Smart Portfolios using Autoencoders

Greg Meyer

MSc Data Science, Department of Statistics, London School of Economics and Political Science, Columbia House, Houghton Street, London, WC2A 2AE, United Kingdom

Abstract

S&P 500 index funds are among the most popular investments in the financial space today. Historically, they have offered investors high returns without the need to take on large risk, minimal costs and exposure to a large number of stocks. However, investing in the S&P 500 has its shortcomings. Index fund units can only be purchased as a bundle and components comprising the S&P 500 cannot be separated. Moreover, the market capitalisation weighted nature of the index concentrates a large proportion of the investment in large corporations resulting in low diversification and higher risk. This paper explores solutions to the shortcomings of index funds by proposing alternative investable portfolios that replicate or outperform S&P 500 returns while achieving a lower risk profile through diversification. Two approaches are identified. First, autoencoders are used as feature extractors to identify high information stocks from which smart portfolios are subsequently constructed. The second approach uses low noise encoded features as inputs to a predictive LSTM network - the stock price predictions are used to construct smart portfolios comprised of stocks with the highest average expected growth. Results show that constructed portfolios achieve higher risk-adjusted returns than the S&P 500 index for periods of the test set.

1 Introduction

Financial markets form a large component of the world economy aiming to preserve and grow the value of the investor's asset. Index funds, the recommended investment vehicle by world-renowned investors like Warren Buffet and David Swanson, are of particular interest due to their high performing historical returns and low risk profile.

A stock index is a measurement of the market capitalisation (total value) of a selected subset of stocks from the stock exchange. A popular index is the S&P 500, a market capitalisation weighted fund comprised of the 500 largest corporations on the New York Stock Exchange. While direct investment into the S&P is infeasible as a result of the large amount of capital required, individuals can gain exposure to indices through index funds or ETFs (exchange-traded funds). Frino and Gallagher show that these S&P 500 index funds, on average, outperform active funds [6].

With the above in mind, there are two important characteristics of index funds that might lead individuals in search of alternative options. Firstly, investment in an index fund leaves little discretion to the investor. These funds are controlled by portfolio managers and individual components on the index cannot be traded. The inflexible nature of such funds means that a portfolio capable of replicating S&P 500 returns would provide investors with a more efficient investable substitute.

A second, more concerning downfall of S&P 500 index funds is that they expose investors to non-negligible levels of risk. The S&P 500 is a market capitalisation weighted fund. In short, this means that companies worth a higher value are given a larger weighting in the fund. This is desirable if high value corporations perform well, however, it exposes investors to high levels of concentration risk. Concentration risk is the result of a lack of diversification over different assets, where an investor stands to lose large sums of money if his highly weighted stocks underperform. For this reason, diversification is a sought after requirement when building portfolios. While the S&P 500 consists of a large number of components, the nature of its weighting scheme heavily skews the performance towards that of the highly valued stocks. The extent of this is that the top 5 companies (1% of the total companies) on the S&P 500 index make up 20% of its value.

A solution to the above shortcomings of S&P 500 index funds is provided by smart portfolios. First described by Markowitz [10], a smart beta index or fund aims to outperform traditional indices by re-weighting the stocks in the index to achieve better risk-adjusted returns. 'Smart' refers to the aim of achieving higher returns whereas 'beta', a financial measure of risk, refers to ensuring the index has a lower risk profile than standard market capitalisation weighted indices. For the purposes of this paper, highly diversified portfolios are considered low risk. By fixing the diversification requirement, the task that remains is to construct a smart portfolio with the same or higher returns benchmarked against the S&P 500.

The natural question that follows is how can one construct such a fund? Portfolio managers have traditionally used financial techniques incorporating value investing. However, in a time where data is abundant, deep learning might be capable of identifying smart portfolios that are not obvious to the human investor. Moreover, the highly non-linear topology present in financial data makes deep models a strong candidate. Autoencoders are of particular interest as a result of their ability to extract informative features from input data.

