Neural Embeddings of Financial Time Series Data

Learning Useful Representations of Financial Time Series

Alik Sokolov Jonathan Mostovoy Brydon Parker Luis Seco

RiskLab - University of Toronto

Abstract

The dominant approaches for financial portfolio construction are reliant on estimating sample covariance and correlations matrices, which serve as an input into a number of classical portfolio construction techniques. These classical approaches are not forward looking, constrained by the ability to estimate covariance and correlation matrices, and inflexible to incorporating additional information. We propose a new approach of utilizing learned representations from deep learning networks to augment such classical techniques. Our approach is able to incorporate learned estimates of future performance, and can be customized to create tailored representations best suited towards varying financial objectives. We showcase one example of such an embedding, compare and contrast it with a classical approaches to portfolio construction, and discuss additional possibilities for applying representation learning in quantitative finance.

1. Introduction

Our work is motivated by viewing covariance and correlation as a special case of a distance metric, or a measure of dissimilarity between securities. Machine learning has been used to learn metrics directly [10], as well as by leveraging deep learning in semi-supervised contexts [11]. Much work has also been put towards developing general purpose representations [3] for common data sources, classical examples being text (words [14], text segments [5], [6]) and images [7].

In parallel, in the realm of finance, the Efficient Market Hypothesis (EMH) is generally accepted: this hypothesis states that all current information is already factored into a security's price. The EMH is particularly well established in terms of returns data (financial time series and their correlation structure), and although some work exists challenging the EMH [17] in this context, the usefulness of pure time-series based models for statistical arbitrage is still questionable.

In contrast, deep learning offers the flexibility of creating useful representations of complex datasets. Even in cases where direct prediction is not necessarily useful (as in the example presented in Section 2.1.1 below), deep representations have proven to be incredibly powerful.

Time series data is used extensively for hedging and portfolio construction, despite the difficulty of forecasting individual price movements. Correlation or covariance matrices computed from these time series, being distance metrics themselves, are frequently used for portfolio optimization [13]. We therefore propose an alternative approach for learning distance metrics that can ultimately be used for tasks beyond just portfolio optimization. Furthermore, our approach has the advantage of incorporation future expectations, as these metrics can be optimized by solving forward-looking tasks. In addition, it offers the flexibility of engineering different metrics by choosing different relevant tasks.

Machine learning has been used extensively in finance, but most applications of machine learning are based around unsupervised techniques. Over the past few years, unsupervised clustering algorithms have found root in portfolio construction [8]. As dealing with time series in general, unsupervised algorithms such as autoencoders, Deep Belief

Networks, or PCA have consistently shown to create more workable feature representation of high-dimensional and unstructured time series data [2]. Unfortunately, however, such approaches are limited in their ability to create useful representations as they are trained to solve tasks (such as reconstruction loss) that are unrelated to the ultimate objective. This flaw is principally why our approach utilizes a supervised algorithm instead.

The remainder of the introduction explains the structure of the paper, while also stating certain general implications of each section.

- Section 2 introduces the conceptual background required for both machine learning and finance.
- Section 3 outlines the preliminary definitions and notation to be used throughout the remainder of the paper.
- Section 4 outlines the general process for leveraging learned representations in finance, and outlines a sample empirical implementation for a transformer network to learn the structure of financial time series of NYSE and NASDAQ listed equities.
- Section 5 discusses how these representations can be evaluated.
- Sections 6 & 7 wrap up the paper with the significance and potential extensions of our work, specifically the potential to operationalize learned representations for a wider variety of tasks in finance using transfer learning techniques.

2. Technical Background

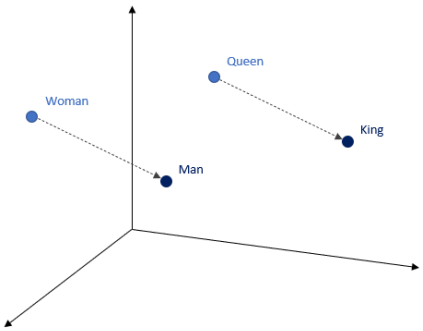
It is our intention to create connections between two vast and well-established fields: finance and machine learning. As such, in hopes of making this paper more accessible to researchers or professionals in either finance or machine learning, we have provided the following short colloquial backgrounds on each discipline included in this paper.

