Attention based Temporal Convolutional Network for stock price prediction

Paolo Janssen

22 April 2022

Master Artificial Intelligence
Utrecht University
Student number: 3802485

Supervisors:
Thijs van Ommen
Zerrin Yumak
Ad Feelders

Abstract

Stock prices are notoriously difficult to predict, making it an excellent testing ground for deep learning models. A recent survey however shows that the TCN is rarely used for stock price prediction with exclusively financial data. In this study we will attempt to improve the TCN performance by looking at the effects of adding attention mechanisms to the TCN for forecasting stock prices. We also propose an architecture called the ATCN, a model that combines temporal and hierarchical attention in the TCN framework. Performance of the TCN is compared to models with hierarchical attention (HA-TCN), temporal attention (TCAN) or both (ATCN). We also evaluate the performance of the attention-based models when using a different number of layers. Results indicate that the TCAN performs best on average and that attention-based models need less filters and layers to perform well. We conclude that attention-based models are preferred over the standard TCN due to significantly faster training times and roughly similar performance, with TCAN the clear winner on this dataset. The ATCN shows some potential but needs to be tested further on more complex datasets. We have made the code used available on https://github.com/PaoloJ36/ATCN/.

1 Introduction

Stock prices are notoriously difficult to predict due to the many factors at play, ranging from predicted earnings and macroeconomic factors all the way to confidence in the company's management and products. Additionally, in recent years large amounts of stock market data has become available to the public for free. This makes the field a great testing ground for deep learning models. Stock prices are generally predicted using two broad types of data: financial data and market sentiment. This study will focus on using financial data exclusively for simplicity's sake.

A survey summarizing 86 recent papers about forex and stock price prediction through deep learning by Hu et al. (2021) shows that there are 5 broad categories of models that are currently used in the field. These are the Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Deep Neural Network (DNN) and Reinforcement Learning (RL). The LSTM makes up the biggest category by far, being the model of choice in 38 out of the 86 papers.

Deep learning practitioners have also generally favored recurrent architectures as a means to tackle sequential tasks up until recently. In a comparative paper by Bai et al. (2018) however, it is shown that the Temporal Convolutional Network (TCN) outperforms these in a large variety of sequential tasks. In addition to that, it is mentioned that the TCN also has other advantages over recurrent architectures. These include being easier and faster to train, having more flexibility in its receptive field and having

a clearer and simpler structure. Bai and colleagues argue that the TCN should be seen as a natural starting point for sequence modeling tasks.

When it comes to stock market prediction however, TCN's are rarely used. The recent survey mentioned earlier by Hu et al. (2021) did not include any papers that used a TCN. There is one recent study that uses TCN's for knowledge driven prediction (Deng et al., 2019), but there seem to be no major studies that use the TCN for financial data. As mentioned, the TCN tends to outperform the currently used recurrent architectures in a variety of sequential tasks. This would make the TCN a good starting point to improve the accuracy of stock market prediction. It would however be more interesting to see if we can improve the results even further than simply applying the TCN to financial data.

One way to improve model performance is to add an attention mechanism (Wang et al., 2016). Attention has shown to improve results in a wide variety of tasks such as image recognition (Mnih, 2014), machine translation (Bahdanau et al., 2014) and video captioning (Gao et al., 2017). Attention-based models have gained even further popularity due to the introduction of the attention-based transformer architecture in a Google paper (Vaswani et al., 2017).

The mechanism of attention can also be used to improve the TCN. In a study about myotonic dystrophy diagnosis, Lin et al. (2019) propose a Hierarchical Attention-based Temporal Convolutional Network (HA-TCN) that significantly outperforms regular TCN and LSTM benchmarks. They show that the HA-TCN is better at highlighting key time series that are useful for classification, while also significantly increasing the performance at a lower number of layers used. The lower number of layers allow the computation time to be reduced significantly without reducing performance. Hao et al. (2020) have found significant performance improvements on language related tasks by using temporal attention. They propose to integrate temporal attention in a new model called the Temporal Convolutional Attention-based Network (TCAN). The HA-TCN and TCAN have shown that the TCN structure can be improved upon by adding different types of attention. We propose to add the two types of attention used in these models to a single novel model which we dub the Attentive Temporal Convolutional network (ATCN). Our prediction is that this will lead to better results, specifically when using a lower number of layers.

In this study, we will combine the TCN architecture with two different forms of attention in order to answer the research question 'How effective is an attention based temporal convolutional network in forecasting individual stock prices?'. The performance of the TCN will be compared to the performance of the HA-TCN and the TCAN. We will also compare the performance to our novel architecture, the ATCN. The experiments will be done using financial stock market data exclusively. We will also be taking an in-depth look at the effect that the different types of attention have

on the performance with a different number of layers. This will allow us to gain understanding of how attention improves the performance of a TCN, and perhaps enable us to catch hints of how to further improve the model's performance in future studies.

First, we will describe the architecture of the TCN. This is followed by the different kinds of attention that are used in this study and how they are used in the ATCN. We will also describe some performance inconsistency issues with the HA-TCN and ATCN and our solution. The experimental setup will then be explained in detail, followed by the results of our experiments and a discussion. We then summarize our results and add some concluding remarks, study limitations and suggestions for future studies.

2 Temporal convolutional networks

We begin by describing the generic architecture of a temporal convolutional network (TCN). This is based on the architecture described by Bai et al. (2018).

