects to hold a portfolio. For example, if the portfolio is rebalanced on a monthly basis, the investors are interested in the *ADM* one month later, and h should be set to 21 (average monthly trading days). We formulate *ADM* prediction model as:

$$\hat{\mathcal{M}}_{t+h} = \mathcal{F}(\mathcal{M}_t^{k,u} | \Theta_{\mathcal{F}}) \tag{5}$$

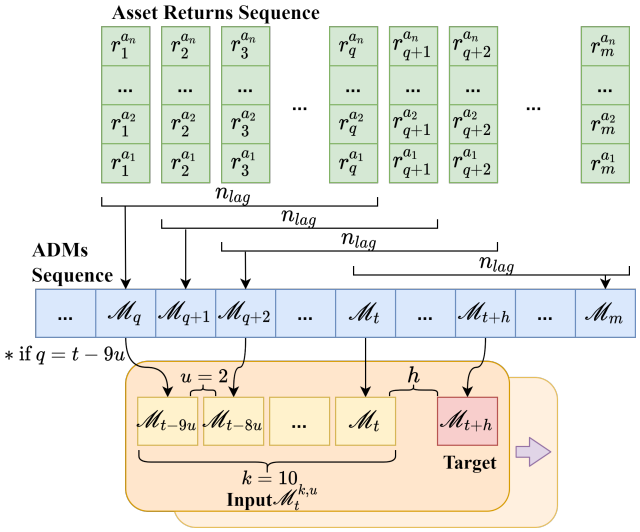


Figure 2: *ADM*s Construction. $r_q^{a_j}$ denotes the return of asset j at time q, and $\mathcal{M}_t$ denotes the *ADM* at time t. We first generate *ADM* sequences from the asset returns sequence. In this figure, $k = 10, u = 2$ and $q = t - (k-1)u = t - 18$. With fixed u, k, h, and shift of current time t, we can obtain multiple input sequences and their corresponding targets.

where $\Theta_{\mathcal{F}}$ denotes the parameters of model $\mathcal{F}$ and $\hat{\mathcal{M}}_{t+h}$ denotes the predicted *ADM*. We will discuss $\mathcal{F}$ in detail in the methodology. To learn the model parameters, the loss is:

$$\begin{aligned} L_{sqr} &= (\mathcal{M}_{t+h} - \hat{\mathcal{M}}_{t+h})^2 \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}(f(a_i,a_j)_{t+h} - \hat{f}(a_i,a_j)_{t+h})^2 \end{aligned} \tag{6}$$

Methodology

ConvLSTM

As introduced above, to forecast future *ADM*, temporal and spatial signals hidden in historical *ADM* need to be carefully modeled. Considering the pioneer contribution and conciseness of the *ConvLSTM* (Shi et al. 2015) in handling spatial and temporal patterns, we choose *ConvLSTM* as the basic building block of our framework. The core unit of the *ConvLSTM* network is *ConvLSTM* cell. All of the input states and intermediate states representations fed to the cell are tensors with the shape $R^{c\times w\times h}$. Denote the input state as X_t, memory state as C_t, hidden state as H_t, and gated signals as i_t, f_t, g_t, o_t at each timestamp for one cell. The future state of a certain pixel in the matrix is determined by the inputs and past states of its local neighbors. This can easily be achieved by using a convolution operator in the state-to-state and input-to-state transitions:

$$\begin{aligned} i_t &= \sigma(W_{xi} * X_t + W_{hi} * \mathcal{H}_{t-1} + W_{ci} \circ C_{t-1} + b_i) \\ f_t &= \sigma(W_{xf} * X_t + W_{hf} * \mathcal{H}_{t-1} + W_{cf} \circ C_{t-1} + b_f) \\ C_t &= f_t \circ C_{t-1} + i_t \circ tanh(W_{xc} * X_t + W_{hc} * \mathcal{H}_{t-1} + b_c) \\ o_t &= \sigma(W_{xo} * X_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \circ C_t + b_o) \\ \mathcal{H}_t &= o_t \circ tanh(C_t) \end{aligned} \tag{7}$$

where '$*$' denotes the convolution operator and '$\circ$' denotes the Hadamard product.