2.1. Machine Learning

Three interrelated concepts are necessary to discuss for this paper: transfer learning, representations, and the transformer network. We do, however, assume a very basic knowledge of deep learning and neural networks.

2.1.1 Representations

A learned representation may be thought of as a transformation of data into a more useful space via a machine learning algorithm. The usefulness of re-representing data into another space comes from the task at hand; common benefits are better performance on tasks such as classification, as well as the ability induce a metric on a dataset that did not originate with an intuitive notion of distance. A classical example is that of word representations, or word embeddings. Historically, a big barrier for natural language processing was the high dimensionality of text data, with no obvious way of representing individual words numerically in a useful manner. It turned out that by training a neural network using a "fill-in-the-blanks" task, the numerical representations created become highly useful [14]. This can be seen in the classical example below, where gender becomes an emergent property of our representation space (as can be seen with the fact that the vector separating "man" and "woman" is the same as the vector separating "king" and "queen"):



Although this result is easy to visualize and understand intuitively, word embeddings have been chiefly responsible for the formidable progress made today in the field of natural language processing. It is worth noting that word embeddings are also incredibly useful, even though the original network used to create these representations has little standalone practical use [14].

The power of deep learning (in comparison to other machine learning algorithms) for the purposes of general representation learning is well known [4]. It has created breakthroughs in fields as diverse

as natural language processing, image processing, and biological research. Furthermore, learning state space representations in reinforcement learning has recently propelled AI to dominance in games such as Go and Chess [16]. This potential is well known to extend beyond the specific task at hand, and learned representations have been shown to be useful across a wide variety of adjacent tasks [20].

In this paper, we will principally be interested in creating a useful representation of financial assets that can generalize across a variety of applications in the domain of finance. Later on in this paper, we empirically construct a representation for financial assets then proceed to demonstrate some of the applications of said representation.

2.1.2 Transfer Learning

Transfer learning may be defined as the use of the resulting embedding function learned via one machine learning algorithm for the purposes of solving a separate, but similar, task. A common example of this might be using a ResNet representation [**resnet**], trained on a substantial amount of images for tackling another, more specific, image classification task.

Transfer learning has become widely used in machine learning, particularly in the fields of natural language and image processing. The significant level of commonality in the data sets and the way one processes such data, regardless of task, has been the primary reason for the wide adoption of transfer learning in these fields. At the same time, transfer learning in finance is relatively under-explored. We hope to show that applying transfer learning techniques to the financial time series domain shows significant promise.

2.1.3 Transformer Networks

Time series data has long been tackled with Recurrent Neural Networks, and specifically LSTMs [15]. The "attention mechanism" has been shown to be able to augment LSTM-based architectures and improve results with NLP tasks [1]. More recently, the Transformer architecture has demonstrated that the recurrent network architecture can be abandoned altogether in favor of self attention, ushering in a revolution in the NLP domain [18]. Transformer networks have in the past shown promising results specifically for time series forecasting [9]. In this paper, we shall utilize the transformer architecture whilst hypothesizing that its ability to capture long-term dependencies and complex interactions will serve for greater accuracy in solving the tasks we assign to our network, and ultimately create better representations.

We refer the reader to a review of the aforementioned papers in this section for a proper introduction to Transformer Networks. For the financially predisposed reader, the main takeaway we'd like to emphasize is simply the adeptness of transformer networks for time series.

2.2. Finance

Arguably the most common quantitative metric/statistic asset managers look at when making asset allocations amongst a collection of assets would be sample covariance and correlation.