The concept of an auto encoder, first described by Ballard [2], revolves around the compression of input data into informative, noiseless features using the encoder. The coded features are then reconstructed with a decoder. An autoencoder's defining characteristic is that targeted outputs are equivalent to the original input. Figure 1 illustrates the structure of a simple autoencoder. This paper uses autoencoders in two ways. First, they are used directly to construct a portfolio of highly informative stocks. Secondly, they are used to denoise the inputs of a predictive LSTM network. In both cases the aim is to construct high performing smart portfolios ca-

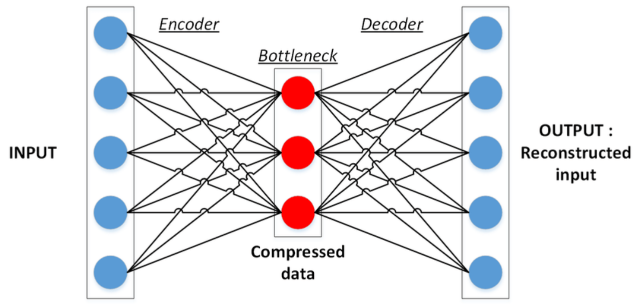

Fig. 1. Deep autoencoder architecture

pable of replicating or outperforming S&P 500 returns while maintaining a lower risk profile through diversification.

The structure of this paper is as follows: First, smart indexing from Deep Learning in Finance [8] is extrapolated to more recent data as an introduction to autoencoders in a financial setting. The paper proceeds to explore a variety of autoencoder architectures as financial feature extractors and compares their suggested portfolios to the S&P 500. Architectures analysed include stacked, convolutional, LSTM and variational autoencoders. Lastly, an LSTM network is trained to predict future stock prices using low noise encoded features as inputs. All constructed portfolio performances are compared to that of the S&P 500 index, which I refer to as the S&P from this point onwards.

2 Dataset

The dataset is comprised of data on S&P 500 component stocks. More specifically, analysis is conducted on price and fundamental stock data. All data is obtained using the Tiingo API. The S&P 500 consists of 505 stocks, updated quarterly based on a market capitalisation ranking. To ensure consistency across results, I make three changes to the actual S&P 500 index values reported by Standard and Poor's. Firstly, I select stock components on the S&P 500 as at 02/05/2020 instead of allowing component updates every quarter. These stocks will be kept constant throughout the study and used as a benchmark against which to measure alternative portfolios.

Secondly, I use the market capitalisation value of the S&P index as a metric as opposed to the figure reported by Standard and Poor's. This figure is calculated by reducing the market capitalisation by a proprietary divisor that Standard and Poor's may change at their discretion. Using market capitalisation as an alternative ensures that results are in line with actual market values.

Lastly, I restrict analysis to 479 of the 505 stocks. The reason for this is that some companies have been significantly restructured and have had their shares diluted in some cases. This leads to inconsistent results that would distort analysis if such stocks were included in a competing portfolio. Two examples of restructured stocks are Arconic Corp (ARNC) and Fox Corp (FOX). Removal of these stocks from the analysis is important to ensure validity and consistency of the study over the entire date range.

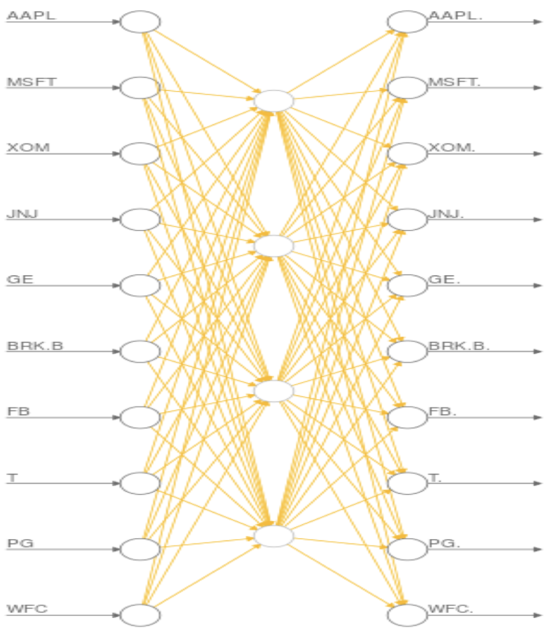

Fig. 2. Representation of 10 inputs and outputs of the full autoencoder

3 Extrapolating Deep Learning in Finance

Heaton et al. propose the use of an autoencoder to identify stocks with the highest communal information [8]. They achieve this by autoencoding all stocks and classifying information content based on a stock's input proximity to its autoencoded output. A portion of the autoencoder is shown in Figure 2.