The TCN is based on two principles: 1) the convolutions in the architecture are causal, meaning that there is no information leakage from future to past; 2) The architecture also maps a sequence of any length to an output of the same length (Bai et al., 2018). The TCN uses dilated convolutions to achieve a large and flexible receptive field, which we will explain in more detail in section 2.2. The TCN also makes use of residual connections to allow the network to learn and modify an identity mapping, preventing exploding and vanishing gradient problems. Residual connections will be explained in section 2.3

2.1 Causal convolutions

The first principle, no information is leaked from the future to the past, is met by using causal convolutions. Causal convolutions differ from regular convolutions by only using input elements up to the current timestep to compute the output of the current timestep. This ensures that no future information is leaked to any of the earlier timesteps.

The TCN uses a 1D fully-convolutional network architecture (Long et al., 2015). In order to keep the subsequent layers the same size as the input layer, zero padding of length (kernel size - 1) is added to the left of the input. This ensures that the second principle is met: a sequence of any length can always be mapped to an output of the same length.

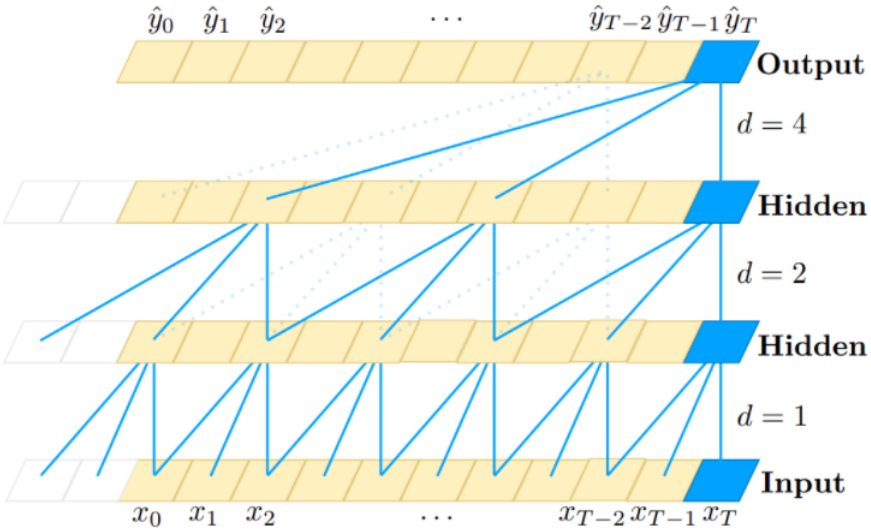

Figure 1: Visual representation of dilated convolutions where dilation factors d = 1, 2, 4 and kernel size k = 3 (Bai et al., figure 1a, p. 4).

2.2 Dilated convolutions

Causal convolutions can only use a history with a size that is linear to the depth of the network. This makes it hard to use on tasks that require a longer history size. Bai et al. (2018) handled this problem by following the work of van den Oord et al. (2016) using dilated convolutions. Dilated convolutions have a dilation factor that allows the model to have an exponentially large receptive field (Yo & Koltun, 2016).

Following the formal definition of Bai et al. (2018), for a 1-D sequence input $x \in \mathbb{R}^n$ and a filter $f : \{0,\ 1,\ ...,\ k-1\} \rightarrow \mathbb{R}$, the dilated convolution operation F on element s of the sequence is defined as

$$F(s) = (x *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot x_{s-d \cdot i}$$

where d is the dilation factor, k is the kernel size, and $s-d \cdot i$ accounts for the direction of the past.

To put it simply, this means that dilation introduces a fixed number of steps between every pair of filter taps. With a dilation factor of 1, this would be equivalent to a regular convolution. Larger dilations however allow the output of higher levels to represent a wider range of inputs to increase the receptive field range of the network. See figure 1 for a visual representation of dilated convolutions.

It is common to increase the dilation factor exponentially with the depth of the network to ensure that there is a filter that hits each input within the ef-

fective history (Bai et al., 2018). This allows the network to take advantage of the entire history in an effective manner. We will also use the exponentially increasing dilation factor (i.e. d = 2^i at level i of the network) in our experiments. When using this exponentially increasing dilation factor, the total receptive field can be calculated by

$$RF = 2^{L-1} * k * b$$

where RF is the receptive field, L is the amount of layers, k is the kernel size and b is the amount of residual blocks, which in this study is always 1.

2.3 Residual connections

Training large networks can be notoriously difficult because of the exploding and vanishing gradient problems first described by Bengio et al. (1994). A study done by He et al. (2016) showed the importance of residual connections when dealing with these problems. A residual connection, also called a shortcut or skip connection, performs an identity mapping on the input, adding the result to the output of other layers (He et al. 2016). Formally, this can be described as follows:

$$o = x + f(x)$$

where o is the output and f is a transformation or series of transformations on the input x. Adding the original input x to the output of a series of transformations allows layers to learn modifications to the identity mapping instead of modifications to the entire transformation. This helps larger and deeper networks stabilize by preventing exploding and vanishing gradient problems.

As a baseline for the TCN architectures, we will be using the implementation of Bai et al. (2018). This consists of a residual block containing multiple layers of dilated causal convolutions and non-linearity using the ReLU function (Nair et al., 2010). The input and output of a residual connection can have different widths. To ensure that the elementwise addition receives a tensor of the same shape, an additional 1x1 convolution is used in the residual connection. See figure 2 for a visual representation of a residual block.

Note that for the HA-TCN and ATCN, the residual connection is changed due to inconsistent performance of the models (see section 3.4 for a more detailed explanation). Because hierarchical attention could not keep the original input intact, the skip connection in these two models is now connected directly from the input to the output layer. This means the skip connection now also skips the within-layer attention layer and the across-layer attention layer.