A prototypical treatment of covariance and or correlation would be as follows (henceforth, "covariance" may may be swapped without loss of generality for "correlation"). First, an asset manager would choose a suitable historical time horizon (say, 3 months worth of daily data) for a collection of time series' associated to a chosen group of assets. Then, over said time horizon, they would compute the sample covariance. This sample (historical) covariance then might be used for a variety of tasks under the assumption that this past covariance will stay constant for the foreseeable future. Examples of how an asset manager might utilize the covariance could be in constructing a minimum variance portfolio, a risk parity / equal risk contributions portfolio, or trying to understand fundamental relationships between assets via PCA. All techniques mentioned have considerable merit, but all are limited in part due to assumptions of constant covariance.

As mentioned previously, the covariance matrix is often used to solve a portfolio optimization task. An approach to solving this oftentimes difficult task is to instead first apply a dimensionality reduction technique. Principal components analysis, PCA, is most commonly chosen in Finance. In contrast to representations learned by neural networks, PCA is a linear technique which learns a transformation of the data with the property that the new representation forms an orthogonal basis in the directions which capture the most amount of sequential variance. We note that PCA has both a probabilistic interpretation (calculated by taking the leading eigenvectors of the covariance matrix as the principal

components) and geometric interpretation (linear manifold learning representing a lower-dimension region of the input space in which data density is maximized). [4].

PCA is quintessentially a representation learning technique and, given its level of familiarity to most readers, may serve as a good intuition for representation learning in general. We are, however, more interested in representations learned by deep learning networks for three main reasons:

- PCA is a linear technique, and therefore cannot be applied iteratively (stacked) to create more complex representations, as opposed to layers in a neural network which learn more abstract representations as the complexity of a network is increased, provided enough training data is available.
- PCA takes as an input a simple matrix. We are therefore unable to input tensors of arbitrary dimensionality, and it is the left to the user of the technique to create an initial feature representation when working with multi-dimensional data, such as tensors of stacked time series in our case. When working with deep learning representations, we are able to use layers such as convolution and self-attention to work with such high-dimensional data more efficiently, and ultimately create a convenient vector representation directly from a complex tensor.
- PCA is an unsupervised technique, and the representation it learns may not necessarily be optimal for all tasks. We create our deep representations by first solving a relevant task, making it more likely that the abstract representation we learn will be useful for related tasks.

The use of deep learning here is also especially pertinent given the well-known need for capturing the nonlinear nature of relationships in financial markets: Bengio et al. [3] emphasizes that deep learning is able to create "abstract" representations: representations that disentangle the factors of variation present in the input.

In further sections, we will demonstrate the utility of our approach in a common portfolio construction techniques of choosing representative securities from a hierarchical clustering of portfolios [8]. We demonstrate that instead of the classical approach of utilizing correlation as an input into the clustering algorithms, one has flexibility in choosing the correct learned representation to define a robust distance metric that fits the user's criteria, using learned representations.

3. Definitions and Notation

Definition 1
We consider a family of *neural networks* that we train to learn a function $f(X) = \hat{y}$, where X is a tensor representing raw financial time series (such as the log returns of an input asset, its sector, etc.), and $\hat{y}$ is the learned prediction. For our problem, the relevant dimensions are as follows: $X \in \mathbb{R}^{T \times d_{\text{feats}}}$, $y \in \mathbb{R}^{d_{\text{trgt}}}$, where T is the length of the input time series, d_{feats} is the number of input time series we consider, and d_{trgt} is the number of targets we want to make for each input (e.g. future returns at time t, future volatility at time t, etc.). Lastly, $f(X)$ is found by solving the standard *supervised learning task* of minimizing the *loss function* $\sum_{j=1}^{d_{\text{trgt}}} L_j(\hat{y_j}, y_j)$.

Definition 2
We define the *representation* for each asset as the learned outputs of our network for the final network layer (prior to the output layer), $X_{\text{embed}} = F_{\text{embed}}(X) = \{f_j(X) : 1 \leq j \leq d_{\text{embed}}\}$, where X is an input tensor, d_{embed} is the dimension of the final neural layer, and $f_j(X)$ is the value neuron j takes for input X.

