Transformers for time series forecasting: How do different positional encodings effect the performance if seasonalities are present

By Tarek Donker, h1253172

Table of Contents

1 Abstract

Transformers for time series forecasting are rising in research interest and achieve state-of-the-art results on many benchmarks. But while the supplementary research of transformers for Natural Language Processing is already built up, for time series forecasting it lacks behind. The positional encoding is an essential component of any transformer and is heavily researched for language tasks. For time series with seasonalities the possibilities expand even more. Nevertheless, for time series forecasting with transformers comparative studies on positional encodings are scarce and the existing research is contradictory to used methods in recent models.

Therefore, the effects on performance of two commonly used absolute positional encodings and a relative positional encoding will be analysed. Also, it will be explored if encoding time features and enriching the positional encoding with these temporal embeddings is superior to using time features as covariates.

1.1 Accomplishments

1. The components of transformers for time series forecasting are reviewed.
2. The performance on a popular benchmark dataset of a transformer model specialized for time series forecasting is improved by up to 20%.
3. Learned and fixed absolute positional encodings, a relative positional encoding and how they interact with a temporal embedding in contrast to time features as covariates are compared. Using temporal embeddings resulted in a slight performance improvement on a traffic dataset. A relative positional encoding achieved the best results. Also, the results showed, that it is possible to omit traditional positional encodings and only use temporal embeddings.

4. A Lookback feature is introduced, which enables the model to build direct dependencies to distant data points without the need and drawbacks of the complete sequence length. This is more complex to model for the positional encoding, because the distance between every datapoint is not constant. The different positional encodings are once again tested on this adapted task. Generally, this experiment confirmed the previous results, but only the relative positional encoding was able to benefit from the Lookback feature.

2 Introduction

2.1 The Success of Transformers for NLP and Image Processing

Transformers have become very popular in the machine learning world and are the predominant architecture for Natural Language Processing (NLP). Since the introduction of the transformer model for machine translation (Vaswani *et al.*, 2017), transformers showed unprecedented performance and attained the state-of-the-art solutions not only in machine translation (Edunov *et al.*, 2018) but also text classification (Devlin *et al.*, 2019), question answering and text generation (Radford *et al.*, 2018).

The deployment of self-attention, the main component of transformers, has superseded prior solutions often based on recurrent neural networks (RNN). Self-attention enables the model to form direct relationships between any element of a sequence of text and therefore improves the ability to capture long-term dependencies.

Recently, the success of transformers even carried over to computer vision by applying self-attention to sequences of image patches (Dosovitskiy et al., 2021). Normally, image processing is dominated by convolutional neural networks (CNN) but transformers also replaced CNNs for optical character recognition (Li et al., 2021) or achieved state-of-the-art on video classification (Arnab et al., 2021).

2.2 The Success of Transformers for Time Series Forecasting?

In contrast, the research on transformers regarding time series just seems to start, even though text generation is very similar to time series forecasting. A time series is a sequence of data points indexed over some time period. Time series forecasting, the task of predicting the value

of a variable based on some observed timely ordered data has been an important area of research and is used as an essential tool to extract information as well as decision making for topics like environmental modelling (Dastorani *et al.*, 2016), medicine (Song *et al.*, 2017; Zeroual *et al.*, 2020) or economic applications such as retail (Qi *et al.*, 2021) and finance (Qin *et al.*, 2017).When the forecast of a model depends on past values of itself, it is called autoregressive. This is the common approach for text generation, for example in the Generative Pre-trained Transformer (GPT) models (Radford *et al.*, 2018, 2019; Brown *et al.*, 2020) and also for time series forecasting (Salinas, Flunkert and Gasthaus, 2019).

Even though language and time series are often processed similarly, in contrast to NLP, the results of deep learning models for time series forecasting were for a long time not competitive to traditional so-called statistical models. Statistical model is a collective term for parametric models, which include the autoregressive integrated moving average (ARIMA) model, popularized in 1970 (Box and Jenkins, 1970; Tsay, 2000) , exponential smoothing (Brown, 1956; Gardner Jr., 1985) or the theta model (Assimakopoulos and Nikolopoulos, 2000). The higher performance of statistical models in comparison to more complex models like neural networks was a conclusion of the popular M3 forecasting competition (Makridakis and Hibon, 2000). This did not change in later challenges, like the NN3 competition (Crone, Hibon and Nikolopoulos, 2011), where deep learning methods even were the main focus and still could not compete with statistical models.

This was explained with the insufficient size of datasets for non-parametric approaches like neural networks as the sample frequency of datasets was often just monthly (Bandara, Bergmeir and Smyl, 2020). Today, the appreciation for data increased and the consideration of its value is omnipresent (*The Economist*, 2017). Additionally, the total amount of data created globally increases rapidly (Holst, 2021) and for example the data produced by internet of things (IoT) devices is expected to grow over five-fold till 2025 in comparison to 2019 (O'Dea, 2020). The main characteristics of such devices are sensors, which capture data over time, and the ability to transmit that data. Sensors are only one source of time series data, and many companies nowadays record and process past observations in a higher frequency to support more fine-grained decisions (Tobback and Martens, 2019).

Generally, deep learning models are more expressive if sufficient data is available. Additionally, a single model can also forecast different time series. This is called a global approach, where a single function approximates every time series in a dataset. Not only simplifies a global approach the modelling if multiple time series are of interest, it has also been shown to improve the forecast accuracy (Montero-Manso and Hyndman, 2021). As a

result, RNNs are commonly used for time series forecasting in research because better data is available and RNNs are global models (Hewamalage, Bergmeir and Bandara, 2021).
Even though RNN based models are predominant in the current literature, in many cases tree-based machine learning models like Gradient Boosting Regression Tree (GBRT) or CatBoost (Daoud, 2019) perform competitively. An example is the recently held M5 competition, where tree-based models were leading the rankings (Makridakis, Spiliotis and Assimakopoulos, 2022). Recently a study empirically compared time series forecasting accuracy of state of the art deep learning models to tree based models and while the RNN versions performed similarly to GBRT, the only transformer based model consistently outperformed all other models (Elsayed *et al.*, 2021).
Although transformers seem to be an easy fit for time series forecasting and show superior performance, the research is still limited. The first model for time series forecasting with self-attention was already able to improve on state-of-the-art performance on clinical data (Song *et al.*, 2017). This paper was followed by ”Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting“ (Li *et al.*, 2020), which explored the disadvantageous computational and memory complexity of transformer models, and adapted and improved the transformer architecture especially for time series forecasting. The authors incorporated a local context for each timestep using CNNs, and therefore this model will be addressed as ConvTransformer. The ConvTransformer was able to improve state-of-the-art results on multiple datasets. These results were only topped by the Temporal Fusion Transformer (TFT) (Lim *et al.*, 2020) introduced by researchers of Google. Recently more transformer based models are proposed for time series forecasting (Qi *et al.*, 2021; Zhou *et al.*, 2021), but comparative analyses, which are common for NLP, on important components are scarce for transformers for time series forecasting.
This work will examine the research question: **“How do different positional encodings effect the performance if seasonalities are present”** by revisiting the ConvTransformer and exploring different choices for the positional encoding, an important component of any transformer.

3 The Components of a Transformer for Time Series Forecasting

The motivation of Vaswani et al. with their paper "Attention is all you need" is to avoid recurrence and convolutions because a crucial factor to capture long-term dependencies is the path length during forward pass and backpropagation of the error. To establish a relationship between two-distant points of an input sequence, for convolutional as well as recurrent mechanisms, the signal has to travel a long way. Instead, self-attention, the elemental building block of transformers, directly models the relations between any input token and can therefore learn long-term dependencies easier.

3.1 Encoder-Decoder-Architecture

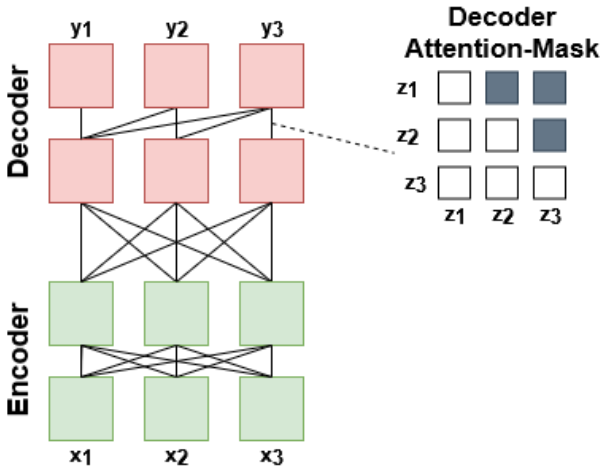

Figure 1:Transformer encoder-decoder structure with a sequence of length three. In the encoder every element of a sequence can attend to any other element and the output of the last encoder-layer is passed on to every step of the decoder. While from one decoder-layer to the next an attention-mask permits an element to attend to future elements of the sequence.

The original transformer of the paper "Attention is all you need" consists of an encoder-decoder architecture and was designed for translation. Based on this encoder-decoder structure with self-attention many state of the art results, for example in translation, question-answering or text-summarizing (Lewis *et al.*, 2019; Raffel *et al.*, 2020) were achieved.
But already in 2014 the general encoder-decoder structure was popularized for sequence to sequence tasks (Cho *et al.*, 2014; Sutskever, Vinyals and Le, 2014) and the general procedure remains identical for transformers. First the encoder compresses the input sequence to a context vector. Then the decoder maps this intermediate representation to a variable-length

output sequence. Before transformers, both the encoder and decoder were recurrent neural networks, mostly Long Short Term Memory networks (LSTM) (Hochreiter and Schmidhuber, 1997), which with their sequential processing captured temporal context. In contrast, in the transformer, where no recurrence is used, the structure of a sequence is mainly established by the conjunction of self-attention and positional encodings.
The transformer's encoder and decoder differentiate by the used context in each module. The transformer encoder works bidirectionally (Lewis *et al.*, 2019) (Figure 1), at each token the model can attend to preceding and subsequent points of the input sequence. On the other hand, the transformer decoder works autoregressively and uses left-context-only. This is achieved by masking the values of the attention mechanism (Figure 1). The output of the encoder is a contextualized version of the input sequence and can be used in full at every timestep in the decoder.

Some transformer based models don't use this two-staged structure. The Bidirectional Encoder Representations from Transformers (BERT) (Devlin *et al.*, 2019) and its numerous variations (Clark *et al.*, 2020; Lan *et al.*, 2020; Sanh *et al.*, 2020) only use an encoder. This is for example applicable for classification tasks, where the fully visible masking pattern in the self-attention is advantageous.

Also, there are models, which only build on decoders and therefore are best suited for generative tasks like text generation or time series forecasting. Most popular examples are the Generative Pre-trained Transformers (GPT) (Radford *et al.*, 2018, 2019; Brown *et al.*, 2020) created under OpenAI. The text generated by these decoder models is close to human level and the fully trained model is therefore not even released out of fear of malicious application (*Better Language Models and Their Implications*, 2019).

While for time series anomaly detection, representation learning or classification (Zerveas *et al.*, 2020; Dang *et al.*, 2021) an encoder-only structure is suitable, for autoregressive time series forecasting a decoder is needed. This is because an encoder in contrast to a decoder is not designed to handle newly generated tokens. On the other hand, an encoder is not essential for time series forecasting. For example, the ConvTransformer consists only of a decoder. But it can under certain circumstances be advantageous to pre-process known covariates in an encoder. Examples are the Aliformer (Qi *et al.*, 2021), which is a transformer for forecasting future e-commerce sales and processes promotional information in an encoder, or the Temporal Fusion Transformer. In the TFT the encoder is itself a complete LSTM encoder-

decoder model, which processes covariates separately. Then the output of the encoder structure is processed with a normal transformer decoder using self-attention.

3.2 Self-Attention

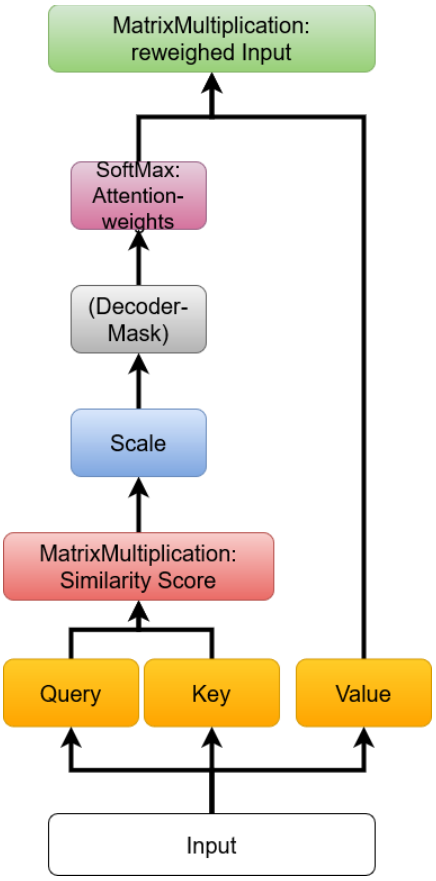

Figure 2: Procedure of the Self-Attention mechanism to reweigh each token of an input sequence depending on its global context. The masking is only applied in a decoder layer.

Self-attention is probably the most renowned mechanism of transformers, even though self-attention was already used earlier (Lin *et al.*, 2017; Paulus, Xiong and Socher, 2017). Omitting recurrent structures in combination with self-attention was then described in "Attention is all you need" and defined as a transformer.

