Deep Science Publishing
https://doi.org/10.70593/978-93-49910-38-6_10

DeepScience
Open Access Books

Chapter 10: Leveraging deep learning for predictive financial modeling

10.1. Introduction to Predictive Financial Modeling

As a result of technological advancement, financial markets have recently become more chaotic and complex. Financial market prediction has drawn significant interest from many academics and institutions in a variety of fields, including data mining and expert systems. Stock price prediction has become a challenging task due to the market's chaotic nature, with various observed factors contributing to price changes. Numerous traditional statistical techniques and recent data mining techniques are used to forecast the price movement direction or value in various time horizons.

Price prediction has been a classic problem in both academia and industry. Financial time series prediction is a long-standing problem, and extensive research is conducted in various disciplines, including statistics, econometrics, and machine learning arenas. Financial price prediction involves predicting the future price series of a specific stock, index, or foreign exchange. Financial price forecasting is the process of forecasting the future price of stocks, bonds, foreign exchanges, commodities, etc. A high accuracy forecasting model can be employed in portfolio management and trading, leading to huge profits. A pricing model prediction can be provided in derivative markets. On the other hand, highly inaccurate or erroneous predictions may cause huge losses in investment.

Forecasting highly volatile financial indices or foreign exchange rates is a complex nonlinear problem, as many uncontrollable factors can change the price. How to select or combine appropriate modeling techniques for price direction movements must be taken into consideration. Generally, two approaches can be differentiated for price forecasting: technical analysis and fundamental analysis. Technical analysis forecasts the prediction price based on prior price. Fundamental analysis, in contrast, aims to forecast the company value by investigating the economy, industry, and company

relevant indicators. However, it should be noted that no single analysis can deliver satisfactory results in price prediction or decisions.

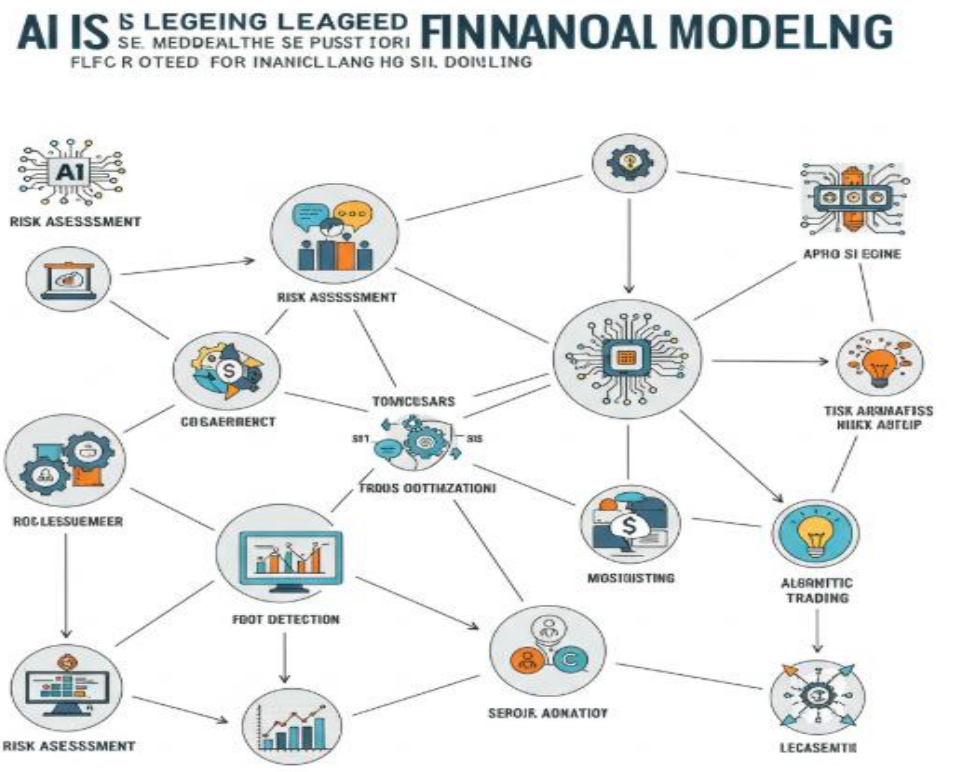

Fig 10.1: Leveraging AI for Financial Modeling

10.1.1. Background and Significance

Deep learning (DL) techniques are a family of machine learning (ML) methods structured as neural networks possessing several dozen or more sequential layers of simple, non-linear signal processors (Kuo, 2011; Brynjolfsson & McAfee, 2017; Davenport & Ronanki, 2018). Despite the tremendous advances achieved over the past years by DL techniques, they have not yet been widely adopted for concrete industrial financial modeling applications. For instance, the latest industry risk prediction models for credit risk, used to estimate default probabilities of commercial counter-parties for creditworthiness assessment and limit setting purposes, still rely on highly hand-tuned stage-wised statistical learning tools, along with a variety of pre- and post-processing optimization techniques. Other applications such as banking and insurance policy modeling, pricing and settlement, lending and trading, risk management, fraud detection,

recommendation, and benchmarking also mostly employ similar statistical learning techniques.