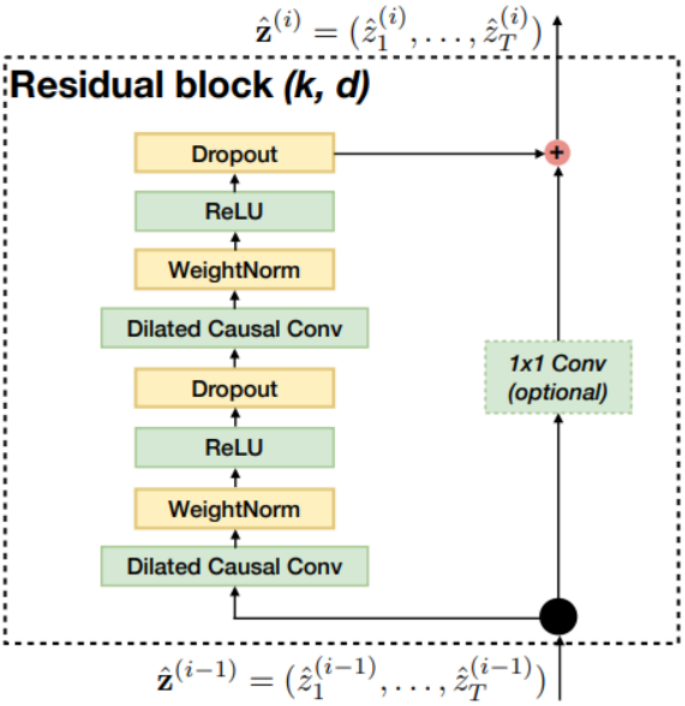

Figure 2: Visual representation of a residual block. The 1x1 convolutional is added when input and output have different dimensions (Bai et al., figure 1b, p. 4).

3 Attention

In order to improve the basic TCN structure, we will use mechanisms of attention. Attention allows the model to compute and use a categorical distribution as a selection process to place more emphasis on certain elements. We will discuss the two specific implementations of attention that are used in this study, hierarchical attention and temporal attention, in more detail.

3.1 Hierarchical attention

In the basic TCN structure, only the activation of the deepest hidden layer is used for the output layer. This means that all information has to be condensed to a C x 1 vector, where C is the number of kernel filters. Hierarchical attention enables the model to also use activations of layers other than the deepest layer, creating additional pathways for the model to use before having to condense the information. This allows the model to more efficiently learn sequential problems without having to increase network depth (Lin et al., 2019).

Formally, suppose the TCN has K hidden layers. Let H_i be the matrix of convolutional activations at layer i where $i = 1,\ 2,\ ...,\ K$; $H_i = [h_1^i,\ h_2^i,\ ...,\ h_T^i] \in \mathbb{R}^{C \times T}$ where C is the number of kernel filters at each layer. The within-layer attention weight $a_i \in \mathbb{R}^{1 \times T}$ is calculated as:

$$a_i = softmax(tanh(w_i^T H_i))$$

where $w_i \in \mathbb{R}^{C \times 1}$ is a trained parameter vector and $(\cdot)^T$ is a transpose operation. Then, the activations of the within-layer attention layers are calculated as:

$$\gamma_i = ReLU(H_i a_i^T))$$

where the activation $\gamma_i \in \mathbb{R}^{C \times 1}$.

After computing the within attention layer, the activations are transformed into $M = [\gamma_0,\ \gamma_1,\ ...,\ \gamma_i,\ ...,\ \gamma_K]$ where $M \in \mathbb{R}^{C \times K}$. Similarly, the across-layer attention then takes matrix M as the input to calculate the across-layer attention layer:

$$a = softmax(tanh(w^T M))$$

$$\gamma = ReLU(Ma^T))$$

where $w \in \mathbb{R}^{C \times 1}$, $a \in \mathbb{R}^{1 \times K}$, $\gamma \in \mathbb{R}^{C \times 1}$. γ is then passed to the output layer of the model.

To state it more simply, the TCN uses a trainable vector to generate weights and then uses these weights to compute within-layer attention. The vectors that this generates are transformed into a matrix and the process is then repeated on this matrix for across-layer attention. See figure 3 for

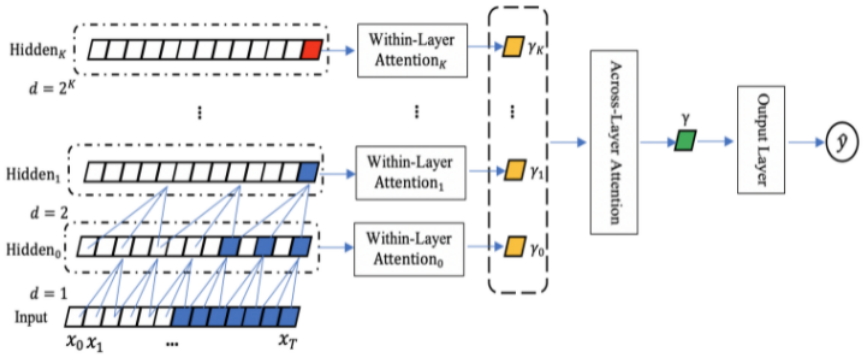

Figure 3: Visual representation of within-layer attention and across-layer attention in a TCN with K hidden layers, kernel filter k = 3 and dilative factor d that increases exponentially with depth (Lin et al., figure 1, p. 2).

a visual representation of how within-layer and across-layer attention are used in a TCN.

Intuitively, this means that within-layer attention is an additional way for the model to capture important information from each layer. In the regular TCN, this information would have to be processed through all of the other convolutional layers first in order to reach the output layer. Because within-layer attention captures this information from every layer, the information can potentially reach the output layer without having to go through more convolutional layers. This could give the higher convolutional layers the option to process different information or the same information in a different way instead. Between-layer attention then allows the model to extract the most important information of all the within-layer attention layers.

