Transformers for Time Series

Not just for Natural Language tasks

Anton Lu

Verena Kain, Michael Schenk, Borja Rodrigo Mateos

Data Science for Beam Operations
Beams Department
CERN

ICFA MLAPA Workshop 2024, Gyeong-ju, Korea

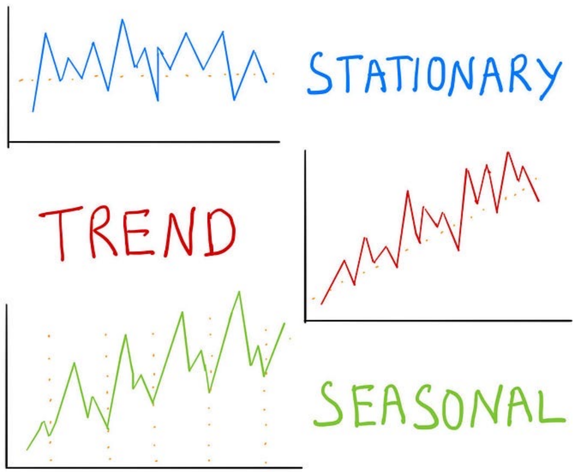
5 March 2024

Contents

Time Series

Come in different forms

- Time series consist of
  - Data points ordered in time
  - Preferably in regular intervals
  - Typically, 3 (or 4) main components
- Time axis not *strictly* needed but helpful to
  - Order data
  - Perform analysis, like in finance and prediction of natural or artificial phenomenon
  - Temporal resolution of the time series depends on the use-case
    - Often in minutes, hours, days, weeks, month, ...
    - Sub-second resolution is not as common



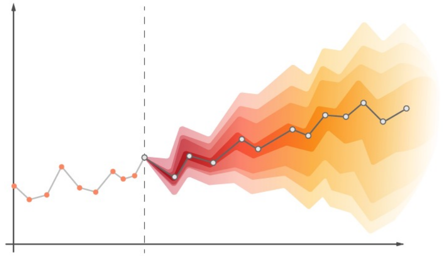
Examples:
- Electricity
- Weather
- Stock prices

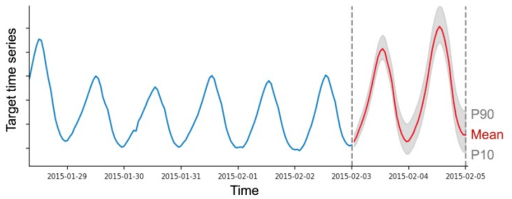
Time Series Forecasting

Predicting the future

Univariate forecasting

- Single time series forecasting (with time)
  - The future may be forecasted just by looking at the past
- Analysis and ML forecasting methods for this is widely researched
  - Simple methods like moving average, regression and basic neural networks can often be sufficient, depending on the accuracy required





Multivariate forecasting

- Forecasting with multiple time series
  - Access to conditional past and future data, both continuous and categorical
- Accurate forecasting is a difficult task
  - Dependency on multiple covariates
  - Long-range dependencies
  - Inherent uncertainty in input and target data
- Long-term Time Series Forecasting (LTSF) a rapidly expanding field in ML
  - But depending on the task, high-precision forecasting is not so commonly seen