Before self-attention was proposed, other methods to calculate attention scores have been used (Luong, Pham and Manning, 2015; Bahdanau, Cho and Bengio, 2016) and differed only slightly in calculation. Both were incorporated in encoder-decoder models with LSTMs for neural machine translation. Attention was used to relieve the information bottleneck problem between encoder and decoder, as the last hidden state of the encoder had to embody the

complete context of the input sentence. For longer sequences this gets challenging, hence attention was implemented to enable the model to directly focus on important time steps in the past and consequently retain longer dependencies.
For the same reason and in a very similar fashion attention is also often used for time series forecasting. For example (Qin *et al.*, 2017) propose a dual-stage attention-based RNN (DA-RNN) for time series prediction, which consists of LSTMs as encoder and decoder. The DA-RNN incorporates the same attention method as proposed in (Bahdanau, Cho and Bengio, 2016), and enables the decoder to adapt the encoder's context vector by reweighing relevant encoder hidden states across all time steps.

In contrast, the application of self-attention in transformers differs significantly. Self-attention is a reweighing of each token in a sequence based on the other data points of the sequence. This happens within one layer and not only to connect encoder and decoder. As a result, self-attention makes it possible to relinquish recurrent structures because the temporal dependencies, which were handled by the sequential processing of RNNs before, can now be mapped by self-attention. This is the essential motivation behind self-attention, because now direct connections between distant tokens can be made instead of trying to retain information over multiple time steps via hidden states of recurrent models.
To calculate the attention scores in a transformer, an input sequence X of n tokens of dimension d, $X \in \mathbb{R}^{n\,x\,d}$ is transformed by multiplying with three learnable weight matrices $W_Q, W_K, W_V \in \mathbb{R}^{d\,x\,d}$, which results in different feature representations called Query Q, Key K and Value V (Equation 1). Then a similarity-score, following (Luong, Pham and Manning, 2015) is calculated between Query and Key. This score is scaled by the Euclidian distance of the dimensions of the Query or Key matrix to prevent large results of the dot product. To calculate the attention weights a row-wise softmax normalization is applied to the scaled similarity-score (Equation 2). Hence, each token of a sequence only depends on elements of their sequence. To achieve that the autoregressive self-attention in the decoder works causally, an attention-mask (Figure 1) is applied prior to the softmax function and a token can only use the similarity-scores of itself and prior elements, the token can not attend to following elements.
The last step is to multiply the final self-attention weights with the Values to receive the new input representation, where each element is reweighted by its dependencies to other tokens.

Definition of Query, Keys and Values:

$$Q = XW_Q\,, \qquad K = XW_K\,, \qquad V = XW_V \tag{EQ1}$$

The reweighing of *V* can be written as

$$SelfAttention * V = softmax\left(\frac{QK^T}{\sqrt{d}}\right)V \tag{EQ2}$$

Or the same operation in the element-wise consideration, to determine the reweighed output representation for element z_i ,

$$z_i = \sum_{j=1}^{n} a_{ij}(x_j W_V) \tag{EQ3}$$

Where each attention weight coefficient, a_{ij} is computed using the softmax function:

$$a_{ij} = \frac{e^{e_{ij}}}{\sum_{k=1}^{n} e^{e_{ik}}} \tag{EQ4}$$

And e_{ij} is the similarity score of each one element of *Q* and *K* :

$$e_{ij} = \frac{(x_i W_Q)\,(x_j W_K)^T}{\sqrt{d}} \tag{EQ5}$$

In the original transformer it is proposed to apply this attention mechanism several times in parallel and is then called multi-head attention. Each attention head uses independent Keys, Queries and Values and can specialize on different dependencies and in total allows the model to learn more complex relationships.

3.3 Positional Encoding

One of the main reasons in the original transformer, beside directly handling distant dependencies, to replace recurrent structures with self-attention is to be able to parallelize computation. Recurrent neural networks are inherently sequential, which impedes parallelization and becomes more critical for longer sequences. In contrast, self-attention operates on calculations of products between matrices and is therefore within a sequence permutation equivariant. This creates a problem as language and maybe even more so, time series, are order dependent. As a solution Vaswani et al. use position encodings.
A position encoding, first introduced in (Gehring *et al.*, 2017), provides positional information. The position is mapped to a vector of continuous numbers and is added to each element of the input sequence of the encoder and decoder before being processed by the attention mechanism.

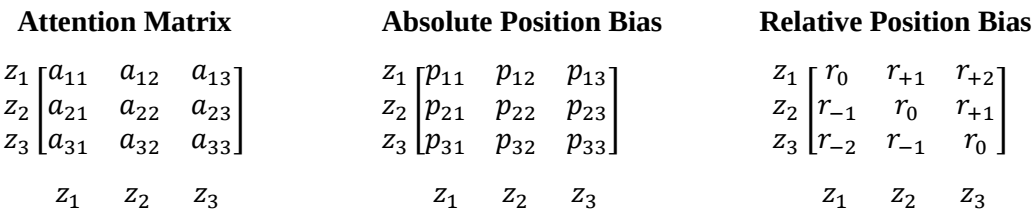

Figure 3: Absolute and Relative position biases which in this example encode the position for a sequence of length three. z_i represents the i_{th} element of the sequence as Query or Key and their dot-product defines the attention score a_{ij} . The Position Biases are added to each element z before the matrix multiplication is calculated and therefore affect the attention score.
For the Absolute Position Bias, to each element their position in a sequence is added, therefore for example the attention score a_{11} of the first element with itself includes the same position encoding twice, once in the Query and once in the Value representation.
In contrast, the Relative Position Bias encodes the relationships directly and includes the distance to other elements. Therefore, for example r_0 stays the same for each element's attention score with itself.

Unlike in time series forecasting research, in NLP positional encodings are explored heavily (Dufter, Schmitt and Schütze, 2021). The research on positional information can be grouped in either absolute or relative encodings. The encodings of the original transformer are absolute and encode each position *p* from 1 to maximum sequence length into a *d*-dimensional vector. Hence, a mapping $f : \mathbb{N} \rightarrow \mathbb{R}^d$ is defined. In the original transformer paper two different

absolute encodings are investigated, an engineered fixed encoding with sinusoidal waves and one where the encoding is learned completely by the model itself.

3.3.1 Absolute Positional Encodings

A common solution for an absolute position encoding is a learned embedding. The position of each element within the input sequence is modelled with a learned lookup table and produces the *d*-dimensional output. An advantage is that the resulting positional encoding is completely data-driven and is possibly able to learn more complex information rather than only about position (Wang and Chen, 2020), which could be especially useful for time series, if temporal information can be incorporated.

The other popular and used in the final version of the original transformer is the sinusoidal encoding (Equation 6). Like in the learned embedding, the sinusoidal function takes the position index *p* of each element of a sequence as input, but here the function is predefined without learnable parameters. It produces *d* waves of different frequencies alternating between sine and cosine.

$$PE_{(p,2j)} = sin\left(\frac{p}{10000^{2j/d}}\right)$$

(EQ6)

$$PE_{(p,2j+1)} = cos\left(\frac{p}{10000^{2j/d}}\right)$$

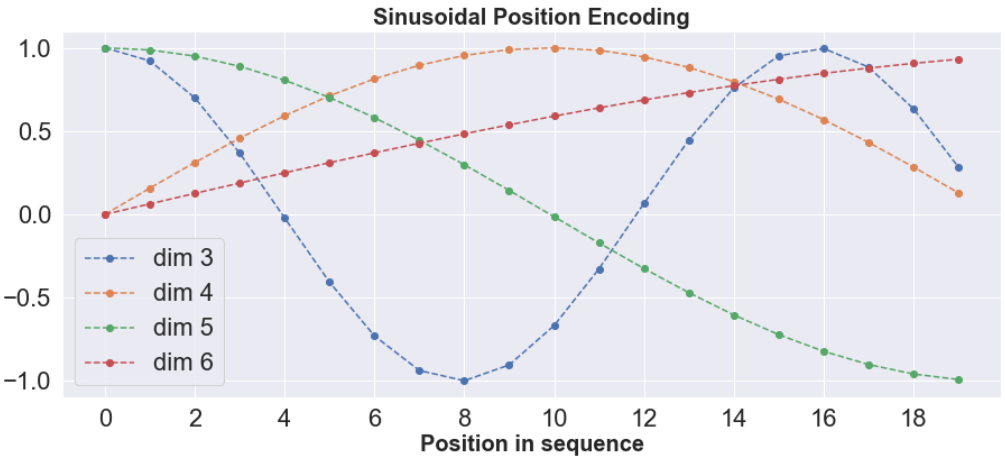

Figure 4: The absolute sinusoidal position encoding uses alternating values of sine and cosine. The wavelength is increasing with higher dimensions of the encoding and therefore avoids identical encodings for different positions even for long sequences.

Both absolute encodings were tested in the original transformer and nearly identical results were observed. The sinusoidal version was chosen because it may allow the model to extrapolate to sequence lengths longer than the ones encountered during training. For time series forecasting this advantage of the sinusoidal encoding is less critical because in contrast to varying sentence length the forecast horizon can be arbitrarily stipulated beforehand. Therefore, it is not clear which positional encoding should be used and it could even be possible that completely data dependent positional encodings are favourable for time series. Nevertheless, very little research has been done to explore the effects of different positional encodings for time series forecasting.

3.3.2 Relative Positional Encodings

The absolute encodings are independent of each other and the main goal is to distinguish different positions, while the relationships are not modelled (Wang *et al.*, 2020).
On the other hand, relative positional encodings (Shaw, Uszkoreit and Vaswani, 2018) encode the relative distance between tokens and represent the pairwise relationships between the current position and other positions.

(Shaw, Uszkoreit and Vaswani, 2018) propose to incorporate relative positional information parameters on the Key as well as the Value level of the self-attention mechanism.

To implement relative positional encodings, one has to introduce the pairwise relationships a_{ij}^{V} , $a_{ij}^{K} \in \mathbb{R}^{\boldsymbol{d}}$ between input elements x_i and x_j .
a_{ij}^{V} , a_{ij}^{K} are edge information which represent the absolute distance between elements modified by learned weight parameters.

To append relative positional information a_{ij}^{V} at the Value level Equation 3 is modified to:

$$z_i = \sum_{j=1}^{n} a_{ij}(x_j W_V + a_{ij}^{V}) \qquad \text{(EQ7)}$$

And for the Key matrix the pairwise relationships a_{ij}^{K} is added to Equation 5:

$$e_{ij} = \frac{(x_i W_Q)\,(x_j W_K + a_{ij}^{K})^T}{\sqrt{d}} \qquad \text{(EQ8)}$$

A drawback of relative position encoding by (Shaw, Uszkoreit and Vaswani, 2018) is the memory complexity of $O(L^2)$, which in contrast to absolute positional encodings is repeatedly calculated in every layer. But the relative position encoding by (Shaw, Uszkoreit and Vaswani, 2018) was recently subject to more research and several improvements also in regard to efficiency were proposed (Huang *et al.*, 2018, 2020; Chen, 2021; Luo *et al.*, 2021).

3.4 Local Context

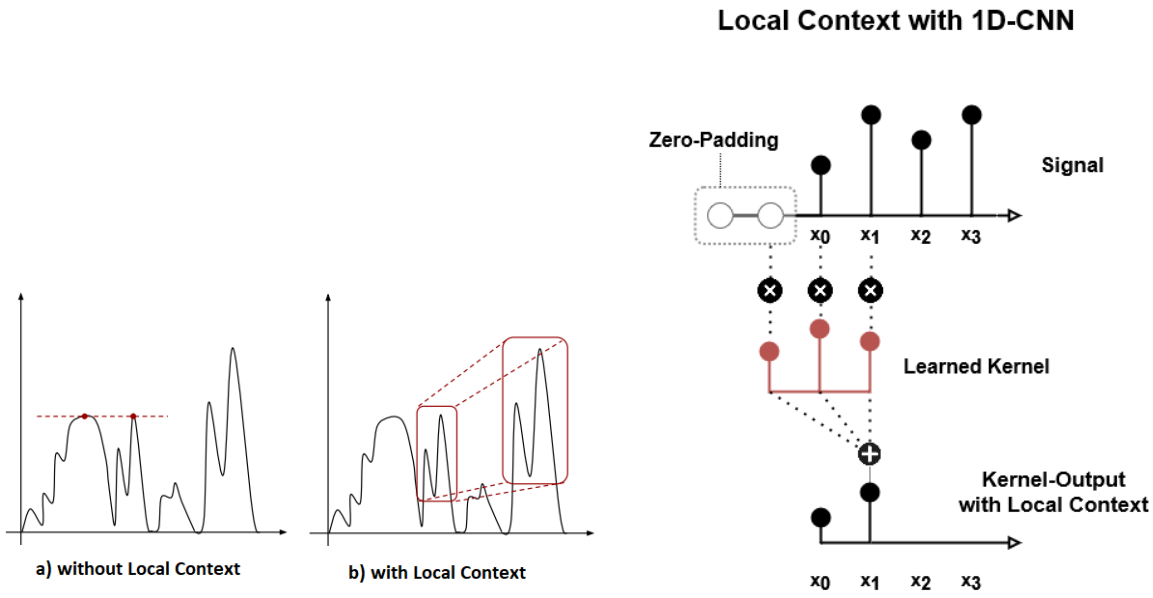

Figure 5: Local Context with 1D-CNNs: Instead of measuring the similarity between points (a), shapes are compared b). Here, the 1D-CNN consists of a learned kernel of size 3. The padding of size kernel_size - 1 = 2 is concatenated to the front to make the 1D-CNN causal and together with a step size of 1 results in an output for each timestep.

Even though a positional encoding is essential, it seems that transformers do not learn the exact token order, instead they learn higher-order distributional statistics (Yang *et al.*, 2019; Sinha *et al.*, 2021) and in combination with self-attention more global dependencies are captured.
The authors (Li *et al.*, 2020) of the ConvTransformer for time series forecasting emphasize this shortcoming of transformers and propose convolutions to capture local context. Convolutions capture relative positions and local dependencies within their receptive field and especially in combination with self-attention are able to model complex relations.
Convolutions are used in Convolutional Neural Networks (CNN) mostly for image processing tasks as a pattern recognition mechanism (Lecun *et al.*, 1998) and CNNs are still competitive in computer vision (Liu *et al.*, 2022) even after recent innovations with transformers (Liu *et al.*, 2021).