Note that we had some issues with performance consistency when using hierarchical attention in models. See section 3.4 for a more detailed explanation of how hierarchical attention can cause these issues and our solution to the problem.

3.2 Temporal attention

Temporal attention can be described as a generic attention mechanism, meaning that it puts more emphasis on certain elements over others. The difference from a generic attention mechanism is the temporal aspect, which means that there can be no information leakage from future to past. We will prove that our implementation of temporal attention has no leakage from future to past at the end of this section. Temporal attention can help the model to capture important information more easily without interfering with the causal aspects of the TCN (see section 2.1).

For temporal attention, the attention weights and the activations are calculated in a different manner than hierarchical attention. The first thing to note is that the trainable parameters are now a matrix $w_i \in \mathbb{R}^{C \times T}$. The temporal attention weights $a_i \in \mathbb{R}^{C \times C}$ are then calculated as:

$$a_i = softmax(tanh(w_i H_i^T))$$

where $(\cdot)^T$ is a transpose operation. Then, the activations of the temporal attention layers are calculated as:

$$\gamma_i = ReLU(H_i^T a_i))$$

where the activation $\gamma_i \in \mathbb{R}^{T \times C}$. The activations are then transposed and used as an input for the the next convolutional layer.

Note that this implementation of temporal attention is different from the implementation in the original TCAN paper by Hao et al. (2020). In the original paper, the authors use queries, keys and values as three different representations of the same data through the self-attention mechanism, but then modified to ensure no information can be leaked from future to past. Self-attention is described as a function which maps a query and a set of key-value pairs to an output (Vaswani et al., 2017). This mechanism aims to capture the complex relationships between words by relating them to each other multiple times (Rosin et al., 2022). For language data such as used in the paper by Hao et al. (2020) this makes sense, as the meaning of a word can depend on other words in the sentence, sentence structure or even different sentences. For financial data however, this is not needed. The meaning of the financial data used in this study does not change according to context. A stock price of 0.8 today will have the same meaning regardless of what the stock price was a week ago. Because the context relationship is significantly less complex for the financial data used in this study, the temporal attention mechanism used here is simplified to only include a single representation.

We prove that our implementation of temporal attention has no leakage from future to past by looking at an example calculation in figure 4. The weights are all set to a value of 1 to ensure that the result does not follow from certain weights being deactivated by having a value of 0. In the input, only the latest timestep has values of 1 while the other timesteps have values of 0. This makes it easy to check for leakage, because any value above 0 in the first 3 rows of the output γ_i would show that there is leakage. In the calculation, we see that the values in these row are 0 just like the input. This shows there is no leakage from future to past.

3.3 Attentive Temporal Convolutional Network

Attentive Temporal Convolutional Network (ATCN) is the novel architecture proposed by this study. The ATCN uses the baseline TCN but implements

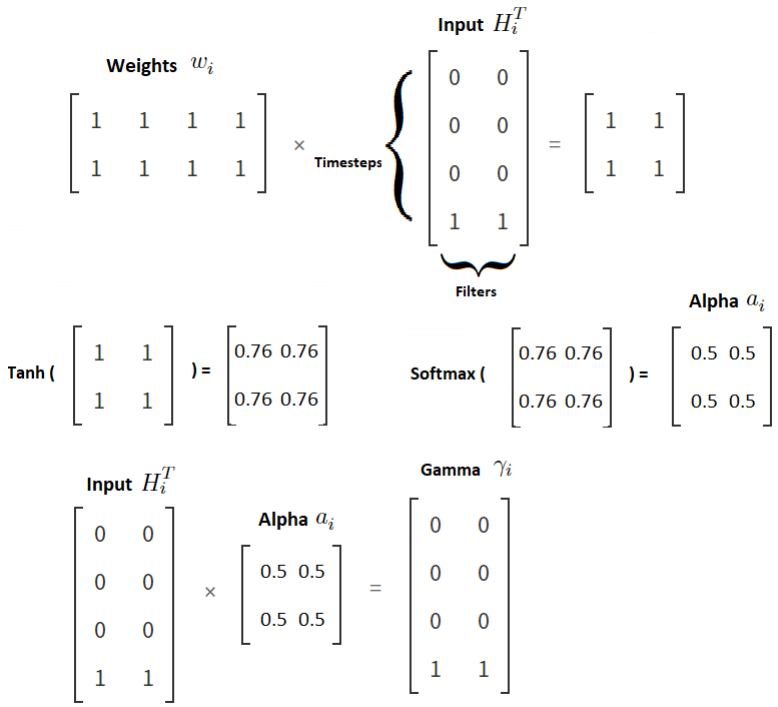

Figure 4: Example calculation to prove this implementation of temporal attention has no leakage from future to past.

both hierarchical (see section 3.1) and temporal (see section 3.2) attention. The model aims to benefit from temporal attention's ability to capture important information more easily, while also applying hierarchical attention to reduce the number of layers used.

In order to achieve this, we adopt a structure that resembles both the TCAN and the HA-TCN closely. Temporal attention is applied first and then the activation is both connected to the next convolutional layer (as used in the TCAN) and to the within-layer attention layer (as used in the HA-TCN). Just like with the HA-TCN, the output of the within-layer attention is then used as input for the across-layer attention layer, which is then connected to the output layer. A skip connected is added directly from the input to the output layer (see section 3.4 for a more in depth analyses of why this was done).