The motivation for choosing F_{embed} to output the layer preceding our output layer is the fact that the final decision function, $F_{\text{output}}(X_{\text{embed}})$, is linear. Due to the linearity, we therefore establish a sense of well-behavedness for our representation. We extract these embeddings by "stopping short" of producing our final output, extracting values $F_{\text{embed}}(X) = X_{\text{embed}}$, with $X \in \mathbb{R}^{T \times d_{\text{feats}}}$ as above, and $X_{\text{embed}} \in \mathbb{R}^d$.

3.1. Finance

Definition 3
If we consider a collection of N assets, we define the *price* and *log returns* (or, for brevity, *returns*) of asset $1 \leq i \leq N$ at time t to be $P_i(t)$ and $r_i(t)$ respectively, where:

$$r_i(t) := \log(P_i(t)) - \log(P_i(t-1))$$

Further, we refer to $\{P_i(t) : t_0 \leq t \leq T\}$ and $\{r_i(t) : t_0 \leq t \leq T\}$ as the *price* and *returns time-series* of asset i. For this paper, we will always assume time series are finite ($T \in \mathbb{N}, T < \infty$).

One of the most important statistics that asset managers look at when making asset allocations amongst a collection of assets would be sample covariance/correlation, defined in the classical way on $\{P_i(t) : t_0 \leq t \leq t_1\}$. Their interest comes from an appeal to minimizing risk - the more correlation in a portfolio, the riskier said portfolio. That said, once a position/portfolio is established, as covariances amongst assets change through time, asset managers will want to update the way they utilize covariance by rebalancing. Therefore, after some period of time, say $k \in \mathbb{N}$, the asset manager will rebalance based on $\{P_i(t) : t_0+k \leq t \leq t_1+k\}$. If we therefore continue looking at covariance in this updated scenario for $k, 2k, 3k, \ldots$, we refer to $|t_1 - t_0|$ as the "window", and k as the period of time for which we "roll-over". The choice of k and $|t_1 - t_0|$ being an arbitrary one, however, is yet another disadvantage of this classical approach.

Definition 4
We train our network by predicting, jointly, several *target metrics* $\{y_j : 1 \leq j \leq d_{\text{trgt}}\}$, using the time series $r_i(t)$. We therefore learn a function $f : \mathbb{R}^{T \times d_{\text{feats}}} -> \mathbb{R}^{d_{\text{trgt}}}$ that maps the feature time our features time series, of dimension $\mathbb{R}^{T \times d_{\text{feats}}}$, to a prediction for each of our targets, of dimension $\mathbb{R}^{d_{\text{trgt}}}$. The definitions of each target we used to train the network described in this paper may be found in Section 4.

Definition 5
In order to predict the targets $\{y_j\}$, we add some additional information to the returns $r_i(t)$. We concatenate the time series $\{r_i(t)\}$ with the time series $\{t\}$ that captures the relative position of each return (instead of positional encoding as used by Vaswani [18]), as well as several additional time series of market returns as additional features (details outlined in the empirical implementation section).

We concatenate all of the time series above into a matrix X, of dimension $T \times d_{\text{feats}}$, with d_{feats} as the number of input time series used.

4. Our Approach and Empirical Implementation

4.1. General Process for Leveraging Learned Representations

In general, our proposal for a new, highly general "learned representation for financial assets" follows the following schema. We describe this proposal from the outlook of a financial analyst.

1. Choose a collection of financial assets for which a financial analyst would be interested in understanding their interrelationships to a higher degree under a particular lens; e.g., portfolio optimization, behaviour in volatile markets, etc.

2. Make a priori and collect *target metrics* or tasks $\{y_j\}$, that should be related to the lens that the analyst would like to look at the interrelationships under; e.g., log returns, covariance, etc.

3. Train a neural network on said target metrics and extract the related representation.

4. Evaluate and use the learned representation in the context that it was created for.

This process therefore allows for the substitution of machine learning for classical statistics found in finance so as to create a highly bespoke/customizable (in terms of the lens an analyst would like through) representation of any collection of financial assets.

We showcase on particular implementation of this process below.

