TRANSFORMER ARCHITECTURE IN DEEP LEARNING FOR FINANCIAL PRICE FORECASTING

Anonym

Humboldt-Universität zu Berlin

School of Business and Economics

Submitted in fulfillment of the requirements for the degree of Master of Science in Business Administration.

Prof. Dr. Stefan Lessmann

Chair of Information Systems

Humboldt-Universität zu Berlin

School of Business and Economics

Berlin, 27. Sept 2021

Abstract

Time series prediction plays an important part in many fields of scientific research and professional applications, including the financial sector. In this paper we examine the recently introduced transformer network architecture for natural language processing and apply a time series adapted transformer model to a financial use case. While providing a brief overview of commonly used methods for predictive tasks on time series data, the paper's focus lies on the adaption of the new transformer architecture to this purpose and a subsequent analysis of its performance. For this, we construct an experiment where the transformer is trained on stock market time series and its subsequent prediction output is used to decide on actions in a downstream trading model. For reference, we compare the transformer's performance to an LSTM's performance on the same data and descent into an ablation study on the transformer's adaptations.

Keywords: Transformer, LSTM, Finance, Time Series Prediction

Table of Contents

TRANSFORMER ARCHITECTURE IN DEEP LEARNING FOR FINANCIAL PRICE FORECASTING

1. Introduction

Often, situations require knowledge about which events are likely to occur in order to make informed decisions for obtaining personal, public or organizational goals. This knowledge is generally not yet given and must be deduced inferential from all sorts of available information. Alongside many others, time series predictive methods, which base their working on past and current serialized observations are one predominantly employed group of tools used in such cases.

Actors in the financial sector rely heavily on the best possible forecasts of future developments to assess risks of business transactions, determine insurance-premiums, or price assets and securities for economically viable and profitable strategies. They therefore long belonged to the early adopters of proven and state-of-the-art predictive methodologies. Due to the nature of available data in the form of e.g. historical production, investment and consumption activities, interest rate and risk levels over time, or long-time stock and index exchange data, which all constitute forms of time-serialized data, many tools for financial analytics fall under the umbrella term of time series forecasting.

Time series forecasting is a data-driven forecasting method, that takes for its input data chronologically ordered observations of the same observation object at several, often periodical and equidistanced, points in time – the time series. It generally establishes correlations, trends or other patterns from the past data to infer and extrapolate these patterns into the future. When grouping the relevant machine learning approaches, a distinction into newly emerging deep learning neural network methods and established classical models for time series is useful.

2. Related Time Series Models

2.1. Classical Models

From the list of classical models, the Box-Jenkins models such as ARMA and ARIMA models are most commonly used for time series forecasting tasks as the one in a financial market setting. These autoregressive-moving-average, respectively for seasonal data autoregressive-integrated-moving-average models are well accepted and offer sound theoretical foundations in statistics. Both methods are parametric and thus assume an implicit underlying distribution, which then helps to extrapolate from the given time series data into future steps (Box & Jenkins, 1970). Although determining these parameters is non-trivial and requires numerical approximating approaches to non-linear problems (Brockwell & Davis, 1991), Box-Jenkins algorithms formed the backbone approach to time series analysis and predictions for several decades, such that ARMA- or ARIMA-forecasts are readily available using respective packages in any well-established statistical software. Despite the ease of use, Box-Jenkins models suffer from the drawback, that the central assumption of stationary time series is generally hard to find in real world data and therefore eroding the effectiveness as well as theoretical grounding of the model (Commandeur & Koopman, 2007).

A second classic statistical approach that extrapolates a found distribution in the data to the future is Bayesian inference. Unlike Box-Jenkins, this methodology does not necessitate stationary data. But whereas the former methodologies arrive at a point prediction, e.g. the predicted closing price of a stock, Bayesian inference forecasting yields a posterior probability derived from the prior probability and likelihood function of the observed data using a specialized Bayesian linear regression. Thus, Bayesian methods are mainly applied in predictive classification tasks, e.g. whether the observed stock will outperform the market. Bayesian methods have gained a large popularity boost with the introduction of the Markovchain and Monte Carlo methods in the 1980s that similarly arrive at a posteri sample distributions through determining estimated equilibrium distributions.

Similarly, Bayesian statistics form the basis to the class of state space models, that again describe a probabilistic dependence of the observed data to a latent distribution. Although first introduced as a modelling framework (Kalman, 1960) modern research regularly extends the field. Time series from the financial sector are usually easily available for a long history and can e.g. combine short-interval data like stock quotes, combined with medium to long interval data i.e. quarterly financial statement reporting, as well as underlying and exogeneous variables like country of company HQ or whether a certain date is a trading day. Recent extensions for Bayesian state space models include the integration of such different data-types (Seeger, et al., 2016) or the handling of large areas of missings in intermittend datasets (Seeger, et al., 2017).

Common among all classical econometric models for time series prediction however is, that they assume a specific underlying model for the way the time series develops. They only determine parameters such that those assumed modelling formulas fit best to the actual data. Financial data, specifically stock market data on the other hand is generally noisy and only follows very complex patterns. It is therefore extremely difficult or even impossible to design appropriate dynamic equations for these models. Hence it becomes unlikely that a parametric econometric model can predict such financial time series best.

2.2. Deep Learning Neural Network Models

On the contrary, neural networks in deep learning do not assume a specific underlying equation they try to fit to the data. Their training algorithm of linkage weight adjustment for chains of otherwise very general mathematical operations is a fundamentally different approach to parameter fitting in econometric models. In theory, a neural network can initially be designed totally agnostic of possible patterns in the data and will nonetheless be able to learn (some of) these patterns in its weights.

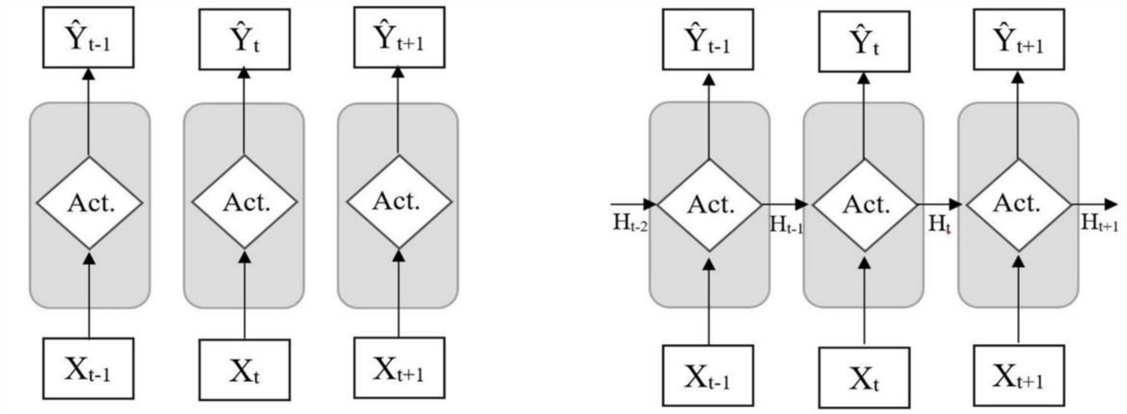


Figure I

Left: schematic view of a feedforward network with a single activation layer. Note that three independent runs for time-steps t-1 to t+1 through the same cell are depicted, not a three-element input.
Right: schematic view of a simple RNN with a single activation layer and linked hidden cell state H. Note that three sequential time-steps through the same cell are depicted. If the single time-steps were to be overlaid, the circular nature of the hidden-state pass becomes evident.

The most basic form of neural networks are simple feedforward networks. A feedforward network transforms an input example in a strictly forward flowing pass through its layers (Figure I). When several examples that are related in the time-dimension, i.e. stock price time series, are fed into such a network, a simple feedforward net will stay agnostic of time. That is, the previous time-steps result will generally not affect later ones. Thus, this simple neural architecture is applicable to time series data in very limited capacity only.

2.2.1. Recurrent Neural Network

In neural network computation time series tasks are often approached through a recurrent neural network (RNN) (Williams, et al., 1986). In an RNN the time series elements at any time-step are successively fed into the same neurons, where they are combined with an internal cell state. This internal state incorporates the neuron's memory of previous time-steps. After activation, the neurons pass on the data to an output layer (Figure I). In comparison to feedforward nets, RNNs have a cyclical structure where the earlier data elements (possibly) loop until the last element is processed. This recurrent information flow equips the network with a simple memory of the past in form of the hidden state in the neurons and hence makes it a compelling tool for time series prediction. Since memory now links the final output to input from all time-steps, the backpropagation links all the time-steps in an accordingly long chain of calculations in order to arrive at gradients for weight updating. RNNs are therefore innately susceptible to training problems from exploding or vanishing gradient descents and early RNN's performance was significantly impeded by backpropagating errors all the way through the different time-steps for long sequences (Hochreiter, 1991).

2.2.2. Long Short-Term Memory

Only with the groundbreaking long short-term memory architecture (LSTM) (Hochreiter & Schmidhuber, 1997) this type of model made a significant improvement in predictive accuracy possible. Subsequent work further refined the LSTM in regard to forgetting by introducing an according forget gate (Gers, et al., 2000), bidirectionality (Graves, et al., 2005), and more recently by proposing wavelet- or attention-supported variants (Yan & Ouyang, 2018-09; Zhang, et al., 2019-07-01). As previous RNNs, the LSTM model fully connects the input sequence to an output sequence through recurrent cells. The key addition in the LSTM over the basic RNN are three input-, output-, and forget-gates and a second long-term memory. While the core cell still memorizes past values in its hidden state, the second dedicated memory component can better store dependencies over arbitrarily longer time intervals. The three gates regulate the information flow in and out of the hidden state and memory. The architecture of a memory cell as the LSTM's core is illustrated in figure II.

The complete LSTM network consists of one or more layers of such memory cells between an input and output layer. The latter are sized in accordance to the specific data requirements as the number of feature variables in the input determine the number of input neurons, while the output neurons correspond to classes in the target space. The core memory cell itself consist in addition to the usual activation component of three small feedforward subnets, that form gates to control the information flow and enables the cell to hold a separate 'long-term' memory state in addition to the normal hidden state of the output. Thus, the model core has three input streams at every time-step: new input X_t, short-term memory H_{t-1}, and

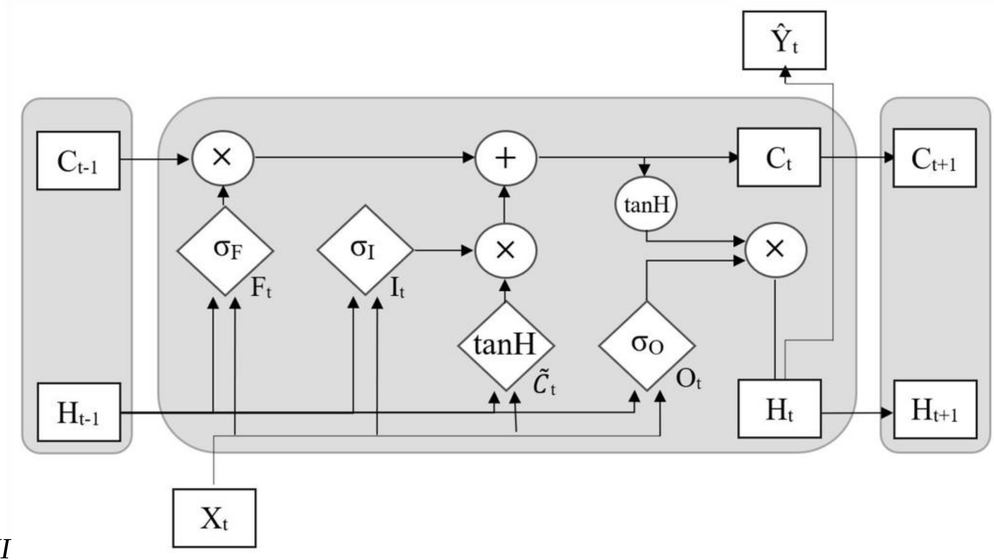


Figure II

Schematic view of the LSTM memory cell's architecture. Depicted is one time-step with input features X_t, σ_F, σ_I nd σ_O are the forget, input and output gates, respectively. The gates themselves consist of a simple, fully connected feedforward network. H_t denotes the hidden state corresponding to that in the basic RNN while C_t is the LSTM's proper long-term memory state. Both H_t and C_t are passed on between recurring steps.

long-term memory C_{t-1}. The forget gate σ_F regulates which short-term memory is removed from the hidden state H, the input gate σ_I controls the flow of new information to the cell's long-term memory state, and gate σ_O decides on the information from new input and previous hidden state that impacts the new hidden state and output.

As LSTMs process sequential data into either another data sequence, a subset of such or single element output, they are naturally suited to be applied to most problems in which sequential data is prevalent. Settings in which mainly but not exclusively sequence-to-point algorithms are utilized can include classification tasks on medical history (Choi, et al., 2016), predicting markets and developments for informed business decisions (Tax, et al., 2017) or one-step ahead stock market prediction as in our comparison paper (Fischer & Krauss, 2017; Islam & Hossain, 2020). A substantial amount of research effort and applications furthermore are in the wide area of natural language processing (NLP) with i.e. speech recognition (Graves, et al., 2005; Graves, et al., 2013), various translation tasks (Wu, et al., 2016), or language correction and completion tasks (Apple Inc. MLR, 2019). The latter pass more often as sequence-to-sequence implementations of the LSTM architecture.

2.2.3. Convolutional Neural Networks

Beside RNNs, the second most common class of networks that was common around 2017 before the later transformer architecture was proposed were convolutional neural networks (CNN) (Abiodun, et al., 2017).

CNNs are regularized multilayer feedforward networks that involve at least one convolutional hidden layer (Figure III). These layers are inspired by the functioning of the visual cortex, as they convolve through the entire input, always only responding to data-stimuli in the currently focused window. This sliding filter or kernel operation concurrently extracts features, i.e. edges in an image processing task or peaks/lows for a time series, and subsequently reduces data-size (Fukushima, 1980).

In contrast to the more general form of multilayer perceptrons, which do not have a convolutional layer and are thus fully connected, CNNs are much leaner in terms of complexity. Whereas the former need regularization techniques like dropouts, skipped connectors or penalization of parameters in the training stage in order to avoid serious overfitting, CNNs have much fewer learnable weights as the different kernel-windows share the same weights. This counters the issues of overfitting as well as problematic exploding or vanishing gradient during training. It also allows the resulting feature map of a convolution to be equivariant under location shifts (Mouton, et al., 2020), which helps recognizing similar patterns at different time in case of time series input.

Since the convolutional kernel is especially well suited to grid-like data such as images, CNNs' predominant application domain are image recognition such as classification (Cireşan, et al., 2012) or face recognition (Alonso-Fernandez, et al., 2021) and in combination with LSTM components also for video analysis (Karpathy, et al., 2014). Time series forecasting on the contrary was usually assumed to be best performed by models within in the domain of RNNs. The last years however saw promising applications of CNN architectures to various

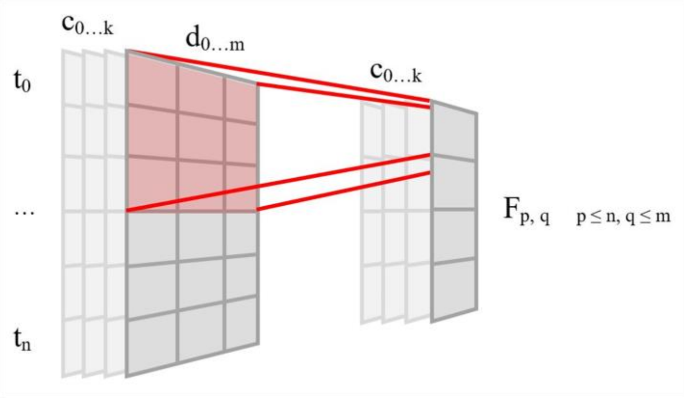


Figure III

Illustration of a sliding 3x3 kernel of a convolutional layer. The input (left) has k channels of a m-dimensional time series with length n and gets mapped to a reduced feature map per input channel. This illustration is without padding or stride.

time series analysis (Zhao, et al., 2017) and prediction tasks, including predictions on financial data (Tsantekidis, et al., 2017). These CNN time series models compare favorably to RNNs in performance (Bai, et al., 2018) as well as applicability. Not only suffer CNNs less from exploding or vanishing gradients but can also be implemented more efficiently than a recursive algorithm (Mittelman, 2015).