Using temporal attention first should help the model capture important information more easily. Temporal attention also has a focus on picking up patterns using the time aspect, with no leakage from future to past. Hierarchical attention on the other hand extracts information quickly across all timesteps, allowing the model to have a reduced number of layers and significantly speeds up the training time. There could be synergy between these two types of attention, where temporal attention picks up the patterns based on time, while hierarchical attention speeds up the information extraction and picks up general patterns using all timesteps. The model also has more flexibility in extracting information as it can choose to use more of the combination of convolutional layers and temporal attention that (resembling the TCAN) or extract the information in the earlier layers more quickly by using hierarchical attention. Using temporal attention's ability to capture relevant information more easily and hierarchical attention's ability to quickly extract that information, this could create a model that has both an enhanced performance and much faster training time.

3.4 Inconsistency of model performance

During the implementation and testing of the models we ran into performance consistency issues with the HA-TCN and ATCN. In some runs they would perform well, in others they would produce validation set results that were several tens of times worse than the baseline of simply taking the stock price at timestep $t-1$. The first idea was that random weight initialization could be the cause of this inconsistency. After trying other initialization methods such as Xavier normal initialization, Xavier uniform initialization and He normal initialization, model performance was still inconsistent. The model should at the very least be able to match the baseline by using one of its inputs, yesterday's stock price. The only way it cannot use the input feature directly is if hierarchical attention is preventing the model from getting the input directly to the output layer. We modified the skip connection to

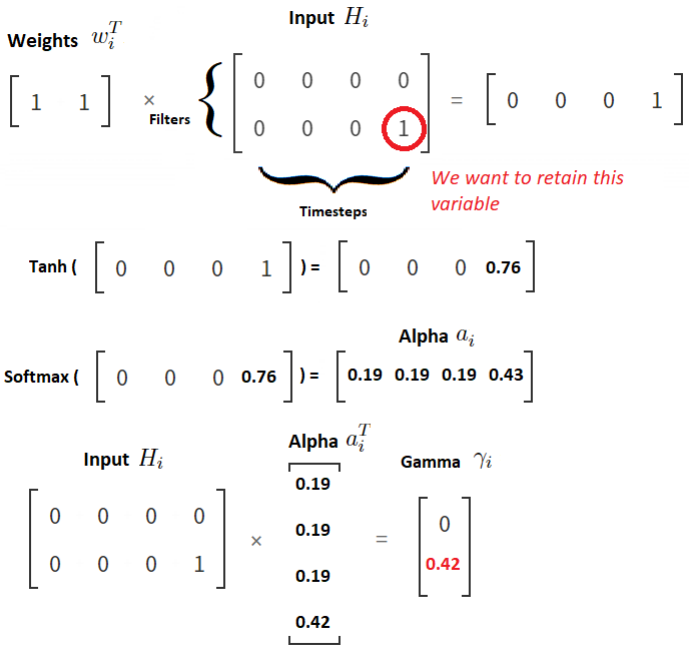

Figure 5: Example calculation of hierarchical attention, attempting to retain an input variable.

directly connect the model inputs to the output layer, and the model performance became consistent (see section 4.2 for the implementation details). This means that hierarchical attention was preventing the input from getting passed on to the output layer.

To figure out why hierarchical attention prevented this, we will take a look at a few examples. In figure 5, you can see a theoretical example of how hierarchical attention works. This example is designed with one goal: retain the input value in such a way that it can be transformed back to the original value with a single linear transformation in the output layer. All other input is set to zero to ensure that it is only the single input value that we end up seeing in γ. The calculations in figure 5 show that an input of 1 leads to γ = 0.42. Repeating this same calculation with different input numbers, we find that an input of 0.5 leads to γ = 0.34, an input of 0.25 leads to γ = 0.31 and an input of 0 leads to γ = 0. There is no linear transformation possible that can transform γ back to the original input in all these example cases. The impossibility of a linear transformation is because the softmax function itself is a non-linear activation function. Since the result is then multiplied with the input, it causes the input to be transformed in a non-linear way. It gets even more difficult if the input has other non-zero values in the same row, because softmax output is relative to other inputs in that row. The only

way a model could get around this limitation is to increase the weight that multiplies with the input to a very high amount, in which case softmax causes the relevant parameter in α to round up to 1. This would keep the input fully intact when multiplying. This is impossible however in this equation due to the tanh function that is applied before the softmax. The tanh can only output values between -1 and 1, meaning that the weights cannot get high enough to set the output of a softmax value to 1, unable to prevent getting the input transformed in a non-linear way. The linear transformation in the output layer cannot revert this change for all cases. Effectively, this means that there is no way for the HA-TCN and ATCN to retain the original input value without a skip connection, which causes the model performance to be inconsistent over multiple runs.

4 Experimental setup

In this study we will run 2 experiments. In both experiments we will predict stock market prices based on financial data. We will first describe the experiments, followed by a more detailed explanation of the dataset. Finally, we specify the implementation of the models used.

In the first and main experiment we predict the stock prices of 3 different individual companies and compare the performance of the TCAN, HA-TCN and ATCN to the performance of the generic TCN structure. This experiment will attempt to answer the main research question *'How effective is an attention based temporal convolutional network in forecasting individual stock prices?'*. Comparing the performance of the generic TCN to models with attention mechanisms added will enable us to track down how effective the attention part of the model is at increasing the performance.

In the second experiment, we will measure how much the performance of the TCAN, HA-TCN and ATCN changes when increasing or decreasing the number of layers. This will allow us to gain some understanding of how the attention mechanism could increase performance. Relatively higher performance increases at a lower number of layers could hint at attention being able to compensate for less computational power, while a performance increase at higher numbers of layers could show that the attention mechanism adds something to the TCN performance that extra layers cannot compensate for.