I apply the above method by fitting a deep autoencoder on 479 stocks using a hidden layer of size 4 (compression factor of 119.75). Visualisations of loss and other metrics can be found in the notebook. After tuning hyperparameters, I run 800 epochs and measure the decoder's ability to reconstruct stock values using a custom metric called bound accuracy. This measure considers a prediction accurate if it falls within a certain threshold of its input. Using a threshold value of 0.01, the model reaches a bound accuracy of 94.83% suggesting that the 4 units in the hidden layer contain a significant amount of information which the decoder is successfully able to interpret and reconstruct. I proceed to extract stocks with the lowest mean absolute error, implying high information content. Once I have obtained the 20 highest information stocks I compare their performance to the S&P index as a whole. The results can be seen in Figure 3. Data to the right of the green line is the test data. It can be seen that high information stocks follow the trend of the S&P at a lower level. The stocks also better replicate the S&P during the early stages of training.

While the experiment provides interesting results, there are some caveats to note. Firstly, basing communal information on the similarity between input and outputs is assumptive - there is no evidence to show that similarity between input and autoencoded stock prices implies high communal informa-

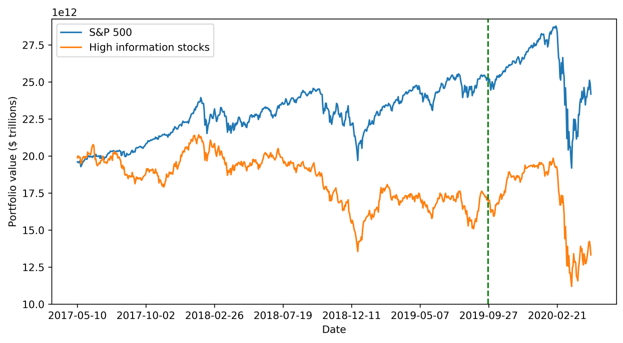

Fig. 3. Portfolio value of 10 highest information stocks compared to the S&P

tion. Secondly, the performance of high information stocks is significantly worse than that of the S&P index. In spite of these caveats, Figure 3 suggests that the information contained in these high information stocks exhibits similar trends to the S&P albeit at a lower level. For this reason, the example serves as motivation for further research into the use of autoencoders as a feature extractor.

4 S&P Replication using Autoencoders

This section further develops the ideas put forward in Section 3 attempting to replicate S&P returns with a lower risk portfolio. This is done by extracting relevant features from stock data using a variety of autoencoder architectures and selecting stocks based on how well they represent the S&P. The logic behind this is that stocks with high information content will contain features that represent the index as a whole (they contain little noise) and therefore can be accurately reconstructed from a compressed set of features. Following on from Section 3, I use the 20 most accurately reconstructed stocks as the comparison portfolio. Before visualising these portfolios, I briefly discuss the different architectures employed.

The data is split up into three portions - training data, validation data and test data. In all cases, I construct autoencoder models using the training data and perform grid searches to select optimal hyperparameters based on performance on the small validation set. These hyperparameters can be found in the relevant Jupyter notebook sections. After obtaining these hyperparameters, I retrain the model on all the training data (including the validation set). Model performance is compared using the test set. While it is difficult to compare test losses across models due to differing loss functions, I still include the results in Table 1. Bound accuracy, however, can be compared across all models. Bound accuracy is the metric I developed to measure the proportion of normalised autoencoded share prices that fall within 0.01 units of the original input. Each model and its corresponding training metrics are discussed below.