The reasons for the slow adoption of DL models are two-fold: (1) Data: most of the state-of-the-art deep neural network models (DNN) have relied on image and language datasets for which upscale or temporal continuity is well-preserved and no intricate pre-processing or engineering is required. In stark contrast, real-world financial data are high-dimensional, sparse, noisy, and extremely imbalanced which makes deep DNN models particularly challenging to train; (2) Model: Deep risk prediction models are typically composed of complicated architectures with sophisticated strategies for data engineering, initialization, training, hyper-parameter tuning, etc. which rely heavily on toolkits that are still maturing, a steep barriers to entry for practitioners. Despite these challenges, a small number of publications and noteworthy models have recently begun replicating some of the success achieved in other fields.

10.2. Fundamentals of Deep Learning

Deep Learning is a consolidated, state-of-the-art Machine Learning tool to fit a function $y = f(x)$ when provided with large data sets of examples $\{(x_i, y_i)\}$. Its flexibility combined with its generally successful performance makes it an obvious choice for high-dimensional, non-linear, and heteroscedastic numerical prediction tasks, such as forecasting stock prices or risk assessment simulation in complex financial and environmental systems. In regression tasks, the straightforward application of Deep Learning (DL) models is equivalent to fitting a (very) non-linear function $y = f(x)$, for which one observation y is given both the actual values of the covariables x and the model hyperparameters a, so as to obtain a point estimate of the target. In the context of forecasting in challenging financial environments, this basic scheme of a DL model does not take into account the uncertainty or dispersion of the underlying stochastic process of financing, and therefore the uncertainty of a prediction is not communicated. This represents a limitation for tasks where there is a cost associated with communicating an erroneous prediction, or when the requirements of the task demand filtering some predictions to provide better calibrated ones. To remedy this, distribution-based predictive Deep Learning models, which predict not only the most likely outcome of the observations but also its full statistical description, had recently emerged. Though there are state-of-the-art distribution-based Deep Learning models for regression tasks, their application to forecasting consumer-use financial tasks is still an open area. This study tackles a challenging real-world problem of forecasting impending financial expenses and incomes of consumers of a financial company. Within the task, which is benchmarked with a small number of standardized baseline models, a long short-term memory sequence-to-sequence predictive Deep Learning was designed for use in a

production environment, assessing the calibration of its predictions by measuring both the area under the calibration curve metric and a normalization for application-specific costs.

Traditionally, people train the weights of linear transformations between layers while keeping the activation functions fixed. Usually, one identical activation function is used for all the neurons on each single layer. Rectifier linear units (Relu) are used as the default choice for the activation in hidden units, while sigmoid and tanh functions are used where output values are bounded. The drawback of Relu activation is the issue of dead units when the input is negative, which makes people introduce functions with non-zero values in the negative range, including leaky-Relu and Elu. As intensive studies are conducted, more types of activation functions such as GeLu and Mish are proposed. There are also activations designed for particular learning tasks such as reinforcement learning, where the activation functions used for neural network function approximation is computed by the sigmoid function multiplying by its input. However, explosion or vanishing gradients in back propagation are issues that harm the performance of the model due to the shape of activations. Techniques such as clipping and batch normalization can be implemented to alleviate these issues.

10.2.1. Neural Network Architectures

On the other hand it has been shown that a multi-layer feed-forward or fully-connected neural network is a powerful nonlinear function approximator (Schreiber et al., 2008; OECD, 2021). The generalized structure of such a network is a hierarchical set of layers, very similar to that of the human brain. Each layer is composed of a finite number of artificial neurons. Two or more neurons of different layers are connected by the weighted connections plus bias weight, and together they end up the three inputs plus bias weight with hidden layer, where hyperbolic tangent is introduced as a bounded nonlinear function. The new value is propagated through the weights and does the next layer's weighted input ones in a similar way. The feed-forward neural networks can also be composed of one or more hidden layers. In this configuration, the hidden neurons enabled the network to provide a sufficiently rich representation space, thereby theoretically achieving any desired approximation or mapping behaviour. However, it has been shown that a standard multi-layer feed-forward neural network yields a straightforward optimization landscape that can easily get stuck in local minima. An estimation error surface is complex with numerous local minima. The complex inputs rely heavily on their lengthy representations, which can only be used by a fully-connected neural network. The network large dataset consists of 1488288 pieces of complex model and testing data. Using all data to train the model would be very inefficient, therefore training and testing data are selected. The model data for training