4.1 Dataset

We will use financial data available on Yahoo finance. The dataset has daily data from 1973 up until now, including the basic variables open price, closing price, highest price, lowest price, dividends, stock splits and trading

volume. Data from 2000 to 2014 will be used for training purposes, while data from 2015 up to but not including October 2021 will be used for testing.

The stocks we will be predicting for the experiments are Apple Inc. (NASDAQ: AAPL), Heineken (AMS: HEIA) and PostNL (AMS: PNL). These 3 companies are chosen because they are widely different: they are in different sectors, have different market cap sizes, different degrees of internationality and have had a varying degree of success. This makes this mix of companies ideal for testing the model architecture's ability to generalize.

In terms of prediction difficulty, Apple is expected to be the easiest to predict. All 3 of the stocks have undergone some drastic deviations around march 2020 due to covid striking fear into the stock markets. Apple however seems to be the stock with the least fluctuations and follows a long term trend upwards. The stock experienced a huge rise during the training data (from $0.44 to $28!) while the stock rose more slowly during the test data, with a temporary dip during covid followed by another big rise. Apart from the sudden covid fall, prediction should be easy relative to the other companies. Heineken is harder to predict due to having more short term fluctuations than Apple, but also tends to follow a slow upward trend in general. Out of the 3 companies, PostNL is expected to be the most challenging company to predict. There seems to be no clear long term trend going on and the price can fluctuate wildly on a day to day basis. It will be interesting to see if the models with an added attention mechanism will have an extra advantage with these fluctuations over the basic TCN.

4.2 Model implementations

We will be using 4 models in the experiments of this study. The baseline model is the TCN introduced by Bai et al. (2018). This follows the structure explained in section 2.

The second model is the TCAN, which uses the baseline TCN and adds an implementation of temporal attention (see section 3.2 for details). Note that this implementation differs from the TCAN by Hao et al. (2020) as the attention is simplified (also see section 3.2 for simplification changes) and we do not use the enhanced residual connection. Instead, we use the same residual connection as in the TCN.

The third model is the HA-TCN, also using the baseline TCN and implementing hierarchical attention (see section 3.1 for details). This implementation follows the structure proposed by Lin et al. (2019) with one exception: a skip connection was added from the input to the output layer. This was done because the model was inconsistent between runs, see section 3.4 for a more detailed explanation.

The fourth model is the ACTN, which is the novel architecture proposed by this study. The ACTN also uses the baseline TCN and adds both temporal and hierarchical attention (see section 3.3 for more details). Like the

HA-TCN, a skip connection is also added from the input to the output layer.

Despite all of these models being based on the TCN, there are some significant differences in what techniques were employed in the TCAN and HA-TCN papers compared to the original TCN. The original TCN paper had a single skip connection connecting the input to the output of a single block. The creators HA-TCN used no residual connection, while the creators of the TCAN used a modified version they dubbed the 'enhanced residual'. Since the main purpose of this study is to research the effect of attention in a TCN, we will keep the residual connection the same for both the TCN and TCAN. We use the skip connection of the original TCN. The plan was to also keep this the same for the HA-TCN and ATCN. This was not feasible however, due to the inconsistency of model performance. In essence, we found that hierarchical attention was unable to keep the original input intact, which meant a residual connection that also skips hierarchical attention was needed (see section 3.4 for a detailed explanation).

Weight normalization (Salimans et al., 2016) and channel dropouts are used in the original TCN paper but not used by the creators of the TCAN and HA-TCN. Due to the relatively small amount of data used in this study, channel dropout is not used. Weight normalization is used for all the convolutional layers in the models.

A grid search is used to find the best hyperparameter options for each model. The possible values for each model were selected from learning rate $\in [0.01, 0.001, 0.0001]$, filters $\in [10, 25, 50, 100]$, epochs $\in [300, 600, 1000]$ and the pair number of layers $\in [2, 3, 6]$ and kernel size $\in [43, 19, 3]$. The number of layers and kernel size pair was chosen to keep the receptive field at 253 in all settings, so models could not gain an edge through different receptive fields sizes (see section 2.2 for calculating the receptive field). This means only a kernel size of 43 could be chosen with 2 layers, 19 with 3 layers and 3 with 6 layers. The results of the grid search is found in table 1. During the grid search we found that for the HA-TCN, the optimal number of layers and kernel size pair varied greatly for each company. The HA-TCN used 2 layers and a kernel size of 43 for PostNL, 3 layers and a kernel size of 19 for Apple and 6 layers and a kernel size of 3 for Heineken. Note that the performance with different number of layers and kernel size is tested in experiment 2, so the numbers stated for these specific hyperparameters are only relevant for experiment 1.

5 Results

We evaluate the 4 models by using them to predict the stock prices of the 3 different companies mentioned in 4.1. All experiments reported in this

Model	Hyperparameters				
	Learning rate	**Filters**	**Epochs**	**Layers***	**Kernel size***
TCN	0,0001	100	1000	3	19
HA-TCN	0,001	25	1000	Varied**	Varied**
TCAN	0,001	25	1000	2	43
ATCN	0,001	25	1000	3	19

Table 1: Resulting hyperparameters of the grid search. Learning rate, filters and epoch numbers are used in experiment 1 and 2. *These layer amounts and kernel sizes are only used for experiment 1 and vary in experiment 2. **The HATCN uses a different number of layers and kernel size for each company, see section 4.2

.

section use the model implementations stated in section 4.2, with only the hyperparameters varying for each model (see table 1 for the hyperparameters used). We also compare the models to the baseline, which is simply using the stock price of the previous day as a prediction.

The results of the comparison between the models on different stocks is shown in table 2. All models perform significantly better than the baseline. The results show that on average, the TCAN was the best performing model. The HA-TCN and ATCN had worse results than the regular TCN when predicting Apple and PostNL, while performing well on predicting Heineken.