3. Transformers

Transformers are a novel network architecture for sequence-to-sequence tasks, first introduced in the paper *Attention is All You Need* (Vaswani, et al., 2017). Originally conceived as a natural language processing model, the transformer was until recently mainly applied to various related NLP tasks. The native application in which the model from the 2017 paper was improving over existing approaches was that of English-German machine translation. Subsequently OpenAI, a research lab for artificial intelligence and Google developed various standard setting NLP models, of which several are publicly available as pre-trained implementations:

GPT (and GPT-2 and GPT-3): less than one year after the original paper on transformer architecture, OpenAI released its first version of a transformer model, that was used for generative pre-training (Radford, et al., 2018) of labeled corpora for language models. The successor GPT-2 was already able to translate and especially generate text on a level, that was in some regards already undistinguishable from that of a human writer (Köbis & Mossink, 2021). The third instantiation again significantly scaled model parameters and training requirements over the first, drastically improving its performance again. This model is currently powering Microsoft's NLP-efforts (Hao, 2020).

BERT: the Bidirectional Encoder Representations from Transformers by Google was published around the same time as OpenAI's GPT and is conceptually similar as it also entails generative pre-training of meaningful vector representations of natural language (Devlin, et al., 2018). Architecturally the model consists of an encoder of parallel and bidirectional self-attention heads similar to those as described in figure IV of the original transformer (Vaswani, et al., 2017).

Shortly afterwards the transformer architecture was transferred to other application domains such as time series forecasts on pandemic data (Wu, et al., 2020), traffic flow (Cai, et al., 2020) or stock volatility and return (Lim, et al., 2020)[1].

3.1. Attention as Transformer Core Mechanism

Unlike LSTMs, transformers are non-recurrent models. Their core functionality feature lies not in a recurring algorithm that keeps selected information in a memory state, but solely in the attention mechanism. This attention mechanism enables the transformer to possibly extract and access data-patterns from every position in an input sequence, regardless of its relative proximity to other parts or the target (Vaswani, et al., 2017; Li, et al., 2019).

The attention mechanism is not a novel feature of transformers but was already a component to extend the former architectures, mainly RNNs, for prediction or other machine learning tasks. There are exemplary applications in natural language problems (Bahdanau, et al., 2015) and time series forecasting, both as extension to the LSTM-model (Cinar, et al., 2017) and to CNN-

---

[1] A summary of key transformer literature is presented in annex A.

based models (Shivam, et al., 2020). Transformers however, were the first popular model class that discarded with the RNN or CNN base altogether and relied on attention only[2].

Conceptually, attention is supposed to give distinctive focus to individual tokens in an input sequence when evaluating their effect on each other and can be understood as a mapping function. This maps a tuple of query-, key-, and value-vectors to an attention output, by summing the elements of the value-vector according to the compatibility of queries with keys. The query, key and value are the embedded and enriched input sequences to the transformer in vector form.

The particular attention function can vary but usually be classified as either additive, respectively Bahdanau attention (Bahdanau, et al., 2015), or as multiplicative dot-product attention. Additive attention scores $a_1, \dots, a_n \in \mathbb{R}$ can be computed from queries $q_1, \dots, q_n \in \mathbb{R}$ and keys $k_1, \dots, k_n \in \mathbb{R}$ as

$$a_i^{additive} = W_3^T \tanh\left(W_1 k_i + W_2 q\right) \in \mathbb{R} \tag{1}$$

with W_1, W_2 and W_3 being weight matrices. Dot-product attention is calculated as

$$a_i^{dot-product} = q^T k_i \in \mathbb{R}. \tag{2}$$

In terms of theoretical complexity both functions are similar, but since in practical programming of neural networks, the latter dot-product attention can be implemented utilizing optimized code for matrix multiplication while additive implementations are through feedforward nets, dot-product attention is computationally faster (Lihala, 2019; Vaswani, et al., 2017).

When scaling up the dimensionality of the key-input however, additive attention outperforms simple dot-product attention in terms of accuracy (Britz, et al., 2017).

---

[2] Around the time of the first transformer publication, there were other pure attention-model architectures, but which did not achieve the same level of popularity in subsequent years (Parikh, et al., 2016).

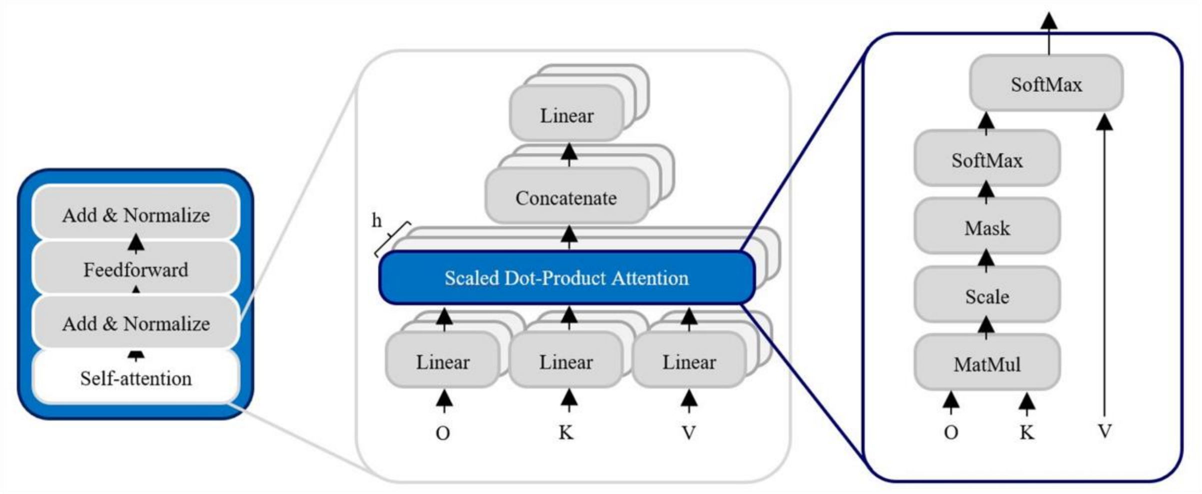


Figure IV

Attention head (right), as part of the multi-head attention (middle) layer in an encoder layer. The mask-component is necessary for the attention module in some positions of the transformer to prevent target-leakage.

The transformer attention is therefore typically a scaled version of the standard dot-product attention, to reduce this performance gap:

$$Attention^{scaled\ dot-product} = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{3}$$

Scaling the dot-products by factor $\frac{1}{\sqrt{d_k}}$ helps avoiding them to grow too large and therefore resulting in extremely low gradients eventually.

3.1.1. Multi-Head Attention

Typically, the attention mechanism is not implemented as a single block, receiving queries, keys, and values of dimension d_{model} as input, but rather in h parallel heads, receiving their own set of input vectors and performing individual scaled dot-product attention. This enables the model to focus on several different aspects of the data simultaneously.

In multi-head attention (MHA), the vectors Q, K and V of dimension d_{model} are thus linearly projected h times to a set of queries, keys and values of dimensions d_q, d_k, d_v as input to the individual attention heads. By assigning separate learnable weights to those $3 * h$ projections, the heads in MHA will each learn different representations of the data and thus be able to attend to several subspaces in the sequence, weighting them differently. If only a single attention head is used, resulting averaging over all noteworthy patterns in the data prevents it capturing any of those effectively.

To combine the individual attention heads, their scaled dot-product attention outputs of dimension d_v are simply concatenated, followed by another linear layer to project them back to an MHA output of d_{model} as depicted in figure IV:

$$MHA(Q, K, V) = concatenate(H_1, \dots, H_h)W^O$$
$$with\ H_i = Attention(QW_i^Q, KW_i^K, VW_i^V). \tag{4}$$

The learnable projection weights are parameter matrices of corresponding dimensionality for the input set $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$, and for the output $W_i^O \in \mathbb{R}^{hd_v \times d_{model}}$.

In addition to making the model more effective in learning dependencies between sequence tokens, MHA also aides more effective computation. When setting $(d_q =)\, d_k = d_v = d_{model}/h$, resulting computation complexity will be close to that of attention on complete dimensionality d_{model} with a single head only (Vaswani, et al., 2017).

3.2. Basic Transformer Architecture

On the highest level the transformer has an encoder-decoder architecture. Such a bipartite structure is already common in other neural network types, e.g. RNNs (Cho, et al., 2014) that are designed to transform an input sequence to a second sequence as their output. The encoder's subcomponents are generally supposed to decompose and combine the sequence into hidden features in a way to find its underlying latent patterns, while the decoder is to generate meaningful output of the desired form from these hidden features.

The transformer as proposed in the original paper (Vaswani, et al., 2017) follows this principle by having $N = 6$ identical layers in the encoder and decoder respectively. Transformer layers consist of point-wise self-attention followed by fully connected feedforward sublayers. A

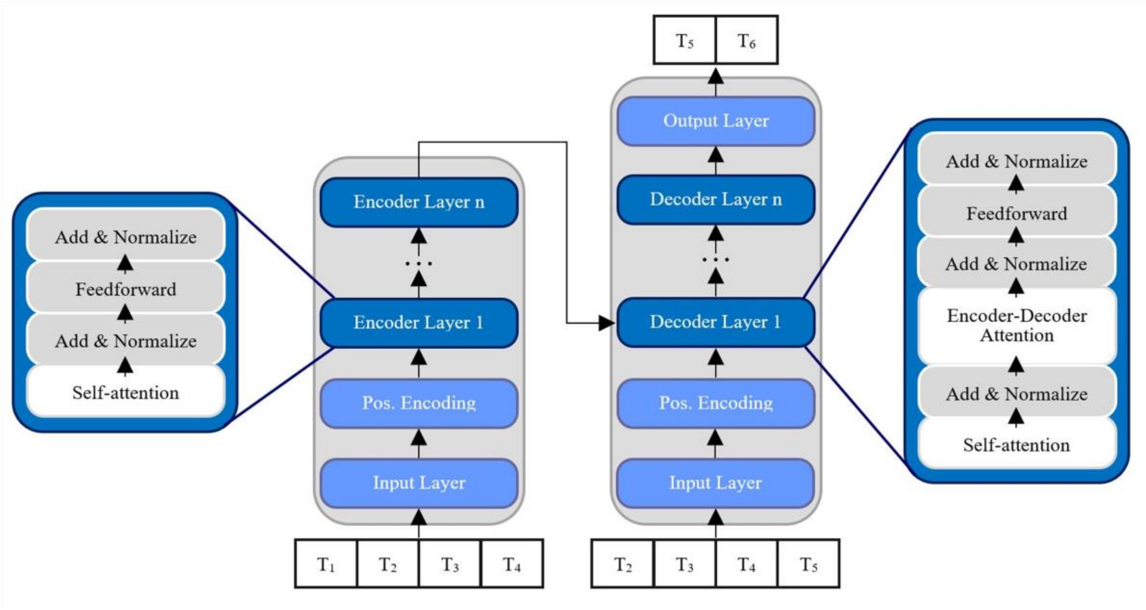


Figure V

Schematic view of a transformer for time series forecasting, including the sublayers of encoder and decoder layers. The left stack of layers for the transformer's encoder with up to n encoder layers and receives an input sequence T_1-T_4. The decoder on the right receives additional activation input T_2-T_5 along the hidden feature mapping from the encoder and maps these to output sequence T_5-T_6.

detailed view of a typical transformer model following the original architecture is depicted in figure V.

3.2.1. Encoder

The encoder is composed of two initial layers for input and positional encoding, followed by N identical but individual encoder layers. The input layer maps the input sequence to encoded input features which the positional encoding layer enriches for positional information. Input features as well as later sublayers' output features for our transformer all have dimension of $d_{model} = 200$ to be able to detect enough relevant patterns in the data, while staying computational feasible. The subsequent encoder layers each consist of the core attention layer and afterwards a fully-connected feedforward sublayer. Both are followed by additional layers for normalization and optionally dropout, if not part of the other layers. The original paper proposes layer normalization (Vaswani, et al., 2017) which is also incorporated in our transformer with learnable scale factor and offset. In addition, feedforward and attention sublayers are both bypassed by residual connections that merge with the normalization output.

3.2.2. Decoder

As the encoder, the decoder begins with an input and positional encoding layer. The decoder's input sequence will generally be right shifted compared to encoder input. This input is set to activate the patterns mapped to the feature space in the encoder. Following these layers the decoder's structure differs from the encoder in that its decoder layers consider of two attention and one feedforward sublayer each. At the end, the decoder has an additional output layer to transform the output sequence into the correct target format. The decoder layers are made up of a self-attention sublayer for the decoder input, followed by an encoder-decoder attention and fully connected feedforward component. All three again have downstream normalization layers and residual connections. The encoder-decoder-attention receives the encoder's feature mapping as values and keys input, while the queries input comes from the previous decoder self-attention.

3.2.3. Feedforward Layers

On top of the attention sub-layers, both encoder and decoder layers stack a fully connected position-wise feedforward layer utilizing a ReLU-activation. Formally it can be described as

$$Feedforward(x) = \max(0, xW_1 + b_1)\, W_2 + b_2 \tag{5}$$

The input and output dimensionality of this layer is d_{model}, while its inner dimensionality is typically considerably larger at $d_{feedforward} = 2048$. This allows the model to find interactions between attention outputs on different model dimensions but within the same sequence position. Positions of the same encoder or decoder layer all share the same layer-weights. Between layers, separate weight-parameters allow for separate learning of the model.

3.2.4. Embedding Layer

Conceptually, the dataset on which the transformer model can be applied can have any form of sequential data: from language sentences, over class sequences to numeric time series data. The pre-transformer embedding of the input data into tokens is therefore case-specific. In the embedding layer of the transformer this pre-embedding gets mapped to data sequences of dimensionality d_{model}. The embedding conversions in the encoder and decoder are via linear transformation using learnable weights as parameters.

3.2.5. Position Encoding Layer

Contrasting to models based on the LSTM or CNN architecture, which inherently capture positional information about the individual tokens of the input sequence, the attention principle of the transformer model rather yields an N-to-N connection of all sequence positions, without having an understanding of time or position. In order to equip the transformer with the ability to also consider the relative position of tokens in the input sequence as well as absolute position of this sequence in an external reference frame, the encoded input sequence is enriched by positional features. This enrichment takes place, both in the encoder and decoder layers.

The position encoding (PE) layer creates a positional encoding of the same length and dimension as the embedded input and subsequently sums them elementwise. The positional encoding is therefore injected into the tokens themselves and not added as a new feature variable.

The default position encoding strategy for transformers is based on a sinus-transformation of relative token positions in the sequence. It indicates the sequentiality and order within the given input sequences. It cannot differentiate between separate instances but only gives a measure of distance between two tokens in the same set. The positional encoding proposed for the original transformer architecture (Vaswani, et al., 2017) constitutes such a relative PE:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10{,}000^{2i/d_{model}}}\right)$$
$$PE_{(pos,2i+1)} = cos\left(\frac{pos}{10{,}000^{2i/d_{model}}}\right). \tag{6}$$

Here, pos refers to the relative position of the element in the sequence, with i being the dimension. For data with dimension $d_{model} > 1$ this leads to PEs based on sine- and cosine-sinusoids alternatingly with increasing wavelength. This systematic differentiation on the time-focus lays the basis for the separate attention heads to later pick up distinctive patterns in the data.

4. Experiment and Methodology

We construct a transformer model for time series prediction and evaluate its predictive capacity for this application based on returns of hypothetical trades derived from model predictions. Inspiration is drawn from a comparable study with the LSTM as model of interest (Fischer & Krauss, 2017). In a second step we further analyze transformer performance and compare results to that of a LSTM applied to the same task for reference. Finally, we compare different variations of the transformer in an ablation study.

4.1. Data

The dataset used for the empirical experiments consists of daily closing prices of major US stocks and is based on the S&P 500 for the years from 1990 to 2020. Current S&P 500 constituent lists and publicly available addition and removal notes for the relevant time were used to construct an experiment dataset closely resembling the S&P 500. This index was chosen since as one of the most popular base-index for hypothetical applications of research models it possibly allows for comparison of later findings. The full list of included stocks can be found in annex B. In total the experiment dataset includes daily data from 530 stocks for 8088 days. This corresponds to the time 01. Jan 1990 until 31. Dec 2021, excluding weekends as general non-trading days.

4.2. Experimental Soft- and Hardware Environment

All predictive and analytical models constructed for the experiments can be obtained from our repository[3] in python code. The transformer model is based on related work and, together with the remaining setup, makes use of popular python-software packages and APIs. For a full list refer to annex G.