A. Stacked Autoencoder. First proposed by Bengio et al., stacked autoencoders (SAEs) are constructed by stacking a

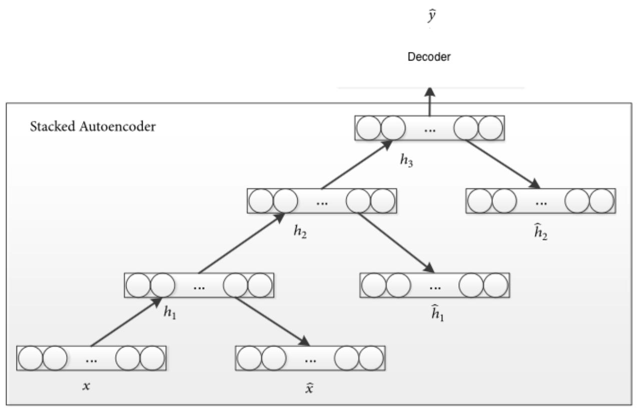

Fig. 4. Stacked autoencoder architecture

sequence of trained single-layer autoencoders layer by layer and fine-tuning the resulting stack [4]. The motivation behind this architecture is that the strategy better optimises and generalises deep networks by initialising upper layers with more accurate representations of high level abstractions. Bao et al. are among the first to integrate stacked autoencoders into a financial framework to be used as feature extractors [3]. Their paper focuses on stacking multiple single layer autoencoders where autoencoders are trained one at a time in order to learn invariant and abstract features.

Using a similar method, I train an SAE by fitting 4 separate autoencoders sequentially following the advice of Chen et al. [5]. After the first autoencoder has been trained, its encoder/reconstruction layer is removed and the encoded middle layer serves as the input to the second autoencoder. This process is repeated until the final autoencoder in the stack, whose output is then mapped to the original input data using a decoding layer. This process is illustrated in Figure 4. Using input data x and output $\hat{x}$ (where $x = \hat{x}$), I train the hidden layer h_1. The output from the trained hidden layer h_1 is then used as input into another autoencoder with hidden layer h_2. This process continues until the final decoder at which point the entire stack of encoders is fine-tuned to obtain the fitted model.

Table 1 suggests that the SAE is fairly successful in reconstructing test input values with a loss of 0.0004 and bound accuracy of 0.6774.

B. Convolutional Autoencoder. Interestingly, there is little past research on the use of convolutional autoencoders to extract financial features. Convolutional autoencoders (CAEs) have performed well on image feature extraction [11]. I extend the concept of a CAE to financial data and observe its ability to extract informative features.

The CAE implemented in this paper encodes input data using a collection of 1D convolutional and max pooling layers followed by a decoder comprising convolutional and upsampling layers. At first glance the CAE seems to perform extremely poorly and unsuccessfully reconstructs convolutionally encoded data with a test bound accuracy of 0.2801. With this in mind, poor reconstruction accuracy is not necessarily indicative of an undesirable model. If the CAE reconstructs inputs that are consistent relative to each other, the encoded output will contain relevant features, even though the absolute reconstructed values will be far off the original input. I expand on this idea in the final part of this section.

Model	Test Loss	Test Bound Accuracy
Stacked AE	0.0004	0.6774
Convolutional AE	0.0023	0.2801
LSTM AE	0.0001	0.8774
Variational AE	0.1677	0.6866

Table 1. Displays the test loss and bound accuracy for different autoencoder architectures.

C. LSTM Autoencoder. First proposed by Hochreiter and Schmidhuber, LSTM networks have achieved the most success in capturing temporal structures [9]. Because the financial data used in this paper is sequential in nature, an LSTM autoencoder seems like a good model choice. Before feeding data to the LSTM autoencoder I reshape the inputs into window periods of 10 days where each data point consists of two trading weeks of adjusted closing prices for all stocks on the S&P (590 training data points with 10 timesteps each and 479 features).

As seen in Table 1, the LSTM autoencoder achieves the lowest loss and highest test accuracy. The LSTM autoencoder clearly fits a model most capable of reconstructing the output from a compressed layer. However, these impressive results do not necessarily translate into high performing portfolio selections. This is further discussed towards the end of the section.

