The Evolution of Transformer Models for Long-Term Forecasting

Zain ul Abideen · Follow

11 min read · May 2, 2023

Listen Share More

Everything you need to know about Long-Term Forecasting and Transformers.

Photo by Chris Liverani on Unsplash

Time series forecasting is an important problem across many domains, including predictions of solar plant energy output, reliable prediction of the online traffic for micro-services, and guidance for dynamic resource allocation to minimize the cost without degrading the performance. Time series often exhibit both short-term and long-term repeating patterns. The main challenge is to capture long-range dependencies while reducing time and space complexity. Different transformer [1] architectures have been proposed to solve this problem of enhancement of locality.

This article discusses 4 novel transformer architectures specialized in long-term series forecasting. Specifically, these are:

1. **LogTrans** [2]
2. **FEDformer** [3]
3. **Pyraformer** [4]
4. **PatchTST** [5]

The self-attention based Transformer has been proposed for sequence modeling and has achieved great success. Several recent works apply it to translation, speech, music and image generation. However, scaling attention to extremely long sequences is computationally prohibitive since the space complexity of self-attention grows quadratically with sequence length. This becomes a serious issue in forecasting time series with fine granularity and strong long-term dependencies

I will briefly introduce transformer architecture here. In the self-attention layer, a multi-head self-attention sublayer simultaneously transforms Y into H distinct query matrices Q (h), key matrices K (h), and value matrices V (h) respectively, with h = 1, ⋯ , H. After these linear projections, the scaled dot-product attention computes a sequence of vector outputs:

$$\mathbf{O}_h = \text{Attention}(\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h) = \text{softmax}\left(\frac{\mathbf{Q}_h \mathbf{K}_h^T}{\sqrt{d_k}} \cdot \mathbf{M}\right) \mathbf{V}_h.$$

Figure 1: Computing matrix of outputs (Source)

Afterward, O(1), O(2), ⋯, O(H) are concatenated and linearly projected again. Upon the attention output, a position-wise feedforward sublayer with two layers of a fully-connected network and a ReLU activation in the middle is stacked.

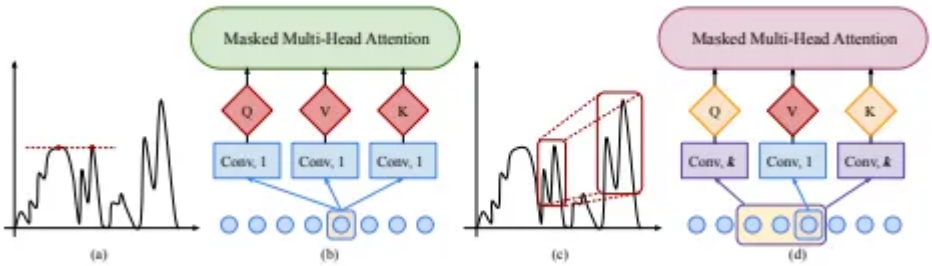
LOGTRANS

This is the first model that enhances the transformer's locality and also breaks the memory bottleneck of transformer. In order to solve the issue of locality and memory, this model uses convolutional self-attention by producing queries and keys with causal convolution so that local context can be better incorporated into attention mechanism. LogSparse Transformer only requires O(L(log L)2) memory cost, improving forecasting accuracy for time series with fine granularity and strong long-term dependencies under constrained memory budget.

Now we will discuss model's key methodologies:

Enhancing the locality of Transformer:

Patterns in time series may evolve with time significantly due to various events, e.g. holidays and extreme weather, so whether an observed point is an anomaly, change point or part of the patterns is highly dependent on its surrounding context. However, in the self-attention layers of canonical Transformer, the similarities between queries and keys are computed based on their point-wise values without fully leveraging local context like shape, as shown in Figure 3(a) and (b).



Comparison between canonical and causal convolutional self-attention layers

This model uses causal convolution of kernel size k with stride 1 to transform inputs (with proper paddings) into queries and keys. Causal convolutions ensure that the current position never has access to future information. By employing causal convolution, generated queries and keys can be more aware of local context and hence, compute their similarities by their local context information, e.g. local shapes, instead of point-wise values, which can be helpful for accurate forecasting.

Breaking the memory bottleneck of Transformer:

In order to reduce the memory cost, the model introduces sparsity in the canonical transformer. LogSparse Transformer needs to calculate O(log L) dot products for

each cell in each layer. Further, we only need to stack up to O(log L) layers and the model will be able to access every cell's information. Hence, the total cost of memory usage is only O(L(log L)2). LogSparse self-attention allows each cell only to attend to its previous cells with exponential step size and itself.

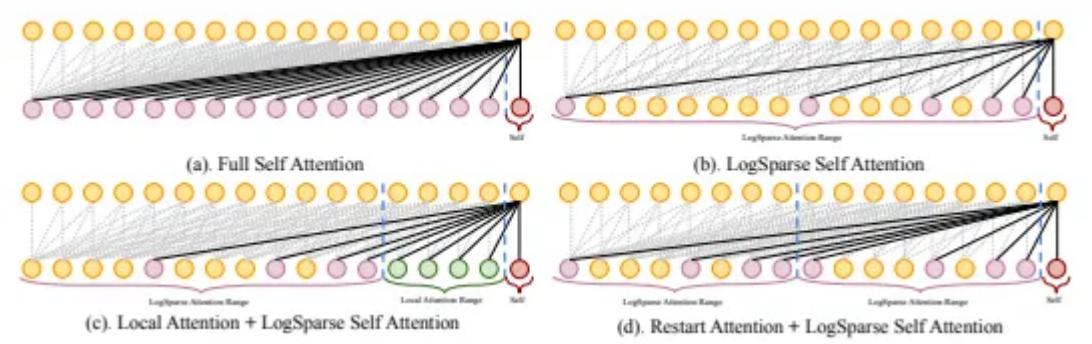


Illustration of different attention mechanism between adjacent layers in Transformer

The model can use Local, Restart or both attentions with LogSparse self-attention. Local Attention allows each cell to densely attend to cells in its left window of size O(log2 L) so that more local information, e.g. trend, can be leveraged for current step forecasting. Restart Attention divides the whole input with length L into subsequences and set each subsequence length L directly proportional to L.

