Farewell RNNs, Welcome TCNs

How Temporal Convolutional Networks are moving in favor of Sequence Modeling — Stock Trend Prediction.

Bryan Tan Aug 31 · 16 min read ★

Photo by Aditya Vyas on Unsplash

Disclaimer: this article assumes that readers possess preliminary knowledge behind the model intuition and architecture of LSTM neural networks.

Overview

1. ***Background of Deep Learning in FTS***
2. ***Noteworthy Data Preprocessing Practices for FTS***
3. ***Temporal Convolutional Network Architecture***
4. ***Example Application of Temporal Convolutional Networks in FTS***

- ***Knowledge-Driven Stock Trend Prediction and Explanation via TCN***

1. Background

Financial Time Series (FTS) modelling is a practice with a long history which first revolutionised algorithmic trading in the early 1970s. The analysis of FTS was divided into two categories: fundamental analysis and technical analysis. Both these practices were put into question by the Efficient Market Hypothesis (EMH). The EMH, highly disputed since its initial publication in 1970, hypothesizes that stock prices are ultimately unpredictable. This has not constrained research attempting to model FTS through the use of linear, non-linear and ML-based models, as mentioned hereafter.

Due to the nonstationary, nonlinear, high-noise characteristics of financial time series, traditional statistical models have difficulty predicting them with high precision. Hence, increased attempts in recent years are being made to apply deep learning to stock market forecasts, though far from perfection. To list a mere few:

Lin *et al.* proposed a method to predict stocks using a support vector machine to establish a two-part feature selection and prediction model and proved that the method has better generalization than conventional methods.

2014

Wanjawa *et al.* proposed an artificial neural network using a feed-forward multilayer perceptron with error backpropagation to predict stock prices. The results show that the model can predict a typical stock market.

2017

Enter LSTM — a surge in studies concerning application of LSTM neural networks to the time series data.

Zhao *et al.*, a time-weighted function was added to an LSTM neural network, and the results surpassed those of other models.

2018

Zhang *et al.* later combined convolutional neural network (CNN) and recurrent neural network (RNN) to propose a new architecture, the deep and wide area neural network (DWNN). The results show that the DWNN model can reduce the predicted mean square error by 30% compared to the general RNN model.

Ha *et al.*, CNN was used to develop a quantitative stock selection strategy to determine stock trends and then predict stock prices using LSTM to promote a hybrid neural network model for quantitative timing strategies to increase profits.

Jiang *et al.* used an LSTM neural network and RNN to construct models and found that LSTM could be better applied to stock forecasting.

2019

Jin *et al.* added investor sentiment tendency in model analysis and introduced empirical modal decomposition (EMD) combined with LSTM to obtain more accurate stock forecasts. The LSTM model based on the attention mechanism is common in speech and image recognition but is rarely used in finance.

Radford *et al.* Precursor to the now hot-stock, **GPT-3**, GPT-2's goal is to design a multitask learner, and it utilizes a combination of pretraining and supervised finetuning to achieve more flexible forms of transfer. Therefore it has 1542M parameters, much bigger than other comparative models.

Shumin *et al*. A knowledge-driven approach using *Temporal Convolutional Network (*KDTCN) for stock trend prediction and explanation. They first extracted structured events from financial news and utilize knowledge graphs to obtain event embeddings. Then, combine event embeddings and price values together to forecast stock trend. Experiments demonstrate that this can (i) react to abrupt changes much faster and outperform state-of-the-art methods on stock datasets. (This will be the focal point of this article.)

2020

Jiayu *et al.* and Thomas *et al.* proposed hybrid attention networks to predict stock trend based on the sequence of recent news. LSTMs with attention mechanisms outperforms conventional LSTMS as it prevents long-term dependencies due to its unique storage unit structure.

Hongyan *et al.* proposed an exploratory architecture referred to *Temporal Convolutional Attention-based Network (TCAN)* which combines temporal convolutional network and attention mechanism. TCAN includes two parts, one is *Temporal Attention (TA)* which captures relevant features inside the sequence, the other is *Enhanced Residual (ER)* which extracts the shallow layer's important information and transfers to deep layers.