Asset Dependency Neural Network

Due to the *ADM* representation problem discussed in the introduction, we need to transform *ADM* to better represent the spatial dependency. Denote the height and width of *ADM* as h and w; the optimal transformation function is defined as $\mathbb{T} : \mathbb{R}^{h \times w} \rightarrow \mathbb{R}^{h \times w}$, and we have:

$$\tilde{\mathcal{M}} = \mathbb{T}(\mathcal{M}) \tag{8}$$

where $\mathcal{M}$ refers to original *ADM* and $\tilde{\mathcal{M}}$ refers to transformed matrix characterized with better spatial property.

One natural way of constructing the transformation $\mathbb{T}$ is to place assets with high dependency values closely in *ADM*, e.g., by various clustering techniques, as a preprocessing step before feeding to spatiotemporal model. However, this two-steps approach has multiple drawbacks: (1) the suitable number of clusters changes with time, and the optimal frequency of clustering is unknown (2) the optimal transformation function $\mathbb{T}$ may involve more complex conversion, which is unable to be captured by only rearranging the position (3) the position transformation generated in preprocessing cannot guarantee that the derived matrices are the optimal data representations for model training. To tackle these problems, we propose an end-to-end framework that composite the transformation function solving the *ADM* representation problem with the spatiotemporal blocks capturing spatial and temporal signals in *ADM*, named as *Asset Dependency Neural Network* (*ADNN*). With this design, the transformation function $\mathbb{T}$ can be trained and updated together with the spatiotemporal block. In this paper, we choose *ConvLSTM* as the spatiotemporal block even though $\mathbb{T}$ can also be composited with other spatiotemporal models.

To approximate the optimal $\mathbb{T}$, a polynomial approximation function is utilized. Since linear functions are not complex enough to model the complexity of financial markets while higher dimensional functions can result in overfitting, we select quadratic function, which is a common practice in function approximation problems (Torokhti and Howlett 2001; Boubekeur and Schlick 2007). The quadratic function conducts the transformation by assigning a new value to each entry based on the whole matrix. It includes the ability to approximate position rearrangement and data scaling if such modifications are optimal from the perspective of representation learning. The quadratic function is combined with the original prediction model and can be written as:

$$\begin{aligned} \mathbb{T}(\mathcal{M}_t^{k,u}) &= a_t \circ \mathcal{M}_t^{k,u} \circ \mathcal{M}_t^{k,u} + b_t \circ \mathcal{M}_t^{k,u} + c_t \\ \hat{\mathcal{M}}_{t+h} &= \mathcal{F}(\mathcal{M}_t^{k,u} | \Theta_{\mathcal{F}}) = \mathcal{C} \odot \mathbb{T}(\mathcal{M}_t^{k,u} | \Theta_{\mathcal{C} \circ \mathbb{T}}) \end{aligned} \tag{9}$$

where a_t, b_t, c_t denotes the coefficient tensor of the quadratic term, the linear term, and the constant term respectively, all of the coefficients are in the shape of $R^{1 \times c \times w \times h}$. $\odot$ denotes the composition of two functions. Note that $\mathbb{T}$ is merged with $\mathcal{C}$ (*ConvLSTM*) and trained together in an end-to-end manner. $a_t = f_a(\mathcal{M}_t^{k,u} | \Theta_a)$, f_a denotes the coefficient function outputting a_t given parameter set Θ_a and input $\mathcal{M}_t^{k,u}$. Coefficients b_t, c_t are output by functions f_b and f_c similarly. This dynamic design can produce different results adapting to different inputs and hence remedying the limitation of the static transformation method. A natural idea is to model coefficient function f by Multilayer Perceptron (*MLP*) (Goodfellow, Bengio, and Courville 2016); however due to the complexity of financial market, *ADM* could be influenced by various market factors like macro-economy, interest rates, industry rotation, shocking events, etc. *MLP* has its limitation in encoding such complex market behavior signals, and as a result can only learn an over-generalized rule for all the market situations. For example, assuming there are k different market states caused by different financial events, and the dependency between two stocks may vary with respect to different states. The utilization of *MLP* on approximating transformation function only allows the model to learn a general rule that applies to all k market situations, which often results in over-generalization of the model and yields unsatisfactory performance.

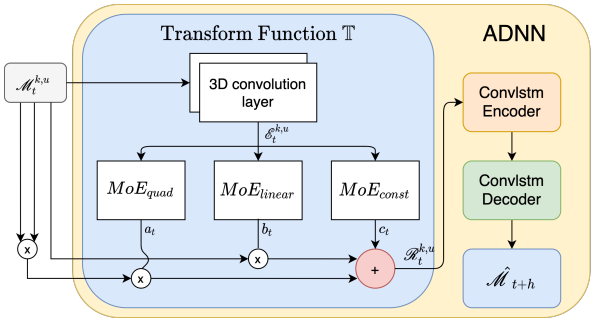


Figure 3: An overview of the *ADNN* architecture.