While two-dimensional convolutions are used for images, one-dimensional (1D) convolutions can also be applied to sequences like time series and are in this context also called temporal convolution. Convolutions extract features with learned kernel vectors of length *s*. The kernels slide over a time series sequence mostly with a step size, also called stride, of one. For autoregressive forecasting it is desirable to have an output of the 1D-CNN for every timestep. Therefore, additionally to a stride of one, a padding precedes the actual sequence. This padding is a concatenation of extra values, often zeros, of length $l = s - 1$. A preceding padding makes the convolutions causal. That means, similarly to self-attention in the decoder, the output of the convolution at a timestep is not dependant on future timesteps.

It is even common to only use 1D-CNNs without any attention mechanism for time series forecasting. Without self-attention and exclusively using simple 1D-CNNs only local patterns would be recognizable, therefore dilated convolutions were introduced (Oord *et al.*, 2016). Dilated convolutions achieve a larger receptive field by skipping input elements with a certain step size, and especially when multiple such layers are stacked, long-term dependencies can be captured.
With self-attention this is not needed and has the advantage in contrast to dilated 1D-CNNs that a direct dependency to every element is modelled in every layer. Ablation studies established that providing some form of local context together with self-attention is crucial for time series forecasting (Li *et al.*, 2020; Lim *et al.*, 2020). Even though these ablation studies did not include positional encodings, (Gehring *et al.*, 2017) showed for a NLP task that

positional encodings are beneficial in combination with 1D-CNNs because they only provide local context and the positional encoding helps to globally distinguish tokens.

For time series, positional encodings could even have a larger effect. Because in contrast to NLP where positional encodings mainly differentiate between words, regularly repeating patterns occur in time series, which are critical to model correctly. Positional encodings could assist the model with a better positional representation if markers for these repeating patterns are incorporated.

4 Time Series with Seasonalities

Time series often exhibit repeating patterns, which are called seasonal if they are of a fixed and known frequency (Hyndman and Athanasopoulos, 2018). Many time series involving humans show such patterns, for example daily patterns because of the day-night rhythm, weekly patterns because of business days and weekend schedule or monthly patterns because of salary payments. Additionally, holiday effects occur and can be of great impact. But they are difficult to model because of the annual frequency and various effects of different holidays. Of course there are also many seasonal patterns of natural origin like periodicities in the microseconds of radio bursts observed by astrophysicists (Andersen *et al.*, 2021) or seasonality of the climate.

Some common subjects in time series forecasting research, where a more accurate forecast has a decisive impact, are electrical load, traffic flow, retail sales, air quality (Sun and Li, 2020), server load (Daraghmeh *et al.*, 2021), trading volume (Arnerić, 2021) or sensor data (Zhou *et al.*, 2021). And with the growth of e-commerce and IoT-devices the potential demand for accurate high-frequency forecasts will rise.

All these examples, if observed in a high enough frequency (e.g., hourly), can show multiple seasonal patterns, which become particularly difficult to forecast because of superposition and possible interactions of the different patterns.

4.1 Traffic Dataset

4.1.1 Data Exploration

The effects of positional encodings on forecasting with the ConvTransformer will be explored using a traffic dataset (Cuturi, 2011b, 2011a). This dataset was selected because the sample frequency is hourly, and it shows homogenous and interesting seasonalities. Additionally, many recent models use this dataset as a benchmark and these results can serve as comparison and validation of the experiments.

The traffic data was originally downloaded from the California Department of Transportation PEMS website and contains 15 month of freeway occupancy rates in San Francisco. The data of 963 sensors is aggregated to hourly usage percentages between 0 and 1 and cover a period from January 1st 2008 to March 30th 2009 but following among others (Salinas, Flunkert and Gasthaus, 2019; Li *et al.*, 2020; Lim *et al.*, 2020) only data till 22nd June 2008 17:00 is used, where the last week contains the test set and is excluded from the training procedure of the model.

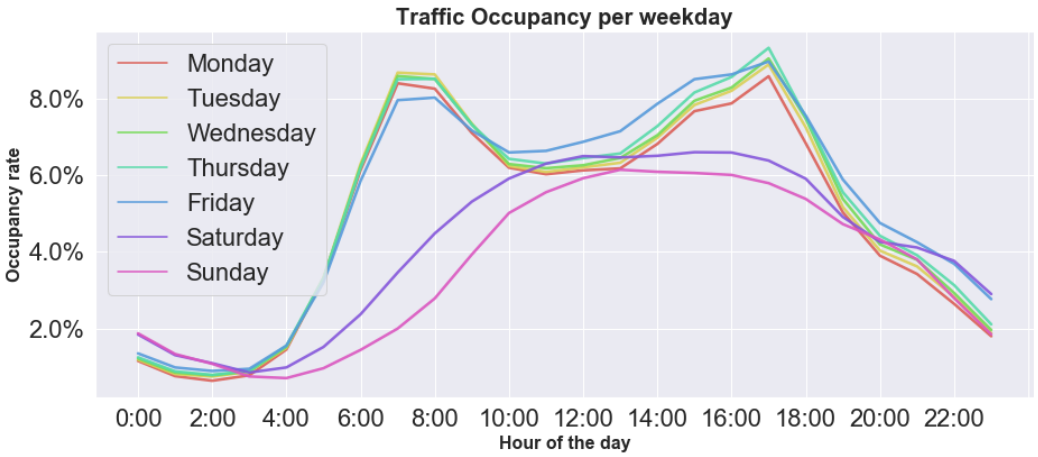

Figure 6: Median traffic occupancy per hour for each weekday across the complete dataset. The distinction between business day and weekend is apparent. Business days show a double peak, in the morning and early evening and are probably caused by people commuting to and from work. At noon on Fridays, it is busier in comparison to other business days. At the weekend, traffic rates increase slower in the morning and stay at a stable level till the evening. Even though Saturday's traffic shows similar characteristics to Sundays, consistently more cars are on the roads on Saturdays. Also, the occupancy rates at late evenings are higher on Fridays and Saturdays.

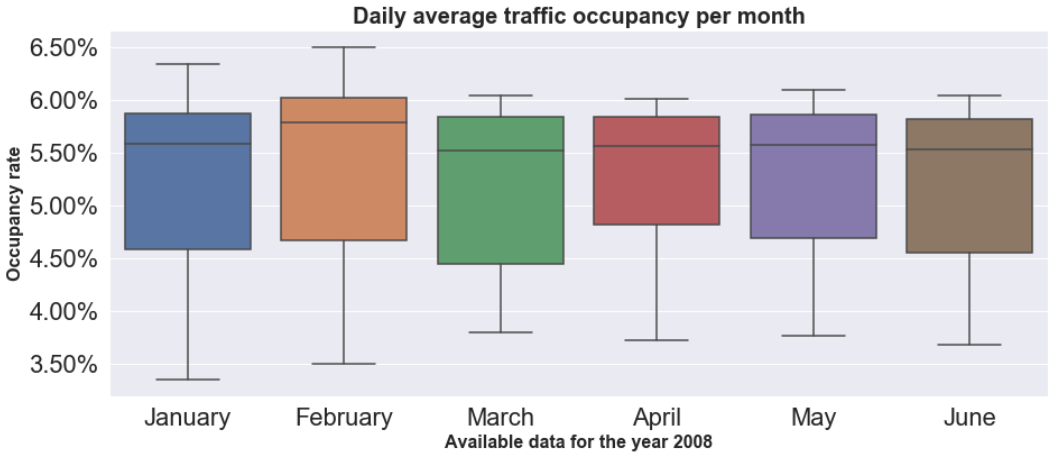

Figure 7: Daily average traffic occupancy distribution for the complete used dataset. Because not a whole year is available, it is difficult to say if there is a yearly seasonality, but in the observed data only small fluctuations can be observed. Therefore, the monthly information will be of subordinate priority for the model performance.

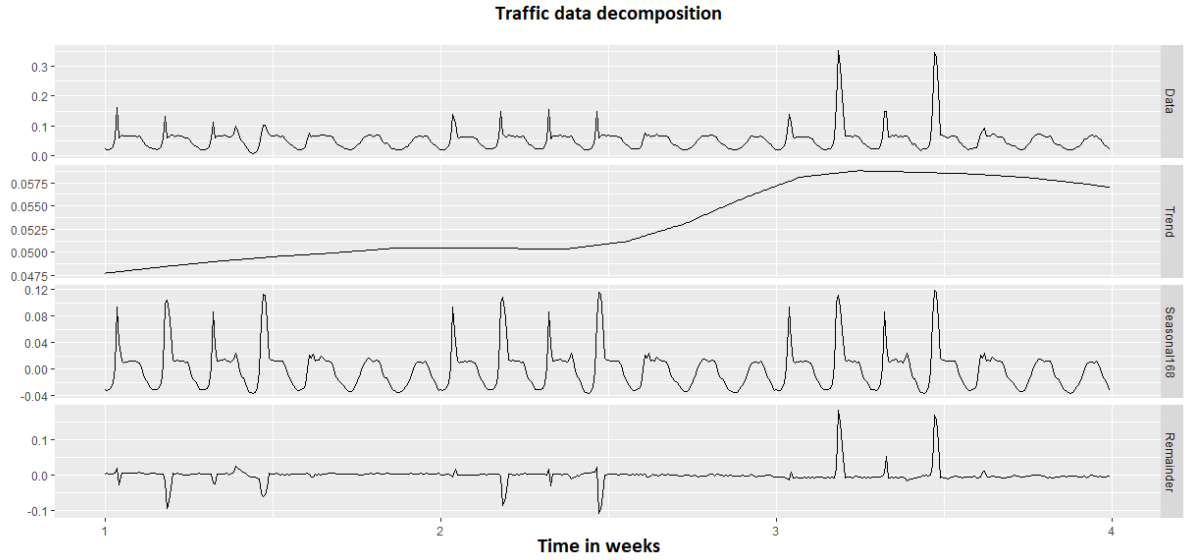

Figure 8: Exemplary data decomposition of one of the 963 accessible roads. The observed data (first row) is split by an additive model into a trend (second row) and a seasonal component (third row), the remainders (forth row) are the error residuals, which can not be explained with the decomposition. 4 weeks of data are displayed and the data starts with midnight of a Monday. It is noticeable, that for this road the seasonal component for Fridays is more similar to the weekend's rates, while in Figure 6 this could not be observed. Also, outliers can be seen in the raw data in week three with high utilization, which can not be modelled with the decomposition and lead to high residuals.

4.1.2 Probabilistic Forecast

As a benchmark this dataset is commonly used to forecast different quantiles. Instead of a single point-forecast, quantile forecasts provide probabilistic information for different chosen thresholds. The quantile loss, also called pinball loss, *QL* for a specific quantile $q \in (0,1)$ is defined as:

$$QL(y, \hat{y}, q) = q(y - \hat{y})^+ + (1 - q)(\hat{y} - y)^+ \qquad \text{(EQ9)}$$

Where y is the observation, $\hat{y}$ the predicted value for that quantile and $(\dots)^+ = max(0, \dots)$
The total quantile loss is then simply the sum of *QL* for all forecasts.

In contrast to ordinary least squares, where the conditional mean is estimated, the 50% quantile estimates the conditional median. Therefore, the predictions are more robust to outliers. But more important, when different quantiles are estimated, one can approximate the distribution of the target and estimate the aleatoric uncertainty (Tagasovska and Lopez-Paz, 2019). Aleatoric uncertainty describes the stochasticity in the data. Such forecasts are especially important for strategic decisions to prepare for different scenarios. For example in retail logistics, the reorder point of goods is often determined with quantile forecasting and the underlying uncertainty in mind (Bruzda, 2020).

Traffic forecasting could be used to predict congestion, travel duration and route selection by a navigation system or when to clean roads with the least disturbance of traffic. Therefore, more accurate forecasts could help the government for better planning and minimize environmental impacts, as well as risk reduction and inconvenience for citizens.

4.2 Methods for Forecasting Seasonal Time Series

Many models were tried for forecasting seasonal data.
Statistical models like ARIMA were modified (Vagropoulos *et al.*, 2016) to better handle seasonal data and in 2011 a popular algorithm with automatic parameter search, a combination of multiple statistical models called “Trigonometric Exponential Smoothing State Space model with Box-Cox transformation, ARMA errors, Trend and Seasonal Components” (TBATS) (De Livera, Hyndman and Snyder, 2011) was proposed to forecast complex seasonalities. Even though TBATS can handle multiple seasonalities a local model for each series has to be fitted.

But, TBATS and other statistical models simply can not compete on these complex datasets with the performance of deep learning and particular transformers (Alexandrov *et al.*, 2019; Wu *et al.*, 2021).

To simplify the forecasting problem deseasonalization can also be applied. Deseasonalization, which is the removal of seasonal patterns, is an approach, so that the model does not have to learn these patterns. But because multiple patterns can interfere with each other, seasonal adjustment is difficult. Nevertheless, often applied is Seasonal and Trend decomposition using Loess (Cleveland *et al.*, 1990) (STL), where Loess is an algorithm for modelling nonlinear relationships. Because STL is only designed to handle a single seasonality, it needs to be executed for each seasonal factor and also for each series separately. Additionally, there can occur problems related to the Gregorian calendar. Because the months are not of equal length, in some cases each month has to be stretched to 31 days by interpolation (Ollech, 2021). This makes STL tedious to apply.
More recently, easier deseasonalization for multiple seasonalities was proposed (Bandara, Hyndman and Bergmeir, 2021; Dokumentov and Hyndman, 2021) and can even be combined with deep learning models.