The timeline above is merely meant to provide a glimpse into the historical context of FTS in deep learning but not downplay on the significant work contributed to by the rest of the sequence model academia during similar time periods.

However, a word of caution is worth mentioning here. It might be the case that academic publications in the field of FTS forecasting are often misleading. Many FTS forecasting papers tend to inflate their performance for recognition and overfit their models due to the heavy use of simulators. Many of the performances claimed in these papers are difficult to replicate as they fail to generalise for future changes in the particular FTS being forecast.

2. Noteworthy Data Preprocessing Practices for FTS

2.1 Denoising

Financial time series data — especially stock prices, constantly fluctuate with seasonality, noise and autocorrection. Traditional methods of forecasting use moving averages and differencing to reduce the noise for forecasting. However, FTS is conventionally non-stationary and exhibits the overlapping of useful signals and noise, which makes traditional denoising ineffective.

Wavelet analysis has led to remarkable achievements in areas such as image and signal processing. With its ability to compensate for the shortcomings of Fourier analysis, it has gradually been introduced in the economic and financial fields. **The wavelet transform has unique advantages in solving traditional time series analysis problems as it can decompose and reconstruct financial time series data from a different time and frequency domain scales**.

Wavelet transform essentially uses multi-scale characteristics to denoise the dataset, effectively separating the useful signal from the noise. In a paper by *Jiayu Qiu, Bin Wang, Changjun Zhou,* they used the coif3 wavelet function with three decomposition layers, and evaluate the effect of the wavelet transform by its signal-to-noise ratio (SNR) and root mean square error (RMSE). **The higher the SNR and the smaller the RMSE, the better the denoising effect of the wavelet transform**:

$$\mathrm{SNR} = 10\log\left[\frac{\sum_{j=1}^{N} x_j^{\,2}}{\sum_{j=1}^{N}\left(x_j - \hat{x}_j\right)^2}\right].$$

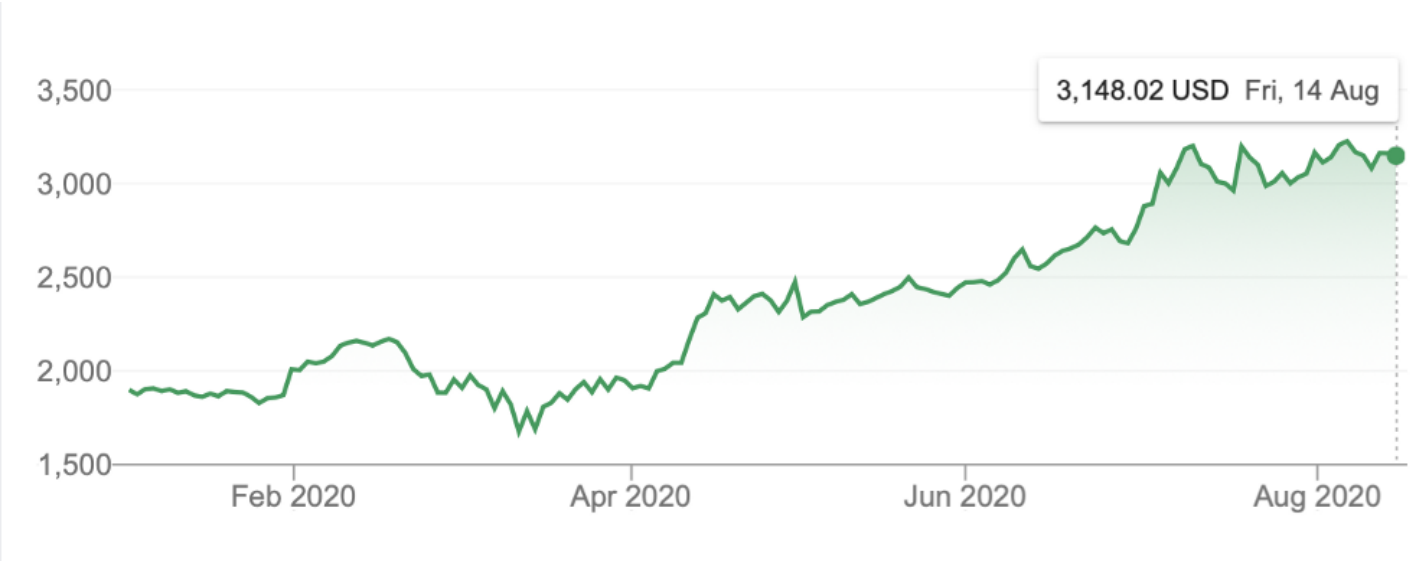
Jiayu Qiu1, Bin Wang, Changjun Zhou

2.2 Data Shuffling

In FTS, the choice of which piece of data to use as the validation set is not trivial. Indeed, there exist a myriad of ways of doing this which must be carefully considered for stock indices of varying volatility.