Inspired by the effectiveness of Mixture-of-Experts (*MoE*) (Jacobs et al. 1991) and recent *MoE* applications (Shazeer et al. 2017; Ma et al. 2018) on solving complex data problem where a single dataset consists of many different data regimes, we expect $\mathbb{T}$ to be capable of transforming input *ADM* into a proper representation with respect to different market states. *MoE* comprises a gating network and multiple subnets, where each subnet is a *MLP* (expert). It achieves improvement especially on the complex dataset as it utilized its gating network to select a sparse combination of experts to process input from different data patterns, and in our setting, *ADM* from different market states. By replacing *MLP* with the *MoE* block in our transformation function, $\mathbb{T}$ can properly transform the input *ADM* into better representation, as different market rules have been learned by different experts, and the gating network can now act as a knowledgeable manager who decides to assign task $\mathcal{M}_t^{k,u}$ to specialized experts and combines their analyses afterward. The critical difference between *MLP* and *MoE* on constructing $\mathbb{T}$ is that *MLP* learns a single rule that generalized all the complex market scenarios while MoE introduces different subnets to specialize in different market scenarios and combined their output to generate better *ADM* representation.

Figure 3 illustrates the whole *ADNN* architecture. The input *ADM* sequence $\mathcal{M}_t^{k,u}$ is fed to $\mathbb{T}$, the transformed result $\mathcal{R}_t^{k,u}$ is sent to the *ConvLSTM* encoder followed by *ConvLSTM* decoder, which then outputs the predicted *ADM*

$\hat{\mathcal{M}}_{t+h} \in R^{1\times c\times h\times w}$. Inside $\mathbb{T}$, in order to better extract the features of the input *ADM* sequence, we use two consecutive 3d convolutional layers (Goodfellow, Bengio, and Courville 2016), and the output $\mathcal{E}_t^{k,u}$ (Equation 10) is fed to the *MoE* layers, where coefficients of the quadratic transformation function are learned by each separate *MoE* block. Transformation function $\mathbb{T}$ can be formulated as:

$$\begin{aligned}
\mathcal{E}_t^{k,u} &= W_{conv2} * (W_{conv1} * \mathcal{M}_t^{k,u}) \\
a_t &= MoE_{quad}(\mathcal{E}_t^{k,u}|\Theta_a) \\
b_t &= MoE_{linear}(\mathcal{E}_t^{k,u}|\Theta_b) \\
c_t &= MoE_{const}(\mathcal{E}_t^{k,u}|\Theta_c) \\
\mathcal{R}_t^{k,u} &= \mathbb{T}(\mathcal{M}_t^{k,u}) = a_t \circ \mathcal{M}_t^{k,u} \circ \mathcal{M}_t^{k,u} + b_t \circ \mathcal{M}_t^{k,u} + c_t
\end{aligned} \tag{10}$$

where [W_{conv1}, W_{conv2}] denote the 3D convolution layers, [MoE_{quad}, MoE_{linear},MoE_{linear}] denote the Mixture of Experts layers used to produce the coefficients a_t, b_t, c_t, $\mathcal{R}_t^{k,u}$ denotes the converted result which will be input for the *ConvLSTM* block for spatial temporal signal extraction.

Experiment

We evaluate *ADNN* from two aspects: *ADM* prediction accuracy and portfolio risk reduction.

Experiment Setting

We construct a pool of real stock prices by combining the daily price data of stocks from S&P-100, NASDAQ-100, and DJI-30, including most influential companies in recent 15 years (from 2005/09/27 to 2020/08/05). We retain stocks with complete closing price data throughout the whole period, resulting in 133 stocks in the pool, each of which having a time series of 3740 price data points.