RNN based models can be global, are more expressive and special solutions for forecasting multi seasonalities were designed. (Bandara, Bergmeir and Hewamalage, 2021) proposed the LSTM-Multi-Seasonal-Net, based on LSTMs with an integrated deseasonalization method for multiple seasonalities. They tested different deseasonalization methods and showed superior results compared to TBATS.

Still, the use for deseasonalization for transformers is debatable as even the authors of the forementioned deseasonalization papers comment, that deep learning models are able to capture homogenous periodic patterns without deseasonalization (Bandara, Bergmeir and Smyl, 2020; Hewamalage, Bergmeir and Bandara, 2021). Furthermore, some calendar effects like moving holidays are still not registered.

DeepAR (Salinas, Flunkert and Gasthaus, 2019) is another RNN based model. It uses LSTMs in an encoder-decoder architecture and is a global model. At its release time DeepAR performed well without deseasonalization in probabilistic forecasting and was also tested on the traffic benchmark and others and the authors emphasized DeepAR's ability to model seasonal data.

Generally, in order for RNNs to be able to predict seasonalities well, the length of the input window has to be long enough and optimally should contain a complete seasonal cycle. A rule of thumb often considered for selecting the length of the input window is
inputSize = 1.25 * *max*(*outputSize; seasonal period*)
(Bandara, Bergmeir and Smyl, 2020; Bandara, Bergmeir and Hewamalage, 2021; Hewamalage, Bergmeir and Bandara, 2021). The same requirement should hold for transformers, as RNNs need the seasonality included in the input sequence to model the dependencies. The advantage of transformers is that these dependencies are modelled directly over arbitrary distances and don't have to be propagated through hidden states, but still the data needs to be included in the sequence.

5 Current Research on Positional Encodings for Time Series Forecasting

Transformers outperform every other model for time series forecasting on many datasets and even though transformers are not explicitly built for multi-seasonality, many of the popular benchmarks, where transformers achieved state-of-the-art results, show multiple seasonalities. Of course, as explained earlier a positional encoding is essential for any transformer. But a positional encoding could be especially important to model the more complex but repeating dependencies in a time series to capture such seasonalities.
In RNNs time information of a timestep is generally provided as an additional input feature (Salinas, Flunkert and Gasthaus, 2019) and position and dependencies are modelled by processing the input sequentially and recursively. While it is possible to process time features as covariates with transformers, recently the idea of incorporating time information to the positional encoding has emerged.

To better portray the temporal information it has been tried to encode the timestamps and enrich the positional encoding with these embeddings (Wu *et al.*, 2021; Zhou *et al.*, 2021).
In contrast, the ConvTransformer simply uses time features as covariates.
Otherwise, it is also possible to omit the position encoding in the conventional way as the TFT employs an LSTM instead, which by sequentially processing the input, incorporates the positional information in the transformer decoder (Lim *et al.*, 2020).

Also, there is no consensus, which general position encoding to use.

(Zhou *et al.*, 2021) employ the sinusoidal encoding of the original transformer (Equation 6) as a basis of their encoding. In contrast, ConvTransformer and the first transformer on clinical time series forecasting (Song *et al.*, 2017) use the learnable positional embedding, which was also proposed as an alternative in the original transformer and explained in Section 3.3.1.

Apart from that, research on positional encodings of transformers for time series forecasting is sparse. And in contrast to transformers for NLP, positional encodings in transformers for time series forecasting were to the author's knowledge never the sole focus of research and even empiric performance comparisons of different encodings are rarely included for the state-of-the-art transformer models.

One of the relevant papers on positional encodings for time series was also focused on forecasting seasonal traffic data with a model called Traffic Transformer (Cai *et al.*, 2020). The proposed architecture is accompanied by an empirical comparison of different solutions for positional encodings. While the authors do not stick to naming conventions and refer to the absolute sinusoidal encoding introduced in Equation 6 as relative position embedding, they only compare different absolute encodings. They propose a global encoding, where the sinusoidal function (Equation 6) is used to encode each timestep and position in the dataset instead of only encoding the position of an input sequence. Additionally, temporal embeddings are tested, where time features are encoded.

Between the absolute sinusoidal position encoding, global encoding only, global encoding with temporal embeddings and absolute sinusoidal with temporal embeddings, only the sinusoidal encoding worked well, when used with conventional self-attention.

Interestingly, the authors report discouraging results for the integration of temporal embeddings to absolute sinusoidal encodings, even though this combination was very successfully used in the architecture of (Zhou *et al.*, 2021). But (Zhou *et al.*, 2021) did not include any empiric comparisons of different positional encodings in their paper.

The authors of the Traffic Transformer also recorded inadequate performance for the global encoding and argue that this does not work well, because the global encodings of the test set were never observed by the model beforehand. Even though, technically the sinusoidal encoding was designed, to extrapolate to unseen sequence length. Therefore, this approach will still be included in the following experiments of this work because by combining the

global encoding with a conventional sequence positional encoding the model is possibly better able to learn about a trend in a dataset.

The relative positional encoding is rarely used for time series forecasting. It is tested in (Mohammdi Farsani and Pazouki, 2021) for the same traffic dataset used in this work and inferior performance in comparison to a sinusoidal encoding is measured. But the authors use a transformer without local context and their transformer does not even beat baselines, which the ConvTransformer outperforms easily.
In contrast, relative positional encoding is successfully used for a recommender system with self-attention (Li, Wang and McAuley, 2020). The model uses historic product interactions of individual costumers as input. These time series are irregularly sampled. With irregular sampled time series, the time intervals between observed elements are varying. The authors use relative positional encodings to model the different relationships between irregular sampled data points.
Even though the traffic dataset, which will be used in this works experiments, is regularly sampled, especially for seasonal data the relationships between points could be influential.

To examine these contradictory references and make a better comparison, this work will analyse the effect on performance of different positional encodings and evaluate if the enrichment of the positional encoding with temporal embeddings or simply using the temporal information as features and not combining them with the positional encoding performs better.

6 Experiments on different Positional Encodings

Because information on performance of different positional encodings for time series forecasting with a competitive transformer architecture is scarce, the focus lies on the basic positional encodings. A comparison of the two absolute positional encodings proposed in the original transformer with and without temporal embeddings and the relative encoding introduced in (Shaw, Uszkoreit and Vaswani, 2018) will be conducted. Also, it will be tested if it is possible to use only the temporal embedding without any additional encoding. This empirical study will be administered on the seasonal traffic dataset.

Later, these experiments will be repeated under more complex conditions on the same dataset to get a better understanding for the positional encodings and review the obtained results.
For this reason, the Lookback Feature will be introduced in Section 6.5, which adds extra data

from further in the past to the input window. This possibly makes the modelling task more difficult and will test the capacities of the different positional encodings.

6.1 Experiment Settings

6.1.1 Implementation of the Positional Encodings

6.1.1.1 Temporal Embedding for the Traffic Dataset

Because for time series forecasting the sequence length is constant and fixed, the traditional positional encodings (e.g. absolute and relative) for each sequence of a batch are equal. Additionally in contrast to NLP with transformers, where a sequence generally starts with the beginning of a new sentence, for time series it may be useful to incorporate temporal information because a subset of the complete time series is randomly sampled and will have different start times (e.g. weekdays).

This is usually done with embeddings. Each temporal feature (e.g. hour of the day, weekday, month, day of month, day of year…) will be encoded in its own lookup table. It is important to only include and encode a feature if it helps the forecast, as otherwise the dimensionality unnecessarily increases and the model is more likely to overfit.

Therefore, only the seasonal time features are encoded, which for the traffic dataset are the hour of the day and the day of the week. While it is possible, that there is a seasonality within a year (Figure 7), for example more traffic in colder months, a specific embedding for each month is not included. Because an embedding is a simple lookup table, it does not extrapolate to unseen values, also it can overfit easily if a seasonality is not repeated multiple times in the training data. For example, if the dataset only includes one year of data, and it was a very warm January, the embedding for January would be very biased. Because the employed traffic data does not even cover a whole year, a month embedding was not utilized. Still, the traffic data shows some variance over the whole time series and in the original implementation of the ConvTransformer the age of each data point was included as a feature, therefore the age will be encoded with sinusoidal waves, like the global encoding introduced in (Cai *et al.*, 2020). Therefore, the sinusoidal function in Equation 6 will be used to encode the age of each datapoint. The input is therefore the global timestamp of a datapoint and is between one to length of the time series instead of one to the length of the sequence for the conventional absolute positional encoding. Because the model is able to extrapolate the sinusoidal

encoding, the model should be able to handle even unseen test data. This characteristic was the main reason, why the sinusoidal encoding was used in the original transformer.

There are different possibilities to combine embeddings. Generally, they are simply added up, but it would also be possible to concatenate them or combine them with a linear layer. Because more complex relationships exist between different temporal embeddings, for example the traffic in the morning (hourly embedding) of a Sunday (weekday embedding) is very different to the morning data of a Monday, the embeddings are combined with a linear layer.

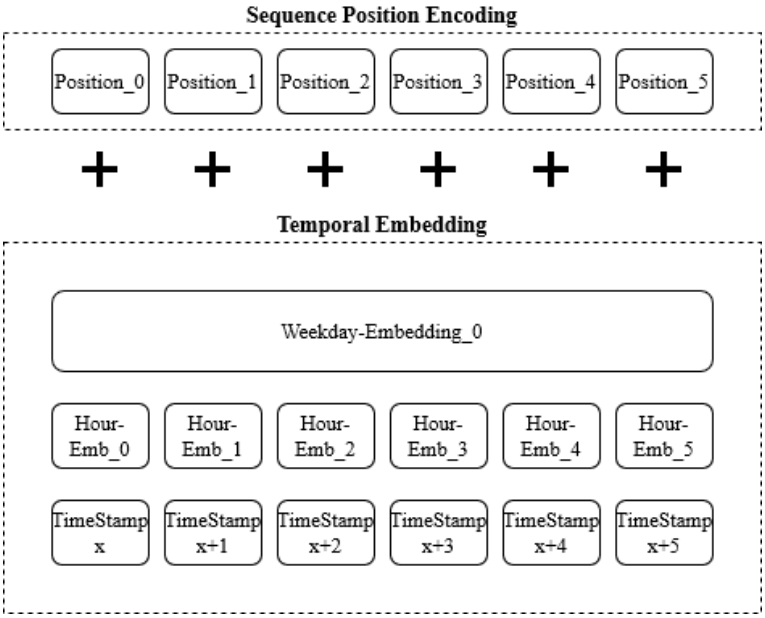

Figure 9: If the time is encoded to enrich the positional encoding, the conventional positional encoding, which defines the order within a sequence, and a temporal embedding are combined. The temporal embedding models the weekday and the hour of the day of each element with learned embeddings and additionally a global timestamp, which defines the point in time on the whole dataset, is added.

6.1.1.2 Conventional Positional Encodings for the Traffic Dataset

The implementations of the absolute positional encodings are relatively straight forward like discussed in section 3.3.1.

For the relative positional encoding, the proposal of an efficient implementation (Shaw, Uszkoreit and Vaswani, 2018) were followed and the learnable relative positional information parameters are used on the Key as well as the Value level of the self-attention mechanism. Additionally, the authors propose to clip the maximum distance at an arbitrary value to enable the model to generalize to longer sequence length not observed during training. For time

series this is not necessary or desirable because the context window is fixed. Also, because of seasonality the distant points often repeat the same patterns as in the forecast window. Therefore, distant points are critical, and their relationships need to be modelled accurately.

6.1.2 The ConvTransformer Model

As a model the ConvTransformer is used. Large portions of code kindly provided by the original authors of (Li *et al.*, 2020) were refactored. Their training loop and other procedures were reused, which include to split the data in training (90%), validation (10%) and test set, where the test set is the agreed upon one week from 6/15/2008 17:00 (included) for this benchmark. In total there are 24*7*963 target timesteps in the test set, which are forecasted by using one week as a context-window (Figure 10) and a 24-hour forecast-window and after each predicted day the context-window is shifted by 24 hours to forecast the next day. For the context-window the true values are used, while for each forecast-window during inference the predicted value is used as further input. During training the true values are always used.
For this work the code was rewritten to be able to use Pytorch lightning, a wrapper for the Python deep learning library Pytorch. Pytorch lightning simplifies many functions needed for research, for example hardware acceleration or code readability.
Experiments were conducted on a machine with an NVIDIA GPU RTX3060 with 12GB VRAM, a 16 core AMD Threadripper 2950x and 64GB RAM using Ubuntu as operating system.

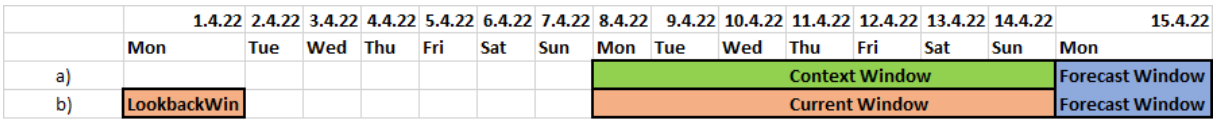

Figure 10: Row a): The forecast window is the target time span, which is to be predicted and is conditional on the given context window. During inference the model operates recursively and uses the forecast x_{i+1} autoregressively to expand the context. During model training the sequence length is the length of the context window plus the length of the forecast window and there is no distinction between context window and forecast window. Therefore, the context window together with the forecast window will be called input window.
For Section 6.5, Row b): If Lookback is used, a relevant time span of the past, which will be called Lookback-window, is included in the input window. The time span directly preceding the forecast window will be called current window. Then, the context window consists of Lookback-window and current-window.

For the experiments the decoder-only ConvTransformer will be used. It follows closely the decoder module of the original transformer and uses 1D – CNNs to provide local context at each time step.