The fixed origin method is the most naive and common method used. Given a certain split size, the start of the data is the training set and the end is the validation set. However, this is a particularly rudimentary method to choose, especially for a high-

growth stock like Amazon. The reason why this is the case is that Amazon's stock price starts off with low volatility and, as the stock grows, experiences increasingly volatile behaviour.

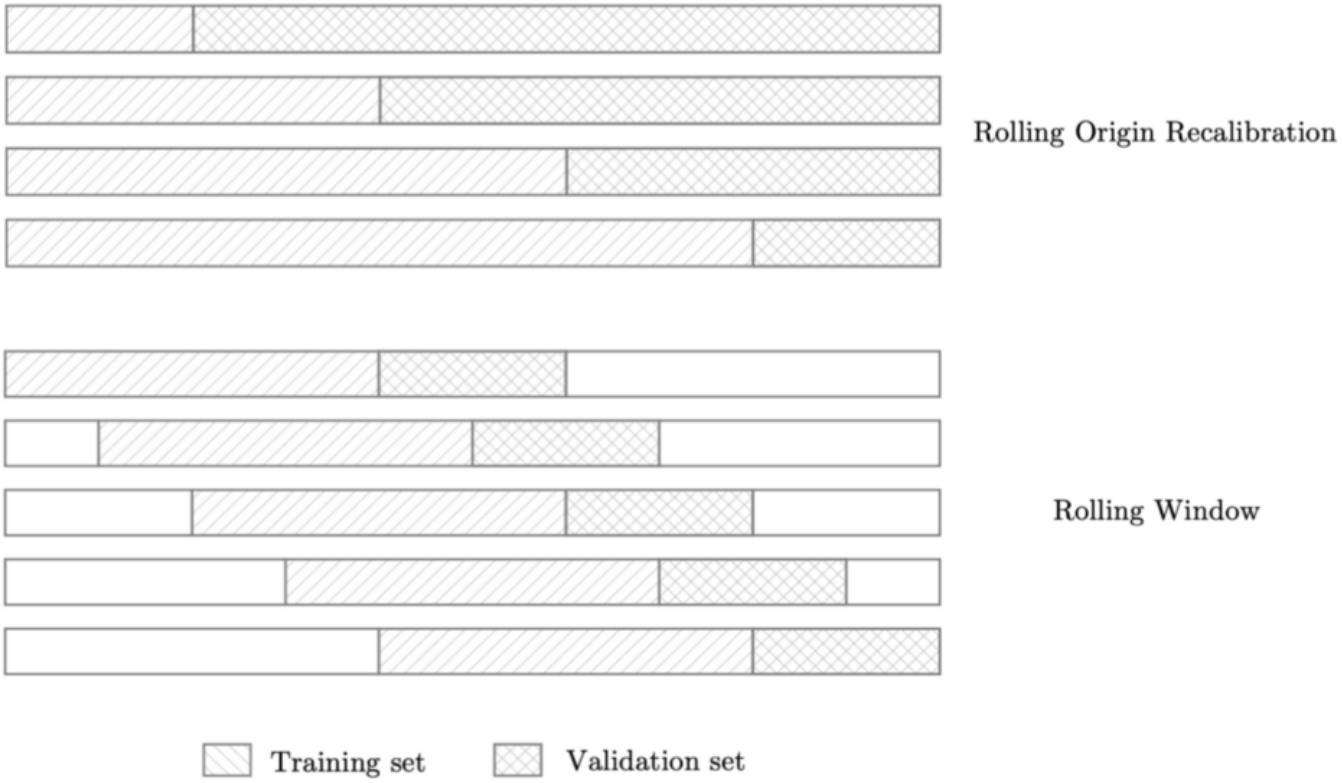


AMZN Year-to-Date prices (source: Google Finance)

We would therefore be training a model on low volatility dynamics and expect it to deal with unseen high volatility dynamics for its predictions. This has indeed shown itself to be difficult and come at a cost in performance for these types of stocks. Therefore our benchmark for validation loss and performance may be misleading if we only consider this. However, for stocks like Intel that are more constant in their volatility (pre-COVID crisis), this method is reasonable.