The *ADM* sequences can be generated from the stock price data as depicted in Figure 2. A general guideline from finance is to choose n_{lag} at least as large as the number of assets n to avoid the out-of-sample problem (Tashman 2000). However, if n_{lag} is too large, the model may be incapable of capturing short-term (e.g. monthly) *ADM* dynamics. To strike a balance, we set $n_{lag} = 42$ and $n = 32$. Since we have a complete pool of 133 stocks, we randomly select stocks from the pool to generate 10 stock datasets, each of which is composed of 32 stocks. We evaluate the performance of the methods on each stock dataset independently to guarantee the generality of the experimental outcome. The full list of stocks in each stock dataset is attached in the technical appendix. In this paper, we study *ADM*s' monthly evolution so $h = u = 21$ (the number of trading days in a month). We set $k = 10$. There are 3487 *ADM* sequences generated for each stock dataset. We select the first 90% of *ADM* sequences as training samples (including validation) and the remaining 10% as testing samples.

We use Adam optimizer and the experiments are run on a server with 4 NVIDIA GeForce RTX 2080-ti Graphic Cards. We utilize the gradual warm-up learning rate scheduler (Goyal et al. 2017) and select cosine annealing as the next scheduler to dynamically set the learning rate.

Baselines

Since we focus on the use of correlation coefficients in *ADM* prediction, we introduce the following baselines used in financial industry. For fairness, methods that utilize external information (e.g., economic indicators) are excluded from the comparison.

- **Constant Correlation Model (CCM)** considers the correlation of returns between any pair of securities to be the same for a given group of securities (Elton and Gruber 1973). It takes the historical means of all the pairwise correlations as the prediction for the next period.
- **DCC-Garch** (Engle 2002) considers the dynamic property of correlation in the model by parameterizing the conditional correlations to forecast the future correlation matrix and utilizes the predicted correlation matrix to generate future covariance matrices.

We also adapt classical spatiotemporal prediction approaches to the sequential *ADM* prediction task:

- **Convolution-3D (Conv3D)**(Gebert et al. 2019) has an encoder-decoder structure with a sequence of 3d convolution layers as encoder and another sequence of 3d convolution layers as decoder.
- **Long-Short-Term-Memory network (LSTM)** (Hochreiter and Schmidhuber 1997) takes a sequence of vectors as input and predicts the target vector. The 2D *ADM*s are first reshaped to 1D vectors for the task.

In addition, we evaluate other transformation architectures that incorporate *ConvLSTM* as the building block.

- **Raw-ConvLSTM** feeds the raw *ADM* sequences to the *ConvLSTM* without any transformation $\mathbb{T}$.
- **P-ConvLSTM** involves static reposition of entries in *ADM*s in the preprocessing stage. Entries of *ADM*s will be clustered using k-means based on the training set, for which similar entries are placed closely together. The placement of each entry is the same for the testing set, and the number of clusters is tuned to 12.
- **T-ConvLSTM** is an end-to-end framework that first transforms *ADM*s through a fully-connected network $\mathbb{T}$ instead of learning a quadratic transformation, then the transformed result is fed to *ConvLSTM*.
- **MLP-ConvLSTM** has a similar architecture to *ADNN* except that it utilizes multi-layer perceptron (*MLP*) to learn the (coefficients of) quadratic transformation function $\mathbb{T}$ instead of *MoE*.

Evaluation of ADM Prediction Accuracy

Evaluation Metrics *ADM* prediction accuracy is measured by the following two metrics:

- **Mean square error (MSE)** is calculated by averaging the squared difference of each entry between the ground-truth matrix and the predicted matrix.
- **Gain** to the previous *ADM*. It is a common practice in finance that takes the *ADM* of the latest period as the future *ADM* (Elton, Gruber, and Urich 1978), where

$\mathbf{n_{exp}}$	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7	Set 8	Set 9	Set 10	Avg.
2 ($\text{top}_k = 1$)	0.0135	0.0133	0.0136	0.0085	0.0144	0.0084	0.0125	0.0130	0.0130	0.0130	0.0123
4 ($\text{top}_k = 2$)	0.0082	0.0087	0.0088	0.0085	0.0091	0.0117	0.0075	0.0078	0.0080	0.0083	0.0086
8 ($\text{top}_k = 4$)	0.0080	0.0085	0.0086	0.0085	0.0090	0.0087	0.0075	0.0075	0.0079	0.0076	**0.0082**
16 ($\text{top}_k = 4$)	0.0081	0.0090	0.0115	0.0087	0.0089	0.0086	0.0076	0.0079	0.0080	0.0074	0.0086

Table 1: *MSE* versus n_{exp}.