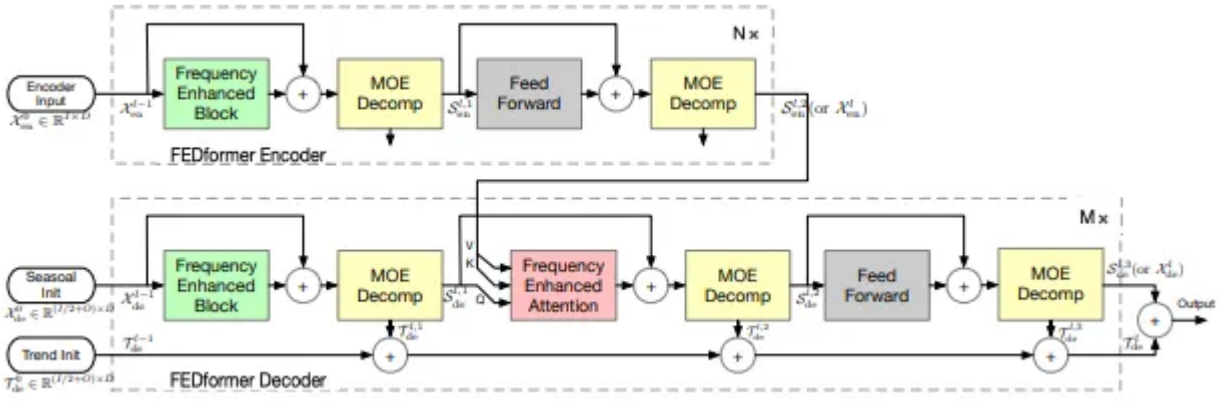
FEDFORMER

This model combines Transformer with the seasonal-trend decomposition method, in which the decomposition method captures the global profile of time series while Transformers capture more detailed structures. As time series tend to have a sparse representation in well-known basis such as Fourier transform, and develop a frequency-enhanced Transformer. Besides being more effective, the proposed method, termed as Frequency Enhanced Decomposed Transformer (FEDformer), is more efficient than standard Transformer with a linear complexity to the sequence length. FEDformer reduces prediction error by 14.8% and 22.6% for multivariate and univariate time series, respectively.

This model presents a special design of network that is effective in bringing the distribution of prediction close to that of ground truth, according to Kologrov-Smirnov distribution test. It also combines Fourier analysis with the Transformer based method. Instead of applying Transformer to the time domain, it applies it to the frequency domain which helps Transformer better capture global properties of time series.

Which subset of frequency components should be used by Fourier analysis to represent time series?

A randomly selected subset of frequency components, including both low and high ones, will give a better representation of time series. This allows us to reduce the computational cost of Transformer from quadratic to linear complexity. Using all the Fourier components allows us to best preserve the history information in the time series, it will potentially lead to overfitting.

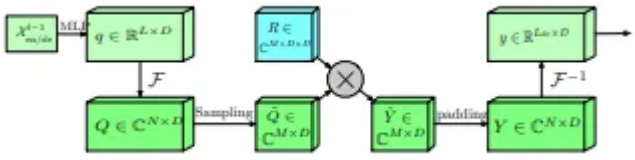


FEDformer Architecture

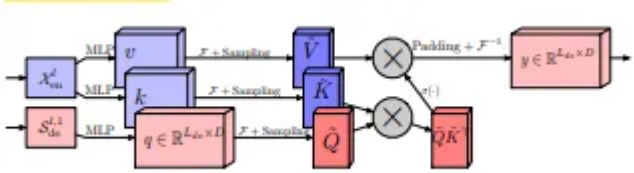
The model has Frequency Enhanced Block (FEB), Frequency Enhanced Attention (FEA) connecting encoder and decoder, and the Mixture Of Experts Decomposition block (MOEDecomp).

Fourier Enhanced Structure:

Fourier Enhanced Structures use discrete Fourier transform (DFT). It includes Frequency Enhanced Block with Fourier Transform (FEB-f) and Frequency Enhanced Attention with Fourier Transform (FEA-f).



Frequency Enhanced Block with Fourier Transform (FEB-f)

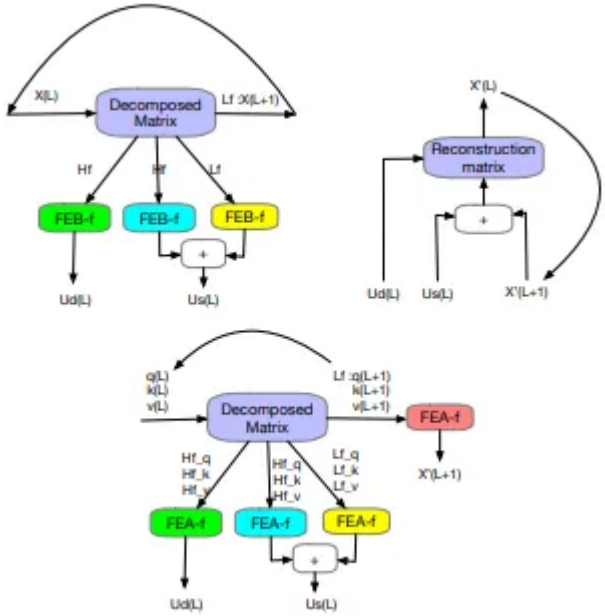


Frequency Enhanced Attention with Fourier Transform (FEA-f)