All models were trained using GPU supported virtual environments provided by Google's Colab-Service. This gives no direct control over the actual utilized resources, but the assignable pool of GPUs includes Nvidia K80s, T4s, P4s and P100s (Google Ireland Ltd., 2021).

4.3. Experimental Setup of Preprocessing and Predicting

In the experiment we seek to analyze the predictive performance of the transformer model and compare it to that of a reference model based on the LSTM architecture over a timespan of three decades. The performance evaluation makes hypothetical trades on the models' predictions.

The experimental setup thus consists of three building blocks. The first handles all the data preprocessing and generates the input data for the predictive models. The predictive models at the core of the experiment receive these inputs and form a prediction of which stocks will perform best for each day in the respective data window. Finally, these predictions are used to calculate a theoretical gain, when following a standardized trading strategy.

[3] Python code available from: https://github.com/Humboldt-WI/dissertations/tree/main/TS_Transformer_Finance. Compatibility is tested for python version 3.8. The repository and annex H include further details of hyperparameters used in the different models.

4.3.1. Data Preparation

The raw data consist of daily closing prices from 01. Jan 1990 to 31. Dec 2020 (8088 days) for 530 stocks. To compare model performance on various input types we compute additional feature variables as input. With p_t being the daily closing prices, these are the simple daily return r_t, the average daily return $\overline{r}_t$ over the last $n = 15$ trading-days and a binary classification of a stock's one-day return being above or below median for the same day. All data is cleaned and treated for missings, non-binary variables are normalized using min-max scaling, and return variables are truncated for extreme outliers.

After preprocessing, the data is sliced into sequences of the specific window-length len_{wind} for the experiment-run, such that we have $8088 - len_{wind}$ sequence sets of all stocks. The actual experiment is then run in several iterations each time on a subset of these slices. The training set includes 240 sequences per stock and the testing set the subsequent 40 sequences. After training, prediction and trading-analysis of model predictions is complete, the cycle is repeated with both, training and testing set shifted by the number of testing sequences. Together with appropriate attention masking, this ensures, that no future data on the prices to be predicted is accessible during model training.

4.3.2. Prediction

For the predictive part of the experiment, we employ a transformer model. This transformer is mainly based on the standard transformer architecture described above with some targeted adjustments. As a benchmark we also conduct the experiment with a reference model based on the LSTM architecture in a separate run.

4.4. Adjustments to Basic Transformer Model Architecture

The above-described model architecture can be regarded as the standard blueprint for transformers as it follows very closely the initially proposed transformer model for natural language as well as it shares most properties with derived models and applications (Vaswani, et al., 2017; Wu, et al., 2016). Subsequent work however showed potential paths of modifying individual aspects of the model to be more suited to time series data.

Financial time series differ in two crucial aspects from sentences of natural languages, for whose translation the transformer was conceived. First, time series, e.g. of stock price history can be much longer sequences than typical sentences. This can lead to serious computation and storage problems given the attention mechanism of transformers. And secondly, sentences have a natural internal structure of which types of tokens (words) usually occur in a sentence, i.e. subject – verb – object – prepositions and usually definite rules as how individual tokens relate to each other, i.e. grammatically correct endings. Stock prices on the other hand are usually accepted to follow a random walk and are only predictable to a very limited amount for individual tokens – at least for humans. Time series data in general however oftentimes reflect a certain cyclicality, i.e. daily or weekly, that is not all that prevalent in natural language. Extended research on time series transformers mainly addresses these two problems (Li, et al., 2019; Cai, et al., 2020; Zhou, et al., 2021).

4.4.1. Sparse Attention

When the input sequence to a LSTM or other recurrent model is too large, this can result in exploding gradients on the one hand, but otherwise only increase computation effort in roughly linear manner. An attention-based transformer however has to compute a $M_{l\,x\,l}$ matrix with l being the input sequence length (Eq. 2). Consequently, large input sequences let the computation and crucially the memory requirements increase quadratically with $O(n \cdot\ l^2)$ and n being the number of layers in the encoder- and decoder stacks. Given our other model meta parameters and experiment hardware this lets the model's memory usage exceed the hardware capabilities in some instances already with roughly $60 \leq l \leq 240$, depending on exact model dimension, number of stacked layers and the assigned virtual machine. Regardless of the exact tipping point in terms of memory usage, this illustrates the general problem. One way to address this issue is by forced down-scaling of other model dimensions or significantly increasing hardware capabilities. While the first will generally deteriorate a model's performance, the latter in essence is, what made the best performing language models in their domain so successful (Brown, et al., 2020). For widespread application without access to vast computing resources is the here proposed change to the attention mechanism more promising.

Annex F shows different forms of attention algorithms. The standard is full self-attention, where all tokens are combined with all tokens in the sequence. Sparse self-attention only calculates the combination of each token with a specified subset of the same sequence tokens, thereby significantly reducing memory usage (Li, et al., 2019; Zhou, et al., 2021). In our experiment model, sparsity is implemented by an algorithm, that retains the $\frac{1}{3}$ rightmost connections but only a subset of the more distant connections with the gaps between connected tokens becoming larger the further in the past the token is. The series of step sizes s_i chosen can be defined as $s_i = \left\lceil \frac{i}{3} \right\rceil + 1$, where i starts at $\frac{2}{3}$ of the attention input sequence, moving backwards and $\lceil \dots \rceil$ denotes the ceiling operation.

Although this results in a possible loss of information, when only viewed at a single attention layer, since output tokens did not attend to all input tokens, stacking multiple attention layers in the encoder or decoder mitigates this. As the attention for the one third rightmost tokens is full, three stacked layers already allow for a theoretical flow of information from any input to any output token.

4.4.2. Convolutional Attention

Convolution mechanisms are not only used in CNNs as model core, but also in conjunction with other algorithms (Bello, et al., 2019) or as additional layer to other architectures.

Li et al. (2019) proposed an adjusted convolutional algorithm mechanism in lieu of simple self-attention. Instead of computing attention scores on Q, V and K-vectors generated as described before, the attention input is channeled through a convolutional layer to obtain these input vectors for self-attention. The dot-product attention mechanism itself remains in place.

The underlying notion is to prevent the mechanism to wrongfully match datapoints of the same value but instead focus it on matching similar 'shapes' in the data. For example, a value y preceded and followed by values $x, z < \hat{y}$ in the sequence – a maximum, should carry a different information content than $x < \tilde{y} < z$ or $\breve{y} < x, z$.

To help our transformer model differentiate between different cases, we added a convolution layer of kernel size 3 to the attention algorithm.

4.4.3. Richer Positional Encoding

The earlier described position encoding strategy encodes the relative position of input tokens. This approach shows the origin of transformers as natural language processing networks. In a sentence, relative word order is important for correct grammar as well as information conveyed. The position of the sentence in a longer section is of secondary importance. For financial time series however there exists no natural 'sentence'. The window of the sequences fed to the model is therefore more arbitrarily chosen. And to interpret e.g. two subsequent one-week series of price-trends, the order of those two weeks is incomparably more important.

Thus, PE for time series data should reflect these differences. Various types of position encoding (PE) suitable for time series data capture different aspects of position or time, such as relative, or global position, periodical time-structures, i.e. days of the week, or possibly categorical, i.e. trading day vs. bank holiday (Cai, et al., 2020). The basis for the first three types is similar.

- Relative position encoding is corresponding to relative PE from natural language and implemented the same (Eq. 6).

- Global position encoding makes it possible for the model to compare two sequences, e.g. encoder source and decoder activation source in regard to position. The formal position assignment is similar to relative position encoding (Eq. 6), with the only difference being that pos refers to the position relative to an external reference point, not necessarily in the currently observed sequence. Global position encoding therefore captures both, global as well as relative continuity in the time series.
- Periodical position encoding can be used to make a model aware of periodicity of a time series. Possible structures in financial time series data might be daily-periodic (0:05, 0:10,…), weekly-periodic (Mon, Tue,…) or yearly-periodic (Jan, Feb,…). Here, pos refers to the index of the series element according to the periodicity timeframe, e.g. $pos \in [0, 6]$ for days of week.

Special time-related features, such as the categorial information, whether the given day was actually a trading day, could theoretically also be encoded in a similar way. As trading of stocks might be a feature individual to those stocks, this information was not encoded in the PE-layer. The above strategies for comparison all refer to the same single timeline for all stocks. Specifically, the information about the ability to trade a certain stock was therefore factored in outside of the transformer model: once the listing of the stock in the sequence selection state for the core transformer model and once in the trading meta-model.

4.4.4. Multivariate Input

The base model computes predictions on a univariate input sequence. For model variations this can be extended to multivariate input sequences.

4.5. LSTM as Benchmark Model

The motivation for using a LSTM-model for benchmarking our transformer's performance is two-fold. For once, the LSTM is one of the most prevalent architectures for time series prediction as described earlier. In addition, it was the object of a similar study as this one (Fischer & Krauss, 2017). Fischer & Krauss discuss the suitability of the LSTM for financial time series prediction, and more importantly, compare the LSTM-performance to a host of predictive models (random forest, dense-neural-network, logistic-regression). By using the same general LSTM topology and similar dataset, we hope to make this experiment easier to align with related research. This experiment's LSTM was consequentially built to the specifications[4] provided by Fischer & Krauss:

- The input layer receives univariate sequences of 80 timesteps. In a model variation we changed this to a multivariate input.
- Following is only one hidden LSTM layer with 25 neurons and an applied dropout of 0.1.
- The final output layer uses softmax activation to classify the samples in two categories – below or above median performance expected.

4.6. Trading

The trading model conducts theoretical trades on the prediction given by the transformer or LSTM model according to a standardized trading strategy. The strategy is for comparative reasons aligned to that of Fischer & Krauss and requires the model to conduct an even number of trades $2 \times k$, with selected $k \in [1, 100]$ for each day predicted. On half of these trades the model must adopt a long position in the chosen stocks while going short on the remaining positions. Opting for a long-short portfolio makes the results less dependent on overall market trends. Assuming a random choice for long and short positions, the expected trading return will be close to zero for bull-markets, as well as bear- and sideways market trends. If only trading in long positions, a random guess would lead to high, vs. negative and zero-

[4] For more detailed description of the reference model please refer to the original paper (Fischer & Krauss, 2017). For better comparability to our transformer models the input length was reduced from 240 to 80 as computational resources did not cope well with sizes above that for the two model types.

returns for these scenarios on average, thus rendering it inappropriate to evaluate the predictive models' performance.

4.7. Experiment Results

As we aim to evaluate the transformer model and its components for the task of financial price prediction as well as compare its performance to that of the LSTM-model our experiment includes data runs using different model configurations. The full set is listed in annex C. Configurations that are altered in the different runs include the sequence length for the lookback window, the type of positional encoding and attention mode in the transformer as well as the variable type of the input sequences.

In accordance with the experiment's objective, we present the results in a tripartite structure. First, we present the performance analysis of the transformer model with specifications No. 1 (annex C) in regard to a random and naïve long portfolio of the underlying portfolio. To enable assessment of the transformer's capability compared to other model types, we will then contrast it to the performance of the LSTM as reference model. Subsequently a brief ablation study comparing different settings for the transformer model to gain an insight into their respective importance concludes the analysis.

4.7.1. Transformer Performance

The base transformer for this part of the performance analysis (No. 1 in annex C) is characterized by full convolutional self-attention with kernel size 3, global position encoding and 80-day input sequences of closing prices. Over the 30-year trading period, with portfolio size $k = 1$ it achieves a mean daily trading return of 0.257% and an average accuracy of 51.80% on the binary above-below-median classification of daily stock performance.

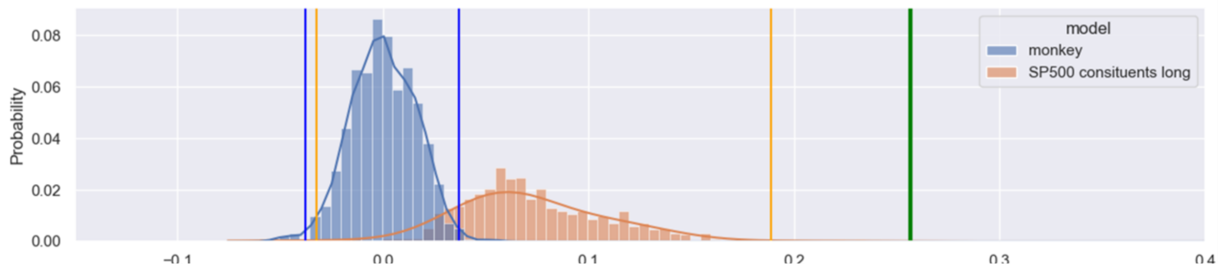


Figure VI

Distribution of average daily returns, 1- and 99-percentile portfolio daily returns for 10,000 empirically sampled random long-short portfolios (blue) and single-stock long positions of every SP500 constituent (orange). The average daily return for the transformer configuration No. 1 is at 0.257% (green).

This daily return from theoretical trades amounts to a 20.1-fold increase to the amount invested over the whole period if gains are distributed after every day[5] (or intermediate losses are remargined) prior to transaction costs. The transformer's average daily return compares favourably to the overall market and outperforms it by a factor of 4.8 as the increase in all stocks of the underlying input entirety amounts to only 0.053% per day on average. A randomly guessed 'monkey'-portfolio of size $k = 1$ achieves a theoretical return of 0% (Malkiel, 2007). As it takes a long or short position in any stock at any given day with the same probability. The transformer also outperforms the 99[th] percentile out of 10,000 sampled monkey portfolios at 0.037% per day by a factor of 6.9 (Figure VI). Distributed cumulative profits from the two reference portfolios amount to 4.31-fold and 2.91-fold increases of the amount invested after 30 years. Employing a standard t-test to test the hypothesis that against these findings the true mean return of the transformer was still zero yields a t-statistic of 5.91 with the critical value

[5] In investment profits are often reinvested and especially the trading structure of this experiment would easily allow for this. Cumulative reinvestment gains are nonetheless not reported for a) by the exponential nature of compound interest, time invested will play a much greater role compared to distributed gains. And b) the disparately higher profit delta results from small insecurities of daily return rates, i.e. during the approx. 7800 days in the trading period a daily $r_1 = 0.1\%$ increases an invested dollar to $\$1 * 1.0010^{7800} = \$2{,}431$, while $r_2 = 0.09\%$ result in less than half that: $\$1 * 1.0009^{7800} = \$1{,}115$.

at 5% significance level being 1.96. The p-value for the transformer's mean return being the result of chance is 0 to eight digits.

Another measure to evaluate the model's performance is its accuracy in correctly classifying the sample stock into those outperforming the median stock and those performing below the median stock for every given day. The transformer's achieved accuracy of 51.80% is equally statistically significant against a hypothesized true value of 50% with the test statistic and p-value being at 15.05 and practically zero for 5% significance. When breaking down the accuracy metric it appears, that the model was best able to avoid false positives (predicting top, when the true class is flop) as table 1 shows. Correct predictions and false negatives all are much closer to their theoretical mean of 0.25. This asymmetry could be a reason for the huge profitability difference in the long and short positions the model takes. If only caring for the returns of the long positions in the combined long-short portfolio, the model would achieve an astonishing mean 0.51% daily return. Returns from short positions however are almost zero at less than 0.01% (Table 2). The difference can partially but by no means completely be explained by the circumstance, that on average, the stocks in the experiment population grew by 0.053% in value per day. A random long trade would therefore be assumedly more profitable than a random short trade by only 0.106%-points per day on average.

Accuracy: 0.5180	True top	True flop
Predicted top	0.2521	0.2078
Predicted flop	0.2670	0.2659

Table 1

Accuracy matrix for the transformer setting 1.