Method		Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7	Set 8	Set 9	Set 10	Mean(Std)
Raw-Conv	MSE	0.00846	**0.00842**	0.00887	0.00855	0.00946	0.00883	0.0152	0.00784	0.00789	0.00772	0.0091(0.0021)
LSTM	Gain	0.210	**0.225**	0.199	0.188	0.179	0.202	-0.487	0.236	0.211	0.234	0.140(0.2096)
P-Conv	MSE	0.00863	0.0112	**0.00851**	0.00908	0.00966	0.00877	0.00780	0.00843	0.00792	0.00829	0.0089(0.0009)
LSTM	Gain	0.193	-0.099	**0.231**	0.138	0.161	0.207	0.237	0.222	0.155	0.176	0.162(0.0927)
T-Conv	MSE	0.01365	0.00965	0.0140	0.0134	0.116	0.0147	0.00943	0.0136	0.0139	0.0132	0.0127(0.0018)
LSTM	Gain	-0.275	0.111	-0.266	-0.270	-0.009	-0.329	0.077	-0.325	-0.398	-0.307	-0.199(0.1754)
MLP-Conv	MSE	0.00862	0.00908	0.0109	0.00866	0.00955	0.00914	0.0104	0.00820	0.00823	0.00918	0.0096(0.0009)
LSTM	Gain	0.188	0.146	-0.003	0.168	0.132	0.160	-0.019	0.192	0.155	0.079	0.095(0.0720)
ADNN	MSE	**0.00802**	0.00852	0.00858	**0.00845**	**0.00897**	**0.00871**	**0.00754**	**0.00752**	**0.00789**	**0.00757**	**0.0082(0.0005)**
	Gain	**0.252**	0.215	0.226	**0.198**	**0.221**	**0.213**	**0.262**	**0.267**	**0.218**	**0.249**	**0.232(0.0223)**
Conv3D	MSE	0.0148	0.0151	0.0152	0.0129	0.0137	0.0136	0.0133	0.0142	0.0142	0.0142	0.0141(0.0007)
	Gain	-0.379	-0.392	-0.374	-0.222	-0.187	-0.231	-0.302	-0.387	-0.421	-0.414	-0.331(0.0833)
LSTM	MSE	0.196	0.215	0.186	0.195	0.194	0.189	0.160	0.181	0.197	0.194	0.205(0.0133)
	Gain	-17.3	-18.8	-15.8	-15.8	-15.8	-16.0	-14.6	-16.7	-18.7	-18.3	-16.9(1.3637)
CCM	MSE	0.0144	0.0143	0.0158	0.0161	0.0177	0.0161	0.0141	0.0142	0.0170	0.0147	0.0154(0.0012)
	Gain	-0.342	-0.314	-0.431	-0.532	-0.534	-0.455	-0.381	-0.385	-0.709	-0.457	-0.454(0.1097)
DCC	MSE	0.0151	0.0152	0.0143	0.0141	0.0150	0.0157	0.0133	0.0155	0.0148	0.0143	0.0147(0.0007)
	Gain	-0.405	-0.398	-0.293	-0.339	-0.302	-0.416	-0.303	-0.514	-0.482	-0.424	-0.388(0.0730)

Table 2: Performance Comparison on Different Datasets.

$\hat{\mathcal{M}}_{t+h} = \mathcal{M}_t$. For convenience, we name this simple strategy as *Previous*. We can use *Previous* method as a benchmark and calculate the gain of the predicted *ADM* $\hat{\mathcal{M}}_{t+h}$ comparing with *Previous* as follows:

$$\text{Gain} = 1 - \frac{\text{MSE}(\hat{\mathcal{M}}_{t+h})}{\text{MSE}(\mathcal{M}_t)} \tag{11}$$

Larger gain indicates better performance over *Previous*.

Parameters Setting We repeatedly run the model with various groups of hyperparameters and use the Bayesian method to fine-tune the proposed framework.

We study the influence of several important hyperparameters based on the aforementioned 10 stock datasets. The inital learning rate for the adaptive learning rate scheduler is set to $5e-4$. We have tested the model with the following batch sizes: $\{128, 256, 384, 512, 640\}$ and finalized the batch size to 512. Horizon h is an application-specific parameter, and since our application is portfolio management with monthly adjustments, we set $h = 21$. Linear and cubic transformation instead of the proposed quadratic one were investigated and the experiment shows that the model incurs higher cost and longer convergence time as the degree increases and the quadratic form reaches the best performance in terms of *MSE* and *gain*.