D. Variational Autoencoder. Recently, variational autoencoders (VAEs) have shown promising results as financial feature extractors and are useful tools in learning latent representations [1]. They differ from standard autoencoders during the encoding step where inputs are encoded into two parameters as opposed to a regular encoded layer. Samples are then drawn from a normal distribution defined by these two parameters (μ and σ). Finally, these samples are decoded with the aim of producing the original input. The loss term for VAEs is comprised of two components. The first component is a reconstruction error term, as seen in all other autoencoders. The second component is the KL divergence between the normal distribution defined by μ and σ and the standard normal distribution. This component imposes some regularisation on the model encouraging all encodings to be distributed evenly around the centre of the latent space. The VAE architecture used in this paper can be seen in Figure 5. The motivation behind the use of VAEs stems from regular autoencoder models being prone to overfitting leading to close points in the latent space being mapped to very different outputs. Encoding a distribution into the latent space removes this obstacle.

The VAE test loss should not be compared to other models as it consists of two terms. The test bound accuracy is similar to that of the stacked autoencoder displaying adequate re-

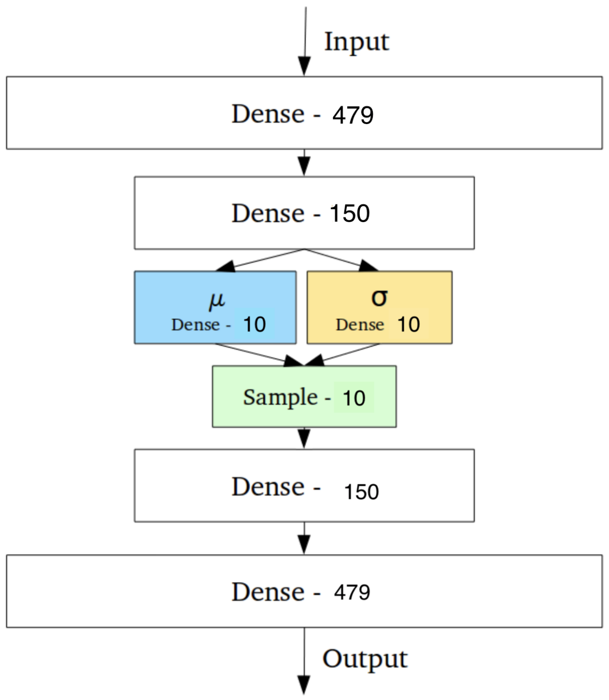

Fig. 5. Variational autoencoder architecture

construction performance. However, my concern is that the randomness introduced as a result of the sampling step might lead to poor replication results. This concern is addressed at the end of this section.

E. Autoencoder Portfolio Performance. Lastly, I visualise how the chosen high information subset of stocks for each architecture compares to the S&P 500 index. To prevent the focus of this paper shifting to a financial risk discussion, I make the assumption that diversified portfolios are equivalent to lower risk portfolios. All constructed portfolios weight the 20 stocks equally and are therefore considered to be better diversified than the S&P as a result of their lower concentration risk. If the autoencoder is able to select a subset of stocks with similar returns to the S&P, the risk-adjusted returns are higher due to the portfolio's lower risk profile. For graphical purposes, I split the visualisation across two plots - the results can be seen in Figure 6 and Figure 7. Portfolio performance on test data is found to the right of the green dotted line. Individual plots for each architecture against the S&P can be found in the Jupyter notebook.