Also, the model setup allows for analysis of varying k from 1 to 100 stocks per trade position. Unsurprisingly both, return as well as risk metrics are highest with $k = 1$ and decreasing for larger k (Figure VII). For smaller k the transformer is able to choose stocks that yield above market average returns. However, this ability appears to be limited to picking only

	Transformer 1	LSTM 9	SP500 long 12
PERFORMANCE MEASURES			
Mean Accuracy:	0.51797	0.50544	-
Mean Return:	0.00257	0.00366	0.00053
Mean Return (long):	0.00511	0.00671	-
Mean Return (short):	0.00020	0.00061	-
Standard error:	0.00043	0.0005	0.00011
t-Statistic:	5.9132	7.35544	4.82093
Minimum:	-0.32668	-0.5081	-0.12126
Quartile 1:	-0.00782	-0.0118	-0.00296
Median:	0.00024	0.0013	0.00053
Quartile 3:	0.00935	0.0178	0.00446
Maximum:	0.70885	0.4994	0.11096
Share: ret > 0:	0.51773	0.54767	0.56041
Share: ret < 0:	0.48227	0.45233	0.43959
Standard dev.:	0.03845	0.04338	0.00994
RISK CHARACTERISTICS			
VaR 1%:	0.08689	0.09726	0.02259
CVaR 1%:	0.09992	0.11197	0.02596
ANNUALIZED RISK-RETURN CHARACTERISTICS			
Return p.a. (distr):	0.66989	0.95614	0.13909
Standard dev. p.a.:	0.64097	0.92206	0.1606
Sharpe ratio p.a.:	0.02014	0.02844	0.68988

Table 2

Selected performance and risk or return statistics for various models. Except for the S&P 500 full market long portfolio, values always refer to a trading size of $k = 1$. Further metrics and model variations are presented in annex E.

a low double-digit number of ‘good’ stocks[6] per day with reasonable accuracy. When extending the trade volume, returns near those of a simple market long position. As the highest trade volume included is 200 stocks in total, that leaves around 300 stocks (varying on the exact trading day) with the worst return expectation to be dropped. The near equality of transformer-returns and market-returns for $k = 100$ hints at a general lack of recognizing non-extreme return potentials effectively. However, it is to note, that between $k = 1$ and $k = 100$ the risk of the more diversified portfolio drops to just 10% of its former level compared to 23% for the return. A risk-return metric driven choice based on Sharpe ratio thus might still favor larger trading volumes.

4.7.2. Contrasting Transformer and LSTM Performance

The previously discussed figure VII also contrasts the transformer against the results from an LSTM model as our reference to other neural networks. Including other types of

[6] Those stocks with return furthest from zero, as a long-short portfolio theoretically benefits from both, extreme positive and negative cases.

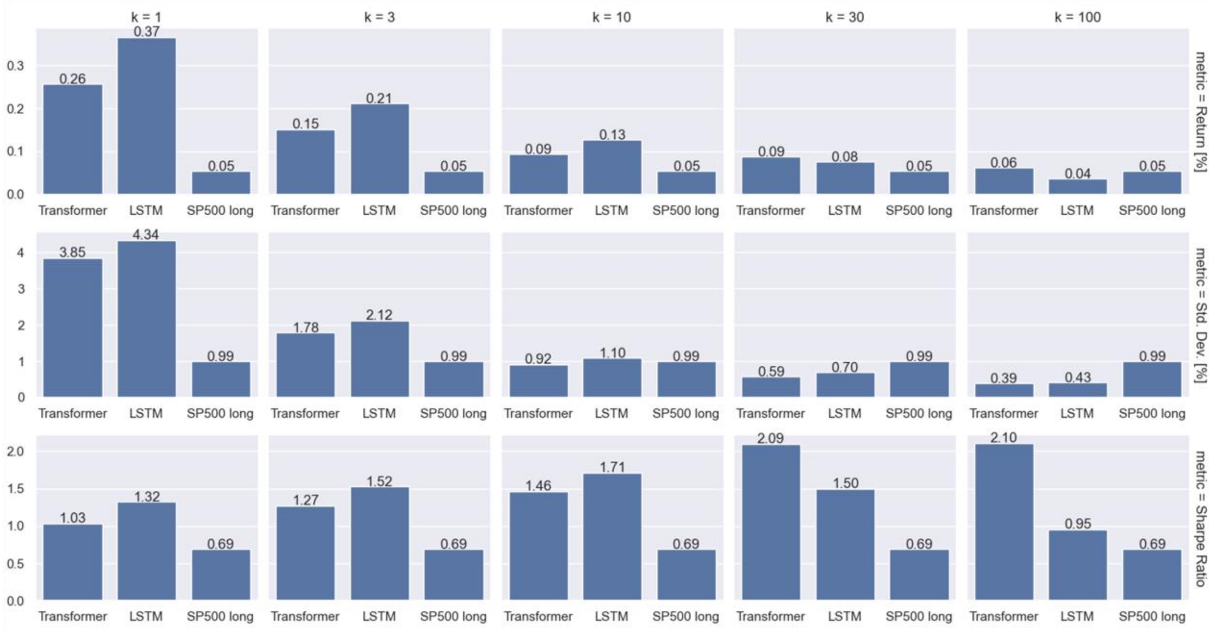


Figure VII

Daily average return, standard deviation and annualized Sharpe ratio for selected trading volumes k per long and short position. The transformer model refers to settings No. 1, the LSTM to settings No. 9. SP500 long No. 12 is included for reference and refers to individual long positions throughout the complete timespan for the complete set of stock in der recreated S&P 500 base portfolio.

alternative predictive architectures for time series data, such as the briefly introduced CNN or classical statistical methods that were left out of scope of the experiment. However, it is hoped, that by choosing such a well-established and extensively analyzed model as the LSTM, a general appreciation up to a certain degree of the transformer against other models can also be made by chain of inference, i.e. with Fischer & Krauss (2017).

Before contrasting the performance differences between transformer and LSTM-models it is important to note, that the outmost layer of our transformer and LSTM experiments are the same. In both cases, raw input consisted of daily stock prices and was converted into trading choices for our analytic purposes. The intermediate layer between predictive model and this outer shell however was eventually adopted differently. For the transformer in its base setting, the core layer receives normalized closing prizes to first predict price predictions, which then are translated into trading choices and a binary classification. The LSTM core on the other hand receives normalized daily returns and directly predicts a binary classification and corresponding confidence. We acknowledge, that this affects comparability. However, as it was observed during model building that both models performed better on this chosen instead of their respective counterparts' specification, it was deemed appropriate in regard to the ultimate goal of comparing financial price predictive performance for trading purposes. Alternative setups are partially presented in the next chapter and annex (see annex C, E).

Comparing average daily returns (Figure VII) gives a clear advantage to the LSTM. At up to 0.37%, for small $k \leq 10$ the LSTM returns are around $\frac{1}{3}$ higher than their counterparts. As we increase k the matter becomes closer, up to the point where the LSTM's return-performance drops below the transformer and even the market long portfolio. Including risk metrics paints a similar picture. Although the LSTM's standard deviation is higher at 4.34% vs. 3.85% for the smallest trade volume, this is compensated for by higher returns such that the Sharpe analysis still favors the LSTM. The comparatively larger decline in profitability for the LSTM however gives the edge to our transformer model for larger k as risk falls unisono for both models. This indicates that the lack to effectively rank the large middle ground of only mediocrely positive/negative return potential is a problem not only the transformer but to an even greater extend also the LSTM suffers from. Related to that, the LSTM's binary accuracy is at 50.5% considerably smaller than the transformer's 51.8% (Table 2). Therefore, it could be reasoned, that the transformer is inferior in identifying the very best candidates for long/short positions, but in correlation with its higher classification accuracy can better avoid picking the wrong side of a trade (shorting a positive return stock and vice-versa). Evidence in form of simple proportion of days with overall positive return vs. those with overall negative return[7]

[7] The positive rate differs from the binary accuracy for the following reason. Binary accuracy is defined as the correct classification of a stock as offering above or below median return for any given day. The median return can be (and in fact is) positive for the majority of trading days due to the general market trend direction.

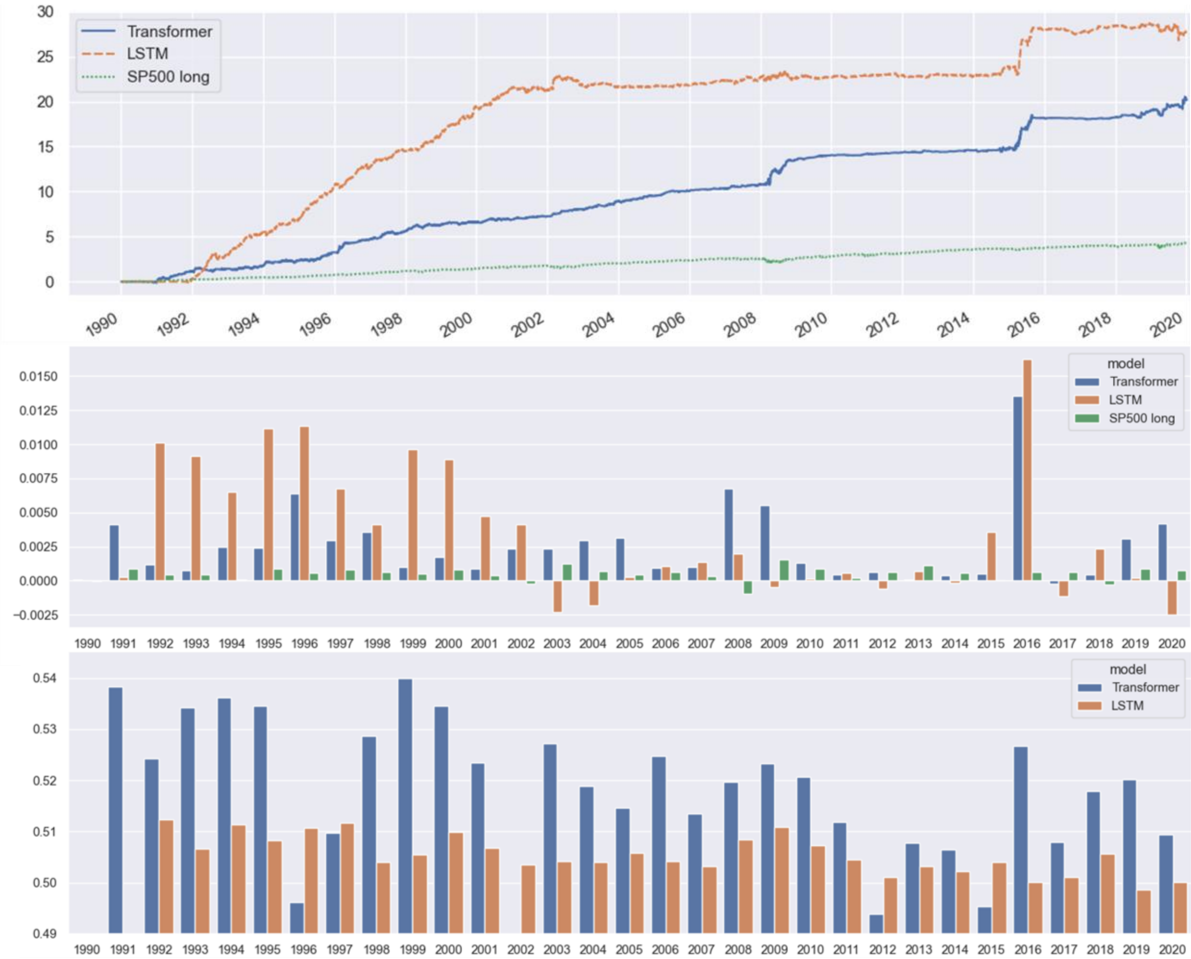


Figure VIII

Development of return and accuracy characteristics over time. The upper panel shows the development of cumulative profits in USD for a daily investment of 1 USD and profit distribution / loss compensation for the $k = 1$ trading portfolios of transformer (No. 1), LSTM (No. 9) and the full market long portfolio (No. 12). The middle graph compares average daily returns for the same portfolios per year (Axis: 0.01 is 1% per day). The third graph shows the average binary accuracy per year for the two models. All graphs start exhibiting data from year 1991 since the first year in the dataset was used for training.

also supports that claim. For $k = 1$ a positive rate of 0.518 (transformer) vs. 0.548 (LSTM) corresponds to the overall LSTM's advantage. With $k = 100$, this reverses to positive rates of 0.584 and 0.551, also supporting the trends seen elsewhere.

Comparing the development of return and accuracy metrics over time shows additional differences in the two models' performances (Figure VIII). Average daily returns for the $k = 1$ portfolios and resulting from that, cumulative profits follow, different time-trends. The LSTM exhibits two distinct phases. In the first decade until 2002 it was clearly outperforming its counterpart. With one exemption yearly average returns per day varied between 0.41% to 1.13% while the transformer's data only reached 0.09% to 0.63%. The total profit accumulated at 2002/12/31 was consequentially almost three times the amount for the LSTM as for the transformer at 22.61 USD vs. 7.83 USD per 1 USD invested. Contrarily, for the remaining years, LSTM-profits practically level out to zero except for a single outlier-year in the experiment's timeframe[8]. During this period, instead of levelling out, the transformer's trading portfolio generally outperforms the LSTM by an although clearly varying, but overall sizeable margin, allowing it to narrow the profitability gap to 27.8 USD vs. 20.15 USD per 1 USD invested after 30 years. This constitutes a relative gain as well as an absolute gain on the LSTM. The only periods, which up to today are a zero-sum game for the transformer are 2013-15 and 2017-19.

The general trend towards lower profitability is also reflected in decreasing accuracy levels for both models. As already seen, the transformer has a higher overall level. And while the LSTM's $k = 1$ portfolio's accuracy seemingly already oscillates around its natural asymptote of 0.5 for the last five years, the transformer's accuracy exhibits a clear trend in the same direction.

As for a possible reason that caused the LSTM's profitability to decline, Fischer & Krauss (2017) hypothesized the feasibility of actually employing advanced neural network methods such as the LSTM for trading started in the early 2000s and let the theoretical advantage the algorithm previously presented disappear. The additional market insights from the LSTM's prediction were already reflected in the price as other market participants also disposed over it. Although, it remains to be supported by additional studies not in scope of this research, it appears possible, that the transformer could provide insights into other aspects than

---

[8] First, this could indicate a possible fault in the experimental code of data. However, this finding is in fact congruent with Fischer & Krauss (2017, p. 18f). Additionally, the last years, not included in that work exhibit common pattern across LSTM and transformer, which experiments did not share the same software code for most parts.

the LSTM, even if it was overall inferior. And that this information edge was also gradually removed by the advancement of data sciences or the introduction of the transformer architecture itself in recent years.

In stark contrast to the general claim of eroding returns stands the performance in the years 2016 (LSTM) and 2008/09, 2016, 2020 (transformer). The 2016's spike for both models is puzzling as global US stock market and global economic data do not offer any ready explanation for the renewed and sudden return plus. Also, inconspicuous accuracy levels indicate the reason to be more fine grained and hidden in the data, but evidently picked up on by both models. In addition to that it seems, the transformer is able to translate phases of relative turmoil on the stock market into trading advantages since both 2008/09 (financial crisis) and 2020 (COVID-19) were the last times the markets significantly contracted. We would consider further research to validate this hypothesis on other data necessary, as the dot-com burst in the early 2000's does not appear to have influenced profits as significantly.

4.7.3. Ablation Study on Model Setup

The results regarding the transformer presented so far pertain to a model setting, where the transformer employs a full self-attention with convolution of kernel size 3 on global PE enriched closing prices. To gain an insight into the effect these various alterations to the original transformer for language modelling (Vaswani, et al., 2017) bring, we performed the same experiment on a multitude of model settings and describe the most relevant in this section.[9]

The first aspect that we added to our transformer on top of the original language transformer is the positional encoding. Other settings equal, we compared the transformers performance with relative (No. 2), weekly-periodic (No. 3) and no PE at all (No. 4) to the

---

[9] For reliability considerations please refer to annex D.

previously analyzed base setting (Figure IX). A comprehensive tabular comparison can be found in annex E. The already presented global-PE model achieved the best trading results, while a model without any PE appears visibly inferior as average daily returns dropped by a fifth to 0.213%. Relative and weekly periodic PE both fall in between at an average daily return of ~0.225%. Considered that no PE obviously carries no additional information content and the relative information from simple relative PE is also included in the global PE, this ordering is not surprising. Weekly periodic encoding appears to also provide some insights to the model, but it remains unclear, whether that hints at strong day-of-week related patterns in return data, or to limited general time succession and distance information this PE provides.

Secondly, we looked into the convolution of the attention module. Results from this part of the study are less clear as this component does not seem to have had a considerably large effect on return performance. The average daily return is at 0.263% and thus very close to the base setting, when the convolution is removed (No. 5). Li et al. (2019) also noted this on some datasets when proposing this convolutional self-attention. However, in their case, that was since the relevant dataset was less challenging and the standard self-attention already performed very well. In the case of this experiment, both statements are improper as stock markets are generally hard to predict and with the LSTM there exists a model, that is more successful so far. At the moment, these experimental findings can therefore not conclusively be explained.