The performance of MoE depends on two crucial parameters, (1) the number of experts n_{exp}, which determines how many experts in total are contained in the network, and (2) top_k, which determines how many experts participate in generating the final transformation function ($\text{top}_k \leq n_{exp}$). Table 1 shows how the two parameters affect the learning of the transformation function $\mathbb{T}$ and in turn the prediction *MSE*. The entry in the table denotes given n_{exp}, the *MSE* returned by the optimal top_k on average in the 10 stock datasets. For example, when $n_{exp} = 8$, on average $top_k = 4$ obtains the optimal *MSE* among the 10 stock datasets. We have multiple observations from the table: (1) the optimal (n_{exp}, top_k) pair is (8,4), (2) n_{exp} and top_k are not the larger the better, which may be due to the fact that initial increase of n_{exp} to a optimal number can successfully capture the complexities of market data, but over-increasing n_{exp} lays a burden on model convergence and harms the performance.

Prediction Accuracy Results Analysis The overall prediction accuracy result is shown in Table 2. The average running time on each dataset is around 3 hours. We can observe that deep learning based methods generally outperform statistical methods (*CCM* and *DCC*), indicating the superiority of data-oriented methods over methods that greatly depend on unrealistic statistical assumptions, especially in the complex and dynamic financial domain. *LSTM*/*Conv3D*, which are simple models that utilize only temporal/spatial information, yields unsatisfactory prediction accuracy. The inferior

performance of *LSTM* further demonstrates the importance of spatiotemporal behaviors in *ADM* prediction compared to the consideration of temporal behavior alone.

By incorporating both spatial and temporal information, *ConvLSTM* improves prediction accuracy. To study if the addition of a transformation layer can help alleviate the limit imposed by the *ADM* representation problem on raw *ConvLSTM*, we evaluate *ConvLSTM* extended with different transformation methods. We can see that the performance improvement of *P-ConvLSTM* and *T-ConvLSTM* are marginal and sometimes even negative. For *P-ConvLSTM*, the static clustering approach failed to capture the time-varying patterns of the market. Even though *T-ConvLSTM* uses an end-to-end architecture that is superior to *P-ConvLSTM* in principle, it displays worse performance, which indicates that without proper design, the utilization of deep layers can degrade the model.

After comparing methods with various transformation function $\mathbb{T}$, we can observe from Table 2 that both *MLP-ConvLSTM* and *ADNN* in quadratic form achieve the best results. Compared with *MLP-ConvLSTM*, *ADNN* could better learn the optimal representation using a group of experts to handle the variety and complexity of the financial market. The superiority of *ADNN* over *MLP-ConvLSTM* lies in the nature of *MoE* which is able to model different market scenarios while *MLP* can only learn a single general model. Notice that the use of transformation function might have minor performance gain on some datasets (e.g. set 2). In these cases, perhaps the raw *ADM*s already exhibit good spatial property to *ConvLSTM* so the transformation would not help much. In conclusion, *ADNN* has gained consistent performance advantages on different datasets over various baselines. It shows the necessity of a well-designed transformation of raw *ADM* to solve the *ADM* representation problem.

Evaluation of Portfolio Risk Reduction

As introduced in the related work, "Modern Portfolio Theory" (Markowitz 1952) calculates the weight of each asset in the portfolio by considering the dynamic property of assets dependencies. Equation 12 briefly summarizes the idea:

$$\begin{aligned} \hat{\mathbf{\Sigma}}_m &= \text{diag}(\mathbf{s}) \cdot \hat{\Gamma}_m \cdot \text{diag}(\mathbf{s}) \\ \omega_m &= \arg\min_{\omega} \omega^T \hat{\mathbf{\Sigma}}_m \omega \\ \sigma_m^2 &= \omega_m^T \mathbf{\Sigma} \omega_m \end{aligned} \tag{12}$$

where $\hat{\Gamma}_m$ denotes the predicted correlation matrix with respect to forecasting method m (here $\hat{\Gamma}_m = \hat{\mathcal{M}}_{t+h}$), $\mathbf{s}$ denotes the ground truth standard deviation vector [2] at time $(t+h)$, diag($\mathbf{s}$) is to create a diagonal matrix with $\mathbf{s}$. We first calculate the predicted covariance matrix $\hat{\mathbf{\Sigma}}$ (first line of Equation 12) and obtain the optimal asset weight vector ω_m (second line). Then, the actual variance of the derived portfolio ω can be computed with the future groundtruth covariance matrix $\mathbf{\Sigma}$ (third line). In Equation 12, portfolio volatility σ_m^2 is a function of $\hat{\mathbf{\Sigma}}_m$, and when $\Sigma = \hat{\mathbf{\Sigma}}_m$ the *optimal* portfolio volatility/risk is obtained.