and testing are constructed based on different-valued real parameters and noise level respectively. Generally input data are segmented equally or randomly from data points for each input-segmenting method. These two feed-forward networks using dissecting algorithms can be called segmenting input with multi-hidden layers and multi-output nodes, respectively. The multi-hidden-layer net would readily exhibit the behavior of a monolithic standard graphical model, for which the regularization and the thresholding division assume the roles of the original static unregularized training construction.

10.2.2. Activation Functions

In a deep neural network, each layer contains a set of neurons and each neuron takes input as a weighted sum of neuron outputs from the previous layer plus a bias. The weights and bias take account of linear transformation of the data flow, while the activation functions bring in non-linearity. Basically, once a task is defined, an architecture is constructed, and weights are initialized, the training phase comes. The weight parameters are updated via gradient descent by back propagating errors during training. The selection of activation functions for different tasks is an issue with importance. It is acknowledged that activation function selection significantly affects model performance. Strategies on activation selection, mostly involving considering prior knowledge of data prior, are developed to release the effort on the time-consuming process of hyperparameter tuning, while model-agnostic approaches simplify the concern with an automatic solution on the choice of activation functions based on reinforcement learning. Generally, there are two main types of activations: the standard functions and the flexible functions.

10.2.3. Loss Functions and Optimization

For mathematical models describing the dynamics of the observed process, state variables xt are used, and a loss function $L(\theta t; \lambda t)$ defines parameters of the model to be estimated. Still, the application of Deep Neural Networks (DNNs), which are trained to approximate a mathematical function given a set of its parameters, follows a different approach. For DNNs the time series that is described, i.e. $\{yt, t = 0, 1, \ldots\}$, is divided into a set of observations X that is fed into the architecture to train parameters Y, then applying this model for the out-of-sample observations Y. Models, like GLM or Autoregressive Moving Average (ARMA), are highly interpretable mathematical functions. In contrast, black-box DNN provides a set of parameters that cannot be directly connected to the data generating process. Thus, DNNs provide less control over the definition of the observable process they try to approximate.

One of the main differences with DNNs is the missing definition of a state representation. The dimension of the input layer must be described but which particular features of the time series must be used (lagged observations, scaled or divided by volatility) cannot be coded directly into the architecture. Instead, bitmaps of the input features are usually used for directly feeding the observations as t × n matrices, where t is a time window and n is the number of the time series. This allows architectures to recognize harmonic cycles or periodicity like in Convolutional Neural Networks. Nevertheless, it imposes the order and definition of the events included for the training.

Two tasks must be accomplished for the optimization of financial models built using Deep Learning: first, a loss function must be defined, i.e. how to assess the quality of transactions taken basing on the output of the model, and second, a big dimension hyperparameter optimization task related to neural architecture searching must be solved. While the latter mimics the structure used in forecasting time series easily, the former is much more complicated to create.

10.3. Data Preprocessing Techniques

Proper data pre-processing is the key to successful implementations of machine learning models, especially for deep learning models. Poor quality, biased, or incomplete data will hinder the performance of a model even with sophisticated architectures and hyperparameter choices. On the other hand, a good dataset will mitigate the impacts of model architecture selection and hyperparameter tuning and is usually good enough to produce acceptable performance levels on fully vanilla models. As real-world finance data is often huge and noisy, this work spends a lot of time and effort surveying the credit risk prediction practices in financial institutions to obtain clean and useful data. This section introduces each step of data pre-processing along with proposed scripts. The proposed data pre-processing approach has the potential to automate the decision-making process of other data pre-processing steps as well, which can greatly benefit other research topics that apply machine learning models to a wide range of data. The following describes the three main steps of data pre-processing in detail, including data cleaning, incorporation of new datasets, and encoding schemes and techniques applied to improve the effectiveness of deep learning point forecasting methods. The section also discusses the weighting schema used for BB and SKU loss functions and the embedding module architectures. It is noted that embeddings with a history length of 125 were used for all data and settings, including baselines. User and item embeddings with different architectures were carefully tuned and cross-validated. The one with the largest validation AUC is reported for each method. Comprehensive ablation analysis is also provided to show insights into the importance of each component involved in the whole framework. In the first part, the careful data pre-processing procedure is first elaborated