As the memory demands of the standard full self-attention mechanism can grow too large quickly, we also tested our implementation of a sparse self-attention (No. 6). Here, instead of a sequence of length 80, the mechanism only received subsets of the elements from these sequences with length 40. As input elements for this sparse-attention model are a real subset of

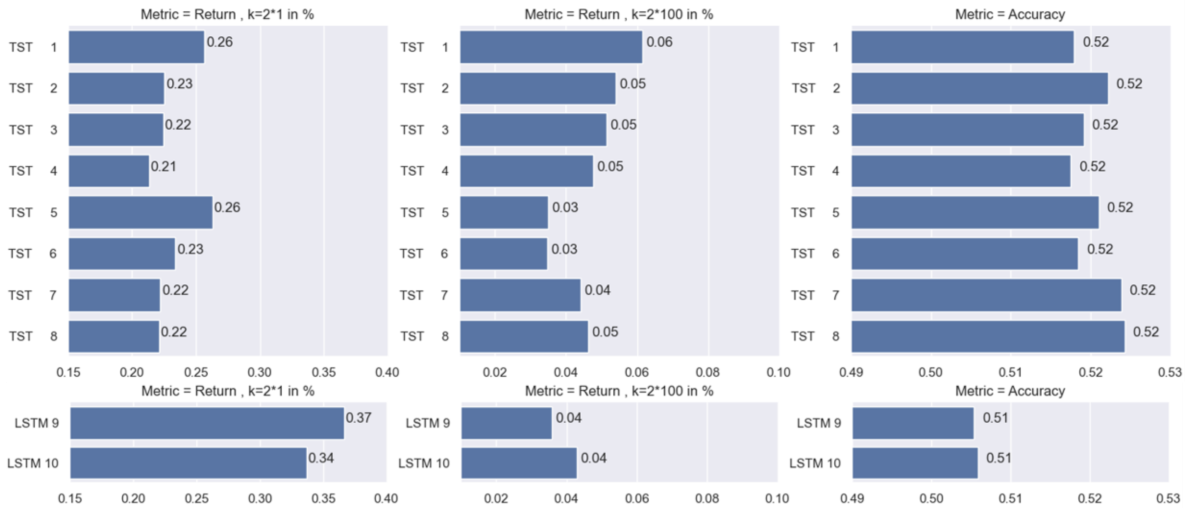


Figure IX

Daily average returns for the $k = 1$ and $k = 100$ portfolios as well as binary accuracies for various settings of our models. The model numbers correspond to settings from annex C.

the inputs to the full attention model (No. 1)[10], we expect its performance to be lower. At only 0.234% average daily return for the $k = 1$ portfolio this is true. However, the relevant aspect is, whether sparse attention can outperform full attention of the same reduced sequence length. For this we compare them to transformer setting No. 7 and can indeed observe a slight advantage for the sparse-attention model over a model with reduced but full-attention. This indicates, that even though sparse self-attention cannot sustain the same performance level as full self-attention, it is a feasible alternative to simply scaling back in feature sequence length.

Finally, we extended our two model types to multivariate input data of closing prices, day-on-day returns and rolling averages for the last 15 trading day returns. Surprisingly, the additional inputs do not convert into strictly better performance. Average daily returns for the smallest portfolio are down 2.4 and 2.8 percentage points for the transformer as well as LSTM. The $k = 100$ portfolio for the transformer is equally performing worse in the multivariate case, with the LSTM by contrast being slightly better than with univariate inputs. Accuracy ratings

---

[10] For a visualization of full versus sparse self-attention, please refer to annex F.

are practically unchanged with minimal improvements of 0.0005. These indecisive results could suggest, that both models are already able to learn relevant patterns from the input sequences and the relatively trivial translation of prices into returns and return averages in preprocessing for the multivariate inputs provide little additional helpful information to the models.

5. Conclusion

In this paper we presented the transformer architecture along a short discussion of a framework of alternative classical statistic and neural network models applicable to time series problems. We adjusted the basic transformer from the NLP background to the specific time series application with three components. First, instead of plain relative encoding, we compared global and periodic encoding strategies and found the global position encoding to be most beneficial. Secondly, canonical attention was generalized to convolutional attention and lastly, the self-attention mechanism at the core of the transformer model was modified to a sparse self-attention to address memory and computational constraints. The effects of convolutional attention were found to be minor, whereas sparse self-attention is inferior to full self-attention but nevertheless an appropriate substitute for memory constraint induced shortening of feature sequences.

To evaluate the architecture's capability in the field of financial forecasting, we implemented an experimental setup where the model's predictions served as input for a simplified trading model and contrasted the performance of various transformer versions to an LSTM model. Although clearly outperforming the market, our transformer models were not able to surpass the LSTM in core return metrics. In terms of accuracy and risk-return metrics on the other hand, the transformer architecture was found to be competitive to the established LSTM-models.

References

Abiodun, O. I. et al., 2017. Comprehensive Review of Artificial Neural Network Applications to Pattern Recognition. *IEEE Access (pre publication),* p. 10.1109/ACCESS.2019.2945545.

Alonso-Fernandez, F. et al., 2021. Facial masks and soft-biometrics: Leveraging face recognition CNNs for age and gender prediction on mobile ocular images. *IET biometrics.*

Apple Inc. MLR, 2019. *Language Identification from Very Short Strings.* [Online] Available at: https://machinelearning.apple.com/research/language-identification-from-very-short-strings
[Accessed 30.07.2021].

Bahdanau, D., Cho, K. & Bengio, Y., 2015. *Neural machine translation by jointly learning to align and translate.* s.l., Proceedings of ICLR.

Bai, S., Kolter, J. Z. & Koltun, V., 2018. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv:1803.01271.*

Bello, I. et al., 2019. Attention Augmented Convolutional Networks. *arXiv:1904.09925.*

Box, G. & Jenkins, G., 1970. Time Series Analysis: Forecasting and Control. In: San Francisco: Holden-Day.

Britz, D., Goldie, A., Luong, M.-T. & Le, Q. V., 2017. Massive exploration of neural machine translation architectures. *arxiv.org,* Issue arXiv:1703.03906.

Brockwell, P. J. & Davis, R. A., 1991. Time Series: Theory and Methods. In: s.l.:Springer-Verlag., p. p. 273.

Brown, T. B. et al., 2020. Language Models are Few-Shot Learners. *arXiv:2005.14165v4.*

Cai, L., Janowicz, K., Mai, G. & Yan, B., 2020. Traffig transformer: Captureing the continuity and periodicity of time series for traffic forecasting. *Transactions in GIS,* Volume 24, pp. 76-755.

Choi, E. et al., 2016. *Doctor AI: Predicting Clinical Events via Recurrent Neural Networks.* s.l., Proceedings of the 1st Machine Learning for Healthcare Conference, in PMLR 56:301-318.

Cho, K. et al., 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *CoRR,* Volume abs/1406.1078.

Cinar, Y. G. et al., 2017. *Position-based content attention for time series forecasting with sequence-to-sequence rnns.* Cham, International conference on neural information processing, pp. 533-544.

Cireşan, D., Meier, U. & Schmidhuber, J., 2012. *Multi-column Deep Neural Networks for Image Classification.* New York, NY, IEEE Conference on Computer Vision and Pattern Recognition.

Commandeur, J. J. F. & Koopman, S. J., 2007. Introduction to State Space Time Series Analysis. *Oxford University Press.*

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K., 2018. BERT: Pre-training of Deep Bidirectional Transformers for. *arXiv:1810.04805v2.*

Fischer, T. & Krauss, C., 2017. Deep learning with long short-term memory networks for financial market predictions. *FAU Discussion Papers in Economics,* Volume 11.

Fukushima, K., 1980. Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological Cybernetics,* 36(4), pp. 193-202.

Gers, F. A., Schmidhuber, J. & Cummins, F., 2000. Learning to Forget: Continual Prediction with LSTM. *Neural Computation,* Volume 12, pp. 2451-2471.

Google Ireland Ltd., 2021. *Colaboratory - Frequently Asked Questions.* [Online] Available at: https://research.google.com/colaboratory/faq.html [Accessed 09.08.2021].

Graves, A., Feandez, S. & Schmidhuber, J., 2005. *Bidirectional LSTM networks for improved phoneme classification and recognition - Lecture Notes.* New York, Springer.

Graves, A., Mohamed, A.-R. & Hinton, G., 2013. *Speech Recognition with Deep Recurrent Neural Networks.* IEEE International Conference on: 6645–6649. arXiv:1303.5778, Acoustics, Speech and Signal Processing (ICASSP).

Hao, K., 2020. *OpenAI is giving Microsoft exclusive access to its GPT-3 language model,* s.l.: MIT Technology Review.

Hochreiter, S., 1991. *Untersuchungen zu dynamischen neuronalen Netzen,* Technische Universität München: Diploma Thesis, Institut für Informati, Lehrstuhl Prof. Brauer.

Hochreiter, S. & Schmidhuber, J., 1997. Long Short-Term Memory. *Neural Computation,* Volume 9, pp. 1735 - 1780.

Islam, M. S. & Hossain, E., 2020. Foreign Exchange Currency Rate Prediction using a GRU-LSTM Hybrid Network. *Soft computing Letters.*

Kalman, R., 1960. A new approach to linear filtering and prediction problems. *Transactions of the ASME-Journal of Basic Engineering,* Volume 82, pp. 35-45.

Karpathy, A. et al., 2014. *Large-scale video classification with convolutional neural networks.* s.l., IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Köbis, N. & Mossink, L. D., 2021. Artificial intelligence versus Maya Angelou: Experimental evidence that people cannot differentiate AI-generated from human-written poetry. *Computers in Human Behaviour,* Volume 114.

Lihala, A., 2019. *Attention and its Different Forms, towardsdatascience.com.* [Online] Available at: https://towardsdatascience.com/attention-and-its-different-forms-7fc3674d14dc
[Accessed 06.08.2021].

Lim, B., Arik, S. Ö., Loeff, N. & Pfister, T., 2020. Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting. *arXiv:1912.09363v3.*

Li, S. et al., 2019. *Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting.* Vancouver, BC, Canada, Proceedings of the 32nd Conference on Neural Information Processing Systems.

Malkiel, B. G., 2007. *A random alk don Wall Street: the time-tested strategy for successful investing.* s.l.:WW Norton & Company.

Mittelman, R., 2015. Time-series modeling with undecimated fully convolutional neural networks. *arXiv:1508.00317.*

Mouton, C., Myburgh, J. C. & Davel, M. H., 2020. Stride and Translation Invariance in CNNs. *Artificial Intelligence Research,* Volume SACAIR 2021. Communications in Computer and Information Science, vol 1342. Springer, Cham. https://doi.org/10.1007/978-3-030-66151-9_17, pp. 267-281.

Parikh, A., Täkström, O., Das, D. & Uszkoreit, J., 2016. A decomposable attention model. *Empirical Methods in Natural Language Processing.*

Radford, A., Narasimhan, K., Salimans, T. & Sutskever, I., 2018. *Open AI: Improving Language Understanding by Generative Pre-Training.* [Online] Available at: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf
[Accessed 30.07.2021].

Seeger, M. et al., 2017. Approximate Bayesian Inference in Linear State. *arXiv:1709.07638v1.*

Seeger, M. W., Salinas, D. & Flunkert, V., 2016. Bayesian Intermittent Demand Forecasting for Large Inventories. *Advances in Neural Information Processing Systems (NIPS),* Volume 29.

Shivam, K., Tzou, J.-C. & Wu, S.-C., 2020. Multi-Step Short-Term Wind Speed Prediction Using a Residual Dilated Causal Convolutional Network with Nonlinear Attention. *Energies,* 13(7).

Tax, N., Verenich, I., La Rosa, M. & Dumas, M., 2017. Predictive Business Process Monitoring with LSTM Neural Networks. *Lecture Notes in Computer Science,* Volume 10253, pp. 477-492.

Tsantekidis, A. et al., 2017. *Forecasting Stock Prices from the Limit Order Book Using Convolutional Neural Networks.* Thessaloniki, GR, IEEE 19th Conference on Business Informatics (CBI).

Vaswani, A. et al., 2017. *Attention Is All You Need.* Long Beach. CA. USA, arXiv:1706.03762v5.

Williams, R. J., Hinton, G. E. & Rumelhart, D. E., 1986. Learning representations by back-propagating errors. *Nature,* Volume 323, pp. 533-536.

Wu, N., Green, B., Ben, X. & O'Banion, S., 2020. Deep Transformer Models for Time Series Forecasting: the Influenza Prevalence Case. *arXiv:2001.08317v1.*

Wu, Y. et al., 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv:1609.08144.*

Yan, H. & Ouyang, H., 2018-09. Financial Time Series Prediction Based on Deep Learning. *Wireless personal communications,* 102(2), pp. 683-700.

Zhang, X. et al., 2019-07-01. AT-LSTM: An Attention-based LSTM Model for Financial Time Series Prediction. *IOP conference series. Materials Science and Engineering,* 569(5), p. 52037.

Zhao, B. et al., 2017. Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics,* 28(1), pp. 162-169.

Zhou, H. et al., 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. *arXiv:2012.07436v3.*

Annex

A Tabular review of related literature on current research on the transformer architecture

Summary of the core aspects of key literature pertaining to possible adaptions of the transformer architecture of components used in our model implementation.

Author, Published	**Topic / Focus**	**Model type**	**Novelty / Proposition**	**Key findings**	**Application area**
Britz et al., 2017	Extensive hyperparameter research for NLP translation models. Among others explores effect of multiplicative, additive or no attention layer between to RNN encoder/decoder stacks.	RNN with attention component	Among others, compares additive to dot-product attention mechanisms in terms of performance and computation cost.	Additive attention generally outperforms multiplicative attention by small margins. No attention is massively inferior giving rise to the question, whether purely attention based models should be explored.	NLP translation
Vaswani et al., 2017	Proposed the novel transformer architecture.	Transformer	Novel pure attention-based model with bipartite encoder-decoder structure.	Transformer model outperforms existing architectures on the NLP translation task. Furthermore, it generalizes well to other areas of application.	NLP translation
Bello et al. 2019	Introducing a two-dimensional self-attention mechanism to use in conjunction with CNNs.	CNN with attention component	CNN component is key for local performance, attention for long distance relations.	Attention augmentation to CNNs significantly improves their object detection and class prediction abilities. Results raise the question to explore possible fully attention-based models to replace CNN structure altogether.	Two-dimensional data classification (i.e. images)
Li et al., 2019	Enhancing the locality of the attention mechanism and addressing the memory bottleneck.	Transformer	Proposed convolutional self-attention to improve locality. Proposes sparse self-attention to address memory requirements.	The convolutional attention has most benefit in applications to challenging datasets. For comparatively less challenging data, it does not offer better performance compared to canonical attention. Sparse attention reduces memory and computation costs, but is generally performing lower than full attention. Exceptions on very complex traffic data suggests however, that sparse attention better detects very long-term relationships.	Time series prediction (traffic, energy, wind)
Chai et al., 2020	Developing a traffic transformer for spatio temporal prediction of complex (traffic) data.	Transformer with GNN (Graph convolutional neural network) component	Proposes enriched positional encoding strategies to enhance time-information content of sequences. I.e. relative, global or periodic PE. Additionally explores intelligent sequence segments composition (adding specified time-steps). The transformer is completed by preceding GNNs in the embedding parts of the encoder and decoder.	Position encoding in periodic ays leads to the worst model performances. Relative PE performs best with slightly better RMSE than global PE. Time series segments strategy performs even better than simpler PE strategies.	Time series prediction (traffic data)

Author, Published	Topic / Focus	Model type	Novelty / Proposition	Key findings	Application area
Lim et al. 2020	Presenting an interpretable (no black box) model for multi-horizon forecasting on complex time series data.	Transformer variant with RNN component	Developed an adaption to the attention mechanism to enhance interpretability. Combine RNN components for local processing with transformer architecture for long-range dependencies.	The model achieves state-of-the-art performance on simple as well as complex data. Furthermore, the interpretable nature enables visualization of learned dependencies and analyze regime changes in the learned patterns.	Time series prediction (electricity, traffic, retail - complex metadata, stock volatility and return)
Wu et al., 2020	Early application of transformer architecture to time series prediction. Benchmarking performance to ARIMA, Seq2Seq and LSTM.	Transformer	Developed a general transformer model for time series prediction, that is conceptually very close to the original transformer.	The transformer model achieves comparable or better results, than LSTM and seq2seq, while all outperform ARIMA. Furthermore, the transformer framework developed can be extended to spatio-temporal space.	Time series prediction (endemic influenza data)
Zhou et al. 2021	Improving the transformer for lang sequence time series forecasting.	Transformer	Addresses memory constraints with propSparse self-attention. Proposes cascading distilling of attention-layer sizes by ahlf to handle extreme long inputs. Proposes novel generative decoder to make multi-step forecasts.	The proposed model outperforms the canonical attention transformer, as well as ARIMA and RNN models. The sparsity distilling improves prediction errors especially for larger inputs compared to canonical attention layers. The model is furthermore able to better predict time steps with lager offsets into the future.	Time series prediction (electricity, elect. Transformer temp., weather)