In the second experiment, We evaluate how the performance of the 3 attention models on the financial dataset changes when different number of layers are used. To compensate for the effects this has on the receptive field, kernel size is adjusted in order to keep the receptive field relatively consistent. Note that the 4 and 5 layer models have a receptive field of 241 and 249 days respectively, while the others have a receptive field of 253 (see section 2.2 for the receptive field calculation).

The results in table 3 show that the TCAN and ATCN perform significantly worse with a higher number of layers. The HA-TCN shows worse results with 3, 4 or 5 layers, but when using 2 layers the performance is close to that of using 6 layers.

Note that Apple was predicted to be the easiest company to predict. If we compare the baseline in table 2 with the best model, we can see that Apple was in fact the hardest company to predict, with a mean square error roughly 7 times lower than the baseline. PostNL was easier with a mean square error roughly 10 times lower than the baseline and Heineken was the easiest, with a mean square error of roughly 25 times lower than the baseline.

Company	Models				
	Baseline	**TCN**	**HA-TCN**	**TCAN**	**ATCN**
Apple	0.01020	0.00206	0.00338	**0.00139**	0.00418
Heineken	0.01120	0.00087	0.00075	0.00056	**0.00044**
PostNL	0.00390	**0.00041**	0.00054	0.00051	0.00060
Average	0.00843	0.00111	0.00166	**0.00082**	0.00174

Table 2: Performance comparison of each model and the baseline on predicting the 3 stocks. Performance is measured through the mean squared error.

Layers	Kernel size	Models		
		HA-TCN	**TCAN**	**ATCN**
2	**43**	0.00078	**0.00056***	0.00130
3	**19**	0.00124	0.00069	**0.00044***
4	**9**	0.00233	0.00600	0.00130
5	**5**	0.00110	0.00179	0.00370
6	**3**	**0.00075***	0.00510	0.00736

Table 3: Performance comparison of the attention models with different number of layers used. Models with a * were the exact variations used in experiment 1. Performance is measured through the mean squared error.

5.1 Discussion

Hierarchical attention seemed to perform relatively poorly in experiment 1 when predicting Apple. When it comes to predicting other companies, the score was roughly the same as the TCN. Note however that this score was achieved with 4 times less filters than the TCN (see table 1 for the hyperparameters), significantly cutting back the amount of trainable parameters in the model. In the original HA-TCN paper by Lin et al. (2019), the HA-TCN was used to highlight a specific period of time in order to more accurately diagnose patients. It may be that for predicting Apple, highlighting a specific period of time is of lesser importance or may even harm the generalizability of the model. This could explain why the HA-TCN and ATCN performed worse than the TCN. During the hyperparameter optimization we noticed that both the HA-TCN and ATCN tended to perform worse when more filters were used. This seems to suggest that models using hierarchical attention are more prone to overfitting when using more filters compared to the regular TCN.

In the second experiment, the HA-TCN had the least difference between the results when using different numbers of layers. In the original HA-TCN paper by Lin et al. (2019), the authors found that the number of layers only has minimal effect on the performance of their model. The difference in this

study is still larger than in the original paper however. The best performing settings were 2 and 6 layers. It may be that the HA-TCN has two different best strategies: generalizing the pattern with a large kernel size and low number of layers, or picking up on specific smaller patterns with a very small kernel size and high number of layers. Unlike temporal attention, since there is no leakage between different filters for hierarchical attention, this could potentially make the model more adept at picking up small specific patterns.

Just like hierarchical attention, temporal attention seems to be able to capture important information easily while using a less filters. Unlike the models with hierarchical attention however, the TCAN seems to perform well predicting any company in experiment 1. When predicting Apple, the hardest company, it performs the best out of all models. Unlike the HA-TCN and ATCN, the model did not perform significantly worse with more filters added during the hyperparameter optimization either. This seems to indicate that the model is quite robust.

In the second experiment, the results of the TCAN varied wildly between lower and higher number of layers. It seems that temporal attention gets significantly worse with a higher numbers of layers. In temporal attention, information doesn't leak between different timesteps, but it does between the filters. It could be that this leakage of information between filters prevents the model from reliably picking up on smaller patterns, which is one of the strengths of picking a smaller kernel size and a higher number of layers.

The ACTN also used significantly less filters than the TCN, which makes it seem that all attention models used in this study are able to capture important information more easily. Interestingly, while the ATCN performed the worst of all 4 models in the first experiment, it was the best model for predicting Heineken. This was the easiest company to predict, so this supports the idea that when there is more to be gained from capturing important information, attention models perform better. The reverse also seems true for hierarchical attention when you look at Apple, the hardest company to predict. Both the ATCN and HA-TCN performed quite poorly compared to the regular TCN, giving the impression that both models pick up specific types of information, but do not always generalize well.

In the second experiment, the results of the ATCN were similar to that of the TCAN, but the error increases more steadily with a higher number of layers. The ATCN gets the most extra parameters with each layer added, which makes overfitting a strong suspect. To test this hypothesis, the ATCN was run again with the exact same settings on 6 layers, but with 10 filters instead of 25. The result of this test was a mean square error of 0.00110, which is significantly better than the original result of 0.00736. This is additional evidence for the ATCN overfitting on higher number of layers.

Apple turned out to be the hardest stock to predict instead of the easiest. This could be because many investing companies create models for predicting Apple, due to the company being large, popular and internation-

ally successful. Heineken is the easiest, but it is hard to pinpoint an exact reason. One possible reason could be that the stock price patterns changed the least during covid compared to the training set.