The FEB-f is used in both encoder and decoder as shown in Figure 2. The input (x) of the FEB-f block is first linearly projected with w, so q = x·w. Then q is converted from the time domain to the frequency domain. The Fourier transform of q is denoted as Q. In FEA-f, we convert the queries, keys, and values with Fourier Transform and perform a similar attention mechanism in the frequency domain, by randomly selecting M modes. We denote the selected version after FourierTransform as Q˜, K˜, V˜.

Wavelet enhanced blocks:

While the Fourier transform creates a representation of the signal in the frequency domain, the Wavelet transform creates a representation in both the frequency and time domain, allowing efficient access of localized information of the signal.



Top Left: Wavelet frequency enhanced block decomposition stage. Top Right: Wavelet block reconstruction stage shared by FEB-w and FEA-w. Bottom: Wavelet frequency enhanced cross attention decomposition stage

Frequency Enhanced Block with Wavelet Transform (FEB-w):

The overall FEB-w architecture is shown in Figure . It differs from FEB-f in the recursive mechanism: the input is decomposed into 3 parts recursively and operates individually. For the wavelet decomposition part, we implement the fixed Legendre wavelets basis decomposition matrix. Three FEB-f modules are used to process the resulting high-frequency part, low-frequency part, and remaining part from wavelet decomposition respectively.

Frequency Enhanced Attention with Wavelet Transform (FEA-w):

FEA-w contains the decomposition stage and reconstruction stage like FEB-w. Here the reconstruction stage remains unchanged. The only difference lies in the decomposition stage. The same decomposed matrix is used to decompose q, k, v signal separately, and q, k, v share the same sets of modules to process them as well.

Mixture of Experts for Seasonal-Trend Decomposition:

Mixture Of Experts Decomposition block (MOEDecomp) contains a set of average filters with different sizes to extract multiple trend components from the input signal and a set of data-dependent weights for combining them as the final trend. We have

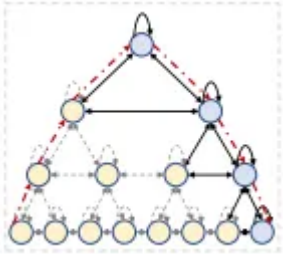
$$\mathbf{X}_{\mathbf{trend}} = \mathbf{Softmax}(L(x)) * (F(x)),$$

MOEDecomp equation

where F(·) is a set of average pooling filters and Softmax(L(x)) is the weights for mixing these extracted trends.

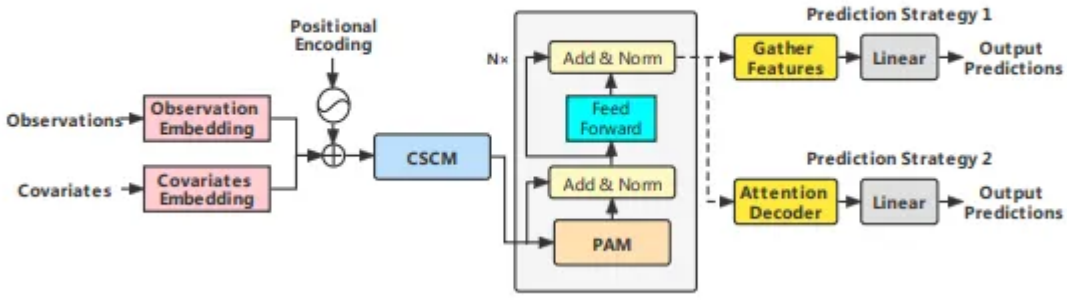
Pyraformer:

This model uses the pyramidal attention module (PAM) in which the inter-scale tree structure summarizes features at different resolutions and the intra-scale neighboring connections model the temporal dependencies of different ranges. Under mild conditions, the maximum length of the signal traversing path in Pyraformer is a constant (i.e., O(1)) with regard to the sequence length L, while its time and space complexity scale linearly with L. It has highest prediction accuracy in both single-step and long-range multi-step forecasting tasks with the least amount of time and memory consumption, especially when the sequence is long.



Pyraformer tree structure

The edges in the above graph can be divided into two groups: the inter-scale and the intra-scale connections. The inter-scale connections build a multiresolution representation of the original sequence: nodes at the finest scale correspond to the time points in the original time series (e.g., hourly observations), while nodes in the coarser scales represent features with lower resolutions (e.g., daily, weekly, and monthly patterns). Such latent coarser-scale nodes are initially introduced via a coarser-scale construction module. On the other hand, the intra-scale edges capture the temporal dependencies at each resolution by connecting neighboring nodes together. As a result, this model provides a compact representation for long-range temporal dependencies among far-apart positions by capturing such behavior at coarser resolutions, leading to a smaller length of the signal traversing path. Moreover, modeling temporal dependencies of different ranges at different scales with sparse neighboring intra-scale connections significantly reduces the computational cost.