B ***Constituent List of Experiment Dataset***

Name	Code	Entry	Name	Code	Entry	Name	Code	Entry
3M	MMM	begin	CABLEVISION SYS. DEAD - DELIST.21/06/16	CVC^F16(P)	begin	EDWARDS LIFESCIENCES	EW(P)	28.03.2000
ABBOTT LABORATORIES	ABT	begin	CABOT OIL & GAS 'A'	COG(P)	08.02.1990	ELECTRONIC ARTS	EA.O	begin
ABBVIE	ABBV.K(P)	10.12.2012	CADENCE DESIGN SYS.	CDNS.O(P)	begin	ELI LILLY	LLY	begin
ABIOMED	ABMD.O(P)	begin	CAMPBELL SOUP	CPB	begin	EMERSON ELECTRIC	EMR	begin
ACCENTURE CLASS A	ACN(P)	19.07.2001	CAPITAL ONE FINL.	COF	16.11.1994	ENDO INTERNATIONAL	ENDP.O(P)	18.07.2000
ACTIVISION BLIZZARD	ATVI.O(P)	25.10.1993	CAPRI HOLDINGS	CPRI.K(P)	15.12.2011	ENTERGY	ETR	begin
ACUITY BRANDS	AYI(P)	03.12.2001	CARDINAL HEALTH	CAH	begin	ENVISION HEALTHCARE DEAD - DELIST.11/10/18	EVHC.K^J18(P)	03.12.1997
ADOBE (NAS)	ADBE.O	begin	CARMAX	KMX(P)	04.02.1997	EOG RES.	EOG	begin
ADT	ADT(P)	19.01.2018	CARNIVAL	CCL	begin	EQUIFAX	EFX	begin
ADV.AUTO PARTS	AAP(P)	29.11.2001	CATERPILLAR	CAT	begin	EQUINIX REIT	EQIX.O(P)	11.08.2000
ADVANCED MICRO DEVICES	AMD.O(P)	begin	CBOE GLOBAL MARKETS(BTS)	CBOE.K(P)	15.06.2010	EQUITY RESD.TST.PROPS. SHBI	EQR	12.08.1993
AES	AES	26.06.1991	CBRE GROUP CLASS A	CBRE.K(P)	10.06.2004	ESSEX PROPERTY TST.	ESS(P)	07.06.1994
AFFILIATED MANAGERS	AMG(P)	21.11.1997	CELANESE	CE(P)	21.01.2005	ESTEE LAUDER COS.'A'	EL(P)	17.11.1995
AFLAC	AFL	begin	CELGENE DEAD - DELIST.21/11/19	CELG.O^K19(P)	begin	EVEREST RE GP.	RE(P)	03.10.1995
AGILENT TECHS.	A	18.11.1999	CENTENE	CNC(P)	13.12.2001	EVERGY	EVRG.K(P)	begin
AGL RESOURCES DEAD - DELIST.01/07/16	GAS^G16(P)	begin	CENTERPOINT EN.	CNP	begin	EVERSOURCE ENERGY	ES	begin
AIR PRDS.& CHEMS.	APD	begin	CERNER	CERN.O(P)	begin	EXELON	EXC.O	begin
AIRGAS DEAD - DELIST.23/05/16	ARG^E16(P)	begin	CF INDUSTRIES HDG.	CF(P)	11.08.2005	EXPEDIA GROUP	EXPE.O(P)	20.07.2005
AKAMAI TECHS.	AKAM.O(P)	29.10.1999	CH ROBINSON WWD.	CHRW.O(P)	15.10.1997	EXPEDITOR INTL.OF WASH.	EXPD.O(P)	begin
ALASKA AIR GROUP	ALK(P)	begin	CHARLES SCHWAB	SCHW.K	begin	EXPRESS SCRIPTS HOLDING DEAD - DELIST.21/12/18	ESRX.O^L18(P)	09.06.1992
ALBEMARLE	ALB(P)	22.02.1994	CHARTER COMMS.CL.A	CHTR.O(P)	02.12.2009	EXTRA SPACE STRG.	EXR(P)	12.08.2004
ALEXANDRIA RLST.EQTIES.	ARE(P)	28.05.1997	CHEVRON	CVX	begin	EXXON MOBIL	XOM	begin
ALEXION PHARMS.	ALXN.O(P)	28.02.1996	CHIPOTLE MEXN.GRILL	CMG(P)	26.01.2006	F5 NETWORKS	FFIV.O(P)	07.06.1999
ALIGN TECHNOLOGY	ALGN.O(P)	26.01.2001	CHUBB	CB(P)	25.03.1993	FACEBOOK CLASS A	FB.O(P)	18.05.2012
ALLEGHENY EN. DEAD - DELIST.28/02/11	AYE^B11	begin	CHURCH & DWIGHT CO.	CHD(P)	begin	FASTENAL	FAST.O(P)	begin
ALLEGION	ALLE.K(P)	18.11.2013	CIGNA	CI	begin	FEDERAL REALTY INV.TST.	FRT(P)	begin
ALLERGAN DEAD - DELIST.17/03/15	AGN^C15	begin	CIMAREX EN.	XEC(P)	30.09.2002	FEDEX	FDX	begin
ALLIANCE DATA SYSTEMS	ADS(P)	08.06.2001	CINCINNATI FINL.	CINF.O	begin	FIDELITY NAT.INFO.SVS.	FIS(P)	20.06.2001
ALLSTATE ORD SHS	ALL	03.06.1993	CINTAS	CTAS.O	begin	FIFTH THIRD BANCORP	FITB.O	begin
ALPHA NATURAL RESOURCES DEAD - DELIST.27/07/16	ANRZQ.PK^G16(P)	15.02.2005	CISCO SYSTEMS	CSCO.O	16.02.1990	FIRST REPUBLIC BANK	FRC(P)	09.12.2010
ALPHABET A	GOOGL.O(P)	19.08.2004	CITIGROUP	C	begin	FIRSTENERGY	FE	begin
ALTRIA GROUP	MO	begin	CITIZENS FINANCIAL GROUP	CFG(P)	24.09.2014	FISERV	FISV.O	begin
AMAZON.COM	AMZN.O(P)	15.05.1997	CITRIX SYS.	CTXS.O	08.12.1995	FLEETCOR TECHNOLOGIES	FLT(P)	15.12.2010
AMBAC FINANCIAL GROUP	AMBC.K	01.05.2013	CLOROX	CLX	begin	FLIR SYSTEMS DEAD - DELIST.17/05/21	FLIR.O^E21(P)	22.06.1993
AMCOR	AMCR.K(P)	11.06.2019	CME GROUP	CME.O(P)	06.12.2002	FLOWSERVE	FLS(P)	begin
AMER.ELEC.PWR.	AEP.O	begin	CMS ENERGY	CMS	begin	FMC	FMC(P)	begin
AMEREN	AEE	begin	COCA COLA	KO	begin	FOOT LOCKER	FL(P)	begin
AMERICAN AIRLINES GROUP	AAL.O(P)	09.12.2013	COGNIZANT TECH.SLTN.'A'	CTSH.O(P)	19.06.1998	FORD MOTOR	F	begin
AMERICAN EXPRESS	AXP	begin	COLGATE-PALM.	CL	begin	FORTINET	FTNT.O(P)	18.11.2009
AMERICAN INTL.GP.	AIG	begin	COLUMBIA PIPELINE GROUP DEAD - DELIST.01/07/16	CPGX.K^G16(P)	17.06.2015	FORTIVE	FTV(P)	13.06.2016
AMERICAN TOWER	AMT(P)	05.06.1998	COMCAST A	CMCSA.O(P)	begin	FORTUNE BNS.HM.& SCTY.	FBHS.K(P)	16.09.2011
AMERICAN WATER WORKS	AWK(P)	23.04.2008	COMERICA	CMA	begin	FOSSIL GROUP	FOSL.O(P)	12.04.1993
AMERIPRISE FINL.	AMP(P)	15.09.2005	CONAGRA BRANDS	CAG	begin	FOX A	FOXA.O(P)	12.03.2019
AMERISOURCEBERGEN	ABC	04.04.1995	CONCHO RESOURCES DEAD - DELIST.19/01/21	CXO^A21(P)	03.08.2007	FRANKLIN RESOURCES	BEN	begin
AMETEK	AME(P)	begin	CONOCOPHILLIPS	COP	begin	FREEPORT-MCMORAN	FCX	10.07.1995
AMGEN	AMGN.O	begin	CONSOLIDATED EDISON	ED	begin	GAMESTOP 'A'	GME(P)	13.02.2002
AMPHENOL 'A'	APH(P)	12.11.1991	CONSTELLATION BRANDS 'A'	STZ(P)	begin	GAP	GPS	begin
ANALOG DEVICES	ADI.O	begin	COOPER COS.	COO(P)	begin	GARMIN	GRMN.O(P)	08.12.2000
ANSYS	ANSS.O(P)	20.06.1996	COOPER INDUSTRIES DEAD - ACQD.BY 903749	CBE^L12(P)	begin	GARTNER 'A'	IT(P)	05.10.1993
ANTHEM	ANTM.K	30.10.2001	COPART	CPRT.O(P)	17.03.1994	GENERAL DYNAMICS	GD	begin
AON CLASS A	AON	begin	CORNING	GLW	begin	GENERAL ELECTRIC	GE	begin
APA	APA.O	begin	CORTEVA	CTVA.K(P)	24.05.2019	GENERAL MILLS	GIS	begin
APARTMENT INV.& MAN.'A'	AIV	22.07.1994	COSTCO WHOLESALE	COST.O	22.10.1993	GENERAL MOTORS	GM(P)	18.11.2010
APPLE	AAPL.O	begin	COTY CL.A	COTY.K(P)	13.06.2013	GENUINE PARTS	GPC	begin
APPLIED MATS.	AMAT.O	begin	COVIDIEN DEAD - DELIST.27/01/15	COV^A15(P)	14.06.2007	GGP DEAD - DELIST.29/08/18	GGP^H18(P)	08.04.1993
APTIV	APTV.K(P)	17.11.2011	CROWN CASTLE INTL.	CCI(P)	18.08.1998	GILEAD SCIENCES	GILD.O(P)	22.01.1992
ARCHER DANIELS MIDLAND	ADM	begin	CSRA DEAD - DELIST.04/04/18	CSRA.K^D18(P)	16.11.2015	GLOBAL PAYMENTS	GPN(P)	12.01.2001
ARCONIC	ARNC.K	01.04.2020	CSX	CSX.O	begin	GLOBE LIFE	GL	begin
ARISTA NETWORKS	ANET.K(P)	06.06.2014	CUMMINS	CMI	begin	GOLDMAN SACHS GP.	GS	04.05.1999
ARTHUR J GALLAGHER	AJG(P)	begin	CVS HEALTH	CVS	begin	H&R BLOCK	HRB	begin
ASSURANT	AIZ(P)	05.02.2004	D R HORTON	DHI	05.06.1992	HALLIBURTON	HAL	begin
AT&T	T	begin	DANAHER	DHR	begin	HANESBRANDS	HBI(P)	16.08.2006
ATMOS ENERGY	ATO(P)	begin	DARDEN RESTAURANTS	DRI	09.05.1995	HARLEY-DAVIDSON	HOG	begin
AUTODESK	ADSK.O	begin	DAVITA	DVA(P)	31.10.1995	HARTFORD FINL.SVS.GP.	HIG	15.12.1995
AUTOMATIC DATA PROC.	ADP.O	begin	DEERE	DE	begin	HASBRO	HAS.O	begin
AUTOZONE	AZO	02.04.1991	DELPHI TECHNOLOGIES DEAD - DELIST.02/10/20	DLPH.K^J20(P)	21.11.2017	HCA HEALTHCARE	HCA(P)	10.03.2011
AVALONBAY COMMNS.	AVB(P)	11.03.1994	DELTA AIR LINES	DAL(P)	26.04.2007	HEALTHPEAK PROPERTIES	PEAK.K(P)	begin
AVERY DENNISON	AVY	begin	DENTSPLY SIRONA	XRAY.O(P)	begin	HELMERICH & PAYNE	HP(P)	begin
BAKER HUGHES A	BKR	begin	DEVON ENERGY	DVN	begin	HENRY SCHEIN	HSIC.O(P)	03.11.1995
BALL	BLL	begin	DIAMONDBACK ENERGY	FANG.O(P)	12.10.2012	HESS	HES	begin
BANK OF AMERICA	BAC	begin	DIGITAL REALTY TST.	DLR(P)	29.10.2004	HEWLETT PACKARD ENTER.	HPE(P)	19.10.2015
BANK OF NEW YORK MELLON	BK	begin	DISCOVER FINANCIAL SVS.	DFS(P)	14.06.2007	HILTON WORLDWIDE HDG.	HLT(P)	12.12.2013
BAXALTA DEAD - DELIST.03/06/16	BXLT.K^F16(P)	15.06.2015	DISCOVERY SERIES A	DISCA.O(P)	06.07.2005	HNTGTN.INGALLS INDS.	HII(P)	22.03.2011
BAXTER INTL.	BAX	begin	DISH NETWORK 'A'	DISH.O(P)	21.06.1995	HOLLYFRONTIER	HFC(P)	begin
BECTON DICKINSON	BDX	begin	DOLLAR GENERAL	DG(P)	13.11.2009	HOLOGIC	HOLX.O(P)	02.03.1990
BEMIS DEAD - DELIST.11/06/19	BMS^F19(P)	begin	DOLLAR TREE	DLTR.O(P)	07.03.1995	HOME DEPOT	HD	begin
BERKSHIRE HATHAWAY 'A'	BRKa(P)	begin	DOMINION ENERGY	D	begin	HONEYWELL INTL.	HON.O	begin
BEST BUY	BBY	begin	DOVER	DOV	begin	HOST HOTELS & RESORTS REIT	HST.O(P)	begin
BIOGEN	BIIB.O(P)	17.09.1991	DOW ORD SHS	DOW(P)	20.03.2019	HP	HPQ	begin
BLACKROCK	BLK(P)	01.10.1999	DTE ENERGY	DTE	begin	HUMANA	HUM	begin
BOEING	BA	begin	DUKE ENERGY	DUK	begin	HUNT JB TRANSPORT SVS.	JBHT.O(P)	begin
BOOKING HOLDINGS	BKNG.O(P)	30.03.1999	DUKE REALTY	DRE(P)	begin	HUNTINGTON BCSH.	HBAN.O	begin
BORGWARNER	BWA(P)	13.08.1993	DUPONT DE NEMOURS	DD	01.09.2017	IDEX	IEX(P)	begin
BOSTON PROPERTIES	BXP(P)	18.06.1997	DXC TECHNOLOGY	DXC(P)	begin	IDEXX LABORATORIES	IDXX.O(P)	24.06.1991
BOSTON SCIENTIFIC	BSX	19.05.1992	E TRADE FINANCIAL DEAD - DELIST.05/10/20	ETFC.O^J20(P)	16.08.1996	IHS MARKIT	INFO.K(P)	19.06.2014
BRIGHTHOUSE FINANCIAL	BHF.O(P)	18.07.2017	EASTMAN CHEMICAL	EMN	14.12.1993	ILLINOIS TOOL WORKS	ITW	begin
BRISTOL MYERS SQUIBB	BMY	begin	EATON	ETN	begin	ILLUMINA	ILMN.O(P)	28.07.2000
BROADCOM	AVGO.O(P)	06.08.2009	EBAY	EBAY.O	25.09.1998	INCYTE	INCY.O(P)	04.11.1993
BROADRIDGE FINL.SLTN.	BR(P)	22.03.2007	ECOLAB	ECL	begin	INGERSOLL RAND	IR(P)	12.05.2017
BROWN-FORMAN 'B'	BFb	begin	EDISON INTL.	EIX	begin	INTEL	INTC.O	begin