4.2. Data and Prediction Task

The data used for our empirical analysis was the daily adjusted-close price history of all stocks listed on the New York & NASDAQ Stock Exchanges from 2000 to 2018. The only filters we applied is to only use tickers with at least 128 days of returns history at the time they are sampled, and removing tickers that had data issues with their adjusted closing prices as observed on Yahoo Finance (tickers that had more than 10 days of over 100% or under -50% returns were removed, as these all had issues with their historical price data).

Upon converting the times series for $N \approx 6,000$ tickers to their log-returns, $R_i(t)$, we use a lag date window of T days in order to incorporate additional longitudinal history for each prediction. In particular, we calculated the following d_{feats} metrics:

- The log returns of the sector corresponding to each stock, calculated using the weighted average stock prices in the sector
- The log returns of the sector corresponding to each stock, calculated using the arithmetic average stock prices in the sector
- The log market log returns, calculated using the weighted average of all the stocks in our dataset
- The log market log returns, calculated using the arithmetic average of all the stocks in our dataset

Our approach involved applying T days of log-return history to predict future stock performance in terms of returns, absolute returns, and volatility at several points in the future. For this paper, our choice of T was taken as 128 days. The network was optimized on the following d_{trgt} joint tasks:

- Predict log returns and absolute log returns 1, 7, 14, and 28 in the future for each reference date
- Predict volatility 7, 14 and 28 days in the future for each reference date

We trained our network based on several concurrent tasks of forecasting both returns and volatility measures (absolute returns and standard deviation), across a variety of time horizons. This is done in order to create more general representations, suited for a wider range of downstream tasks. Finally, we calculate the total loss as the sum of L_1 losses for each individual prediction.

4.3. Neural Network Architecture

We utilize the *Transformer encoder layer*, per Vaswani [18], as the core component of our neural network. This choice for the architecture is due to the recent successes it has had in model sequential natural language data, as well as superior scalability in GPU environments and a slight advantage in performance on our task over comparable recurrent architectures.

We utilize multi-head attention and position-wise feedforward networks inside each encoder layer per Vaswani, but do not use positional encodings. Instead, we concatenate the ordinal position of each return $r_i(t)$ to our features vector as an additional feature.

As per Vaswani, the inputs and outputs of each layer are of dimension d_{model}, and the output of each sub-layer is $\text{LayerNorm}(x + \text{Sublayer}(x))$. We also utilize a *modified Transformer encoder layer* as the first encoder layer, where residuals connections are not utilized, and the output of each sublayer is simply $\text{LayerNorm}(x)$.

The motivation behind the *modified Transformer encoder layer* is to use an intermediate representation (not the learned residual) in order to create some internal features within the neural network, aiding our model in learning features representative of the global statistics of the time series (e.g. mean and variance). Although the transformer network excels at finding interactions between log-returns of individual days (as each transformer layer looks at pairwise interaction between the tokens, or in our case, days), it can struggle to find global structure across all tokens / times, which we know to be important in financial markets. In order to help the network capture these long-term dependencies, we connect the output of the first transformer block to the embedding layer of the final encoder block, computing N_{features} additional metrics based on the learned representation corresponding to the final N_{tokens} tokens of the *Modified Transformer Layer* (this adds an additional $N_{\text{tokens}} * N_{\text{features}}$ neurons to embedding layer). In our experiments, we observe that this helped the network converge much faster and to a better local minimum, allowing it to learn global features around the overall time series more effectively.

Our final architecture uses n_{blocks} transformer layers (1 with the inner residual connections removed, and $(n_{\text{blocks}} - 1)$ standard layers per Vaswani) with n_{heads} attention heads each. We remove the inner residual connections from the first transformer block in order to retain the full information in the first layer encodings, which are aggregated to capture the global time series structure. We use an embedding dimension of n_{embed} and encoder feed-forward network dimension $n_{\text{feedforward}}$, and a dropout of $P_{\text{attention\_drop}}$ for the transformer attention layer, and $P_{\text{residual\_drop}}$ for the transformer residual layer. This is all consistent with the Transformer network used in the original paper [18], with slightly lower network dimensionality,

which we found was the appropriate size given the size of our training data sample, the dynamics of our problem.