Because the 963 univariate traffic occupancy rates are already between 0 and 1 a dataset normalization is not necessary. For this benchmark dataset the conventions are to use one week as context-window and predict the following day. Therefore, each of the 963 time series is split into eight day long input sequences, which are selected randomly to fill the batch of size 64. In the original implementation each sequence is scaled by 1+ the average value of their context window of length t_0 : $scale = 1 + \frac{1}{t_0}\sum_{t=1}^{t_0} v_t$. The authors adopted this scaling mechanism from (Salinas, Flunkert and Gasthaus, 2019). Instead, in this work the + 1 was omitted and each sequence was only scaled by the average of their context-window. Because only the context-window is used no data leakage occurs. This change brought a considerable improvement in convergence time and some gains in general performance (Table1: MeanScaling).

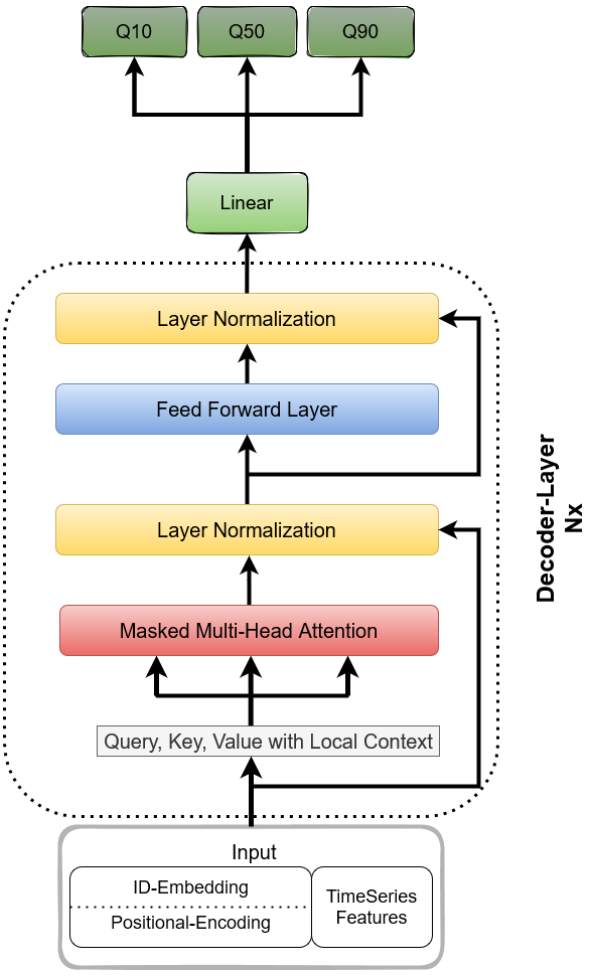

Figure 11: Architecture of the ConvTransformer and directly outputting the different quantiles using the pinball loss. In the original implementation N=3 layers were used.

For each eight-day sequence an ID embedding of the origin sensor is provided, and a positional encoding is added. Then this embedding sum is concatenated to the value of that timestep and constitutes the input for the first decoder.

The data is then processed in the self-attention mechanism. First, Query, Keys and Values are calculated. To add local context for the Query and Keys each time step is processed with a causal 1D-CNN of kernel-size 9. The Values are processed with a 1D-CNN of kernel-size 1 and therefore are not enriched with local context. The output-size of the 1D-CNNs is equal to the input-size times the number of attention heads. For all experiments 8 heads are used. The resulting Query, Key and Values are split and transformed with the masked multi-head self-attention. Then the eight resulting weighted Values are merged.

A residual connection is employed, which combines the new weighted input representation with the original input of the self-attention mechanism. This mitigates the vanishing gradient problem if many layers are used. Then, Layer Normalization (Ba, Kiros and Hinton, 2016) is used on the combined inputs, which stabilizes training and is very important for transformers and still, often special optimizers or very small learning rates are needed (Xiong *et al.*, 2020). For future research it would be interesting to test the research accomplishments of transformers for NLP in this regard on time series forecasting tasks, because for best convergence of this work's model very small learning rates were necessary. For example, PowerNorm (Shen *et al.*, 2020) is an alternative to Layer Normalization, which significantly improved performance on multiple NLP tasks. Another solution could be DeepNorm (Wang *et al.*, 2022), which recently enabled training of very deep transformers.

The normalized output is then processed with a feedforward layer and once more normalized. In the feedforward layer a Rectified Linear Unit $f(x) = max(0, x)$ is used as activation function. Again, for future research it would be interesting to test different options of activation functions like it has been done for NLP (Du, 2020; Dubey, Singh and Chaudhuri, 2022).

This concludes a pass through one decoder layer. Usually, multiple layers are used, where each consecutive layer takes the output of the previous layer as input. In the original implementation of the ConvTransformer only three layers were used but slightly better results can be achieved with more layers, therefore five layers were implemented for the following experiments (Table 1: Pinball5L).

To compute the actual quantile forecast, there are two general possibilities, either the parametric or the non-parametric approach.

The original implementation of the ConvTransformer used the parametric approach, the model predicts the parameters, namely the mean μ and standard deviation σ of a Gaussian distribution and during training, the negative log-likelihood, which is defined as $L = -log\left((2\pi\sigma^2)^{-1/2}\, e^{[-(y-\mu)^2/(2\sigma^2)]}\right)$ and is used as a loss function. During inference, multiple samples from the final distribution are drawn and the forecasts for the different quantiles can be read out from the ordered samples and loss for the benchmark is then calculated with the pinball loss (Equation 9).

The non-parametric solution is to directly predict the target quantile points via quantile regression and use the accumulated pinball losses for each quantile as training loss, which makes the optimization target and the benchmark score more similar.

While it depends on the dataset, which approach is more successful, in the experiments of (Chen *et al.*, 2020) the non-parametric outperformed the sampling of a Gaussian distribution. Also, in the state-of-the-art result on this Traffic dataset of the Temporal Fusion Transformer a non-parametric approach was chosen. Therefore, as a loss function the quantile loss for the quantiles $Q = \{0.1, 0.5, 0.9\}$ is used and summed across the three quantile outputs and jointly minimized.

	ConvTrans		MeanScaling		Pinball3L		Pinball5L		WithoutTime		**My_CTrans**	
Q50	0.122	-	0.1182	3%	0.1004	18%	0.0999	18%	0.1062	13%	**0.0974**	20%
Q90	0.081	-	0.0797	2%	0.0757	7%	0.0751	7%	0.0808	0%	**0.0748**	8%

Table 1: Ablations: 50% (Q50) and 90% (Q90) Quantile losses on the traffic dataset followed by the relative change (in colour) in comparison to the reported results of the original ConvTransformer (ConvTrans). Always ceteris paribus to the previous model. MeanScaling describes the improvement when the sequence scaling was changed. In Pinball3L the loss is additionally changed to quantile loss, the most important adaptation to the original implementation. Pinball5L additionally changes the number of layers from 3 to 5. WithoutTime is an ablation of all time features. My_CTrans includes all adaptations and implements relative positional encoding with temporal embeddings.

	ARIMA	ETS	DeepAR	MQRNN	Seq2Seq	ConvTrans	**My_CTrans**	TFT
Q50	0.223 *129%*	0.236 *142%*	0.161 *65%*	0.117 *20%*	0.105 *8%*	0.122 *25%*	**0.0974** -	0.095 *-2%*
Q90	0.137 *83%*	0.148 *98%*	0.099 *32%*	0.082 *10%*	0.075 *0%*	0.081 *8%*	**0.0748** -	0.070 *-6%*

Table 2: 50% (Q50) and 90% (Q90) Quantile losses on the traffic dataset reported from (Lim et al.*, 2020) followed by the relative change (in colour) in comparison to the best results of this work's version of the ConvTransformer with the described changes and the relative positional encoding enriched with the temporal embedding. ARIMA and exponential smoothing (ETS) belong to the classical statistical models for time series forecasting. DeepAR, MQRNN and Seq2Seq are modern models for time series forecasting based on RNNs and ConvTransformer and temporal fusion transformer (TFT) represent the transformer models.*

After implementing the pinball loss for the ConvTransformer, the results improved significantly, and the Q50 performance was increased by 20%, which is close to the state-of-the-art performance of the TFT. This makes the results of the following experiments on positional encodings more valid and useful because the model is more competitive and performance differences of the experiments are less likely a result of fluctuations of a badly designed model. And most importantly, it makes the research results applicable for practical use cases and future research, because if a transformer with positional encodings still performs significantly worse in comparison to the state-of-the-art model, which does not use positional encodings, there would be no incentive to finetune for the best achievable positional encoding. Because the ConvTransformer with adaptations performs only two percent worse in the 50% quantile forecast in comparison to the TFT, this work's results could be helpful for future research, as there are many possibilities to further improve the ConvTransformer like adaptations to the 1D-CNN (Chen *et al.*, 2020; Tang *et al.*, 2021) for a better local context or an additional variable selection layer, which was proposed for the TFT and had a significant effect on the forecast for the traffic dataset.

Also, an important difference of the ConvTransformer to the TFT is, that it abstains from the use of RNNs, which is more in line with the original convictions of any transformer to waive recurrent structures for better parallelization. Additionally, in Section 6.5 it will be explained that a transformer for time series forecasting without RNNs is able to jump over periods in time, which would be more difficult if RNNs are used.

To summarize, three components in comparison to the original implementation of the ConvTransformer were changed:

- The scaling mechanism of each sequence in the batch
- The number of layers from three to five
- The loss function to a non-parametric quantile loss, which had the highest influence on performance

Otherwise, the source code of the ConvTransformer was followed and the tuned hyperparameters were adopted except for the learning rate, which was reduced to 0.0002 for all experiments for stable training.

6.1.3 Overview of the tested Positional Encodings

The ConvTransformer used an absolute learned position embedding and time features were provided as input. The time features for the traffic dataset are the hour, weekday, month and age of each data point. Each time feature was standard scaled. These features were also used in the benchmark on the traffic data by DeepAR (Salinas, Flunkert and Gasthaus, 2019), which is based on LSTMs .
It will be explored if it makes a difference to either simply concatenate the time features or to enrich the positional encoding with the temporal embedding. Also, the performance of absolute learned position embeddings, sinusoidal position encodings and the relative position encoding by (Shaw, Uszkoreit and Vaswani, 2018) will be compared. Additionally, it will be tested if it is possible to omit any traditional position encoding and only use the temporal embedding.

In total six different combinations of positional encodings are compared:

1. Learned absolute embedding with time features concatenated, the original encoding
2. Fixed sinusoidal absolute encoding with time features concatenated
3. Learned absolute embedding enriched with the temporal embedding
4. Fixed sinusoidal absolute encoding enriched with the temporal embedding
5. Temporal embedding without any additional conventional positional encoding
6. Relative positional encoding enriched with the temporal embedding

For each solution a model will be trained three times with different random seeds and for each seed five model checkpoints of the epochs with the best accuracy on the validation set are kept. Therefore, each positional encoding will be evaluated 15 times on the test set. Except from the positional encoding all hyperparameters stay the same and each model is trained for 15 epochs. The training and validation loss for each model was observed to ensure that every model with a different positional encoding is fitted equally well. The losses showed the same characteristics and reached similar values for every positional encoding, therefore it can be concluded, that it is a fair comparison even without finetuning the hyperparameters for each positional encoding individually.

6.2 First Experiment: Regularly Spaced Time Series

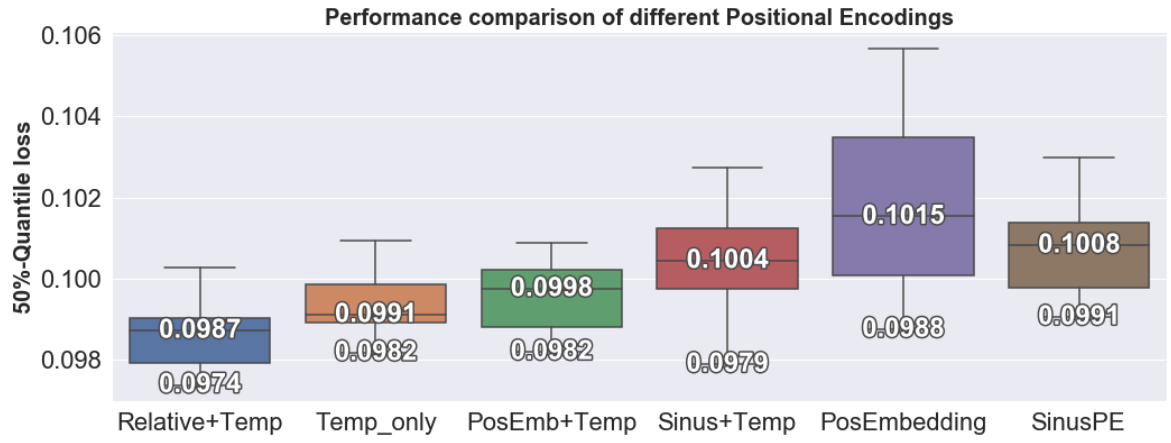

Figure 12: Median and best results of the 50%-Quantile forecast for the tested positional encodings. The relative positional encoding combined with the temporal embedding (Relative+Temp) performs best, followed by omitting conventional positional encodings and only using the temporal embedding (Temp_only). Combining a learned embedding with the temporal embedding (PosEmb+Temp) performs similar. With a slightly worse median but a larger variance is the sinusoidal encoding enriched with the temporal embedding (Sinus+Temp). The worst results are the learned embedding (PosEmbedding) and the fixed sinusoidal encoding (SinusPE) encodings with temporal information provided as covariates.