B (cont.) Constituent List of Experiment Dataset

Name	Code	Entry	Name	Code	Entry	Name	Code	Entry
INTERCONTINENTAL EX.	ICE(P)	16.11.2005	NEWELL BRANDS (XSC)	NWL.O	begin	STATE STREET	STT	begin
INTERNATIONAL BUS.MCHS.	IBM	begin	NEWFIELD EXPLORATION DEAD - DELIST.14/02/19	NFX^B19(P)	12.11.1993	STRYKER	SYK	begin
INTERNATIONAL PAPER	IP	begin	NEWMONT	NEM	begin	SUNTRUST BANKS DEAD - DELIST.09/12/19	STI^L19	begin
INTERPUBLIC GROUP	IPG	begin	NEWS 'A'	NWSA.O(P)	19.06.2013	SVB FINANCIAL GROUP	SIVB.O(P)	begin
INTL.FLAVORS & FRAG.	IFF	begin	NEXTERA ENERGY	NEE	begin	SYNCHRONY FINANCIAL	SYF(P)	31.07.2014
INTUIT	INTU.O	12.03.1993	NIELSEN	NLSN.K(P)	26.01.2011	SYNOPSYS	SNPS.O(P)	26.02.1992
INTUITIVE SURGICAL	ISRG.O(P)	13.06.2000	NIKE 'B'	NKE	begin	SYSCO	SYY	begin
INVESCO	IVZ(P)	04.12.2007	NISOURCE	NI	begin	T ROWE PRICE GROUP	TROW.O	begin
IPG PHOTONICS	IPGP.O(P)	13.12.2006	NOBLE ENERGY DEAD - DELIST.06/10/20	NBL.O^J20(P)	begin	TAKE TWO INTACT.SFTW.	TTWO.O(P)	15.04.1997
IQVIA HOLDINGS	IQV(P)	09.05.2013	NORDSTROM	JWN	begin	TAPESTRY	TPR	05.10.2000
IRON MOUNTAIN	IRM(P)	01.07.1997	NORFOLK SOUTHERN	NSC	begin	TARGET	TGT	begin
J M SMUCKER	SJM(P)	begin	NORTHERN TRUST	NTRS.O	begin	TE CONNECTIVITY	TEL(P)	14.06.2007
JACK HENRY AND ASSOCIATES	JKHY.O(P)	begin	NORTHROP GRUMMAN	NOC	begin	TECHNIPFMC	FTI(P)	17.01.2017
JACOBS ENGR.	J(P)	begin	NORWEGIAN CRUISE LINE HDG.	NCLH.K(P)	18.01.2013	TELEFLEX	TFX(P)	begin
JEFFERIES FINANCIAL GROUP	JEF(P)	begin	NOV	NOV(P)	29.10.1996	TEXAS INSTRUMENTS	TXN.O	begin
JOHNSON & JOHNSON	JNJ	begin	NRG ENERGY	NRG(P)	03.12.2003	TEXTRON	TXT	begin
JOHNSON CONTROLS INTL.	JCI(P)	21.08.1991	NUCOR	NUE	begin	THE HERSHEY COMPANY(FRA)	HSY.F	27.07.1999
JOY GLOBAL DEAD - DELIST.06/04/17	JOY^D17(P)	08.06.2001	NVIDIA	NVDA.O	22.01.1999	THE TRAVELERS COMPANIES BDR	TRVC34.SA	25.05.2016
JP MORGAN CHASE & CO.	JPM	begin	O REILLY AUTOMOTIVE	ORLY.O(P)	23.04.1993	THERMO FISHER SCIENTIFIC	TMO	begin
JUNIPER NETWORKS	JNPR.K	25.06.1999	OCCIDENTAL PTL.	OXY	begin	TIFFANY & CO DEAD - DELIST.07/01/21	TIF^A21	begin
KANSAS CITY SOUTHERN	KSU(P)	begin	OMNICOM GROUP	OMC	begin	TJX	TJX	begin
KELLOGG	K	begin	ONEOK	OKE(P)	begin	T-MOBILE US	TMUS.O(P)	19.04.2007
KEURIG GREEN MOUNTAIN DEAD - DELIST.04/03/16	GMCR.O^C16(P)	21.09.1993	ORACLE	ORCL.K	begin	TOTAL SYSTEM SERVICES DEAD - DELIST.18/09/19	TSS^I19(P)	begin
KEYCORP	KEY	begin	PACCAR	PCAR.O	begin	TRACTOR SUPPLY	TSCO.O(P)	18.02.1994
KEYSIGHT TECHNOLOGIES	KEYS.K(P)	20.10.2014	PACKAGING CORP.OF AM.	PKG(P)	28.01.2000	TRANSDIGM GROUP	TDG(P)	15.03.2006
KIMBERLY-CLARK	KMB	begin	PARKER-HANNIFIN	PH	begin	TRANSOCEAN	RIG(P)	28.05.1993
KIMCO REALTY	KIM(P)	22.11.1991	PAYCHEX	PAYX.O	begin	TRIPADVISOR 'A'	TRIP.O(P)	07.12.2011
KINDER MORGAN	KMI(P)	11.02.2011	PAYPAL HOLDINGS	PYPL.O(P)	06.07.2015	TWITTER	TWTR.K(P)	07.11.2013
KLA	KLAC.O	begin	PENTAIR	PNR(P)	begin	TYSON FOODS 'A'	TSN	begin
KOHL'S	KSS	19.05.1992	PEOPLES UNITED FINANCIAL	PBCT.O(P)	begin	U S BANCORP 100 DS	USB_pa	16.06.2010
KRAFT FOODS GROUP DEAD - ACQD.BY 9801CJ	KRFT.O^G15(P)	17.09.2012	PEPSICO	PEP.O	begin	UDR	UDR(P)	begin
KRAFT HEINZ	KHC.O(P)	06.07.2015	PERKINELMER	PKI	begin	ULTA BEAUTY	ULTA.O(P)	25.10.2007
KROGER	KR	begin	PERRIGO	PRGO.K(P)	18.12.1991	UNDER ARMOUR A	UAA(P)	18.11.2005
L BRANDS	LB	begin	PETSMART DEAD - DELIST.12/03/15	PETM.O^C15(P)	23.07.1993	UNION PACIFIC	UNP	begin
L3HARRIS TECHNOLOGIES	LHX	begin	PFIZER	PFE	begin	UNITED AIRLINES HOLDINGS	UAL.O(P)	25.01.2006
LABORATORY CORP.OF AM. HDG.	LH(P)	begin	PHILIP MORRIS INTL.	PM(P)	17.03.2008	UNITED PARCEL SER.'B'	UPS	10.11.1999
LAM RESEARCH	LRCX.O(P)	begin	PHILLIPS 66	PSX(P)	12.04.2012	UNITED RENTALS	URI(P)	18.12.1997
LAMB WESTON HOLDINGS	LW(P)	01.11.2016	PINNACLE WEST CAP.	PNW	begin	UNITEDHEALTH GROUP	UNH	begin
LEGGETT&PLATT	LEG	begin	PIONEER NTRL.RES.	PXD(P)	08.08.1997	UNIVERSAL HEALTH SVS.'B'	UHS(P)	begin
LEIDOS HOLDINGS	LDOS.K(P)	13.10.2006	PNC FINL.SVS.GP.	PNC	begin	UNUM GROUP	UNM	begin
LENNAR 'A'	LEN(P)	begin	PPG INDUSTRIES	PPG	begin	V F	VFC	begin
LEVEL 3 COMMS. DEAD - DELIST.01/11/17	LVLT.K^K17(P)	10.10.1997	PPL	PPL	begin	VALARIS	VAL(P)	03.05.2021
LINCOLN NATIONAL	LNC	begin	PRINCIPAL FINL.GP.	PFG.O	23.10.2001	VALERO ENERGY	VLO	begin
LINDE	LIN	17.06.1992	PROCTER & GAMBLE	PG	begin	VARIAN MEDICAL SYSTEMS DEAD - DELIST.15/04/21	VAR^D21(P)	begin
LKQ	LKQ.O(P)	03.10.2003	PROGRESSIVE OHIO	PGR	begin	VENTAS	VTR(P)	begin
LOEWS	L	begin	PROLOGIS REIT	PLD(P)	21.11.1997	VERISIGN	VRSN.O(P)	30.01.1998
LORILLARD DEAD - DELIST.12/06/15	LO^F15(P)	01.02.2002	PRUDENTIAL FINL.	PRU	13.12.2001	VERISK ANALYTICS CL.A	VRSK.O(P)	07.10.2009
LOWE'S COMPANIES	LOW	begin	PUB.SER.ENTER.GP.	PEG	begin	VERIZON COMMUNICATIONS	VZ	begin
LUMEN TECHNOLOGIES	LUMN.K	begin	PUBLIC STORAGE	PSA(P)	begin	VERTEX PHARMS.	VRTX.O(P)	24.07.1991
LYONDELLBASELL INDS.CL.A	LYB(P)	28.04.2010	PULTEGROUP	PHM	begin	VIATRIS	VTRS.O(P)	begin
M&T BANK	MTB	begin	PVH	PVH(P)	begin	VISA 'A'	V(P)	19.03.2008
MACERICH	MAC(P)	11.03.1994	QEP RESOURCES (NYS) DEAD - DELIST.17/03/21	QEP^C21(P)	16.06.2010	VORNADO REALTY TRUST	VNO	begin
MACY'S	M	05.02.1992	QORVO	QRVO.O(P)	03.06.1997	VULCAN MATERIALS	VMC	begin
MALLINCKRODT PUB	MNKKQ.PK(P)	17.06.2013	QUALCOMM	QCOM.O	13.12.1991	WABTEC	WAB(P)	16.06.1995
MARATHON OIL	MRO	16.04.1991	QUANTA SERVICES	PWR(P)	13.02.1998	WALGREENS BOOTS ALLIANCE	WBA.O	begin
MARATHON PETROLEUM	MPC(P)	24.06.2011	QUEST DIAGNOSTICS	DGX	17.12.1996	WALMART	WMT	begin
MARKETAXESS HOLDINGS	MKTX.O(P)	05.11.2004	RALPH LAUREN CL.A	RL(P)	12.06.1997	WALT DISNEY	DIS	begin
MARRIOTT INTL.'A'	MAR.O	23.03.1998	RANGE RES.	RRC(P)	16.04.1990	WASTE MANAGEMENT	WM	begin
MARSH & MCLENNAN	MMC	begin	RAYMOND JAMES FINL.	RJF(P)	begin	WATERS	WAT	17.11.1995
MARTIN MRTA.MATS.	MLM(P)	17.02.1994	RAYTHEON TECHNOLOGIES	RTX	begin	WEC ENERGY GROUP	WEC(P)	begin
MASCO	MAS	begin	REALTY INCOME	O(P)	18.10.1994	WELLCARE HEALTH PLANS DEAD - DELIST.24/01/20	WCG^A20	01.07.2004
MASTERCARD	MA(P)	25.05.2006	REGENCY CENTERS	REG.O(P)	29.10.1993	WELLS FARGO & CO	WFC	begin
MAXIM INTEGRATED PRDS.	MXIM.O(P)	begin	REGENERON PHARMS.	REGN.O(P)	02.04.1991	WELLTOWER	WELL.K(P)	begin
MCCORMICK & COMPANY NV.	MKC	begin	REGIONS FINL.NEW	RF	begin	WESTERN DIGITAL	WDC.O(P)	begin
MCDONALDS	MCD	begin	REPUBLIC SVS.'A'	RSG(P)	01.07.1998	WESTERN UNION	WU	20.09.2006
MCKESSON	MCK	10.11.1994	RESMED	RMD(P)	02.06.1995	WESTROCK	WRK	24.06.2015
MEDTRONIC	MDT	begin	ROBERT HALF INTL.	RHI	begin	WHIRLPOOL	WHR	begin
MERCK & COMPANY	MRK	begin	ROCKWELL AUTOMATION	ROK	begin	WILLIAMS	WMB	begin
METLIFE	MET	05.04.2000	ROLLINS	ROL(P)	begin	WILLIS TOWERS WATSON	WLTW.O(P)	12.06.2001
METTLER TOLEDO INTL.	MTD(P)	14.11.1997	ROPER TECHNOLOGIES	ROP(P)	13.02.1992	WPX ENERGY DEAD - DELIST.07/01/21	WPX^A21(P)	12.12.2011
MGM RESORTS INTL.	MGM(P)	begin	ROSS STORES	ROST.O(P)	begin	WW GRAINGER	GWW	begin
MICHAEL KORS HDG. (SWX) DEAD - 12/09/17	KORS.S^I17(P)	13.11.2014	ROYAL CARIBBEAN GROUP	RCL(P)	28.04.1993	WYNN RESORTS	WYNN.O(P)	28.10.2002
MICROCHIP TECH.	MCHP.O(P)	19.03.1993	S&P GLOBAL	SPGI.K	begin	XCEL ENERGY	XEL.O	begin
MICRON TECHNOLOGY	MU.O	begin	SALESFORCE.COM	CRM(P)	23.06.2004	XEROX HOLDINGS	XRX	begin
MICROSOFT	MSFT.O	begin	SBA COMMS.	SBAC.O(P)	16.06.1999	XILINX	XLNX.O	12.06.1990
MID-AMER.APT COMMUNITIES	MAA(P)	28.01.1994	SCHLUMBERGER	SLB	begin	XYLEM	XYL(P)	13.10.2011
MOHAWK INDUSTRIES	MHK(P)	01.04.1992	SEAGATE TECHNOLOGY HOLDINGS	STX.O(P)	11.12.2002	YUM! BRANDS	YUM	11.09.1997
MOLSON COORS BEVERAGE COMPANY B	TAP	begin	SEALED AIR	SEE	begin	ZIMMER BIOMET HDG.	ZBH	25.07.2001
MONDELEZ INTERNATIONAL CL.A	MDLZ.O(P)	13.06.2001	SEMPRA	SRE	begin	ZIONS BANCORP.	ZION.O	begin
MONSTER BEVERAGE	MNST.O(P)	begin	SHERWIN-WILLIAMS	SHW	begin	ZOETIS A	ZTS(P)	01.02.2013
MOODY'S	MCO	19.06.1998	SIGNET JEWELERS	SIG(P)	begin			
MORGAN STANLEY	MS	23.02.1993	SIMON PROPERTY GROUP	SPG	14.12.1993			
MOSAIC	MOS(P)	begin	SKYWORKS SOLUTIONS	SWKS.O(P)	begin			
MOTOROLA SOLUTIONS	MSI	begin	SL GREEN REALTY	SLG(P)	15.08.1997			
MSCI	MSCI.K(P)	15.11.2007	SMITH (AO)	AOS(P)	begin			
NASDAQ	NDAQ.O(P)	01.07.2002	SNAP-ON	SNA	begin			
NAVIENT	NAVI.O(P)	17.04.2014	SOUTHERN	SO	begin			
NEKTAR THERAPEUTICS	NKTR.O(P)	03.05.1994	SOUTHWEST AIRLINES	LUV	begin			
NETAPP	NTAP.O	21.11.1995	STANLEY BLACK & DECKER	SWK	begin			
NETFLIX	NFLX.O(P)	23.05.2002	STARBUCKS	SBUX.O	26.06.1992			

Note: The experiment dataset consists of daily closing prices for the above 530 stocks. Data period is from 01.01.1990 to 31.12.2020 excluding weekends. Stock selection is based on constituents of the S&P 500 major stock index for the same period. Provided codes are according to the Refiniv EIKON convention for data retrieval. Date of entry refers to the first day the experiment data set has non-Null data for the respective stock. *Source:* Refinitiv Eikon.

C List of Experiment Sessions of Varying Model Settings

Transformer Models						
id	*type*	*lookback window*	*positional encoding*	*convolution*	*attention mode*	*input variables*
1	Transformer	80	global	kernel size 3	full self-attention	closing price
2	Transformer	80	relative	kernel size 3	full self-attention	closing price
3	Transformer	80	weekday (Mo-Fr)	kernel size 3	full self-attention	closing price
4	Transformer	80	none	kernel size 3	full self-attention	closing price
5	Transformer	80	global	none	full self-attention	closing price
6	Transformer	40 out of 80	global	kernel size 3	sparse self-attention	closing price
7	Transformer	40	global	kernel size 3	full self-attention	closing price
8	Transformer	80	global	kernel size 3	full self-attention	closing price, one-day return, 3-week return average
LSTM Models						
id	*type*	*lookback window*				*input variables*
9	long-short Term Memory	80				one-day return
10	long-short Term Memory	80				closing price, one-day return, 3-week return average
Other						
id	*type*					
11	Monkey (random choice)					
12	S&P 500 constituents long (market portfolio)					

D Experimental Reliability and Problems

The experiment's code was run on Google's Colab-Professional Service. While the gives access to generally reliable servers with GPU support, it is still limited in its capabilities when considering very large datasets or complex models. For one aspect, it does not give control over the actual hardware the virtual machine (VM) is set up on, nor does it guarantee access to GPU-support or the same number of VMs to use concurrently (usually one, sometimes two). Additionally, runtimes of the VMs are restricted to an again not explicitly specified and varying length of approximately up to 12-20h maximum, albeit shorter in case internet or browser communication was instable.