We added two hidden layers after the transformer blocks, with dimensionality of d_{hidden}, and dropout of $P_{\text{hidden\_drop}}$ applied to each layer. We then use the output of the final feedforward layer to extract the embeddings of each particular stock for our downstream analysis tasks, simulating the use of such an embedding as a learned distance metric. These are added primarily to reduce the dimensionality of our final representation, as well as add some additional expressiveness to our network to further linearize our final representation.

We utilized RADAM [12] as our optimizer, with a learning rate of r_{learning}, in order to speed up convergence and simplify the hyper-parameter search for the optimal learning rate. We found that convergence was fast and did not benefit from a warm-up epoch when using the RADAM optimizer, and that we the hyper-parameter search was significantly simplified as we were able to set a narrower and higher range of learning rates.

Finally, we experimented with our batch size and early stopping criteria. The hyperparameter choices for our final architecture are summarized below:

Hyperparameter Description	Final Choice
Lag window of T days	128
Number of *feature time series* d_{feats}	7
Number of *target metrics* d_{trgt}	11
Embedding dimension n_{embed}	256
Number of transformer blocks n_{blocks}	6
Transformer block number of heads n_{heads}	16
Transformer block feed-forward network dimension $n_{\text{feedforward}}$	1024
Transformer block attention dropout $P_{\text{attention\_drop}}$	0.1
Transformer block residual dropout $P_{\text{residual\_drop}}$	0.1
N_{tokens} for feature computation	3
Internal features computed based on token embeddings	Standard deviation, L1, L2, and L3 norms ($N_{\text{features}} = 4$)
Hidden layer dimensions d_{hidden}	256, 128
Hidden layer dropout $P_{\text{hidden\_drop}}$	0.25
Optimizer	RADAM
Learning rate r_{learning}	5×10^{-5}
Batch size	32
Evaluate every	200 steps
Early stopping	no improvement after 3 consecutive evaluations

The best model found, based on the final hyperparameter choices described above, had an aggregate L_1-loss of 1.447×10^{-2}, 19.25% lower compared to a statistically derived benchmark loss of 1.792×10^{-2}. Our approach for calculating the benchmark was estimating future returns and volatility using statistical measures based on historical averages during our lag window of T days, using statistical estimates for each task as follows.

This tells us that the network is able to learn a representation that successfully captures the structure of the time series, and is able to make predictions on future performance reasonably well. Most of the gain over a statistical estimate comes from predicting volatilities and absolute log returns, with the predictions for 1, 7, and 28 day log returns being only marginally better than utilizing the mean. This is consistent with literature and the known difficulty of predicting returns; but, as we will see, does not prevent us from learning representations having properties that can be generally useful for a range of different tasks.

5. Evaluating the Representations

We evaluate our representations by using them to supplement a common approach in quantitative finance of findings clusters of stocks using clustering techniques such as hierarchical clustering [8], which can then be used for downstream tasks such as hedging and portfolio construction.

We compare two approaches towards creating 10 hierarchical clusters on the first trading day of each month from January 2000, to August 2018; for consistency and simplicity, we sample the 1,000 largest stocks in each period by market capitalization for our experiment. The first approach being from quantitative finance:

1. Calculate the correlation matrix of the stocks in each period, again using our standard 128 day window
2. Utilize a modified correlation matrix (in our case, 2 - Corr) to create a pseudo-distance matrix D
3. Use the matrix D to calculate hierarchical cluster assignments for each period using the Ward criterion [19]

For comparison, we apply an alternative approach utilizing our learned representations:

1. Evaluate the embedding of each stock in each period by doing a forward pass through our trained network up to the final fully connected layer

2. Calculate a distance matrix in the embedding space using the standard Euclidean distance between each stock's embedding
3. Use the matrix 2 to calculate hierarchical cluster assignments for each period using the Ward criterion [19]

We note that the representations we create generates clusters that are more similar than correlation clusters in terms of their average risk-return profile in the 28 days following the first day of each period. This is, of course, natural given our training scheme, which optimizes these representations to solve tasks related to this outcome. Still, it is instructive that this structure is captured in the relative representations of each stock in the embedding space. These results are summarized below.