on in detail and then the modelling process, hyperparameter setting, and optimization process are introduced. An important step in preparing a dataset with chronological time-ordering for machine learning and forecasting is to curate it from behalf of the actual use cases and commercial implementations of prediction models. Outside well-established datasets are unlikely to reveal the actual real-world decision-making processes and reflect a business's concerns or objectives. Thus, this work first studied in-depth credit risk detection practices in large financial institutions to prepare a dataset that contains high-quality information to investigate how deep learning and machine learning can better support credit risk prediction in terms of and precision.

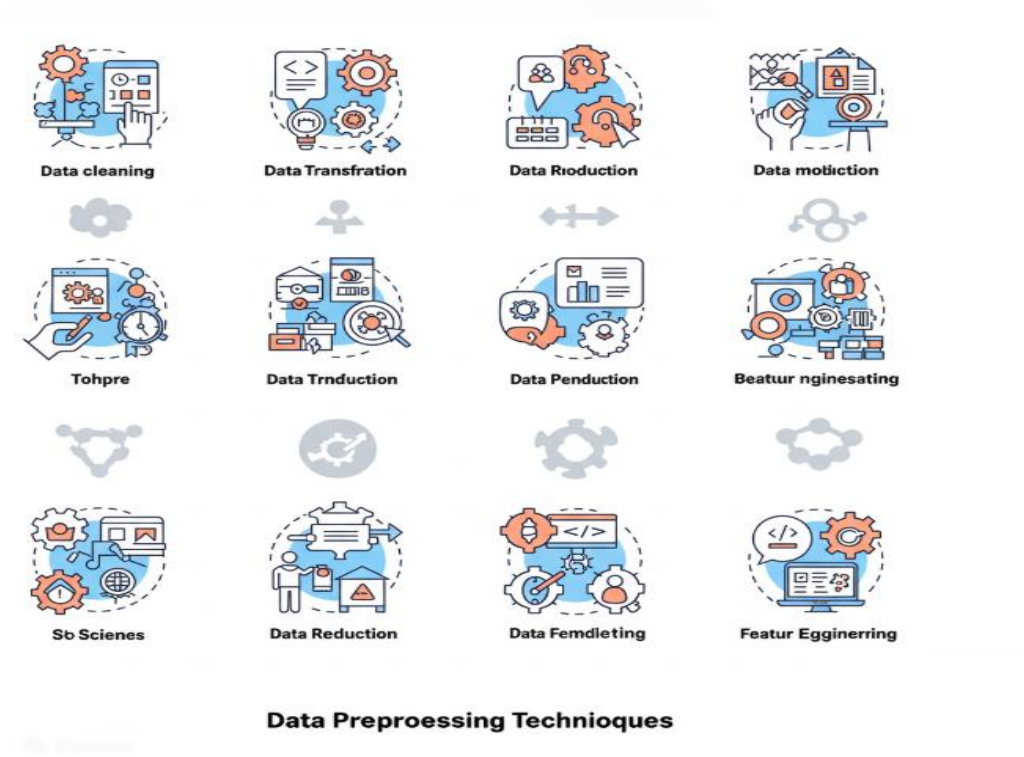

Fig 10.2: Data Preprocessing Techniques

10.3.1. Data Cleaning and Transformation

This study collected lending record columns related to the risk of loan repayment from the authentic loan platform in China, including loan related information, loaner's account basic information, and loaner's account transaction record information. For each record, there were 95 types of features and over 20 types of records available, and more

than 170 million transaction records were processed. In addition, collaborations with data scientists and financial experts from the loan company also helped in understanding the domain knowledge. It is obviously infeasible to model all 170 million transaction records and 95 features due to the high computation cost and high dimensionality during acquiring model weights. Due to the collaborative work with domain knowledge experts, initially, the feature selection algorithms were implemented and the following features were filtered: 44 types of loan features, 18 types of loaner's account basic information features, and 12 types of loaner's account transaction features.