As the dataset included 530 stocks over a time of 31 years that adds to around 4.2m individual stock closing prices. Slicing and combining this into the sequences of window size 80 for the models brings that number up to 332m (although not simultaneously in memory) that passed through the network repeatedly as the experimental frame rolled through the complete timespan of experimental data. As this made the experiments very time-consuming a typical experiment run took more than one virtual machine session to complete.

This introduced two challenges, that should be noted and could potentially affect the experiments' reliability.

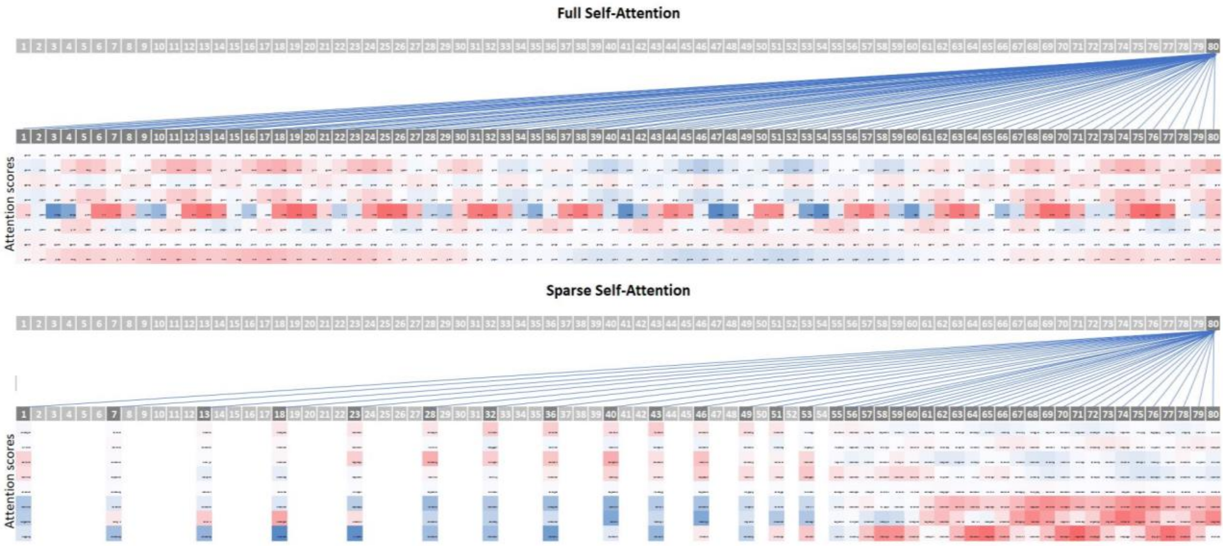
As VM-sessions could be interrupted uncontrollably and thus also during the continuous saving of results to external memory, these files were observed to be compromised on at least one occasion. Also, the disruption necessitated the splitting of the experiment into several parts and eventually merging the datafiles. Although detailed documentation of experiments progress was made and the files carefully reviewed, as small residual risk cannot be one hundred percent excluded.

Secondly, time-demand and resource constraints forbade it to run every setting plenty of times to compare robustness. This might be a challenge to the experiments reliability when compared to other research. However, it is believed, that the number of days and therefore prediction-based trades in the experiment's timeframe provide a sufficiently large number for stability.

E *Performance, Return and Risk Metrics for Model Settings from Ablation Study*

Model	Transformer 1		Transformer 2		Transformer 3		Transformer 4		Transformer 5		Transformer 6		Transformer 7		Transformer 8		LSTM 9		LSTM 10	
--- PERFORMANCE MEASURES ---																				
Mean Accuracy:	0.5180		0.5222		0.5192		0.5175		0.5211		0.5185		0.5239		0.5242		0.5054		0.5059	
Trades (long/short each)	1	100	1	100	1	100	1	100	1	100	1	100	1	100	1	100	1	100	1	100
Mean Return:	0.0026	0.0006	0.0023	0.0005	0.0022	0.0005	0.0021	0.0005	0.0026	0.0004	0.0023	0.0004	0.0022	0.0004	0.0022	0.0005	0.0037	0.0004	0.0034	0.0004
Mean Return (long):	0.0051	0.0014	0.0047	0.0013	0.0045	0.0013	0.0043	0.0012	0.0050	0.0011	0.0047	0.0011	0.0045	0.0013	0.0047	0.0013	0.0067	0.0012	0.0064	0.0012
Mean Return (short):	0.0000	-0.0001	-0.0002	-0.0003	-0.0001	-0.0003	0.0000	-0.0003	-0.0001	-0.0003	0.0000	-0.0004	-0.0001	-0.0004	-0.0003	-0.0004	0.0006	-0.0004	0.0004	-0.0004
Standard error:	0.0004	0.0000	0.0004	0.0000	0.0004	0.0000	0.0004	0.0000	0.0004	0.0001	0.0004	0.0000	0.0005	0.0000	0.0005	0.0000	0.0005	0.0001	0.0005	0.0001
t-Statistic:	5.9132	13.9845	5.0839	13.0081	5.1545	11.9764	4.9056	11.6370	6.2394	6.9667	5.3673	8.8431	4.8959	10.9970	4.8820	11.1623	7.3554	7.3706	6.9144	8.5431
Minimum:	-0.3267	-0.0481	-0.3267	-0.0463	-0.3247	-0.0495	-0.3267	-0.0495	-0.2729	-0.0753	-0.3267	-0.0425	-0.3255	-0.0481	-0.3247	-0.0421	-0.5081	-0.0450	-0.5125	-0.0468
Quartile 1:	-0.0078	-0.0010	-0.0085	-0.0010	-0.0086	-0.0010	-0.0082	-0.0010	-0.0103	-0.0013	-0.0084	-0.0011	-0.0100	-0.0010	-0.0096	-0.0012	-0.0118	-0.0013	-0.0114	-0.0014
Median:	0.0002	0.0004	0.0003	0.0003	0.0003	0.0003	0.0000	0.0003	0.0001	0.0002	0.0003	0.0002	0.0002	0.0002	0.0001	0.0003	0.0013	0.0002	0.0011	0.0003
Quartile 3:	0.0094	0.0020	0.0101	0.0020	0.0100	0.0019	0.0095	0.0019	0.0124	0.0020	0.0098	0.0017	0.0108	0.0017	0.0110	0.0020	0.0178	0.0020	0.0174	0.0022
Maximum:	0.7089	0.0626	0.7089	0.0550	0.7089	0.0585	0.7089	0.0585	0.5169	0.0558	0.7089	0.0519	0.7236	0.0528	0.7236	0.0550	0.4994	0.0382	0.5000	0.0423
Share: positive return	0.5177	0.5842	0.5184	0.5790	0.5175	0.5734	0.5112	0.5733	0.5133	0.5481	0.5169	0.5445	0.5139	0.5555	0.5110	0.5579	0.5477	0.5507	0.5436	0.5596
Share: negative return	0.4823	0.4158	0.4816	0.4210	0.4825	0.4266	0.4888	0.4267	0.4867	0.4519	0.4831	0.4555	0.4861	0.4445	0.4890	0.4421	0.4523	0.4493	0.4564	0.4404
Standard dev.:	0.0385	0.0039	0.0393	0.0037	0.0385	0.0038	0.0386	0.0036	0.0373	0.0045	0.0386	0.0035	0.0402	0.0036	0.0401	0.0037	0.0434	0.0043	0.0425	0.0044
Skewness:	5.3221	1.2365	4.8476	0.9909	5.3037	1.0035	5.1496	0.8407	4.2207	-1.0717	5.1518	0.7964	4.7275	0.7260	4.5721	0.9336	0.2765	0.3687	0.6776	0.0495
Kurtosis:	83.2543	28.2571	76.7980	24.9826	83.4658	35.5613	82.0731	31.3495	60.2211	40.2093	81.4423	25.4088	71.5028	29.1173	70.1751	19.8809	30.8749	14.6309	33.7330	15.0483
--- RISK CHARACTERISTICS ---																				
VaR 1%:	0.0869	0.0085	0.0891	0.0080	0.0872	0.0083	0.0877	0.0080	0.0842	0.0100	0.0875	0.0078	0.0913	0.0078	0.0911	0.0081	0.0973	0.0095	0.0955	0.0098
CVaR 1%:	0.0999	0.0098	0.1024	0.0093	0.1003	0.0096	0.1008	0.0092	0.0968	0.0115	0.1006	0.0089	0.1049	0.0090	0.1047	0.0093	0.1120	0.0110	0.1100	0.0112
VaR 5%:	0.0607	0.0058	0.0623	0.0055	0.0610	0.0057	0.0614	0.0055	0.0588	0.0070	0.0612	0.0054	0.0639	0.0054	0.0638	0.0056	0.0677	0.0066	0.0666	0.0068
CVaR 5%:	0.0768	0.0074	0.0788	0.0070	0.0771	0.0073	0.0775	0.0070	0.0744	0.0088	0.0773	0.0068	0.0807	0.0069	0.0805	0.0071	0.0858	0.0084	0.0843	0.0086
Max. drawdown:	-27.149	-0.2560	-27.149	-0.1738	-1.2496	-7.1566	-0.9107	-0.3375	-27.149	-0.2538	-5.8525	-0.2553	-5.6822	-0.2705	-6.9984	-0.3026	-8.2447	-7.8049	-3.8534	-4.8431
Max. drawdown adj.:	-0.3684	-0.1348	-0.2444	-0.1583	-0.6199	-0.3497	-0.9107	-0.2241	-0.4908	-0.2130	-0.3275	-0.1170	-0.2727	-0.1069	-0.2079	-0.1084	-0.2042	-1.8936	-0.3033	-0.7594
--- ANNUALIZED RISK-RETURN CHARACTERISTICS ---																				
Return p.a. (distr):	0.6699	0.1607	0.5882	0.1408	0.5856	0.1343	0.5568	0.1244	0.6860	0.0913	0.6105	0.0908	0.5796	0.1151	0.5768	0.1207	0.9561	0.0939	0.8809	0.1120
Excess return p.a. (distr):	0.6410	0.1332	0.5581	0.1132	0.5533	0.1059	0.5250	0.0962	0.6595	0.0636	0.5803	0.0633	0.5494	0.0875	0.5483	0.0930	0.9221	0.0678	0.8506	0.0854
Standard dev. p.a.:	0.6212	0.0630	0.6344	0.0594	0.6214	0.0614	0.6240	0.0588	0.6029	0.0718	0.6238	0.0563	0.6492	0.0574	0.6479	0.0593	0.7009	0.0687	0.6870	0.0707
Downside dev.:	0.0201	0.0023	0.0214	0.0022	0.0206	0.0023	0.0207	0.0023	0.0203	0.0031	0.0206	0.0022	0.0219	0.0022	0.0221	0.0023	0.0284	0.0028	0.0275	0.0030
Downside dev. p.a.:	0.3254	0.0374	0.3459	0.0359	0.3325	0.0376	0.3338	0.0370	0.3272	0.0505	0.3325	0.0360	0.3536	0.0357	0.3575	0.0366	0.4594	0.0457	0.4442	0.0476
Sharpe ratio p.a.:	1.0328	2.1012	0.8825	1.8956	0.8969	1.7284	0.8470	1.6353	1.0909	0.8765	0.9334	1.1103	0.8493	1.5126	0.8466	1.5585	1.3238	0.9549	1.2412	1.1840
Sortino ratio p.a.:	1.9736	3.5612	1.6205	3.1506	1.6785	2.8405	1.5844	2.6064	2.0121	1.2596	1.7530	1.7565	1.5607	2.4490	1.5359	2.5423	2.0241	1.4773	1.9239	1.7996

F Visualization of Attention Scores



The figure color codes attention scores for predicting the same stock for a given day with full self-attention and sparse self-attention. In full self-attention every element attends to all other elements in the sequence. In sparse self-attention every element only attends to a subset of the sequence. Depicted are only the attention relations of the last element (80) for the first attention layer from the encoding stack. The model's MHA-block comprised 8 heads, which we can see picking up different aspects from the data in the full as well as in the sparse self-attention mode.

G ***Python Packages and Contributions by Others***

Standard python libraries:

Matplotlib 3.3.3, John D. Hunter, Michael Droettboom, https://matplotlib.org

NumPy 1.21.2, Travis E. Oliphant et al., https://www.numpy.org

Pandas 1.3.3, Pandas Development Team, https://pandas.pydata.org

Torch 1.9.0, PyTorch Team, https://pytorch.org

SciPy 1.7.1, https://www.scipy.org

Seaborn 0.11.2, Michael Waskom, https://seaborn.pydata.org

Sklearn 0.0, https://www.python.org/pypi/scikit-learn/

TensorFlow Keras 2.4.3, Google Inc., https://keras.io

Torch 1.9.0, PyTorch Team, https://pytorch.org

Further code sources

Repo-2021, Rubens Zimbres, https://github.com/RubensZimbres/Repo-2021/tree/main/Transformer,

[Accessed 16.09.2021]: foundation for time series transformer

Transformer-time-series-predictions, Oliver Guhr, https://github.com/oliverguhr/transformer-time-series-prediction,

[Accessed 16.09.2021]: ideas and reference

Keras-transformer, Kirill Mavreshko, https://github.com/kpot/keras-transformer,

[Accessed 16.09.2021]: ideas and reference

Informer2020, Haoyi Zhou, https://github.com/zhouhaoyi/Informer2020,

[Accessed 16.09.2021]: ideas and reference

Transformers for Time Series, Max Cohen, https://github.com/maxjcohen/transformer,

[Accessed 16.09.2021]: ideas and reference

H *Parameter Settings*

Model	Transformer 1	Transformer 2	Transformer 3	Transformer 4	Transformer 5	Transformer 6	Transformer 7	Transformer 8	LSTM 9	LSTM 10
Experiment	Transformer	Transformer	Transformer	Transformer	Transformer	Transformer	Transformer	Transformer	LSTM	LSTM
EID	indiv.	indiv.	indiv.	indiv.	indiv.	indiv.	indiv.	indiv.	indiv.	indiv.
	MODEL PARAMETERS									
window_size	80	80	80	80	80	80	40	80	80	80
V	200	200	200	200	200	200	200	200	-	-
N	6	6	6	6	6	6	6	6	-	-
dropout	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
convolve	3	3	3	3	0	3	3	3	-	-
pe_type	'global'	'relative'	'weekday'	None	'global'	'global'	'global'	'global'	-	-
attention_type	'full'	'full'	'full'	'full'	'full'	[79,73,67,62,57,52,48,44,40,37,34,31,29,27,25,24,23,22,21,20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0]	'full'	'full'	-	-
mode	'no;5'	'no;5'	'no;5'	'no;5'	'no;5'	'no;5'	'no;5'	'no;5'	-	-
	TRAINING PARAMETERS									
criterion	nn.MSELoss()	nn.MSELoss()	nn.MSELoss()	nn.MSELoss()	nn.MSELoss()	nn.MSELoss()	nn.MSELoss()	nn.MSELoss()	ks.losses.Binary Crossentropy()	ks.losses.Binary Crossentropy()
learning	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
epochs	20	20	20	20	20	20	20	20	20	20
patience	3	3	3	3	3	3	3	3	3	3
nbatch	1	1	1	1	1	1	1	1	1	1
minibatch	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	-	-
m_init_freq	1	1	1	1	1	1	1	1	1	1
	SEQUENCE PARAMETERS									
block_size	320	320	320	320	320	320	320	320	320	320
step_size	40	40	40	40	40	40	40	40	40	40
	DATA PARAMETERS									
start	0	0	0	0	0	0	0	0	0	0
end	8088	8088	8088	8088	8088	8088	8088	8088	8088	8088
var_order	['price','price']	['price','price']	['price','price']	['price','price']	['price','price']	['price','price']	['price','price']	['ret',price','average','price']	['ret','binclass']	['ret',price','average','binclass']
trg_in_src	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
d_tgt	1	1	1	1	1	1	1	1	-	-
	LOCATION PARAMETERS									
data_path	'resources\SP500_Price_Inputdata.csv'									
time_path	'resources\Timeline_010190_300621.csv'									

Parameter names refer to the experiment code available from https://github.com/Humboldt-WI/dissertations/tree/main/TS_Transformer_Finance.