Time Series Forecasting

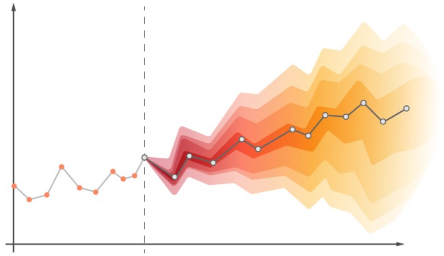
Predicting the future

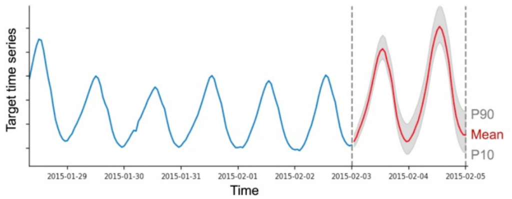
Univariate forecasting

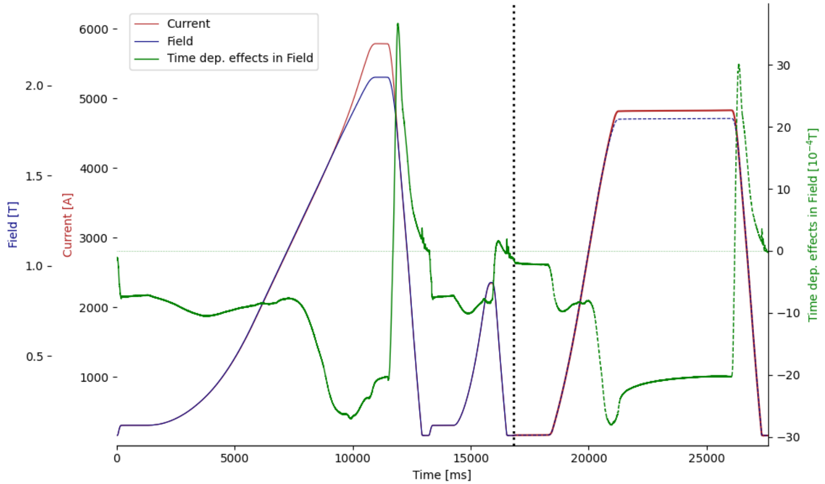
- Single time series forecasting (with time)
  - The future may be forecasted just by looking at the past
- Analysis and ML forecasting methods for this is widely researched
  - Simple methods like moving average, regression and basic neural networks can often be sufficient, depending on the accuracy required

Multivariate forecasting

- Forecasting with multiple time series
  - Access to conditional past and future data, both continuous and categorical
- Accurate forecasting is a difficult task
  - Dependency on multiple covariates
  - Long-range dependencies
  - Inherent uncertainty in input and target data







Sequence-to-Sequence neural models

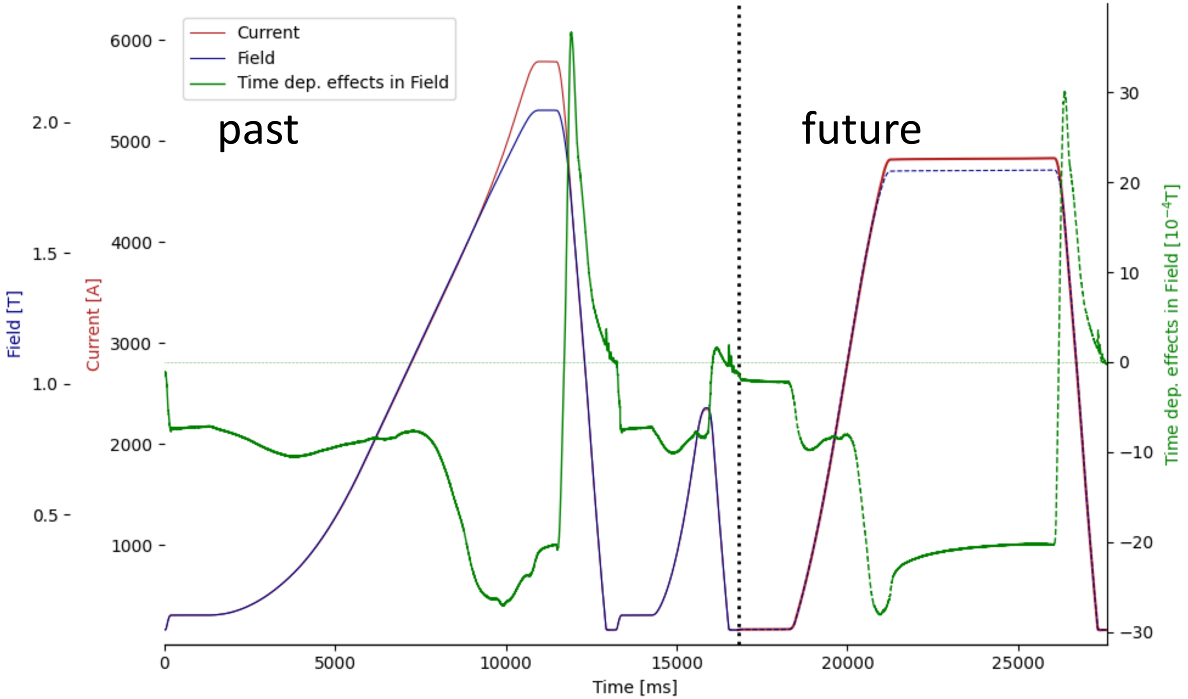
Is this Forecasting?