The rolling origin recalibration method is slightly less vulnerable than fixed origin as it allows the validation loss to be computed by taking the average of various different splits of the data to avoid running into unrepresentative issues with high volatility timeframes.

Finally, the **rolling window method** is usually one of the most useful methods as it is particularly used for FTS algorithms being run for long timeframes. Indeed, this model outputs the average validation error of multiple rolling windows of data. This means the final values we get are more representative of recent model performance, as we are less biased by strong or poor performance in the distant past.



Shuffling techniques visualized (Thomas Hollis, Antoine Viscardi, Seung Eun Yi, 2018)

A study done by Thomas Hollis, Antoine Viscardi, Seung Eun Yi indicated that both rolling window (RW) and rolling origin recalibration (ROR) describe very slightly better performances (58% and 60%) than that of the simple fixed origin method. This suggests that for volatile stocks like Amazon, using these shuffling methods would be inevitable.

Loss (RW)	Accuracy (RW)	Loss (ROR)	Accuracy (ROR)
0.000692	0.538	0.000810	0.555
0.000693	0.530	0.000825	0.571
0.000725	0.575	0.000978	0.607
0.000755	0.551	0.000989	0.563
0.000780	0.514	0.001001	0.579
0.000788	0.583	0.001014	0.538

Performance comparison of shuffling methods

3. Temporal Convolutional Network

Temporal Convolutional Networks, or simply TCN, is a variation of Convolutional Neural Networks for sequence modelling tasks, by combining aspects of RNN and CNN architectures. Preliminary empirical evaluations of TCNs have shown that a simple convolutional architecture outperforms canonical recurrent networks such as LSTMs

across a diverse range of tasks and datasets while demonstrating longer effective memory.

The distinguishing characteristics of TCNs are:

1. **The convolutions in the architecture are causal, meaning that there is no information "leakage" from future to past.**
2. **The architecture can take a sequence of any length and map it to an output sequence of the same length, just as with an RNN. TCNs possess very long effective history sizes (i.e., the ability for the networks to look very far into the past to make a prediction) using a combination of very deep networks (augmented with residual layers) and dilated convolutions.**

3.1 Model Architecture Overview

3.1.1 Causal Convolutions

As mentioned above, the TCN is based upon two principles: the fact that the network produces an output of the same length as the input, and the fact that there can be no leakage from the future into the past. To accomplish the first point, the TCN uses a 1D fully-convolutional network (FCN) architecture, where each hidden layer is the same length as the input layer, and zero padding of length (kernel size − 1) is added to **keep subsequent layers the same length as previous ones**. To achieve the second point, the TCN uses *causal convolutions*, convolutions where output at time *t* is convolved only with elements from time *t* and earlier in the previous layer.