We first evaluate how separated stocks are along some of our key metrics across clusters. To calculate this, we:

1. Calculate the weighted average $\mu_{t,j}$ of each metrics y_j (log returns, volatility, etc.) across all clusters at time t
2. Calculate the intra-cluster volatilities at each time t as $\frac{V_1}{(V_1^2 - V_2)} \times \sum_{i \in t} w_i (y_{t,i} - \mu^*_{t,j})^2$, weighted by the number of stocks in each cluster, and for each metric

The chart below shows the weighted intra-cluster volatility of the mean stock volatilities, as a measure of how well each clustering approach is able to group stocks that will have similar volatilities in the future. As expected, our approach creates clusters that are consistently better separated in terms of future volatility:

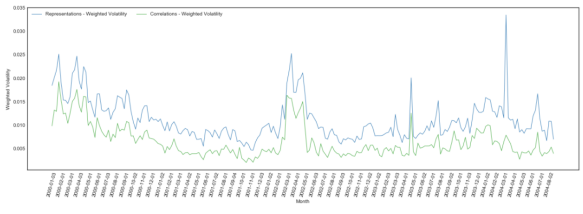


Figure 1: Intra-Cluster Volatility: Stock Volatilities

This is not the case for future returns; this is is also to be expected, as predicting future returns based on time series data is known to be ineffective due to rapid repricing. Nevertheless, we believe adding such tasks can induce the learned embeddings to be more useful, even when the performance on some of the initial tasks is lacking.

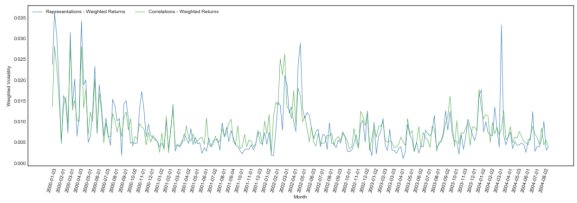


Figure 2: Intra-Cluster Volatility: Stock Returns

6. Significance

We show that modern machine learning approaches can be used to learn distance metrics at scale, and offer some examples on how such an approach may differ from using historical correlations. Learning tailored representations based on specific tasks is an approach that has shown tremendous value in domains of natural language processing and image processing, yet has not been utilized in finance. Financial time series data, however, is comparable with those two domains through its abundance of training data to learn rich representations. It is perhaps set to benefit even more from learning such representations due to the prevalence of distribution drift, and the abundance of varying financial objectives that warrant learning more diverse representations of the same data.

The flexibility offered up can help create general-purpose embeddings that serve the financial needs of creating stable, robust portfolios that account for overall market conditions. It is natural to engineer tasks for pre-training that will create desirable properties in representations of time series data, such as stability of representations over time and incorporation of additional market data (including non-time series data such as financial news and fundamental data). All of these advantages make deep neural networks a highly promising approach towards creating new ways to process market data, creating rich representations suitable for improving existing financial strategies and creating new ones.

7. Further Work / Limitations

Our future work will focus on additional improvements for stability, capturing additional market information, and learning representations that best capture the behavior of the market jointly.

Creating time series representations that control for stability of representations over time is another promising area of research, which can help

create systems of hedging and portfolio construction that are stable over time (and hence more testable), helping reduce some of the risk associated with potentially unknown failure points created by highly nonlinear techniques such as deep learning.

Although such approaches can be criticized for their "black-box" nature, we argue that the downside compared to the traditional correlation-driven approach is low. An additional metric needs to be interpreted, but the portfolio construction and hedging work can be performed as is. The additional flexibility neural networks offer of tailoring the representations to specific pre-training tasks, and our ability to inject domain expertise into how these tasks are specified, will over time lead to wider adoption of such systems.

References

[1] Bahdanau, D., Cho, K., Bengio Y. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2014. arXiv: 1409.0473 [cs.CL].