Interestingly, the stacked and convolutional autoencoders result in portfolios that perform in line with the S&P whereas the LSTM and VAE models display poorer results. The close replication of the CAE portfolio is particularly surprising due to the low bound accuracy achieved on test data. The success of this model can be explained partially by the structure of a convolutional layer. In a similar way to how each convolution detects edges and features in an image, the convolutional layer detects significant features present in the financial data. While convolutions make absolute reconstruction of the original output difficult, the relative accuracy among stocks

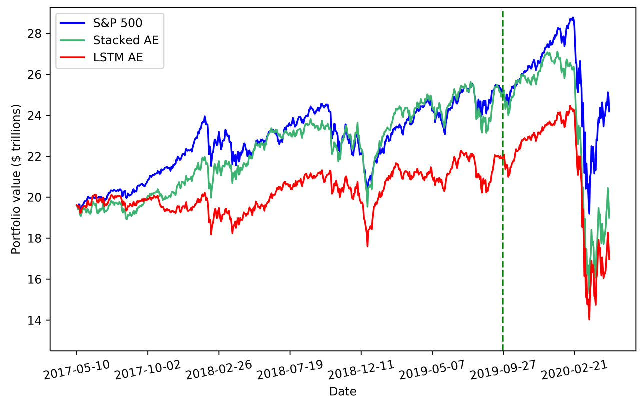

Fig. 6. Stacked and LSTM AE portfolio performance benchmarked against the S&P 500

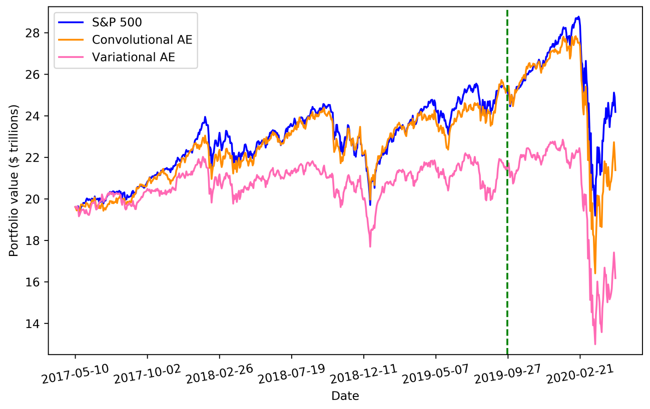

Fig. 7. Convolutional and Variational AE portfolio performance benchmarked against the S&P 500

seems to be in line with an information content ordering.
In contrast to the CAE, the VAE performs poorly and only replicates the S&P in the early training stages. I believe this can be explained by the tendency of VAEs to regularise the latent space. The latent space is essentially a distribution from which points are sampled and then decoded which ensures that points close together in the latent space are likely to be decoded in a similar way. This is desirable when building a generative process, however, the forced regularisation of the latent space can often result in the loss of important distinct features.
Another surprisingly poor model is the LSTM autoencoder. With the highest bound accuracy and best ability to accurately reconstruct input, one would think that the LSTM autoencoder should perform best. The model's poor performance is succinctly explained by Goodfellow et al. LSTM models are designed such that not all aspects of the past sequence are encoded to the hidden layer. This creates a problem where the lossy encoded summary of input data might not contain potentially significant features from the past sequence [7]. While the large number of parameters in this paper's LSTM autoencoder allow for accurate reconstruction, the LSTM cell structure results in a latent space with less informative features.
Contrastingly, the stacked autoencoder does a good job of closely replicating the S&P. SAE success can be attributed to the sequential training of each layer within the autoencoder which helps optimisation by initialising weights near local minima [4]. The SAE can be seen as a better initialised version of the simple autoencoder from Section 3 resulting in more relevant feature extraction.
In conclusion to this section, it can be argued that portfolios proposed by SAEs and CAEs achieve a good level of replication with similar trends and market capitalisation values to the S&P across the entire date range. Their performance is far superior to the example discussed in Section 3. However, the results are still not fully satisfactory. Although the equally weighted portfolios are better diversified than the S&P and hence lower risk, their marginally lower returns prevents them being classified as true smart portfolios. A possible reason for this is that an accurately reconstructed stock is not necessarily indicative of a high information stock.

The section that follows attempts a different approach in constructing portfolios where the focus is shifted from replicating the S&P to outperforming it while taking on a lower risk profile.