In general, due to the lower number of filters required for the models with attention, training times were significantly lower. HA-TCN was almost twice as fast as the regular TCN, while the TCAN and ATCN were also about a third faster than the TCN. For the HA-TCN, this creates an interesting trade-off where the model sacrifices some performance on average to be able to train faster. The TCAN however just seems more solid all-around, training relatively fast while having similar or better performance than the regular TCN. It also seems that for all attention models trained on this dataset, a lower number of layers is better or equal to a higher number of layers. Since training time also increases with higher number of layers, a lower number of layers is preferred.

6 Conclusion

We have presented an empirical evaluation of the effectiveness of attention based temporal convolutional networks in forecasting individual stock prices. In order to achieve this, we tested the performance of the TCN, HA-TCN, TCAN and a novel architecture dubbed the ATCN on three different individual stocks. The experimental results show that all models with attention mechanisms added are more easily able to capture information, needing less layers and filters to perform at a comparable level to the TCN. Since the attention models needed a lower number of filters and layers, parameter size and training time are greatly reduced compared to the TCN. Models that use hierarchical attention (HA-TCN and ATCN) performed worse on average due to issues predicting Apple, but seemed to perform much better when the stock price was more easily predicted from financial data used in this study, as was the case with the other 2 companies. The HA-TCN in particular seemed adept at picking up small patterns with a higher number of layers used and a lower kernel size. The TCAN was the best model on average and seemed to achieve this by being able to generalize well with a low number of layers. The performance dropped drastically on higher number of layers and lower kernel size, suggesting that temporal attention is less useful for picking up small patterns. The ATCN was the worst performer on average mainly due to issues with predicting the hardest company (Apple). The model was also more prone to overfitting, as the performance dropped significantly on a higher number of layers while performance increased when the filter size was lowered.

These results suggest that using attention models is preferred due to similar results and reduced parameter size and training time. Models with hierarchical attention are seemingly better at capturing smaller patterns, while

models with temporal attention are to be used when more general patterns are to be extracted. While the ATCN should in theory benefit from both, it performed relatively poorly on this dataset.

There are some limitations however, as the dataset used for these models was on the small side and exclusively contained financial data. Further research has to show if these results hold up on larger, more complex datasets and can be generalized to other types of data. Another limitation of this study is that the covid correction took place in the timeframe used for the test set. We do not know if the covid correction differs significantly from the other stock market corrections in this dataset, so the results might be affected.

The performance of the ATCN shows some potential but needs to be tested further on more complex datasets than the one used in this study. Since temporal attention and hierarchical attention seem to be better at different aspects, using both of them in a model in forms other than the ATCN might also yield further improvements. One possibility is to combine these into an ensemble model of both the TCAN and the HATCN. In theory, this could allow the HA-TCN to focus on small patterns while the TCAN focuses on the more broad general patterns, creating the possibility for the best of both worlds.

7 References

Bai, Shaojie, J. Zico Kolter, and Vladlen Koltun. “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling.” *arXiv preprint arXiv:1803.01271* (2018).

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate.” *arXiv preprint arXiv:1409.0473* (2014).

Bengio, Yoshua, Patrice Simard, and Paolo Frasconi. “Learning long-term dependencies with gradient descent is difficult.” *IEEE transactions on neural networks 5.2* (1994): 157-166.

Deng, Shumin, et al. “Knowledge-driven stock trend prediction and explanation via temporal convolutional network.” *Companion Proceedings of The 2019 World Wide Web Conference*. 2019.

Gao, Lianli, et al. “Video captioning with attention-based LSTM and semantic consistency.” *IEEE Transactions on Multimedia* 19.9 (2017): 2045-2055.

Hao, Hongyan, et al. “Temporal convolutional attention-based network for sequence modeling.” *arXiv preprint arXiv:2002.12530* (2020).

He, Kaiming, et al. “Deep residual learning for image recognition.” *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

Hu, Zexin, Yiqi Zhao, and Matloob Khushi. “A survey of forex and stock price prediction using deep learning.” *Applied System Innovation 4.1* (2021): 9.

Mnih, Volodymyr, Nicolas Heess, and Alex Graves. “Recurrent models of visual attention.” *Advances in neural information processing systems*. 2014.

Nair, Vinod, and Geoffrey E. Hinton. “Rectified linear units improve restricted boltzmann machines.” *Icml.* 2010.

Lin, Lei, et al. “Medical Time Series Classification with Hierarchical Attention-based Temporal Convolutional Networks: A Case Study of Myotonic Dystrophy Diagnosis.” *CVPR Workshops*. 2019.

Long, Jonathan, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation.” *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

Oord, Aaron van den, et al. “Wavenet: A generative model for raw audio.” *arXiv preprint arXiv:1609.03499* (2016).

Rosin, Guy D., and Kira Radinsky. “Temporal Attention for Language Models.” *arXiv preprint arXiv:2202.02093* (2022).

Salimans, Tim, and Durk P. Kingma. “Weight normalization: A simple reparameterization to accelerate training of deep neural networks.” *Advances in neural information processing systems 29* (2016): 901-909.

Srivastava, Nitish, et al. “Dropout: a simple way to prevent neural networks from overfitting.” *The journal of machine learning research 15.1* (2014): 1929-1958.

Vaswani, Ashish, et al. “Attention is all you need.” *Advances in neural information processing systems*. 2017.

Wang, Yequan, et al. “Attention-based LSTM for aspect-level sentiment classification.” *Proceedings of the 2016 conference on empirical methods in natural language processing*. 2016.

Yu, Fisher, and Vladlen Koltun. “Multi-scale context aggregation by dilated convolutions.” *arXiv preprint arXiv:1511.07122* (2015).