Architecture of Pyraformer

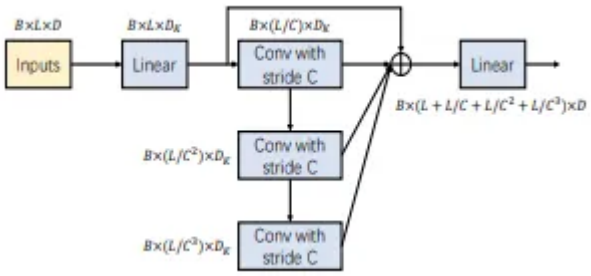
Pyramidal Attention Module (PAM):

The model leverages a pyramidal graph to describe the temporal dependencies of the observed time series in a multiresolution fashion. Such a multiresolution structure has proved itself an effective and efficient tool for long-range interaction

modeling in the field of time series. Every node only pays attention to a limited set of keys in the PAM, corresponding to the pyramidal graph.

Coarser-Scale Construction Module (CSCM):

CSCM targets at initializing the nodes at the coarser scales of the pyramidal graph, so as to facilitate the subsequent PAM to exchange information between these nodes. Specifically, the coarse-scale nodes are introduced scale by scale from bottom to top by performing convolutions on the corresponding children nodes $C^{(s)}_{\ell}$. As demonstrated in Figure 3, several convolution layers with kernel size C and stride C are sequentially applied to the embedded sequence in the dimension of time, yielding a sequence with length L/Cs at scale s. The resulting sequences at different scales form a C-ary tree. We concatenate these fine-to-coarse sequences before inputting them to the PAM. In order to reduce the amount of parameters and calculations, we reduce the dimension of each node by a fully connected layer before inputting the sequence into the stacked convolution layers and restore it after all convolutions. Such a bottleneck structure significantly reduces the number of parameters in the module and can guard against over-fitting.



Coarser-scale construction module

PATCHTST

This model is for multivariate time series forecasting and self-supervised representation learning. It is based on two key components: segmentation of time series into subseries-level patches which are served as input tokens to Transformer and channel-independence, where each channel contains a single univariate time series that shares the same embedding and Transformer weights across all the series.

Patching design naturally has three-fold benefit: local semantic information is retained in the embedding; computation and memory usage of the attention maps are quadratically reduced given the same look-back window; and the model can

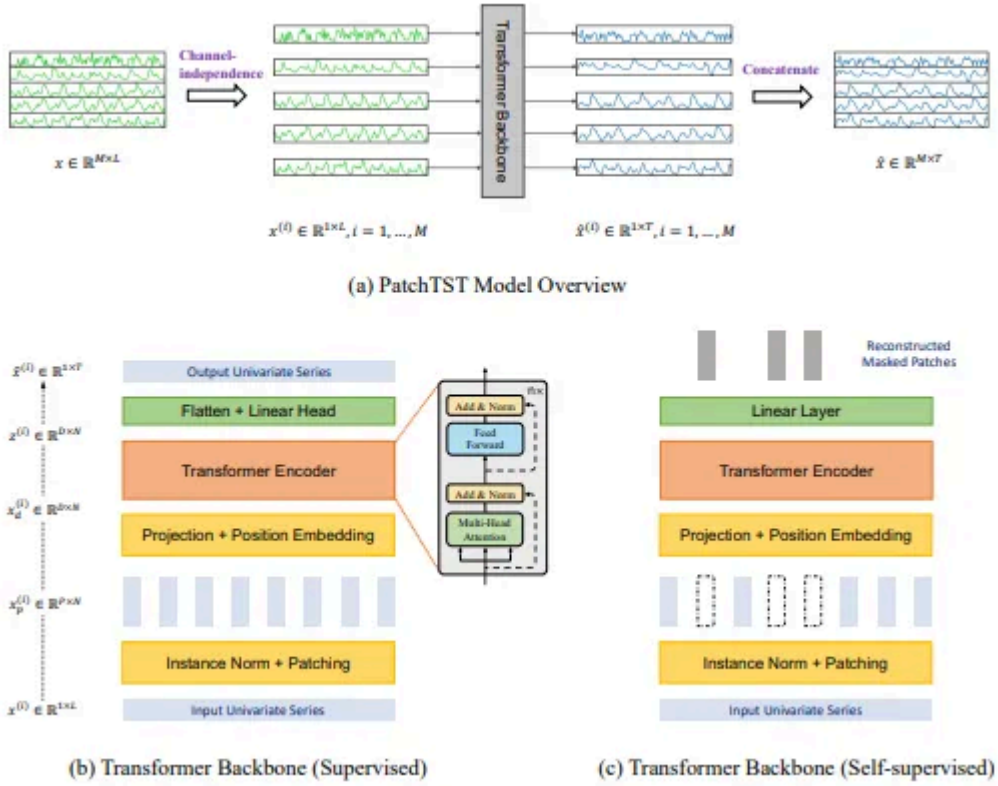
attend longer history. Channel-independent patch time series Transformer (PatchTST) improves the long-term forecasting accuracy significantly.

Key Designs:

1. Patching: Time series forecasting aims to understand the correlation between data in each different time steps. However, a single time step does not have semantic meaning like a word in a sentence, thus extracting local semantic information is essential in analyzing their connections. In contrast, the model enhances the locality and capture comprehensive semantic information that is not available in point-level by aggregating time steps into subseries-level patch.
2. Channel-independence: A multivariate time series is a multi-channel signal, and each Transformer input token can be represented by data from either a single channel or multiple channels. Depending on the design of input tokens, different variants of the Transformer architecture have been proposed. Channel-mixing refers to the latter case where the input token takes the vector of all time series features and projects it to the embedding space to mix information. On the other hand, channel-independence means that each input token only contains information from a single channel.