5 LSTM Portfolio Construction

The direct application of LSTMs to raw financial data has not been particularly successful in past literature. High noise-to-signal ratio found in financial data leads to a model with little predictive power that trivially predicts a value very close to the previous day. However, the use of low noise financial feature inputs can harness the predictive capability of LSTMs. Autoencoders are capable of extracting these features which can then be fed into an LSTM network.
This section follows a different line of analysis from Section 4 where instead of assessing a stock's information content, encoded features are used to train an LSTM network that aims to predict adjusted closing stock prices for the first 60 trading days of the test set. Portfolios are then constructed using stocks expected to show the highest levels of growth.

A. Input data. The training data fed into the network has shape (samples, timesteps, features). In all cases there are 481 samples (471 for the LSTM autoencoder), each with 60 timesteps (representing 60 days of data) and a differing number of features depending on the dimension of the architecture's encoded space.

B. Network architecture. The network used for analysis is a many-to-many LSTM model. I use two stacked LSTM layers with 32 and 16 hidden neurons respectively followed by a TimeDistributed dense layer. The TimeDistributed wrapper allows for the dense layer to be applied to each timestep. An important part of the architecture is the stateful argument which is set to true in each LSTM layer. Setting this to true implies that there is time dependency between samples, which is the case for this dataset considering it is essentially one time series. This will force the propagation of states through batches. Finally, I manually reset states after every epoch to ensure the model doesn't treat each new epoch as an extension of the original time series. It is also

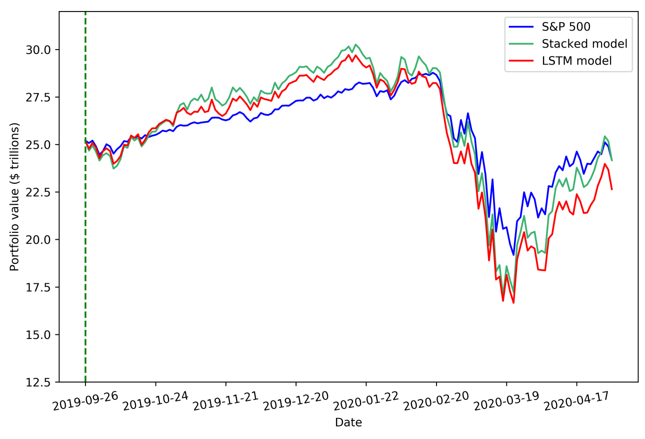

Fig. 8. Predictive LSTM portfolio performance for stacked and LSTM encoded features

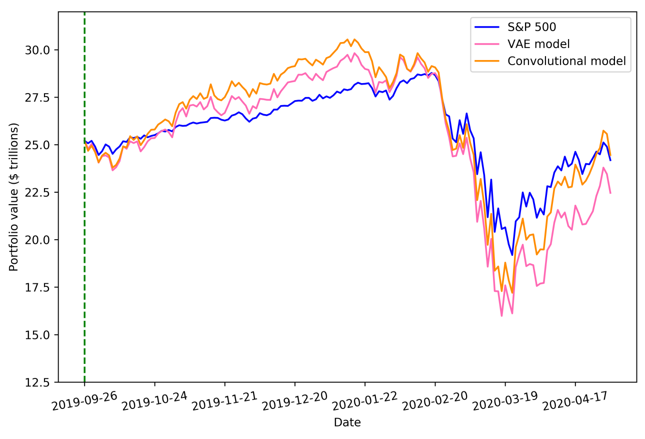

Fig. 9. Predictive LSTM portfolio performance for variational and convolutional encoded features

important not to shuffle data when training so that the time dependency is maintained. I train all networks for 20 epochs. Network outputs have the same format as each input sample with the exception of features. This dimension is expanded to 479 (476 for the convolutional model) via the dense layer to ensure predictions are obtained for each stock.