It is shown that for rebalancing the highly imbalanced classes, both down-sampling of majority classes as well as cost-sensitive learning approaches can be useful. A down-sampling method is proposed, where the majority class samples are rebalanced by randomly deleting samples. However, this method can potentially lose information regarding training samples from the majority classes. To prevent the loss of majority class samples, a cost-sensitive SVM is trained with a re-weighted objective function, with weights inversely proportional to the class frequencies. The effectiveness of using cost-sensitive class weighting prior is verified. Finally, with multiple available machine learning models, it is shown that for imbalanced data, performance varies significantly by the underlying classifier, evaluation measures, imbalanced ratios and even between different datasets. It is shown that there is no single best classifier and model training strategy that is appropriate under all situations. Thus, robust methods and learning strategies regardless of models are necessary to improve the stability of detection performance for imbalanced data.

As consumer lending has become a lucrative business for financial technology companies in the past decades, there is an urgent need of developing a machine learning risk prediction system. To do so, a variety of pre-processing and data mining techniques are proposed to address challenges encountered in risk modeling using real-world business data. This study finds that deep learning approaches outperform classical statistical learning methods by a significant margin and at present models are in production, facilitated with better interpretability of the model prediction outputs.

10.3.2. Feature Selection and Engineering

One important preprocessing step for signals containing a large number of features is feature selection. Only selecting a smaller feature subset that preserves predictive power reduces the need for memory and computation, leading to a more understandable, easier to tune, and generally better trained model. Feature selection may be related to Xavier initializing most of the heavy convolution lenses in the DNN feature extractor in trained bias-variance trade off. Feasibly selecting features before they are fed into the DNN helps keep the number of neurons exponentially smaller than that of input features, and

is essential for the analyses of sparse inputs such as stock event representation. The feature selection step, e.g., using a linear model or information gain rank-based selection, can be constructed ahead of training DNNs on an input set.

Selecting signal features increases the interpretability of the solution while training the model to predict on a raw feature input space may not. Processes with lesser memory footprint cost, e.g., limit DNN input parameters, and training diminutive features are easier at avoiding spurious minima. Adding a final class-based pooling operation in this framework can account for how much output nods will receive an interpretation in terms of input features.

10.3.3. Normalization and Scaling

Normalization is required if time series from different sources or with disparate value ranges are fed into the same model as multiple input features. Data that vary greatly in scale can lead to the issue of exploding gradients during training. Thus, large magnitude values overshadow smaller magnitude values, causing the neural network not to learn effectively. Data normalization is a common pre-processing step used to prevent the model from converging into an undesired local minimum in financial time series forecasting. Standardization and scaling flat to a certain bound are common practices when normalizing the training features .

An ad-hoc way to normalize financial time series input features is to transform them to the interval of (−1, 1). Hence, the inputs, returns, volume, and labels are all rescaled by a simple linear transformation to values between −1 and 1. Then, the transformed inputs are fed into a simple feed-forward fully connected one stratum dense network, comprising three hidden layers, with hyperbolic tangent activations. During implementation, approximately 20 percent of past data, including up to one and a half months of price movement, is utilized to predict the last day. The mean square error (MSE) loss function is used to measure the predictive performance. The Mean Absolute Error (MAE) error is used as an evaluation metric while providing the reference directions and probabilities for model selection.

The output probability is then fed into the loss function to compute the binary cross-entropy loss and its gradient. Then it propagates backward through the computational graph of the prediction to update all learnable parameters, including the weight matrices, kernel matrices, and projection matrices learned by the training or fine-tuning MLP models, either end-to-end during training or off-line during modelling. Each training pass for the financial time series of a single minute is regarded as one batch, and the model parameters are updated after all samples in the training set are used once to form

one epoch. The training is terminated when reaching the maximum threshold of 100 epochs or when the updating approaches zero.

10.4. Deep Learning Models in Finance

Deep learning is a promising technique to deal with the heterogeneous, highly variable patterns in finance. It is a standard econometric framework to describe the data-generating processes in macroeconomics, finance, and beyond. In conducting econometric analysis, several ways must be decided upon, including the choice of targets to be modeled, fitting and inference methods, sources of information, and control for endogeneity. The goal of this paper is to improve upon this standard by addressing the following aspects of deep learning in the context of applied econometrics.

The proposed deep learning model can be perfectly described and estimated in this framework. Still, they are perfectly aware of the practical issues that arise in implementing it. They deal with the model class in its theoretical generality. Still, they illustrate the computational feasibility of the deep learning model in a large application dealing with important and complex safety and stock pricing.

The experiments demonstrate predictive ability for both intra-day and daily price movements, and conditions under which this ability depends on background stock information. They emphasize that stock price prediction may be ill-posed, highlighting interesting directions for future research, such as recursive training and model averaging. The usefulness of the empirical exercise resides in the potential extension to a portfolio of stocks. These and other applications of the methodology in forecasting financial time series will take longer than typical econometric applications of similar size and complexity. Still, they highlight a new way of thinking about predictions that may be fruitfully applied in finance.