To put it simply: **TCN = 1D FCN + causal convolutions.**

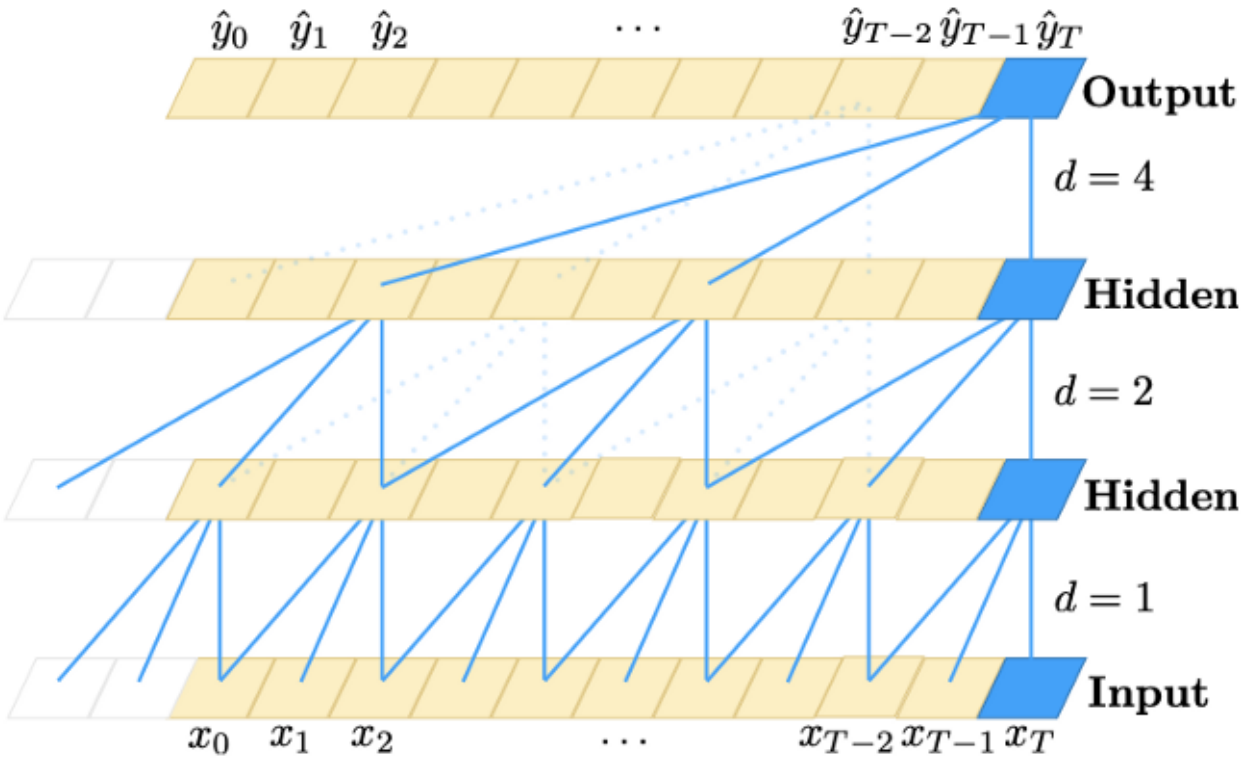
3.1.2 Dilated Convolutions

A simple causal convolution is only able to look back at a history with size linear in the depth of the network. This makes it challenging to apply the aforementioned causal convolution on sequence tasks, especially those requiring a longer history. The solution implemented by *Bai, Kolter and Koltun (2020)*, was to employ dilated convolutions that enable an exponentially large receptive field. More formally, for a 1-D sequence input $x \in R^n$ and a filter $f:\{0,\ldots,k-1\}\rightarrow R$, the dilated convolution operation F on element s of the sequence is defined as:

$$k-1$$

$$F(s) = (\mathbf{x} *_d f)(s) = \sum_{i=0} f(i) \cdot \mathbf{x}_{s-d \cdot i}$$

where d is the dilation factor, k is the filter size, and $s - d \cdot i$ accounts for the direction of the past. Dilation is thus equivalent to introducing a fixed step between every two adjacent filter taps. When $d = 1$, a dilated convolution reduces to a regular convolution. Using larger dilation enables an output at the top level to represent a wider range of inputs, thus effectively expanding the receptive field of a ConvNet.



A dilated causal convolution with dilation factors d = 1, 2, 4 and filter size k = 3. The receptive field is able to cover all values from the input sequence.

3.1.3 Residual Connections

Residual blocks effectively allow layers to learn modifications to the identity mapping rather than the entire transformation, which has repeatedly been shown to benefit very deep networks.

Since a TCN's receptive field depends on the network depth n as well as filter size k and dilation factor d, stabilization of deeper and larger TCNs becomes important.

3.2 Pros & Cons

Several advantages of using TCNs for sequence modelling:

Parallelism. Unlike in RNNs where the predictions for later timesteps must wait for their predecessors to complete, convolutions can be done in parallel since the same filter is used in each layer. Therefore, in both training and evaluation, a long input sequence can be processed as a whole in TCN, instead of sequentially as in RNN.