[2]The reason of assuming that future standard deviation vector is known is to eliminate the extra variability brought by $\mathbf{s}$ since the main purpose of evaluation here is to examine the performance of *ADM* (in this paper "correlation") prediction.

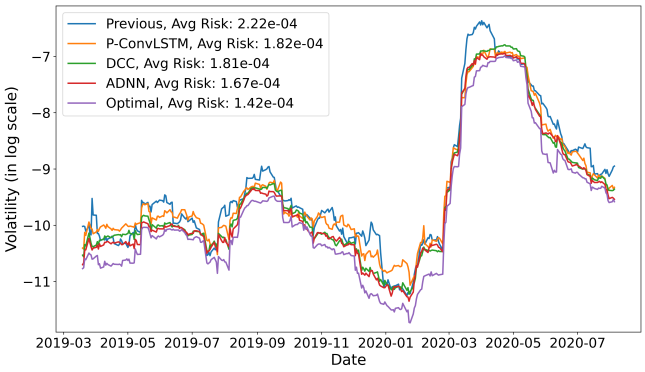


Figure 4: Portfolio Example using Dataset 7.

We randomly choose one stock dataset (set 7) as the source for constructing a diversified portfolio based on the predicted *ADM*s. All of the methods under evaluation build up their portfolios from the same 32 stocks in the stock dataset, which are then evaluated based on their respective portfolios' volatility σ_m^2. For baselines, we choose (1) the intuitive but commonly used *Previous* method (introduced in the beginning of evaluation of *ADM* prediction accuracy), (2) the well-known statistical method *DCC-Garch*, and (3) the second best deep learning method *T-ConvLSTM*. All baselines and the proposed *ADNN* will be compared to *Optimal* ($\hat{\mathbf{\Sigma}} = \Sigma$). For simplicity, we hold the assumption that no short selling is allowed (i.e. asset weight $\omega_i \in [0, 1]$).

The comparison results are shown in Figure 4. We can observe that, as expected, both *DCC* and *ADNN* are superior to the method *Previous*, and they display similar trends. *ADNN* outperforms *DCC* with an average risk of 1.67×10^{-4}, which ranks only second to the optimal portfolio. In terms of average risk, *ADNN* achieves an improvement of 24.8% over *Previous* and 13.9% over *DCC*. In conclusion, the *ADNN* method is capable of generating a more diversified portfolio compared with the baselines.

Conclusion

The prediction of asset dependency matrix (*ADM*) has been extensively studied in the financial industry. In this paper, we formulate the problem as sequential *ADM* prediction and solve the *ADM* representation and forecasting problem by proposing *ADNN*, which incorporates *MoE* for transforming *ADM* and facilitating the *ConvLSTM* model to predict future *ADM*s. We validate the effectiveness of *ADNN* on real stock data by comparing it with various baselines and apply it to the portfolio diversification task.

For future work, we plan to extend the current research in the following three directions: (1) design position transformation in an end-to-end manner and visualize the process to improve explainability, (2) apply the framework to more kinds of *ADM* (e.g., covariance matrix), and (3) compete against well-established portfolios with consideration of more realistic factors (e.g., transaction cost).

Acknowledgements

Research reported in this paper was supported by the Research Grants Council HKSAR GRF (No. 16215019) and UIC Research Grant (No. R201705). We appreciate the anonymous reviewers for their helpful comments on the manuscript.

References

Ane, T.; and Kharoubi, C. 2003. Dependence structure and risk measure. *The journal of business*, 76(3): 411–438.

Babaeizadeh, M.; Finn, C.; Erhan, D.; Campbell, R. H.; and Levine, S. 2017. Stochastic variational video prediction. *arXiv preprint arXiv:1710.11252*.

Baesel, J. B. 1974. On the assessment of risk: Some further considerations. *The Journal of Finance*, 29(5): 1491–1494.

Bengio, Y.; Courville, A.; and Vincent, P. 2014. Representation Learning: A Review and New Perspectives. arXiv:1206.5538.

Blume, M. E. 1975. Betas and their regression tendencies. *The Journal of Finance*, 30(3): 785–795.