- In physics we often build transfer functions $f: x \rightarrow y$ where the the dynamics are time and past dependent, i.e. $y_t = f(x_t, x_{t-1}, \dots, y_{t-1}, \dots)$
  - Represent the data as a time series and use sequence-to-sequence models
  - Can be seen as multivariate time-series forecasting

Example: predict future magnetic field B as a function of excitation current I

Find the relationship $f: I \rightarrow B$, or $B_t = f(I_t, I_{t-1}, \dots, B_{t-1}, \dots)$

The relationship is non-trivial as static and dynamic effects make the field difficult to model

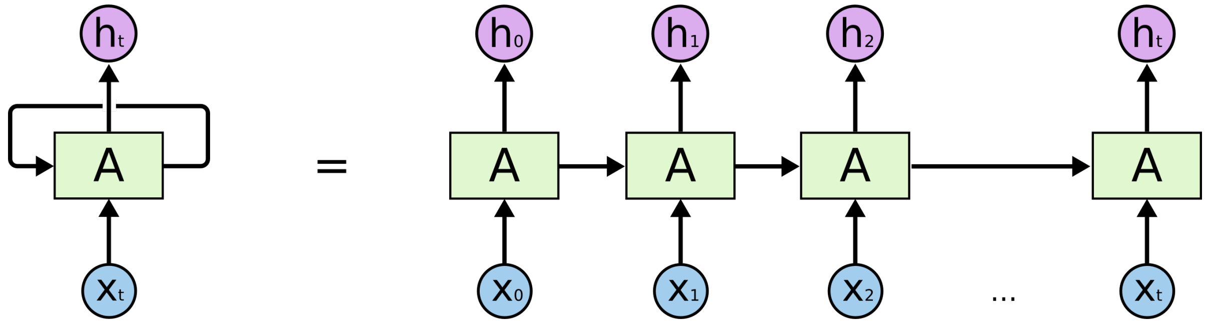


Sequence-to-Sequence neural models

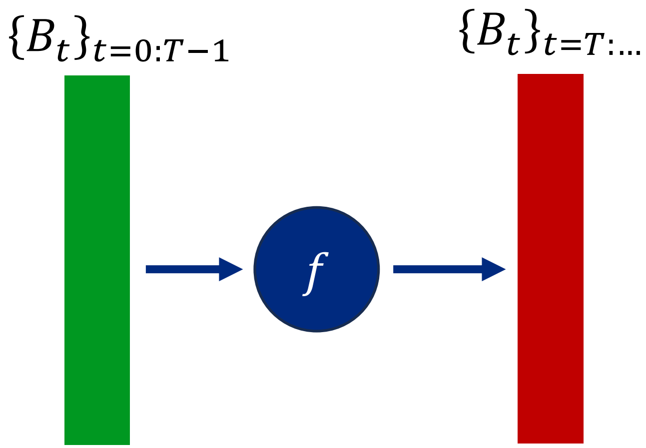
Recurrent Neural Networks

- In physics we often build transfer functions $f: x \rightarrow y$ where the the dynamics are time and past dependent, i.e. $y_t = f(x_t, x_{t-1}, \ldots, y_{t-1}, \ldots)$
  - Represent the data as a time series and use sequence-to-sequence models

- RNNs like GRUs and LSTMs represent the previous state-of-the-art sequence-to-sequence modeling