C. Portfolio performance. Portfolio stocks will be chosen on the basis of the highest average expected growth defined below:

Average Growth = $\frac{1}{60}\Sigma_{i=0}^{60}(\frac{x_i - x_0}{x_0})$

x_0 is the last known share price
x_i is the predicted share price on day i of the test set where $i \in (1, 60)$

I create an equally weighted portfolio using 20 stocks expected to produce the highest average growth during the first 60 days of the test set. This portfolio's performance is then compared to the S&P over the test period. I repeat this process for all encoded features obtained from the architectures discussed in Section 4. The results can be seen in Figures 8 and 9. As was the case in Section 4, the stacked and convolutional predictive portfolios perform best. It is also interesting to note that all encoded varieties construct portfolios with similar trends. For most of the training set, these portfolios outperform the S&P while maintaining a lower risk profile resulting from their highly diversified structure. There is, however, an unusual period where all portfolios lose value but the S&P seems to lose less value than the constructed portfolios. This dip in the market can be explained by the current coronavirus crisis - the dip began around mid-February 2020, a few weeks after the pandemic was declared. The S&P's smaller level of loss over this crisis can be explained by its heavy weighting in technology companies. The top 5 stocks on the S&P are all tech companies which were able to continue operating during periods of lockdown resulting in a quick recovery of their share prices. The diversified constructed portfolios have a smaller weighting in the tech sector and therefore suffered larger losses.
In conclusion to this section, the results show that the models created above provide superior risk-adjusted returns to the S&P index for a large part of the test period and improve on the models used in Section 4. While these models are by no means the secret to risk-free high returns, they provide a solid basis for the argument that LSTM networks with encoded inputs provide promise in the field of portfolio construction.

6 Conclusion

This paper set out to build smart portfolios with risk-adjusted returns that beat the S&P 500 index using autoencoders as feature extraction tools. First, results from Deep Learning in Finance were extrapolated as a justification for the use of autoencoders in the field. Second, the paper compared the effectiveness of a variety of autoencoder architectures in constructing portfolios. It was found that stacked and convolutional autoencoders best replicated the S&P 500, however, returns were slightly lower than the S&P. Aiming to achieve higher returns with the same risk profile, the paper proceeded to implement a predictive LSTM network using encoded input data to identify high growth stocks. With the exception of the coronavirus market shock, the constructed portfolios perform with superior returns to the S&P while maintaining a lower risk profile as a result of diversification.
While the models proposed in this paper do not always result in smart portfolios with superior risk-adjusted returns to the S&P 500, there is sufficient evidence to suggest that the combination of autoencoders and LSTM predictors are valuable tools in the financial market space. Other than further refinement of such models, the major challenge facing future researchers involves accounting for extreme events like coronavirus. Such events are not included in past data and are therefore absent in the encoded feature space. Although rare, these events' impact on financial markets are significant - future research into methods of building previously unseen crises into the encoded space is central to the model's success.

References

[1] Soheila Abrishami et al. “Enhancing Profit by Predicting Stock Prices using Deep Neural Networks”. In: *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE. 2019, pp. 1551–1556.

[2] Dana H Ballard. “Modular Learning in Neural Networks.” In: *AAAI*. 1987, pp. 279–284.

[3] Wei Bao, Jun Yue, and Yulei Rao. “A deep learning framework for financial time series using stacked autoencoders and long-short term memory”. In: *PloS one* 12.7 (2017).

[4] Yoshua Bengio et al. “Greedy layer-wise training of deep networks”. In: *Advances in neural information processing systems*. 2007, pp. 153–160.

[5] Yushi Chen et al. “Deep learning-based classification of hyperspectral data”. In: *IEEE Journal of Selected topics in applied earth observations and remote sensing* 7.6 (2014), pp. 2094–2107.

[6] Alex Frino and David R Gallagher. “Tracking S&P 500 index funds”. In: *The Journal of Portfolio Management* 28.1 (2001), pp. 44–55.

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[8] JB Heaton, Nicholas G Polson, and Jan Hendrik Witte. “Deep learning in finance”. In: *arXiv preprint arXiv:1602.06561* (2016).

[9] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[10] Harry Markowitz. “Portfolio Selection”. In: *The Journal of Finance* 7 (1952). DOI: 10.2307/2975974. URL: https://www.jstor.org/stable/2975974.

[11] Jonathan Masci et al. “Stacked convolutional auto-encoders for hierarchical feature extraction”. In: *International conference on artificial neural networks*. Springer. 2011, pp. 52–59.