Model Structure:

Each input univariate time series x(i) is first divided into patches which can be either overlapped or non-overlapped. Denoting the patch length as P and the stride — the non-overlapping region between two consecutive patches as S. The model uses the MSE loss to measure the discrepancy between the prediction and the ground truth. Instance Normalization helps in mitigating the distribution shift effect between the training and testing data. It simply normalizes each time series instance x(i) with zero mean and unit standard deviation.



(a) PatchTST Model Overview

(b) Transformer Backbone (Supervised)

(c) Transformer Backbone (Self-supervised)

Architecture

In principle, a longer look-back window increases the receptive field, which potentially improves the forecasting performance.

Representation Learning:

This technique is conceptually simple: a portion of input sequence is intentionally removed at random and the model is trained to recover the missing contents. masking is applied at the level of single time steps. The masked values at the current time step can be easily inferred by interpolating with the immediate proceeding or succeeding time values without high level understanding of the entire sequence, which deviates from our goal of learning important abstract representation of the whole signal. Transferring of masked pre-trained representation on one dataset to others also produces SOTA forecasting accuracy.

PatchTST/64 implies the number of input patches is 64, which uses the look-back window L = 512. PatchTST/42 means the number of input patches is 42, which has the default look-back window L = 336. Both of them use patch length P = 16 and stride S = 8. Thus, we could use PatchTST/42 as a fair comparison to DLinear and other Transformer-based models, and PatchTST/64 to explore even better results on larger datasets.

Closing Remarks

Taking all the above into consideration, Transformers have undoubtedly revolutionized the landscape of time series forecasting. From the aforementioned models, PatchTST is the most promising and has achieved SOTA results. It perfectly captures local semantics, and global patterns while quadratically reducing memory and computation usage.

Thank you for reading!

Follow me on LinkedIn!

References

[1] [1706.03762] Attention Is All You Need (arxiv.org)

[2] Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting: https://arxiv.org/abs/1907.00235

[3] Pyraformer: Low-Complexity Pyramidal Attention for Long-Range Time Series Modeling and Forecasting | OpenReview

[4] [2201.12740] FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting (arxiv.org)

[5] [2211.14730] A Time Series is Worth 64 Words: Long-term Forecasting with Transformers (arxiv.org)

Machine Learning | Time Series Analysis | Transformers | Deep Learning | Forecasting

https://huggingface.co/abideen

More from Zain ul Abideen

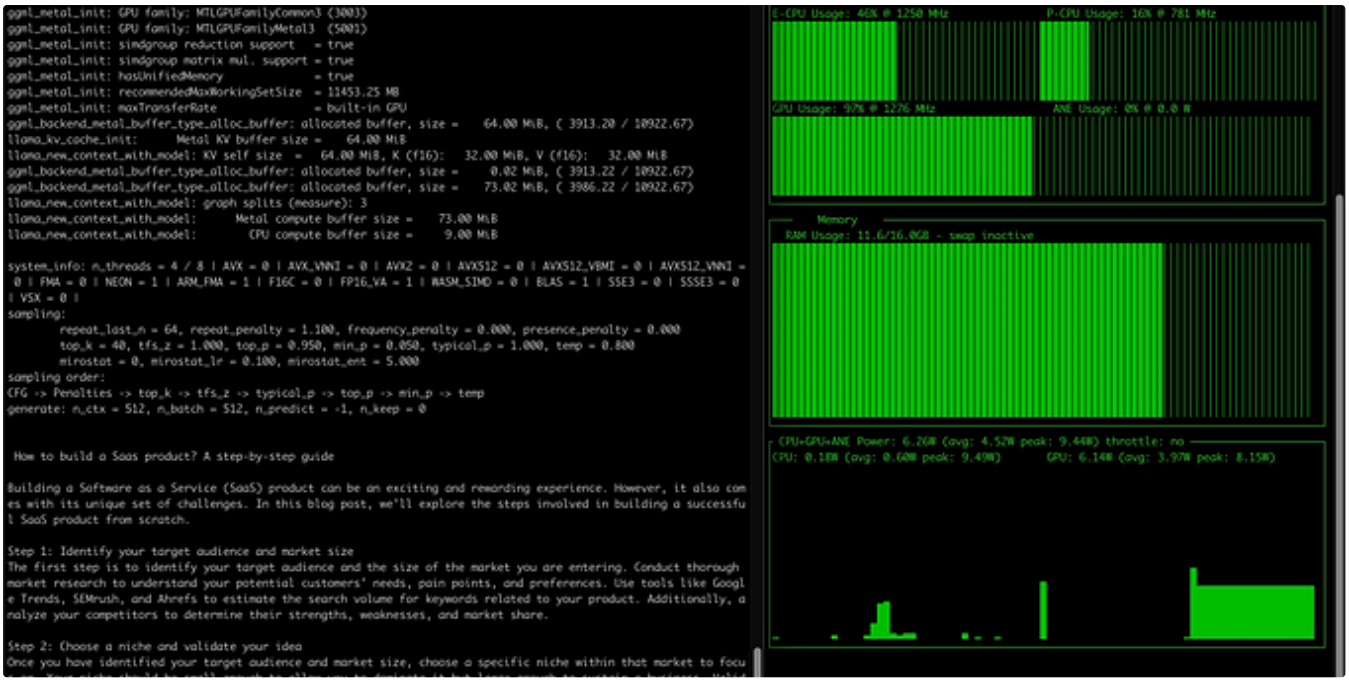
Zain ul Abideen

Llama-Bitnet | Training a 1.58 bit LLM

What is 1 bit LLM and How to train 70M Llama-Bitnet?