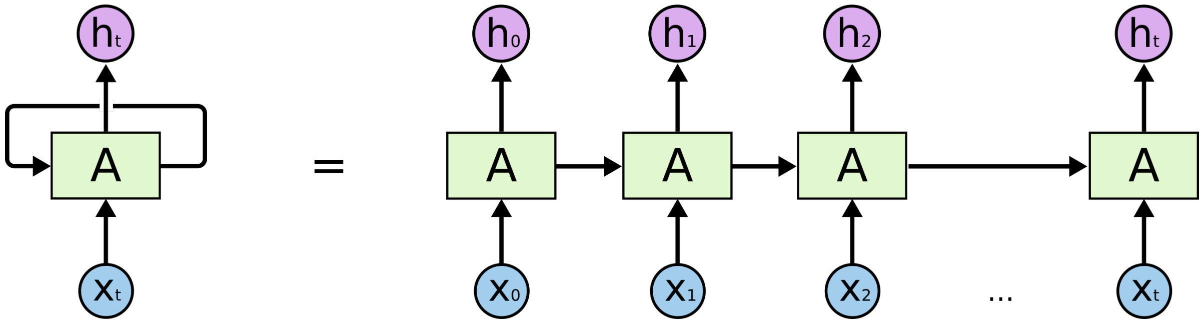


Example: $B \rightarrow B$

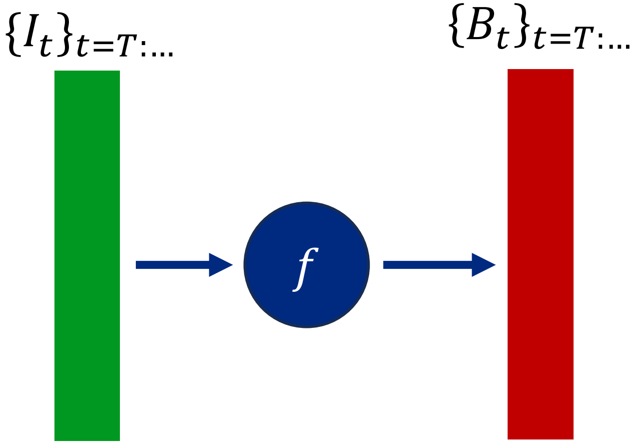


Sequence-to-Sequence neural models
Recurrent Neural Networks

- In physics we often build transfer functions $f: x \to y$ where the the dynamics are time and past dependent, i.e. $y_t = f(x_t, x_{t-1}, \dots, y_{t-1}, \dots)$
  - Represent the data as a time series and use sequence-to-sequence models
- RNNs like GRUs and LSTMs represent the previous state-of-the-art sequence-to-sequence modeling



Example: $I \to B$

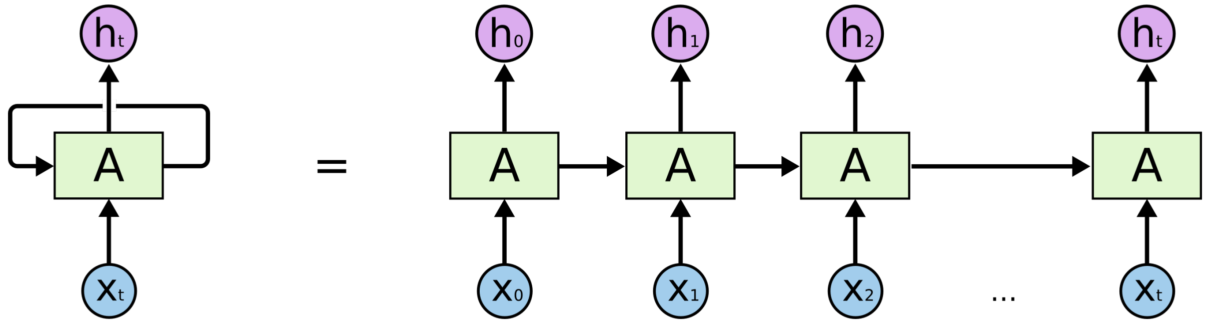


Sequence-to-Sequence neural models

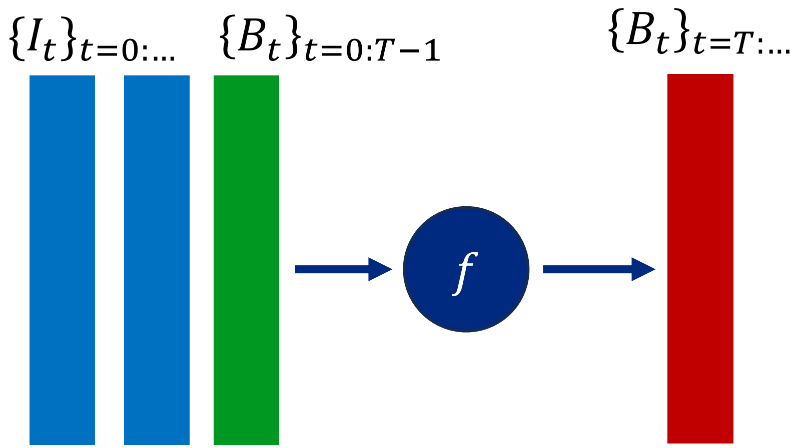
Recurrent Neural Networks

- In physics we often build transfer functions $f: x \rightarrow y$ where the the dynamics are time and past dependent, i.e. $y_t = f(x_t, x_{t-1}, \dots, y_{t-1}, \dots)$
  - Represent the data as a time series and use sequence-to-sequence models

- RNNs like GRUs and LSTMs represent the previous state-of-the-art sequence-to-sequence modeling