Flexible receptive field size. A TCN can change its receptive field size in multiple ways. For instance, stacking more dilated (causal) convolutional layers, using larger dilation factors, or increasing the filter size are all viable options. TCNs thus afford better control of the model's memory size and are easy to adapt to different domains.

Stable gradients. Unlike recurrent architectures, TCN has a backpropagation path different from the temporal direction of the sequence. TCN thus avoids the problem of exploding/vanishing gradients, which is a major issue for RNNs (and which led to the development of LSTM and GRU).

Low memory requirement for training. Especially in the case of a long input sequence, LSTMs and GRUs can easily use up a lot of memory to store the partial results for their multiple cell gates. However, in a TCN the filters are shared across a layer, with the backpropagation path depending only on network depth. Therefore in practice, it was found that gated RNNs are likely to use up to a multiplicative factor more memory than TCNs.

Variable length inputs. Just like RNNs, which model inputs with variable lengths in a recurrent way, TCNs can also take in inputs of arbitrary lengths by sliding the 1D convolutional kernels. This means that TCNs can be adopted as drop-in replacements for RNNs for sequential data of arbitrary length.

Two notable disadvantages to using TCNs:

Data storage during evaluation. TCNs need to take in the raw sequence up to the effective history length, thus possibly requiring more memory during evaluation.

Potential parameter change for a transfer of domain. Different domains can have different requirements on the amount of history the model needs in order to predict. Therefore, when transferring a model from a domain where only little memory is needed (i.e., small k and d) to a domain where much longer memory is required (i.e., much larger k and d), TCN may perform poorly for not having a sufficiently large receptive field.

3.3 Benchmark

Evaluation of TCNs and recurrent architectures on typical sequence modelling tasks that are commonly used to benchmark RNN variants (Bai, Kolter and Koltun, 2020)

Executive summary:

The results strongly suggest that the generic TCN architecture *with minimal tuning* outperforms canonical recurrent architectures across a broad variety of sequence modelling tasks that are commonly used to benchmark the performance of recurrent architectures themselves.

4. Knowledge-Driven Stock Trend Prediction and Explanation via TCN

4.1 Background

Most of the deep neural networks in stock trend prediction have two common drawbacks: (i) **current methods are not sensitive enough to abrupt changes of stock trend**, and (ii) **forecasting results are not interpretable for humans**. To address these two problems, Deng et al., 2019 proposed a novel *Knowledge-Driven Temporal Convolutional Network (KDTCN)* for stock trend prediction and explanation, by incorporating background knowledge, news events and price data into deep prediction models, to tackle the problem of stock trend prediction and explanation with abrupt changes.

To address the problem of prediction with abrupt changes, events from financial news are extracted and structurized into event tuples, e.g., “Britain exiting from EU ” is represented as (*Britain*, *exiting from*, *EU*). Then entities and relations in event tuples

are linked to KGs, such as Freebase and Wikidata. Secondly, structured knowledge, textual news, as well as price values are vectorized respectively, and then concatenated together. Finally, feed these embeddings into a TCN-based model.

Experiments demonstrate that KDTCN can (i) react to abrupt changes much faster and outperform state-of-the-art methods on stock datasets, as well as (ii) facilitate the explanation of prediction particularly with abrupt changes.

Furthermore, based on prediction results with abrupt changes, to address the problem of making explanations, the effect of events are visualized by presenting the linkage among events with the use of Knowledge Graphs (KG). By doing so, we can make explanations of (i) how knowledge-driven events influence the stock market fluctuation in different levels, and (ii) how knowledge helps to associate events with abrupt changes in stock trend prediction.

Note the section below merely summarizes into a broad overview of the paper Shumin et al., if you could refer to the paper if you seek further technical details.

4.2 Model Architecture Overview

The fundamental TCN model architecture mentioned here is derived from *Section 3* above —a generic TCN architecture consisting of causal convolutions, residual connections and dilated convolutions.

The overview of KDTCN architecture is shown below:

Illustration of the KDTCN framework

Original model inputs are **price values** X, **news corpus** N , and **knowledge graph** G. The price values are normalized and mapped into the price vector, denoted by

where each vector pt represents a real-time price vector on a stock trading day t, and T is the time span.