The best overall results of 0.0974 Q50 and 0.0748 Q90 were achieved with a relative positional encoding in combination with the temporal embedding (Figure 12: Relative+Temp). This is an improvement of 1.4% in the 50% quantile forecast in comparison to the best result of the originally used encoding, where a learned embedding with time covariates (Figure 12: PosEmbedding) was used the best run was 0.0988 Q50 and 0.0763 Q90.
The Q90 of all positional encodings was generally very similar and the average for each encoding was slightly above 0.075, but of course there can be some trade-off between Q50 and Q90 as both affect the total loss of the model. Therefore, it is also interesting to look at the median Q50 result of each encoding to avoid skewed results.

Generally, all positional encodings perform well and the difference of the best Q50 loss between best and worst performing method is less than 2%. This improvement range was to be expected, as for example in the research by (Shaw, Uszkoreit and Vaswani, 2018) the relative position encoding outperformed the absolute encoding in most experiments by less than 2%.
Therefore, the findings of the experiments in the Traffic Transformer (Cai *et al.*, 2020), where only a sinusoidal encoding performed well, are not generalizable. When they tested a sinusoidal encoding in combination with a temporal embedding or a sole temporal embedding, their performance decreased significantly. At least for the dataset used in this work's experiments this could not be replicated, and it seems, that many position encodings can be used with similar performance.

6.2.1 Time Features as Covariates vs Temporal Embedding

The temporal embedding appears to outperform traditional feature preparation. The median results and also the best runs of the absolute encodings without temporal enrichment are worse than all other positional encodings. The increase of computational complexity by learnable embeddings is a drawback but of small effect. Another disadvantage is, that the embeddings need to be designed deliberately and the design has to be tailored to the specific dataset, which comes with additional hyperparameter tuning. But for time series forecasting it is often the case that a model is continuously used and often it will make sense to find the right embedding to improve all future forecasts.

Also, the effect if no time information at all is included was measured, and the decrease was significant (Table1: WithoutTime). Conclusively, time information is crucial for an accurate forecast and the best performing solution is a temporal embedding.

6.2.2 Absolut Encoding: Learnable vs Fixed

The authors intuition was that the learnable positional embedding would be superior to a fixed sinusoidal encoding because it could possibly incorporate more dataset dependent information. But while the median Q50 result was slightly better for a learned embedding together with a temporal embedding (PosEmb+Temp) than the one with sinusoidal waves (Sinus+Temp), the distributions overlap and the best Q50 result of Sinus+Temp was superior to the PosEmb+Temp, and more experiments are needed for a conclusive judgement.

6.2.3 Relative Position Encoding and Temporal Embeddings Only

The best performing median results are for the relative encoding and the position encoding only depending on temporal embeddings. Which shows that a traditional absolute encoding is not needed and that the constructed temporal embedding can replace these encodings. This is contrary to (Cai *et al.*, 2020) where they tested a very similar temporal encoding but reported a strong performance decrease. Their loss almost doubled, which is unlikely to have only one cause, but in this work better results are noticeable when combining the sinusoidal global timestamp with the weekday and hour embeddings with a linear layer, which in combination with the global encoding seems to produce a better representation. For example, in the presented traffic benchmark the input window is eight days long and therefore, when only hour and day are encoded, a position is not uniquely identifiable. The global timestamp makes each position unique and by combining global timestep, hour and time embedding with learned parameters, the model is able to produce a better representation.

The relative positional encoding produced the best results and seems promising for future research on time series forecasting. Because the relative embedding parameters are learned autonomously according to the characteristics of the data, it is suitable for time series. Relative positional encodings are based on the distance of the datapoints, but the importance can still be learned. This harmonizes well with time series forecasting with seasonalities, because there, the distance between periodicities is constant and the importance for such

distant points is high. Admittedly, the additional improvement to temporal embeddings is only minimal and more experiments are needed.

6.3 Second Experiment: Irregularly Spaced Time Series; Introducing Lookback

To further examine how positional encodings affect time series forecasting with seasonalities another experiment with more complex temporal dependencies within the input sequence will be conducted. The context window will be irregularly spaced, the distance between each datapoint in a sequence will not be equal. But each individual sequence will have the same structure, therefore it is not completely irregular.

Excluding insignificant data from a sequence could improve performance. Formulated in a different manner, additionally to the time window directly preceding the forecast window, a lookback window, which is further in the past, will be added and some data will be skipped (Figure 10). This lookback window repeats a seasonality or could even include the seasonality for the first time and could thereby help modelling existing patterns. The excluded time frame will always be equal and therefore each sampled sequence will show the same temporal structure. The most important prerequisite is the positional encoding because it has to model the similarity of the lookback window to the forecast window but also has to portray that the lookback window is further in the past.

This lookback feature is not only a way to explore the capacities of positional encodings but could generally be useful for forecasting time series with seasonalities. It takes advantage of the non-sequential processing of transformers and simultaneously limits the biggest drawback of transformers.

6.3.1 Drawback of Transformers

The transformer excels because with self-attention it computes long term dependencies directly and independently. If the data is sampled in a high frequency or generally seasonalities occur over a long time frame, it would be necessary to have a very long input sequence, which includes all seasonalities, as otherwise the self-attention can not attend to the relevant points in the past. This is a common problem, as for example yearly holidays have a strong effect on many types of time series.

The question arises, why the input to the transformer can not contain a whole year to include the annual seasonality. In contrast to RNNs there is no bottleneck and with self-attention direct dependencies even over a long sequence could be learned.

Unfortunately, with self-attention the most critical drawback of transformers occurs as a side-effect. Because every element attends to every other element of a sequence the computation and memory scales with the sequence length quadratically (Qiu *et al.*, 2020). Many solutions with NLP tasks in mind were suggested (Tay *et al.*, 2020) and also in time series forecasting with transformers this was addressed early on (Li *et al.*, 2020).

The most common solution to alleviate the quadratic time and memory complexity is to restrict the self-attention.
Many transformer based models succeeded to use some form of attention matrix sparsification without performance degradation (Child *et al.*, 2019). Since the attention scores without limitations already form a long tail distribution, after the softmax function these values are of minor impact anyway (Zhou *et al.*, 2021). The objective of a sparsification is then to only exclude the dispensable values.
For example, (Zhou *et al.*, 2021) define ProbSparse self-attention for time series forecasting. Instead of computing the entire dot product between Key and Query, they define an easier to compute likeness function to find the dominant pairs and allow each Key to only attend to an arbitrary number of dominant Queries.

Li et al. (2019) propose the LogSparse self-attention, as a decoder-only model in which each token only attends to itself and its previous tokens they limit the attention mask further by only attending to elements with an exponential step size resulting in $O(log L)$ complexity instead of $O(L^2)$. If at least $O(log L)$ stacked layers are used the model then can access every cell's information. Additionally, they propose further expansions which slightly loosen up the limitations. A local attention range lets an element attend to directly preceding data points and restart attention resets the LogSparse attention within the sequence.

6.3.2 The Lookback Feature

The Lookback Feature enables a different approach. To still include all seasonalities, less informative parts of the input window will be skipped by applying prior knowledge about seasonalities. This would enable the model to form direct dependencies even for temporal distant data points without being exposed to full window length quadratic complexities.

This is not easily doable with RNNs because RNNs process data sequentially and assume that the time distance between two consecutive points is equally long. This is mostly an advantage because it incorporates relative relationships. Therefore, to the authors knowledge it has not been tried.

Also, for NLP this would be a very difficult and less applicable task, and even if it were applicable, it would require lots of manual pre-processing to mark unimportant passages. On the other hand, for time series with periodicities over long sequences, it is much easier. Normally, one would know about periodicities in the data or use autocorrelation and data visualization to uncover such behaviours, which enables exclusion of uninformative subsets.

The attention matrix sparsification could even be used together with the Lookback feature, but first a suitable position encoding has to be implemented. To analyse if this is possible, the Lookback Feature will be used on the traffic dataset. The traffic dataset does not even cover a whole year and might not even show significant annual periodicities, instead the weekday of the forecast-window will be included once again. To reiterate, the traffic benchmark uses one week as context-window to forecast the following day, therefore the input sequence is eight days long and the weekday of the forecast-window will be included once in the context-window. In the following experiments an additional day will included, which is two weeks before the forecast-window, hence the input sequence is nine days long and the context-window repeats the weekday of the forecast-window twice (Figure 10). Of course, with this dataset there is no additional seasonality included in this lookback window, therefore any possible improvement in comparison to only using one week as context would be small.

6.3.3 Model Adaptations for the Lookback Feature

All hyperparameters are identical to Section 6.4 and again each experiment is performed three times and evaluated on the test set for the five best validation losses per run for a total of 15 results for each positional encoding. This also helps to validate the prior experiments with more data.

The only adaptation to the model is regarding the 1D-CNN for the local context. The 1D-CNN also models relative position in its receptive field and assumes therefore similar to RNNs that all timesteps in a sequence are of equal distance. Because between the Lookback-window and current- window is a gap in time, one can not simply concatenate the sequences and apply the 1D-CNN. Instead, the model processes the Lookback-window separately as described in Section 3.4 and then a new zero padding is concatenated in front of the current-window.

6.3.4 Results of the Lookback Experiments

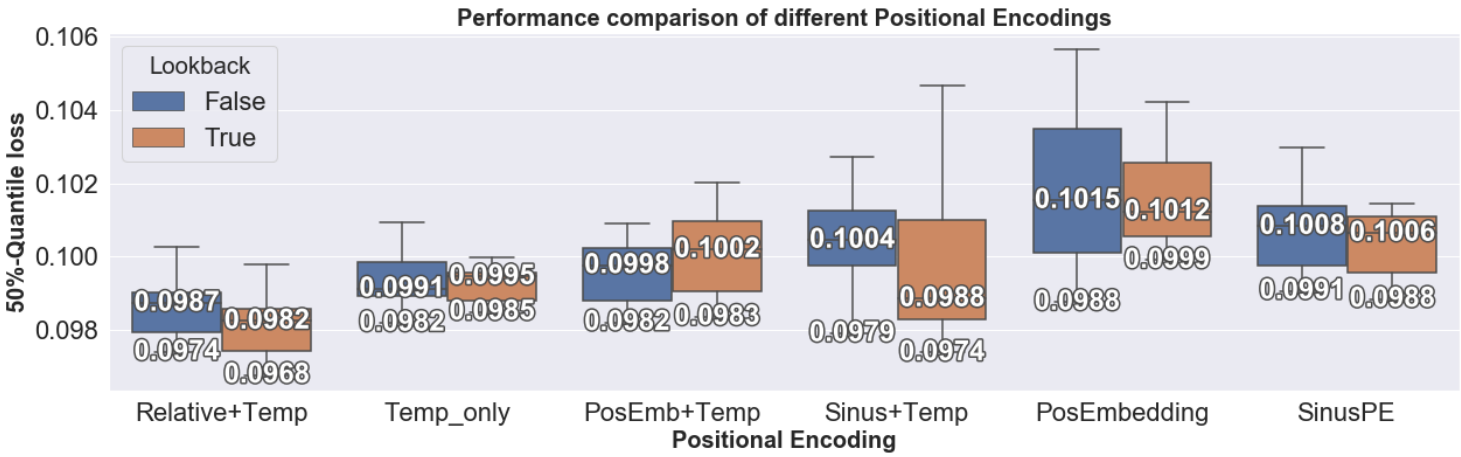

Figure 13: Median and best results of the 50%-Quantile forecast for the tested positional encodings with and without the additional lookback-window. Without Lookback one week is used as context, but with Lookback the day two weeks before the day of the forecast-window is concatenated to the context. Therefore, with Lookback the weekday of the forecast target is repeated twice in the context.

As expected, the positive effect of the additional lookback-window was small, because with this dataset it does not add an unseen seasonality. Still, by repeating the day of the forecast and therefore allowing the self-attention to form dependencies to two days, which are similar to the target, Lookback could have a smoothening effect. If the day a week before the forecast was atypical, Lookback could have a positive influence.
The result of the relative positional encoding with Lookback shows a slight improvement of 0.5%. That the relative positional encoding was able to improve with the Lookback was expected, as relative distance, which relative positional encoding models, is crucial for irregular time series (Sun *et al.*, 2021). But even though the relative distance is not constant between every datapoint, it is constant within every sequence. The time jump always occurs after the 24^{th} element because then the lookback-window ends, therefore it was surprising that the other position encodings were not able to model and improve with the lookback-window. The only other position encoding, which showed performance improvement, was the sinusoidal enriched with the temporal embedding. But it also displayed a larger variance, which makes it more difficult to assess this conclusively and it could be that the result of the first experiment was already skewed. The sinusoidal encoding enriched with the temporal embedding already showed a large variance in the first experiment and the best Q50 result also showed a good performance.

An explanation for the superior performance of the relative positional encoding in all experiments could be, that it is advantageous to force the model to encode each relationship. But it is also possible, that this improvement is achieved because more learned parameters are available as the relative positional encoding is freshly introduced in every layer in contrast to absolute encodings. Another possibility is that relative positional encodings perform better because the positional information is inserted into the attention mechanism instead into the input, which is argued in (Chen *et al.*, 2021).

In general, the lookback experiment confirms the conclusions of the first experiment and enriching the position encodings with temporal embeddings again outperformed time covariates. In future experiments this result could be reviewed for deseasonalized time series and time series without seasonalities.
Otherwise, the relative position encoding and solely the temporal embeddings performed superior and are very consistent. This is an interesting result because both are practically never used for time series forecasting and these results directly contradict the findings of the TrafficTransformer (Cai *et al.*, 2020), where they tested a very similar version of only using

temporal embeddings with a global timestep and this solution was significantly outperformed. Of course, the obtained results have to be verified on different datasets, but it seems to be possible to omit a traditional positional encoding.
Also, a relative positional encoding seems promising for time series forecasting as it performed the best, admittedly the computational drawbacks are significant, because it is repeatedly calculated in every layer. But the recent advancements in research regarding relative positional encodings alleviate this drawback.
Moreover, it is often worthwhile to finetune the position encoding for a time series problem, even if it improves the performance only slightly. Because time series models are generally continuously used over a long time frame, these improvements add up and could be significant.