5 min read · Apr 4, 2024

113 1



Zain ul Abideen

Apple MLX vs Llama.cpp vs Hugging Face Candle Rust for Lightning-Fast LLMs Locally

Mistral-7B and Phi-2 to experiment fastest inference/generation speed across libraries.

6 min read · Jan 31, 2024

470 2

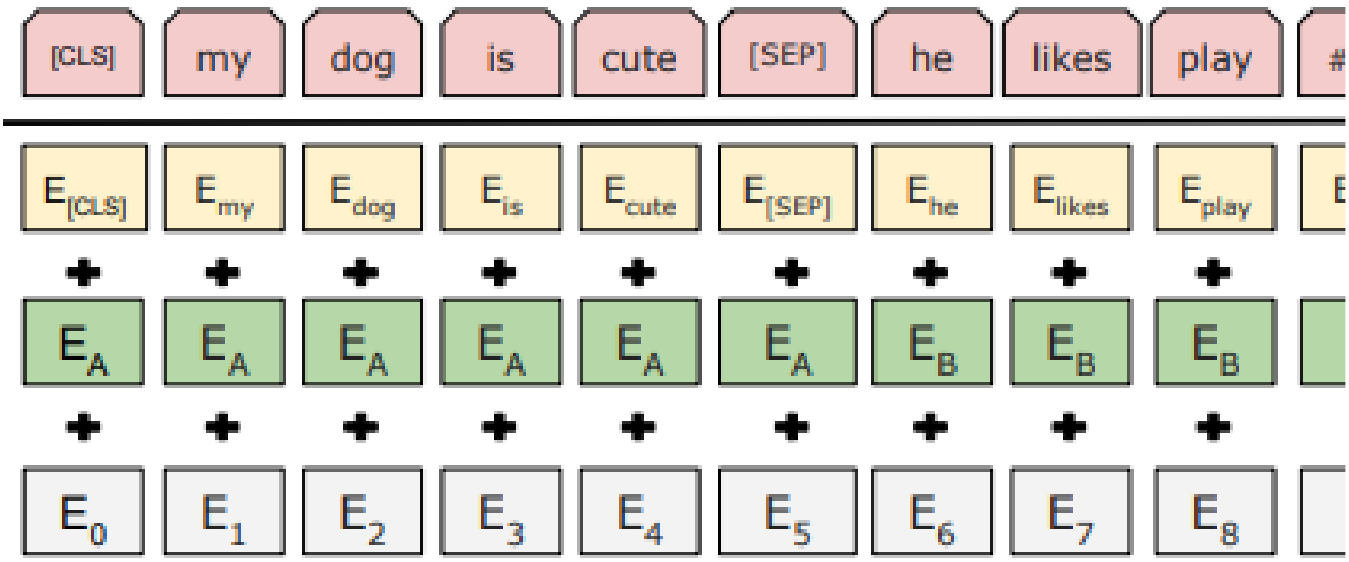
Zain ul Abideen

ORPO Outperforms SFT+DPO | Train Phi-2 with ORPO

Train Phi-2 with ORPO with LazyOrpo

5 min read · Mar 22, 2024

99 3



Zain ul Abideen

A Comparative Analysis of LLMs like BERT, BART, and T5

Exploring Language Models

6 min read · Jun 26, 2023

146 1

See all from Zain ul Abideen

Recommended from Medium

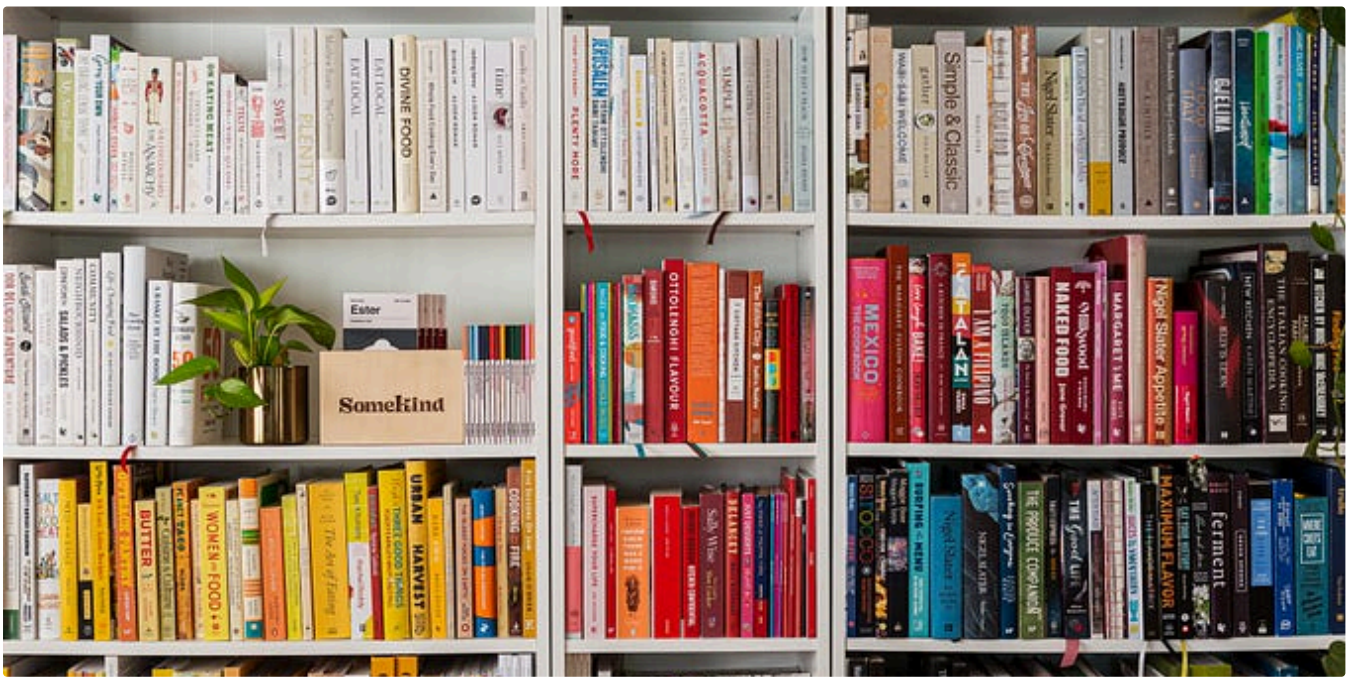
Kaan Aslan

Time Series Forecasting with a Basic Transformer Model in PyTorch

Time series forecasting is an essential topic that's both challenging and rewarding, with a wide variety of techniques available to...

10 min read · Jan 12, 2024

91 1



Mouna Labiadh in DataNess.AI

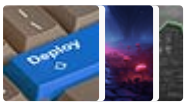
Forecasting book sales with Temporal Fusion Transformer

A step-by-step guide on how to use Temporal Fusion Transformer for book sales forecasting.