Boubekeur, T.; and Schlick, C. 2007. QAS: Real-Time Quadratic Approximation of Subdivision Surfaces. In *15th Pacific Conference on Computer Graphics and Applications (PG'07)*, 453–456.

Caporin, M.; and McAleer, M. 2013. Ten things you should know about the dynamic conditional correlation representation. *Econometrics*, 1(1): 115–126.

Conover, C. M.; Jensen, G. R.; Johnson, R. R.; and Mercer, J. M. 2008. Sector rotation and monetary conditions. *The Journal of Investing*, 17(1): 34–46.

Elton, E. J.; and Gruber, M. J. 1973. Estimating the dependence structure of share prices–implications for portfolio selection. *The Journal of Finance*, 28(5): 1203–1232.

Elton, E. J.; Gruber, M. J.; and Padberg, M. W. 1977. Simple rules for optimal portfolio selection: the multi group case. *Journal of Financial and Quantitative Analysis*, 329–345.

Elton, E. J.; Gruber, M. J.; and Urich, T. J. 1978. Are betas best? *The Journal of Finance*, 33(5): 1375–1384.

Engle, R. 2002. Dynamic conditional correlation: A simple class of multivariate generalized autoregressive conditional heteroskedasticity models. *Journal of Business & Economic Statistics*, 20(3): 339–350.

Engle, R. F.; Ledoit, O.; and Wolf, M. 2019. Large dynamic covariance matrices. *Journal of Business & Economic Statistics*, 37(2): 363–375.

Gebert, P.; Roitberg, A.; Haurilet, M.; and Stiefelhagen, R. 2019. End-to-end Prediction of Driver Intention using 3D Convolutional Neural Networks. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, 969–974.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Goyal, P.; Dollár, P.; Girshick, R.; Noordhuis, P.; Wesolowski, L.; Kyrola, A.; Tulloch, A.; Jia, Y.; and He, K. 2017. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*.

Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation*, 9(8): 1735–1780.

Hsieh, J.-T.; Liu, B.; Huang, D.-A.; Fei-Fei, L.; and Niebles, J. C. 2018. Learning to decompose and disentangle representations for video prediction. *arXiv preprint arXiv:1806.04166*.

Jacobs, R. A.; Jordan, M. I.; Nowlan, S. J.; and Hinton, G. E. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1): 79–87.

Ma, J.; Zhao, Z.; Yi, X.; Chen, J.; Hong, L.; and Chi, E. H. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1930–1939.

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

Mathieu, M.; Couprie, C.; and LeCun, Y. 2015. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*.

Oprea, S.; Martinez-Gonzalez, P.; Garcia-Garcia, A.; Castro-Vargas, J. A.; Orts-Escolano, S.; Garcia-Rodriguez, J.; and Argyros, A. 2020. A Review on Deep Learning Techniques for Video Prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–1.

Rangel, J. G.; and Engle, R. F. 2012. The Factor–Spline–GARCH model for high and low frequency correlations. *Journal of Business & Economic Statistics*, 30(1): 109–124.

Shazeer, N.; Mirhoseini, A.; Maziarz, K.; Davis, A.; Le, Q.; Hinton, G.; and Dean, J. 2017. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. arXiv:1701.06538.

Shi, X.; Chen, Z.; Wang, H.; Yeung, D.-Y.; Wong, W.-K.; and Woo, W.-c. 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *arXiv preprint arXiv:1506.04214*.

Tashman, L. J. 2000. Out-of-sample tests of forecasting accuracy: an analysis and review. *International journal of forecasting*, 16(4): 437–450.

Torokhti, A.; and Howlett, P. 2001. On the best quadratic approximation of nonlinear systems. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 48(5): 595–602.

van Binsbergen, J. H. 2011. Anticipating Correlations: A New Paradigm for Risk Management. 49(1): 150–150.

Vasicek, O. A. 1973. A note on using cross-sectional information in Bayesian estimation of security betas. *The Journal of Finance*, 28(5): 1233–1239.

Wang, Y.; Jiang, L.; Yang, M.-H.; Li, L.-J.; Long, M.; and Fei-Fei, L. 2018. Eidetic 3d lstm: A model for video prediction and beyond. In *International conference on learning representations*.

Wang, Y.; Long, M.; Wang, J.; Gao, Z.; and Yu, P. S. 2017. PredRNN: Recurrent Neural Networks for Predictive Learning using Spatiotemporal LSTMs. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.