6.3.5 Lookback for Retail

Even though the Lookback Feature did not deliver significant improvements, the Lookback feature had some positive effect on the forecast performance with a relative positional encoding. Which shows, that with the correct positional encoding a transformer can model jumps in time. For different datasets this improvement could even be larger because it would otherwise not be possible to include the complete time frame. This could be the case for high frequency data or long-range periodicities like annual holidays.

For retail in particular, holidays have a strong effect but are rarely considered except by a holiday marker as a feature for a specific timestep. The explored temporal embedding could be expanded to also include holiday information and transformers could utilize this information much better in contrast to RNNs. First with the Lookback-window the seasonality could be included in the context-window but also with a good positional encoding it would be possible to capture effects which are caused by holidays but happen prior to the event.
For example, as stores are usually closed on Sundays and on some holidays, customers anticipate closed stores and might buy before such days. Of course, there can also be a general surplus of sales for the event, e.g. gift purchases for Christmas. (Qi *et al.*, 2021) observe a similar effect with planned promotions and propose Aliformer. Aliformer is a transformer for time series forecasting designed to incorporate plans for promotions. Before an announced promotion starts, the sales volume decreases and then increase when the product is cheaper. To enable the use of this information, the model needs to process the information in the

encoder. In the encoder the self-attention is bidirectional and can use the knowledge of future promotions at a certain timestep. Of course, this makes only sense if the customers know of events beforehand. Otherwise, they can not anticipate, and the effect will not occur. Holidays are always announced, and the effects would differ depending on which weekday they fall on. For example, if stores are closed on Monday, and shopping is limited on Sunday, a surplus can probably already be observed on Saturday. Maybe because Aliformer is focused on e-commerce, holidays are not separately considered. But in future research, it would be interesting to process holidays with the Lookback feature with a relative positional encoding and temporal embeddings in combination with bidirectional attention.

7 Conclusions

In this work an empirical comparison of different positional encodings on the performance of transformer for time series forecasting was conducted. Different absolute positional encodings and a relative encoding were evaluated on a popular benchmark traffic dataset. Two different experiments on this dataset were evaluated. The first followed the common standards for the benchmark and a probabilistic forecast was made. The second experiment altered the input to the model and made the input irregularly spaced. This experiment helped to examine the effect on performance of the positional encodings under different conditions. The results and conclusions to both experiments were similar.

The answer to the research question: “How do different positional encodings effect the performance if seasonalities are present” is, that different positional encodings have only a small impact on performance and all encodings delivered usable forecasts, which in itself is an interesting conclusion and contrary to some research. Still, some encodings performed slightly better.

Performance gains of temporal embeddings over the traditional use of time covariates were recorded. Even though, modern transformers for time series forecasting implement positional encodings enriched with temporal embeddings, comparative studies are hard to find. Moreover, some research, which was published before these models were introduced, showed poor performance for absolute positional encodings with temporal embeddings and temporal embeddings as a sole encoding. Both could not be confirmed.

Furthermore, a relative positional encoding, which is rarely used in research for time series forecasting, performed better than any absolute positional encoding. These gains in

comparison to absolute encodings are similar to performance improvements measured for NLP tasks.

The relative positional encoding had some influence on the performance improvement of a recent time series transformer model by up to 20%.

The largest impact on the increase of accuracy for the probabilistic forecast had the change of the loss function to the quantile loss. This effect was shown in other studies, but to the authors knowledge had not been documented for the ConvTransformer. This shows, that while different positional encodings can be explored for finetuning, other components of a model have a bigger influence on performance.

Still, the advancements of research on transformers for NLP are also beneficial for time series forecasting. Because time series and language processing are very similar and the state-of-the-art solutions for both are based on the same architecture, the problems and solutions are transferable. The rise of the strongest language generation models based on transformers goes hand in hand with the new influence of transformers for time series forecasting. But the research on time series forecasting is much more limited and modular components, which brought improvement for NLP transformers, have not been explored. With the conducted experiments on positional encodings, the effects on performance were analysed for one dataset, but in this process more interesting possibilities for future research were revealed. For example, the improvements on Layer Normalization with PowerNorm and DeepNorm could be easily used with transformers for time series forecasting and help with the required small learning rates.

8 References

Alexandrov, A. *et al.* (2019) 'GluonTS: Probabilistic Time Series Models in Python', *arXiv:1906.05264 [cs, stat]* [Preprint]. Available at: http://arxiv.org/abs/1906.05264 (Accessed: 15 February 2022).

Andersen, B.C. *et al.* (2021) 'Sub-second periodicity in a fast radio burst', *arXiv:2107.08463 [astro-ph]* [Preprint]. Available at: http://arxiv.org/abs/2107.08463 (Accessed: 14 February 2022).

Arnab, A. *et al.* (2021) 'ViViT: A Video Vision Transformer', *arXiv:2103.15691 [cs]* [Preprint]. Available at: http://arxiv.org/abs/2103.15691 (Accessed: 11 January 2022).

Arnerić, J. (2021) 'Multiple STL decomposition in discovering a multi-seasonality of intraday trading volume', *Croatian Operational Research Review*, 12(1), pp. 61–74. doi:10.17535/crorr.2021.0006.

Assimakopoulos, V. and Nikolopoulos, K. (2000) 'The theta model: A decomposition approach to forecasting', *International Journal of Forecasting*, 16, pp. 521–530. doi:10.1016/S0169-2070(00)00066-2.

Ba, J.L., Kiros, J.R. and Hinton, G.E. (2016) 'Layer Normalization', *arXiv:1607.06450 [cs, stat]* [Preprint]. Available at: http://arxiv.org/abs/1607.06450 (Accessed: 6 March 2022).

Bahdanau, D., Cho, K. and Bengio, Y. (2016) 'Neural Machine Translation by Jointly Learning to Align and Translate', *arXiv:1409.0473 [cs, stat]* [Preprint]. Available at: http://arxiv.org/abs/1409.0473 (Accessed: 5 February 2022).

Bandara, K., Bergmeir, C. and Hewamalage, H. (2021) 'LSTM-MSNet: Leveraging Forecasts on Sets of Related Time Series with Multiple Seasonal Patterns', *IEEE Transactions on Neural Networks and Learning Systems*, 32(4), pp. 1586–1599. doi:10.1109/TNNLS.2020.2985720.

Bandara, K., Bergmeir, C. and Smyl, S. (2020) 'Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach', *Expert Systems with Applications*, 140, p. 112896. doi:10.1016/j.eswa.2019.112896.

Bandara, K., Hyndman, R.J. and Bergmeir, C. (2021) 'MSTL: A Seasonal-Trend Decomposition Algorithm for Time Series with Multiple Seasonal Patterns', *arXiv:2107.13462 [stat]* [Preprint]. Available at: http://arxiv.org/abs/2107.13462 (Accessed: 16 February 2022).

Better Language Models and Their Implications (2019) *OpenAI*. Available at: https://openai.com/blog/better-language-models/ (Accessed: 3 February 2022).

Box, G.E.P. and Jenkins, G.M. (1970) *Time Series Analysis: Forecasting and Control*. Holden-Day.

Brown, R.G. (1956) *Exponential Smoothing for Predicting Demand*. Little.

Brown, T.B. *et al.* (2020) 'Language Models are Few-Shot Learners', *arXiv:2005.14165 [cs]* [Preprint]. Available at: http://arxiv.org/abs/2005.14165 (Accessed: 3 February 2022).

Bruzda, J. (2020) 'Multistep quantile forecasts for supply chain and logistics operations: bootstrapping, the GARCH model and quantile regression based approaches', *Central European Journal of Operations Research*, 28(1), pp. 309–336. doi:10.1007/s10100-018-0591-2.

Cai, L. *et al.* (2020) 'Traffic transformer: Capturing the continuity and periodicity of time series for traffic forecasting', *Transactions in GIS*, 24(3), pp. 736–755. doi:https://doi.org/10.1111/tgis.12644.

Chen, P. (2021) ‘PermuteFormer: Efficient Relative Position Encoding for Long Sequences’, in, pp. 10606–10618. doi:10.18653/v1/2021.emnlp-main.828.

Chen, P.-C. *et al.* (2021) ‘A Simple and Effective Positional Encoding for Transformers’, *arXiv:2104.08698 [cs]* [Preprint]. Available at: http://arxiv.org/abs/2104.08698 (Accessed: 8 February 2022).

Chen, Yitian *et al.* (2020) ‘Probabilistic Forecasting with Temporal Convolutional Neural Network’, *arXiv:1906.04397 [cs, stat]* [Preprint]. Available at: http://arxiv.org/abs/1906.04397 (Accessed: 14 June 2021).

Child, R. *et al.* (2019) ‘Generating Long Sequences with Sparse Transformers’, *arXiv:1904.10509 [cs, stat]* [Preprint]. Available at: http://arxiv.org/abs/1904.10509 (Accessed: 13 February 2022).

Cho, K. *et al.* (2014) ‘Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation’, *arXiv:1406.1078 [cs, stat]* [Preprint]. Available at: http://arxiv.org/abs/1406.1078 (Accessed: 20 January 2022).

Clark, K. *et al.* (2020) ‘ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators’, *arXiv:2003.10555 [cs]* [Preprint]. Available at: http://arxiv.org/abs/2003.10555 (Accessed: 22 January 2022).

Cleveland, R. *et al.* (1990) ‘STL: A seasonal-trend decomposition procedure based on loess (with discussion)’, *Journal of Official Statistics*, 6(1), pp. 3–33.

Crone, S., Hibon, M. and Nikolopoulos, K. (2011) ‘Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction’, *International Journal of Forecasting*, 27, pp. 635–660. doi:10.1016/j.ijforecast.2011.04.001.

Cuturi, M. (2011a) ‘Fast Global Alignment Kernels’, in *ICML*.

Cuturi, M. (2011b) *UCI Machine Learning Repository: PEMS-SF Data Set*. Available at: https://archive.ics.uci.edu/ml/datasets/PEMS-SF (Accessed: 24 February 2022).

Dang, W. *et al.* (2021) ‘TS-Bert: Time Series Anomaly Detection via Pre-training Model Bert’, in Paszynski, M. et al. (eds) *Computational Science – ICCS 2021*. Cham: Springer International Publishing (Lecture Notes in Computer Science), pp. 209–223. doi:10.1007/978-3-030-77964-1_17.

Daraghmeh, M. *et al.* (2021) ‘Time Series Forecasting using Facebook Prophet for Cloud Resource Management’, in *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*. *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 1–6. doi:10.1109/ICCWorkshops50388.2021.9473607.

Dastorani, M. *et al.* (2016) ‘Comparative study among different time series models applied to monthly rainfall forecasting in semi-arid climate condition’, *Natural Hazards*, 81(3), pp. 1811–1827. doi:10.1007/s11069-016-2163-x.

De Livera, A.M., Hyndman, R.J. and Snyder, R.D. (2011) ‘Forecasting Time Series With Complex Seasonal Patterns Using Exponential Smoothing’, *Journal of the American Statistical Association*, 106(496), pp. 1513–1527. doi:10.1198/jasa.2011.tm09771.

Devlin, J. *et al.* (2019) ‘BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding’, *arXiv:1810.04805 [cs]* [Preprint]. Available at: http://arxiv.org/abs/1810.04805 (Accessed: 22 January 2022).

Dokumentov, A. and Hyndman, R.J. (2021) 'STR: Seasonal-Trend Decomposition Using Regression', *arXiv:2009.05894 [stat]* [Preprint]. Available at: http://arxiv.org/abs/2009.05894 (Accessed: 16 February 2022).

Dosovitskiy, A. *et al.* (2021) 'An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale', *arXiv:2010.11929 [cs]* [Preprint]. Available at: http://arxiv.org/abs/2010.11929 (Accessed: 9 January 2022).

Du, H. (2020) *SOTA Attention Mechanism and Activation Functions on XLNet*, p. 11. Available at: https://www.semanticscholar.org/paper/SOTA-Attention-Mechanism-and-Activation-Functions-Du-Xie/17a56053a934aa0b6c85ee004bd5c684a325d6be (Accessed: 6 March 2022).

Dubey, S.R., Singh, S.K. and Chaudhuri, B.B. (2022) 'A Comprehensive Survey and Performance Analysis of Activation Functions in Deep Learning', *arXiv:2109.14545 [cs]* [Preprint]. Available at: http://arxiv.org/abs/2109.14545 (Accessed: 6 March 2022).

Dufter, P., Schmitt, M. and Schütze, H. (2021) 'Position Information in Transformers: An Overview', *arXiv:2102.11090 [cs]* [Preprint]. Available at: http://arxiv.org/abs/2102.11090 (Accessed: 17 February 2022).

Edunov, S. *et al.* (2018) 'Understanding Back-Translation at Scale', *arXiv:1808.09381 [cs]* [Preprint]. Available at: http://arxiv.org/abs/1808.09381 (Accessed: 10 March 2022).

Elsayed, S. *et al.* (2021) 'Do We Really Need Deep Learning Models for Time Series Forecasting?', *arXiv:2101.02118 [cs, stat]* [Preprint]. Available at: http://arxiv.org/abs/2101.02118 (Accessed: 20 December 2021).

Gardner Jr., E.S. (1985) 'Exponential smoothing: The state of the art', *Journal of Forecasting*, 4(1), pp. 1–28. doi:10.1002/for.3980040103.

Gehring, J. *et al.* (2017) 'Convolutional Sequence to Sequence Learning', *arXiv:1705.03122 [cs]* [Preprint]. Available at: http://arxiv.org/abs/1705.03122 (Accessed: 8 February 2022).