As for news corpus, pieces of news are represented as event sets, ε; then, structured into event tuple $\boldsymbol{e = (s, p, o)}$, where $\boldsymbol{p}$ is the action/predicate, $\boldsymbol{s}$ is the actor/subject and $\boldsymbol{o}$ is the object on which the action is performed; then, each item in the event tuples is linked to KG, correspond to entities and relations in KG; lastly, event embeddings V are obtained by training both event tuples and KG triples. A more detailed of this process is documented in Shumin *et al*.

Finally, event embeddings, combined with price vectors are input into a TCN-based model.

4.2.1 Datasets & Baselines

Datasets:

1. **Time-series Price Data** X: The price dataset of daily value records of DJIA index

2. **Textual News Data *N*:** News dataset composed of historical news headlines from Reddit WorldNews Channel (top 25 posts based on votes).

3. **Structured Knowledge Data *G*:** A sub-graph constructed from the structured data of two commonly used open knowledge graphs for research — Freebase and the Wikidata.

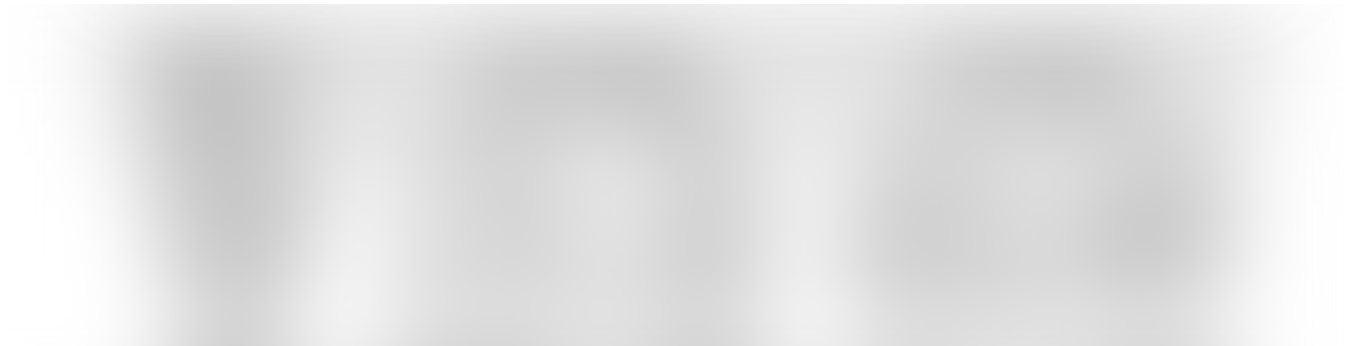
Baselines:

Baseline models with different inputs. In the first column, prefix WB means word embeddings, EB means event embeddings, PV means the price vector, and KD means knowledge-driven. Note that **event embedding (a)** and **event embedding (b)** denote event embedding without and with KG respectively.

4.4 Prediction Evaluation

Performance of KDTCN was benchmarked in three progressive aspects: (i) evaluation of basic TCN architecture, (ii) influence of different model inputs with TCN, and (iii) TCN-based model performance for abrupt changes.

Basic TCN Architecture:



Stock trend prediction results over the DJIA index dataset with different basic prediction models.

Note that all experiments reported in this part are only input with **price values**.

TCN greatly outperforms baseline models on the stock trend prediction task. TCN achieves much better performance than either traditional ML models (ARIMA), or deep neural networks (such as LSTM and CNN), indicating that TCN has more obvious advantages in sequence modeling and classification problems.

Different Model Inputs with TCN:

Stock trend prediction results over the overall DJIA index dataset with different inputs on TCN-based models.

As seen, WB-TCN and EB-TCN both get better performance than TCN, indicating **textual information** helps to improve forecasting.

KDTCN gets both the highest accuracy and F1 scores, and such a result demonstrates the validity of model input integration with structured knowledge, financial news, and price values.

Model Performance for Abrupt Changes:

Stock trend prediction results over the local DJIA index dataset of abrupt changes, with different model inputs.

It was observed that models with knowledge-driven event embedding input, such as KDEB-TCN and KDTCN, can greatly outperform numerical-data-based and textual-data-based models. These comparison results indicate that knowledge-driven models have advantages in reacting to abrupt changes in the stock market swiftly.