[2] Bao, W., Yue, J., Rao, Y. "A deep learning framework for financial time series using stacked autoencoders and long-short term memory". In: *PLOS One* (2017). DOI: 10.1371/journal.pone.0180944.

[3] Bengio, Y. "Deep Learning of Representations for Unsupervised and Transfer Learning". In: *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*. Ed. by Isabelle Guyon et al. Vol. 27. Proceedings of Machine Learning Research. Bellevue, Washington, USA: PMLR, Feb. 2012, pp. 17–36.

[4] Bengio, Y., Courville, A., Vincent, P. *Representation Learning: A Review and New Perspectives*. 2012. arXiv: 1206.5538 [cs.LG].

[5] Cer, D., Yang, Y., Kong, S., Hua, N., Limtiaco, N., St. John, R., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Strope, B., Kurzweil, R. "Universal Sentence Encoder for English". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 169–174. DOI: 10.18653/v1/D18-2029. URL: https://www.aclweb.org/anthology/D18-2029.

[6] Devlin, J., Chang, M., Lee, K., and Toutanova, K. "BERT: Pre-training of deep bidirectional transformers for language understanding". In: *North American Association for Computational Linguistics (NAACL)* (). arXiv: 1810.04805 [cs.CL].

[7] He, K., Zhang, X., Ren S., Sun, J. "Deep Residual Learning for Image Recognition". In: (2015). arXiv: 1512.03385.

[8] León, D. Aragón, A., Sandoval J., Hernández, G., Arévalo, A., Niño, J. "Clustering algorithms for Risk-Adjusted Portfolio Construction". In: *Procedia Computer Science* 108 (2017). International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland, pp. 1334–1343. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2017.05.185. URL: http://www.sciencedirect.com/science/article/pii/S187705091730772X.

[9] Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y., Yan, X. *Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting*. 2019. arXiv: 1907.00235 [cs.LG].

[10] Li, W., Huo J., Shi, Y., Gao Y., Wang, L., Luo, J. *Online Deep Metric Learning*. 2018. arXiv: 1805.05510 [cs.CV].

[11] Li, X., Yin, H., Zhou, K., Zhou, X. "Semi-supervised clustering with deep metric learning and graph embedding". In: *World Wide Web* (2019). DOI: 10.1007/s11280-019-00723-8.

[12] Liu, L., Jiang, H., He, P., Chen, W., Liu, X., J. Gao, J., and Han J. "On the variance of the adaptive learning rate and beyond". In: (2019). arXiv: 1908.03265 [cs.LG].

[13] H. Markowitz. "Portfolio Selection". In: *The Journal of Finance* 7.1 (1952), pp. 77–91. ISSN: 00221082, 15406261. URL: http://www.jstor.org/stable/2975974.

[14] Mikolov, T., Chen K., Greg C., Dean J. "Efficient Estimation of Word Representations in Vector Space". In: (2013). arXiv: 1301.3781.

[15] Siami-Namini, S., Tavakoli, N., Siami Namin, A. *A Comparative Analysis of Forecasting Financial Time Series Using ARIMA, LSTM, and BiLSTM*. 2019. arXiv: 1911.09512 [cs.LG].

[16] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, Ma., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., Hassabis, D. "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play". In: *Science* 362.6419 (2018), pp. 1140–1144. ISSN: 0036-8075. DOI: 10.1126/science.aar6404. eprint: https://science.sciencemag.org/content/362/6419/1140.full.pdf. URL: https://science.sciencemag.org/content/362/6419/1140.

[17] Varaku, K. *Stock Price Forecasting and Hypothesis Testing Using Neural Networks*. 2019. arXiv: 1908.11212 [q-fin.ST].

[18] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., Polosukhin, I. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL].

[19] Ward, J. "Hierarchical Grouping to Optimize an Objective Function". In: *Journal of the American Statistical Association* 58.301 (1963), pp. 236–244. DOI: 10.1080/01621459.1963.10500845. eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1963.10500845. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1963.10500845.

[20] Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., He Q. *A Comprehensive Survey on Transfer Learning*. 2019. arXiv: 1911.02685 [cs.LG].