11 min read · Aug 2, 2023

141 2

Lists

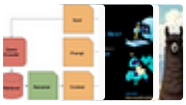


Predictive Modeling w/ Python

20 stories · 1142 saves

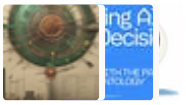
Practical Guides to Machine Learning

10 stories · 1374 saves



Natural Language Processing

1417 stories · 912 saves



data science and AI

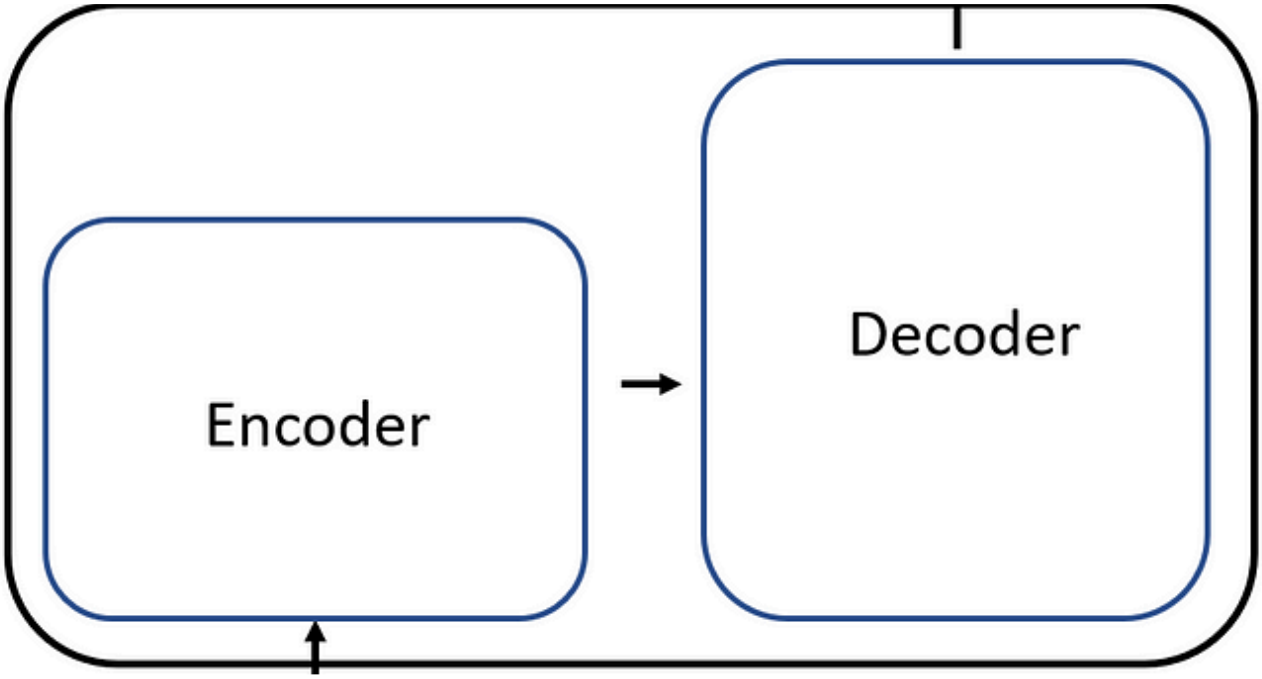
40 stories · 140 saves

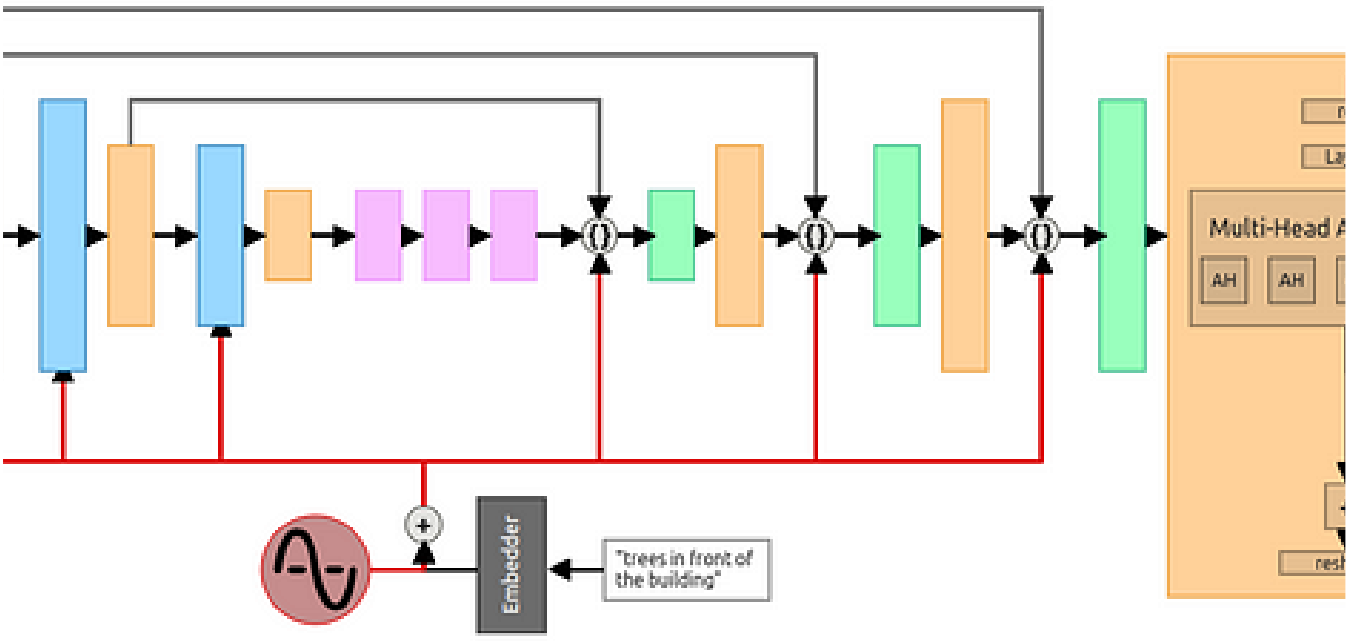
Chris Kuo/Dr. Dataman in Dataman in AI

Temporal Fusion Transformer for Interpretable Time Series Predictions

We shop at retail businesses like Walmart, Target, and Best Buy. We buy from department stores like Macy's, Nordstrom, and Sears. We visit...