The analysis of the pluggable deep learning architecture compares predictive ability under various assumptions. In its basic form, it treats stocks as independent and identically distributed. While the model remains general and applicable in this way, it misses the factor structures and correlations that are fundamental to the pricing of stocks as complex systems. As these biases can be dealt with and acknowledged, this choice is comparatively tractable and illustrates typical selection biases that forecast horizon varies with prediction interval length. Under this assumption, the pairs of inputs and targets are constructed using the interarrival time difference. In its simplest form, the model architecture treats the selected prediction pairs independently. To expose the idea most simply, no recurrent viscosity is assumed. However, if delays or long-memory processes are suspected, previous inputs one or more time steps back can also be incorporated.

10.4.1. Recurrent Neural Networks (RNNs)

Recurrent deep learning architectures have received a lot of attention in the past decades in the computational intelligence community. These models have been employed in a wide range of applications spanning from text classification to speech recognition and stock price forecasting. Recent advancements in deep learning, combined with increasing processing power, have made it possible to train RNNs on massive amounts of data. RNNs break the limitations of hidden layer architectures widely employed in feedforward networks. RNNs accept sequences as inputs and, by doing so, preserve the information from previous items in the sequence. This work considers the formulation of Gated Recursive Networks based on recurrent architectures for supervised learning.

RNNs differ from standard feed-forward architectures in which a neural network receives as input a single item and produces as output a single target value. On the contrary, the RNN accepts a sequence observation of items as input and produces a sequence of estimates. To this end, a hoisted feedback operation is applied to the hidden layer, resulting in a presentation of the hidden state that no longer depends exclusively on the input. The application of the feedback traverses the hidden layer itself. Since RNNs consider the whole sequence in predictions a bulk of memory is required, limiting their usability. Thus, it is important to analyze a portion of the output history and feedback of observations to attain a more computationally acceptable architecture.

In financial engineering, forecasting is important in many different applications, portfolio optimization, regulatory compliance, risk exposure assessment, and management procedures. In times of stress, financial predictions become unreliable creating a challenge that must be addressed. Financial time series exhibit both short and long spikes, depending sharply on unexpected events. RNNs seem to be a good candidate for stock price predictions. They are able to capture temporal relationships, and model sequences of inputs, they are also quite resistant to noisy inputs and fluctuation of the data.

10.4.2. Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a type of neural networks that are mostly known as good image classifiers. CNNs consist of convolutional layers, which have convolutional filters, pooling layers, normalization layers, and fully connected layers. To forecast time series using CNNs, one can either take the original time series and feed it as a 1D vector to the model or transform the input data through Image Processing and generate pictures. In this latter method, 2D images are generated so that CNN can treat them as pictures. Recently, CNNs have been used for the purpose of forecasting financial time series, both to forecast fundamental variables such as interest rates or currencies

(i.e. 1D time series), or to forecast price series from stock indices or financial assets (i.e. 2D images). As to the first kind, it is possible to feed a CNN model with numerical data (unprocessed) and retrieve good results. In this sense, CNNs have been used to forecast the volatility of financial time series.

On the second kind of input data, financial information is transformed into images through different mathematical methods, and then fed to the CNN model. The transformation of numerical financial data through Image Processing, more specifically through Transforming Recurrence Graphs into images, is used to forecast its volatility in a stock index. This model is primarily based on two types of Convolutional Neural Networks: the first model consists of a Conv2D layer which is applied to time information converted into pictures (images) through the RG transformation; and the second model is a simple CNN used to forecast time series data (TSDs) unprocessed.

Aside from the differences in data treatment, both models have several common aspects. The first is common to both model specifications, which is the data set used to test these models. The second is that once the image treatment of RG is performed, no more pre-processing is needed; all these images are used as input for both models. A third common aspect is the splitting of the dataset into a training set for training models and a validation or testing set for future predictions. The fourth common point is that the models are implemented in Python using Keras, with a TensorFlow backend, and run in a machine with an Nvidia GTX 1070 with 8GB of memory for a graphical processing unit (GPU).

10.4.3. Generative Adversarial Networks (GANs)

Research of GANs has flourished rapidly, which finds a wide range of applications in computer vision and beyond. GANs simply have two competing neural networks: a generator and a discriminator. The generator draws samples from a space, and the discriminator judges whether a sample comes from the training dataset or is generated by the generator, which correspondingly gives the generator a reward. During competitive training, the distribution of the generated samples eventually converges to the distribution of the training dataset. Both the generator and discriminator are typically neural networks, which are complex nonlinear functions that can approximate their target distributions well. Compared to those non-competitive training methods, the advantage of GANs training is that the algorithms need significantly larger iterations to achieve the same level of accuracy.