Additional note on how the degree of stock fluctuation is quantified below.

First, get time intervals of abrupt changes by figuring out the difference of stock fluctuation degree D(*fluctuation)* between two adjacent stock trading days

where x at time t denotes the stock price value on the stock trading day t. Then the difference of fluctuation degree C is defined by:

If |Ci | exceeds a certain threshold, it can be considered that the stock price abruptly changes at the *i*th day.

4.1.4 Explanation for Predictions

Explanations for why knowledge-driven events are common sources of abrupt changes to human without ML expertise are accomplished in two aspects: (i) **visualizing effects of knowledge-driven events** on prediction results with abrupt changes, and (ii) **retrieving background facts of knowledge-driven events by linking the events to external KG**.

Effect Visualization of Events:

The prediction result in the figure below is that trend of DJIA index will drop. Note that the bars of the same colour have the same event effect, the height of bars reflects the degree of effects, and the event popularity declines from left to right. Intuitively, events with higher popularity should have greater effects on stock trend prediction with abrupt changes, but not always.

Examples of event effect on stock trend prediction

Nearly all other events with negative effect are related to these two events, *e.g.*, (*British Pound, drops, nearly 5%*) and (*Northern Ireland, calls for poll on United Ireland*).

Although there are also some events have positive effects of predicting stock trend to rise, and have high popularity, *i.e.*, (*Rich, Getting, Richer*), the total effect is negative. Therefore, abrupt changes of the stock index fluctuation can be viewed as the combined result of effects and popularity of events.

Visualization of Event Tuples Linked to KG:

Illustration of triples in KG linked to events

First, the event tuples with great effects or high popularity in stock trend movements were searched. Then, backtrack to the news texts containing these events. Finally, retrieve associated KG triples linked to event tuples by entity linking. In the above figure, each event tuple is marked in blue, and entities in it are linked to KG.

These listed event tuples, such as (*Britain, exiting from, EU*), (*United Kingdom, votes to leave, European Union*), (*British Pound, drops, nearly 5%*), (*J. K. Rowling, leads the charge for, Scottish independence*), and (*Northern Ireland, calls for poll on United Ireland*), are not strongly relevant literally. However, with the linkage to KG, they can establish an association with each other, and strongly related to events of Brexit and EU Referendum. By incorporating explanations of event effects, it could be justified that knowledge-driven events are common sources of abrupt changes.

5. Conclusion

The preeminence enjoyed by recurrent networks in sequence modelling may be largely a vestige of history. Until recently, before the introduction of architectural elements such as dilated convolutions and residual connections, convolutional architectures were indeed weaker. The recent academic research has indicated that with these elements, a simple convolutional architecture is more effective across diverse sequence modelling tasks than recurrent architectures such as LSTMs. Due to the comparable clarity and simplicity of TCNs, it was proposed in *Bai, S., Kolter, J. and Koltun, V., 2020,* that convolutional networks should be regarded as a natural starting point and a powerful toolkit for sequence modelling.

Furthermore, as seen in the application of TCNs in stock trend prediction above, by incorporating news event and knowledge graphs, TCNs could significantly outperform canonical RNNs.

Citations and References

[1] Hollis, T., Viscardi, A. and Yi, S. (2020). *"A Comparison Of Lstms And Attention Mechanisms For Forecasting Financial Time Series".*

[2] Qiu J, Wang B, Zhou C. (2020). *"Forecasting stock prices with long-short term memory neural network based on attention mechanism".*

[3] Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. (2020). *"Neural Machine Translation By Jointly Learning To Align And Translate".*

[4] Bai, S., Kolter, J. and Koltun, V., 2020. *"An Empirical Evaluation Of Generic Convolutional And Recurrent Networks For Sequence Modeling".*

[6] Deng, S., Zhang, N., Zhang, W., Chen, J., Pan, J. and Chen, H., 2019. *"Knowledge-Driven Stock Trend Prediction and Explanation via Temporal Convolutional Network".*

[5] Hao, H., Wang, Y., Xia, Y., Zhao, J. and Shen, F., 2020. *"Temporal Convolutional Attention-Based Network For Sequence Modeling".*


Download on the
App Store

GET IT ON
Google Play