· 23 min read · Apr 18, 2024





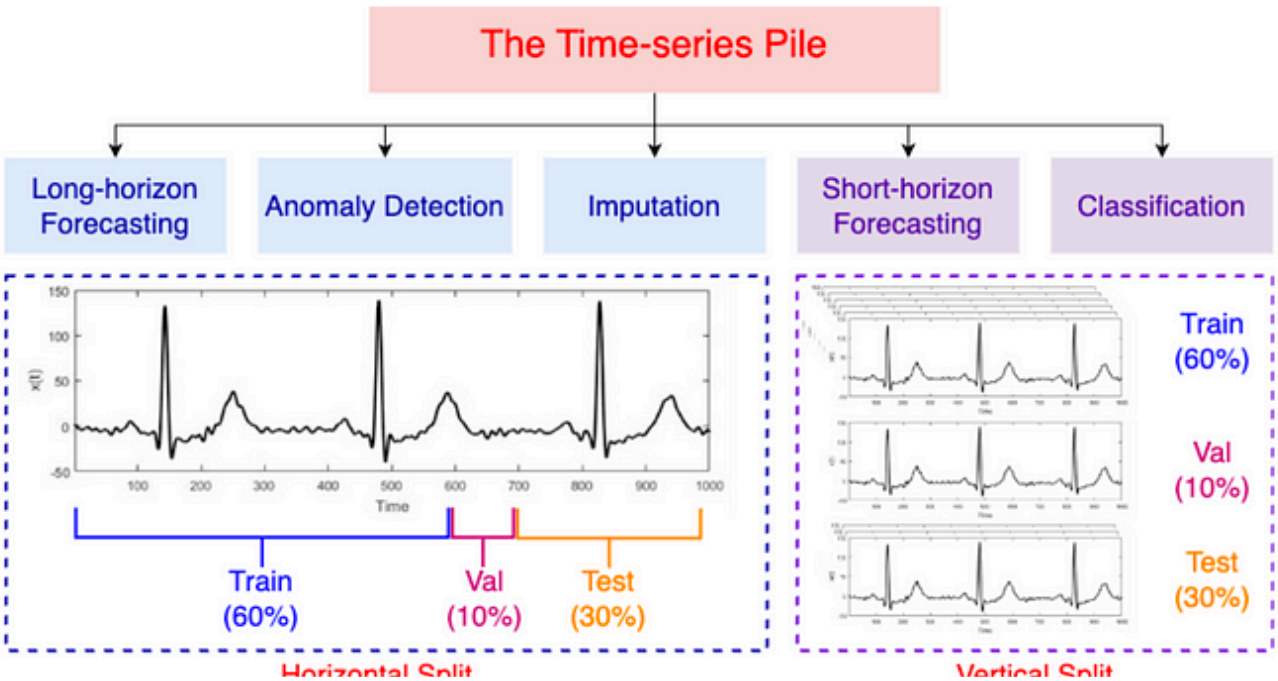
Kemal Erdem (burnpiro)

Step by Step visual introduction to Diffusion Models.

How the diffusion models works under the hood? Visual guide to diffusion process and model architecture.

15 min read · Nov 10, 2023

329 2



samuel chazy in Artificial Intelligence in Plain English

Moment: A Family of Open Time-Series Foundation Models

Pre-training large models on time-series data is challenging because of the absence of a large and cohesive public time-series repository...

4 min read · Mar 4, 2024

54 1