The first paper is exploring strategies that are effective against a set of adversarially bounded policies. For the concern of action-space continuous MDPs, a GANs-based algorithm called APAC-Net is proposed, which solves certain classes of MFGs in dimension up to 100. The algorithms trained under such a supervision are robust against

adversarially perturbed policies, which is expected to have applications in safety-critical sequential decision making. The second reformulation of this paper is widely accepted, or formulated as a min-max game, which leads to familiar GANs in various settings. Two frameworks in which GANs have been adopted in the mathematical finance literature are identified. In the first framework, the no-arbitrage condition is exploited to recast a constrained control and optimization problem as a minimax problem to estimate the SDF.

10.5. Time Series Analysis with Deep Learning

To build the end-to-end DL-based Financial Trading System, input data is first transformed to be compatible with the analyzer network. Data from a financial market is collected over a time window. Given this time window, rates are sampled for a set of assets, each containing a 10 minutes resolution time series input. Each input time series is transformed using one of two DL networks, a CNN and an LSTM adapted to energy consumption forecasts, to obtain an estimation of the future market movement regarding a specific asset. Given all above N assets and respective sample predictions, a confidence estimator loads a trained CNN, providing the necessary information to infer trade entry and exit points. A Probabilistic Neural Network is trained to assign each trade a market movement confidence score. Then, given inputs consisting of the movement prediction made on the chosen asset and general market prediction of other assets, the network attempts to classify the market movement into one of three potential scenarios (uptrend, downtrend or no-trend).

Time series data analysis is a critical problem in various fields, including financial, climate, social and biological sciences. Time series data is a sequence of observations indexed or ordered by time. Commonly, the objective of time series analysis is estimating meaningful statistics and patterns of the observed sequential data. In particular, the forecasting task aims to estimate the future values of a time series data based on previously observed associated measurements.

Fig : Deep Learning for Financial Time Series Prediction

10.5.1. Understanding Time Series Data

Failing to account for the time element can render human reasoning and prediction futile in numerous situations where the evolutionary characteristics of reality play an important role. In such cases, countless pieces of historical information will be represented as time series data. Each data point can be uniquely described by two sets of characterizing features: the learning input/output pairs (past/future states), and the covariates (possibly weighted real-valued inputs) that best describe them. Nevertheless, as the real-valued time series grow longer, the dimensionality of the input space grows exponentially, thus rendering it very difficult to analyze the data and work with them. Creating surrogate models, which have a lower dimension than the original cached data space and can predict the output using its own inputs, are still useful alternatives.

Financial time series, which represent the money commodities flowing in and out of various financial markets, on the one hand exhibit a highly volatile, wealth-seeking behavior due to the speculation performed by investors, companies, and other

organizations in the ongoing, global visualized 24/7 trading rooms. Just as one cannot expect the perfect prediction of the arrival times of vehicles boarding and disembarking from a station, accurately predicting future prices in financial markets is impossible due to the multitude of simultaneous influences exerted on price levels, which trail back to very different historical moments in time. Nevertheless, a better understanding of such markets can have a huge impact on forecasting price levels, which is the task aimed for in this paper, where financial time series are understood as graphs. On the other hand, time series can exhibit periodic behavior, such as seasons or tidal waves in weather or underwater observations to monitor drop-offs in oceans. Also, similar evolutions should produce similar results. This is why scientists proliferated models of temporal evolutions, alternative time-domain representations, and equivalent data set treatments. This was also the case with financial time series.

10.5.2. Techniques for Time Series Forecasting

In the financial world, time series forecasting has become an integral part of decision-making since it enables the prediction of future trends using historical time series data. With the development of information technology and a growing number of applications, high-frequency time series forecasting with a large number of records has gained attention. Time series forecasting is an extremely challenging problem since the values are sequentially dependent on one another. In recent years, deep learning methods have achieved state-of-the-art performances in various domains by capturing complex non-linear temporal relationships. However, the capability of these methods with regard to time series forecasting has been overlooked. Traditional statistical models, such as ARIMA or Exponential Smoothing, generally assume a linear relationship in the data and are thus rarely suitable for complex time series data. Recently, machine learning algorithms, such as SVM, are capable of learning non-linear relationships and have been applied to time series forecasting applications.