Hewamalage, H., Bergmeir, C. and Bandara, K. (2021) 'Recurrent Neural Networks for Time Series Forecasting: Current status and future directions', *International Journal of Forecasting*, 37(1), pp. 388–427. doi:10.1016/j.ijforecast.2020.06.008.

Hochreiter, S. and Schmidhuber, J. (1997) 'Long Short-Term Memory', *Neural Computation*, 9(8), pp. 1735–1780. doi:10.1162/neco.1997.9.8.1735.

Holst, A. (2021) *Total data volume worldwide 2010-2025*, *Statista*. Available at: https://www.statista.com/statistics/871513/worldwide-data-created/ (Accessed: 7 January 2022).

Huang, C.-Z.A. *et al.* (2018) 'Music Transformer', *arXiv:1809.04281 [cs, eess, stat]* [Preprint]. Available at: http://arxiv.org/abs/1809.04281 (Accessed: 27 December 2021).

Huang, Z. *et al.* (2020) 'Improve Transformer Models with Better Relative Position Embeddings', *arXiv:2009.13658 [cs]* [Preprint]. Available at: http://arxiv.org/abs/2009.13658 (Accessed: 26 February 2022).

Hyndman, R.J. and Athanasopoulos, G. (2018) '2.3 Time series patterns', in *Forecasting: Principles and Practice (2nd ed)*. Available at: https://Otexts.com/fpp2/ (Accessed: 14 February 2022).

Lan, Z. *et al.* (2020) 'ALBERT: A Lite BERT for Self-supervised Learning of Language Representations', *arXiv:1909.11942 [cs]* [Preprint]. Available at: http://arxiv.org/abs/1909.11942 (Accessed: 23 January 2022).

Lecun, Y. *et al.* (1998) 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE*, 86(11), pp. 2278–2324. doi:10.1109/5.726791.

Lewis, M. *et al.* (2019) 'BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension', *arXiv:1910.13461 [cs, stat]* [Preprint]. Available at: http://arxiv.org/abs/1910.13461 (Accessed: 22 January 2022).

Li, J., Wang, Y. and McAuley, J. (2020) 'Time Interval Aware Self-Attention for Sequential Recommendation', in *Proceedings of the 13th International Conference on Web Search and Data Mining*. New York, NY, USA: Association for Computing Machinery, pp. 322–330. Available at: https://doi.org/10.1145/3336191.3371786 (Accessed: 5 March 2022).

Li, M. *et al.* (2021) 'TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models', *arXiv:2109.10282 [cs]* [Preprint]. Available at: http://arxiv.org/abs/2109.10282 (Accessed: 9 January 2022).

Li, S. *et al.* (2020) 'Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting', *arXiv:1907.00235 [cs, stat]* [Preprint]. Available at: http://arxiv.org/abs/1907.00235 (Accessed: 9 December 2021).

Lim, B. *et al.* (2020) 'Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting', *arXiv:1912.09363 [cs, stat]* [Preprint]. Available at: http://arxiv.org/abs/1912.09363 (Accessed: 9 December 2021).

Lin, Z. *et al.* (2017) 'A Structured Self-attentive Sentence Embedding', *arXiv:1703.03130 [cs]* [Preprint]. Available at: http://arxiv.org/abs/1703.03130 (Accessed: 5 February 2022).

Liu, Z. *et al.* (2021) 'Swin Transformer: Hierarchical Vision Transformer using Shifted Windows', *arXiv:2103.14030 [cs]* [Preprint]. Available at: http://arxiv.org/abs/2103.14030 (Accessed: 9 February 2022).

Liu, Z. *et al.* (2022) 'A ConvNet for the 2020s', *arXiv:2201.03545 [cs]* [Preprint]. Available at: http://arxiv.org/abs/2201.03545 (Accessed: 9 February 2022).

Luo, S. *et al.* (2021) 'Stable, Fast and Accurate: Kernelized Attention with Relative Positional Encoding', in. *Advances in Neural Information Processing Systems*. Available at: https://openreview.net/forum?id=X7XNPor93uG (Accessed: 19 February 2022).

Luong, M.-T., Pham, H. and Manning, C.D. (2015) 'Effective Approaches to Attention-based Neural Machine Translation', *arXiv:1508.04025 [cs]* [Preprint]. Available at: http://arxiv.org/abs/1508.04025 (Accessed: 5 February 2022).

Makridakis, S. and Hibon, M. (2000) 'The M3-Competition: results, conclusions and implications'. doi:10.1016/S0169-2070(00)00057-1.

Mohammdi Farsani, R. and Pazouki, E. (2021) 'A Transformer Self-attention Model for Time Series Forecasting', *Journal of Electrical and Computer Engineering Innovations (JECEI)*, 9(1), pp. 1–10. doi:10.22061/jecei.2020.7426.391.

Montero-Manso, P. and Hyndman, R.J. (2021) 'Principles and Algorithms for Forecasting Groups of Time Series: Locality and Globality', *arXiv:2008.00444 [cs, stat]* [Preprint]. Available at: http://arxiv.org/abs/2008.00444 (Accessed: 10 May 2021).

O'Dea, S. (2020) *Global IoT connections data volume 2019 and 2025*, *Statista*. Available at: https://www.statista.com/statistics/1017863/worldwide-iot-connected-devices-data-size/ (Accessed: 7 January 2022).

Ollech, D. (2021) 'Seasonal Adjustment of Daily Time Series', *Journal of Time Series Econometrics*, 13(2), pp. 235–264. doi:10.1515/jtse-2020-0028.

Oord, A. van den *et al.* (2016) 'WaveNet: A Generative Model for Raw Audio', *arXiv:1609.03499 [cs]* [Preprint]. Available at: http://arxiv.org/abs/1609.03499 (Accessed: 31 May 2021).

Paulus, R., Xiong, C. and Socher, R. (2017) 'A Deep Reinforced Model for Abstractive Summarization', *arXiv:1705.04304 [cs]* [Preprint]. Available at: http://arxiv.org/abs/1705.04304 (Accessed: 5 February 2022).

Qi, X. *et al.* (2021) 'From Known to Unknown: Knowledge-guided Transformer for Time-Series Sales Forecasting in Alibaba', *arXiv:2109.08381 [cs]* [Preprint]. Available at: http://arxiv.org/abs/2109.08381 (Accessed: 20 February 2022).

Qin, Y. *et al.* (2017) 'A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction', *arXiv:1704.02971 [cs, stat]* [Preprint]. Available at: http://arxiv.org/abs/1704.02971 (Accessed: 5 February 2022).

Qiu, J. *et al.* (2020) 'Blockwise Self-Attention for Long Document Understanding', *arXiv:1911.02972 [cs]* [Preprint]. Available at: http://arxiv.org/abs/1911.02972 (Accessed: 12 February 2022).

Radford, A. *et al.* (2018) 'Improving Language Understanding by Generative Pre-Training', p. 12.

Radford, A. *et al.* (2019) 'Language Models are Unsupervised Multitask Learners'. Available at: https://www.semanticscholar.org/paper/Language-Models-are-Unsupervised-Multitask-Learners-Radford-Wu/9405cc0d6169988371b2755e573cc28650d14dfe (Accessed: 3 February 2022).

Raffel, C. *et al.* (2020) 'Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer', *arXiv:1910.10683 [cs, stat]* [Preprint]. Available at: http://arxiv.org/abs/1910.10683 (Accessed: 21 January 2022).

Salinas, D., Flunkert, V. and Gasthaus, J. (2019) 'DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks', *arXiv:1704.04110 [cs, stat]* [Preprint]. Available at: http://arxiv.org/abs/1704.04110 (Accessed: 2 June 2021).

Sanh, V. *et al.* (2020) 'DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter', *arXiv:1910.01108 [cs]* [Preprint]. Available at: http://arxiv.org/abs/1910.01108 (Accessed: 23 January 2022).

Shaw, P., Uszkoreit, J. and Vaswani, A. (2018) 'Self-Attention with Relative Position Representations', *arXiv:1803.02155 [cs]* [Preprint]. Available at: http://arxiv.org/abs/1803.02155 (Accessed: 8 February 2022).

Shen, S. *et al.* (2020) 'PowerNorm: Rethinking Batch Normalization in Transformers', in *Proceedings of the 37th International Conference on Machine Learning*. *International Conference on Machine Learning*, PMLR, pp. 8741–8751. Available at: https://proceedings.mlr.press/v119/shen20e.html (Accessed: 15 January 2022).

Sinha, K. *et al.* (2021) 'Masked Language Modeling and the Distributional Hypothesis: Order Word Matters Pre-training for Little', *arXiv:2104.06644 [cs]* [Preprint]. Available at: http://arxiv.org/abs/2104.06644 (Accessed: 1 February 2022).

Song, H. *et al.* (2017) 'Attend and Diagnose: Clinical Time Series Analysis using Attention Models'. Available at: https://arxiv.org/abs/1711.03905v2 (Accessed: 1 June 2021).

Sun, C. *et al.* (2021) 'TE-ESN: Time Encoding Echo State Network for Prediction Based on Irregularly Sampled Time Series Data', *arXiv:2105.00412 [cs]* [Preprint]. Available at: http://arxiv.org/abs/2105.00412 (Accessed: 19 February 2022).

Sun, W. and Li, Z. (2020) 'Hourly PM2.5 concentration forecasting based on feature extraction and stacking-driven ensemble model for the winter of the Beijing-Tianjin-Hebei area', *Atmospheric Pollution Research*, 11(6), pp. 110–121. doi:10.1016/j.apr.2020.02.022.

Sutskever, I., Vinyals, O. and Le, Q.V. (2014) 'Sequence to Sequence Learning with Neural Networks', *arXiv:1409.3215 [cs]* [Preprint]. Available at: http://arxiv.org/abs/1409.3215 (Accessed: 20 January 2022).

Tagasovska, N. and Lopez-Paz, D. (2019) 'Single-Model Uncertainties for Deep Learning', *Advances in Neural Information Processing Systems*, 32. Available at: https://papers.nips.cc/paper/2019/hash/73c03186765e199c116224b68adc5fa0-Abstract.html (Accessed: 23 June 2021).

Tang, W. *et al.* (2021) 'Rethinking 1D-CNN for Time Series Classification: A Stronger Baseline', *arXiv:2002.10061 [cs, stat]* [Preprint]. Available at: http://arxiv.org/abs/2002.10061 (Accessed: 22 May 2021).

Tay, Y. *et al.* (2020) 'Efficient Transformers: A Survey', *arXiv:2009.06732 [cs]* [Preprint]. Available at: http://arxiv.org/abs/2009.06732 (Accessed: 5 December 2021).

The Economist (2017) 'The world's most valuable resource is no longer oil, but data', 6 May. Available at: https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data (Accessed: 7 January 2022).

Tobback, E. and Martens, D. (2019) 'Retail credit scoring using fine-grained payment data', *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 182(4), pp. 1227–1246. doi:10.1111/rssa.12469.

Tsay, R.S. (2000) 'Time Series and Forecasting: Brief History and Future Research', *Journal of the American Statistical Association*, 95(450), pp. 638–643. doi:10.2307/2669408.

Vagropoulos, S.I. *et al.* (2016) 'Comparison of SARIMAX, SARIMA, modified SARIMA and ANN-based models for short-term PV generation forecasting', in *2016 IEEE International Energy Conference (ENERGYCON)*. *2016 IEEE International Energy Conference (ENERGYCON)*, pp. 1–6. doi:10.1109/ENERGYCON.2016.7514029.

Vaswani, A. *et al.* (2017) 'Attention Is All You Need', *arXiv:1706.03762 [cs]* [Preprint]. Available at: http://arxiv.org/abs/1706.03762 (Accessed: 5 January 2022).

Wang, B. *et al.* (2020) 'Encoding word order in complex embeddings', *arXiv:1912.12333 [cs]* [Preprint]. Available at: http://arxiv.org/abs/1912.12333 (Accessed: 8 February 2022).

Wang, H. *et al.* (2022) 'DeepNet: Scaling Transformers to 1,000 Layers', *arXiv:2203.00555 [cs]* [Preprint]. Available at: http://arxiv.org/abs/2203.00555 (Accessed: 6 March 2022).

Wang, Y.-A. and Chen, Y.-N. (2020) 'What Do Position Embeddings Learn? An Empirical Study of Pre-Trained Language Model Positional Encoding', *arXiv:2010.04903 [cs]* [Preprint]. Available at: http://arxiv.org/abs/2010.04903 (Accessed: 8 February 2022).

Wu, H. *et al.* (2021) 'Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting', *arXiv:2106.13008 [cs]* [Preprint]. Available at: http://arxiv.org/abs/2106.13008 (Accessed: 29 December 2021).

Xiong, R. *et al.* (2020) 'On Layer Normalization in the Transformer Architecture', *arXiv:2002.04745 [cs, stat]* [Preprint]. Available at: http://arxiv.org/abs/2002.04745 (Accessed: 6 March 2022).

Yang, B. *et al.* (2019) 'Assessing the Ability of Self-Attention Networks to Learn Word Order', in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. *ACL 2019*, Florence, Italy: Association for Computational Linguistics, pp. 3635–3644. doi:10.18653/v1/P19-1354.

Zeroual, A. *et al.* (2020) 'Deep learning methods for forecasting COVID-19 time-Series data: A Comparative study', *Chaos, Solitons & Fractals*, 140, p. 110121. doi:10.1016/j.chaos.2020.110121.

Zerveas, G. *et al.* (2020) 'A Transformer-based Framework for Multivariate Time Series Representation Learning', *arXiv:2010.02803 [cs]* [Preprint]. Available at: http://arxiv.org/abs/2010.02803 (Accessed: 14 January 2022).

Zhou, H. *et al.* (2021) 'Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting', *arXiv:2012.07436 [cs]* [Preprint]. Available at: http://arxiv.org/abs/2012.07436 (Accessed: 3 January 2022).