Yet, it is often hard to model sequential data. Some recurrent neural networks (RNNs)-based architectures have been proposed for time series forecasting, including long short-term memory networks (LSTM), Gated Recurrent Unit networks (GRU), and deep recurrent neural networks (DRNNs). Nonetheless, RNNs based models have problems with difficulty in parallel computations, the vanishing gradient, and overfitting. Furthermore, although convolutional neural networks (CNNs) ignore the sequence order of time series data, many 1D CNNs based models have been developed in the literature for time series forecasting. Extensive empirical studies demonstrate that they can compete with current state-of-the-art methods and achieve superior performance on several public datasets. With the remarkable advance of deep learning techniques, various methods have been applied to time series forecasting, including seasonal

decomposition, fully connected networks, RNNs-based networks, CNNs-based networks, and hybrid models.

In recent years, deep learning applications in time series forecasting have attracted much attention as a plethora of data sets become available and deep learning techniques develop rapidly. Diverse methods have been proposed for time series forecasting, including CNN-based networks, RNN-based networks (GRU or LSTM), and hybrid models. In particular, convolutional neural networks (CNNs) are effective in capturing both local and global patterns in time series. It leverages deep convolution and downsampling operations that lead to fewer parameters as well as a larger receptive field. In addition, some deep learning methods can also model complex temporal dynamics. For computational strategy, RNN-based networks typically suffer from a huge number of parameters and inefficient computations. The CNNs-based networks enable parallel computation in training and adopting fewer parameters compared to RNN-based methods.

10.6. Conclusion

A new set of momentum factor models that enhance the classical Smart Beta strategy by incorporating additional asset classes, modeling frequencies, and loss functions is presented. All models are end-to-end trained and specialized towards addressing a diverse set of risk types at numerous points throughout the architecture, from risk prediction to risk-aware asset pricing. Extensive empirical evidence shows that momentum factor models built within the proposed framework significantly outperform alternatives also employed in practice.

A new approach to prediction markets that sustains liquidity and prevents manipulation is presented. Participants individually manage a fund that places wagers in order to maximize their expected returns. Liquidity providers continuously supervise funds, measuring their performance and contribution to liquidity. If a fund is badly performing, the liquidity provider either withdraws its support, in which case the corresponding market closes, or manipulates the fund’s prices. Both cases destroy liquidity whereas the former is irreversible.

10.6.1. Emerging Trends

With the continuous rapid economic development and improvement of financial globalization, there are more and more securities and financial tools appearing in the market, so speculation has become a common phenomenon. Stock price prediction is to utilize numerous original data, such as company stocks, macroeconomic conditions, and

investment situation, and through deep learning and machine learning models, to reasonably integrate the features into a model to predict future stock trends. At present, researchers mainly focus on studying the internal peacefulness of the stock market. However, this paper makes use of stock-related news and focuses on stock prediction from the perspective of external influences. A blending ensemble model is proposed, and the three sub models are composed of a two-layer deep learning model and a basic SVM classifier. Such a well-trained model ensemble is demonstrated to be capable of stock price trend prediction sufficiently by optimizing a strictly defined ensemble cost function. The new proposed core ensemble algorithm can not only be applied to stock price-related domains but also to various domains, input types, or sub-models. These methods have been considered the hottest trend and have been widely used. This paper uses sampling data from October 2014 to October 2020 to train the model designed using various deep learning algorithms. Based on the well-trained model using all model outputs as the input of SVM, the test data from October 2020 to November 2022 is used to obtain the final predictions. To evaluate the quality of the new model, the model dependent on only price data is also trained, and the comparison results show the evident improvement of prediction performance. Additionally, not only the LSTM, CNN, and SVM are used, but also other machine learning methods are considered so recursively for the comparison of the correlation of the model outputs proposed.

References

Brynjolfsson, E., & McAfee, A. (2017). *Machine, platform, crowd: Harnessing our digital future*. W. W. Norton & Company.

Davenport, T. H., & Ronanki, R. (2018). Artificial intelligence for the real world. *Harvard Business Review*, 96(1), 108–116.

Kuo, A. M.-H. (2011). Opportunities and challenges of cloud computing to improve health care services. *Journal of Medical Internet Research*, 13(3), e67. https://doi.org/10.2196/jmir.1867

OECD. (2021). *Tax Administration 2021: Comparative Information on OECD and Other Advanced and Emerging Economies*. OECD Publishing. https://doi.org/10.1787/0e3c3016-en

Schreiber, G. et al. (2008). *Knowledge Engineering and Management: The CommonKADS Methodology*. MIT Press.