Example: $I \rightarrow B$



Sequence-to-Sequence neural models

Enter the Transformer

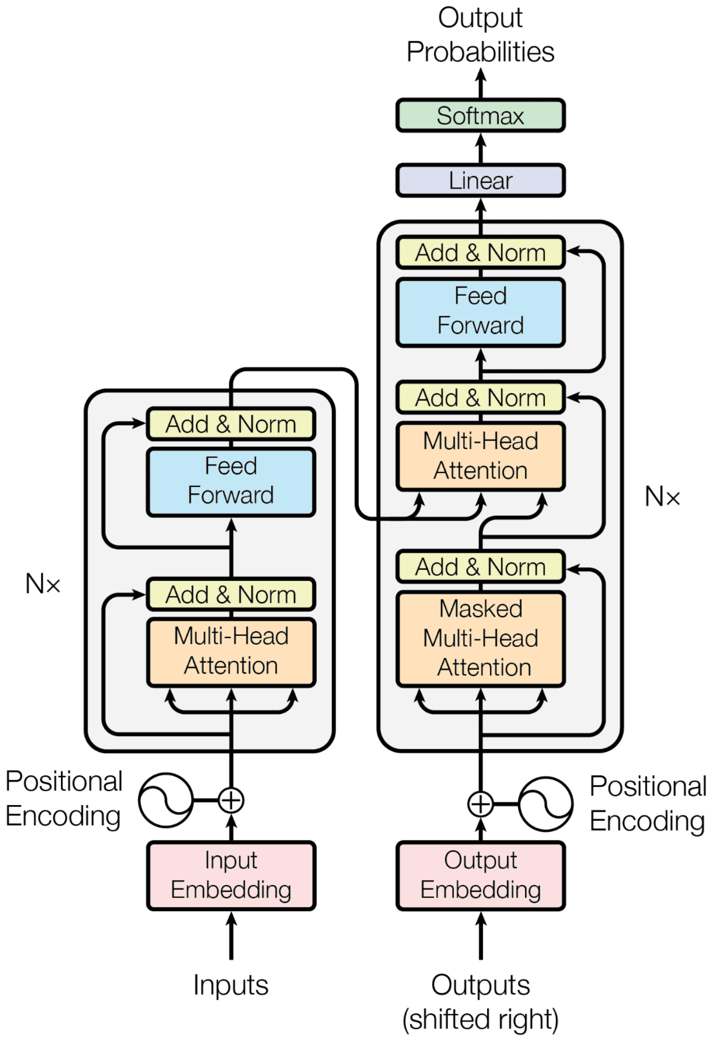
- The State-of-the-Art for sequence modeling
  - Self attention
  - No-recurrent units, allowing parallel computation
  - Widely used in almost all language tasks now
    - Machine translation
    - Text generation
    - Question answering

arXiv:1706.03762

Sequence-to-Sequence neural models

Enter the Transformer

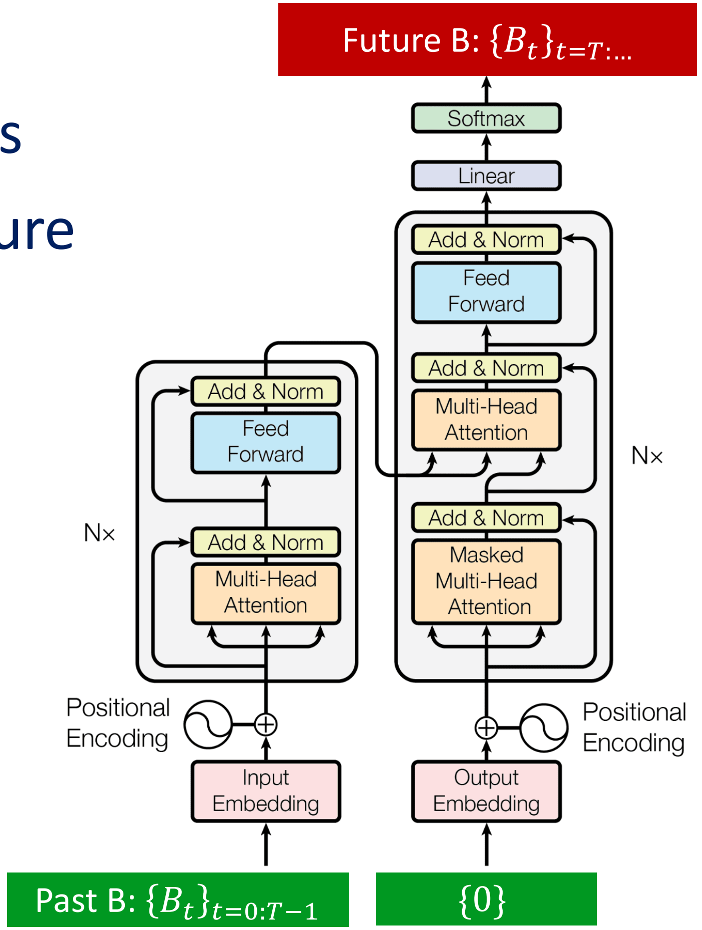
- The State-of-the-Art for sequence modeling
  - Powered by the multi-headed self-attention
  - No-recurrent units, allowing parallel computation
  - As opposed to RNNs, transformers have **two** inputs



Sequence-to-Sequence neural models

Enter the Transformer

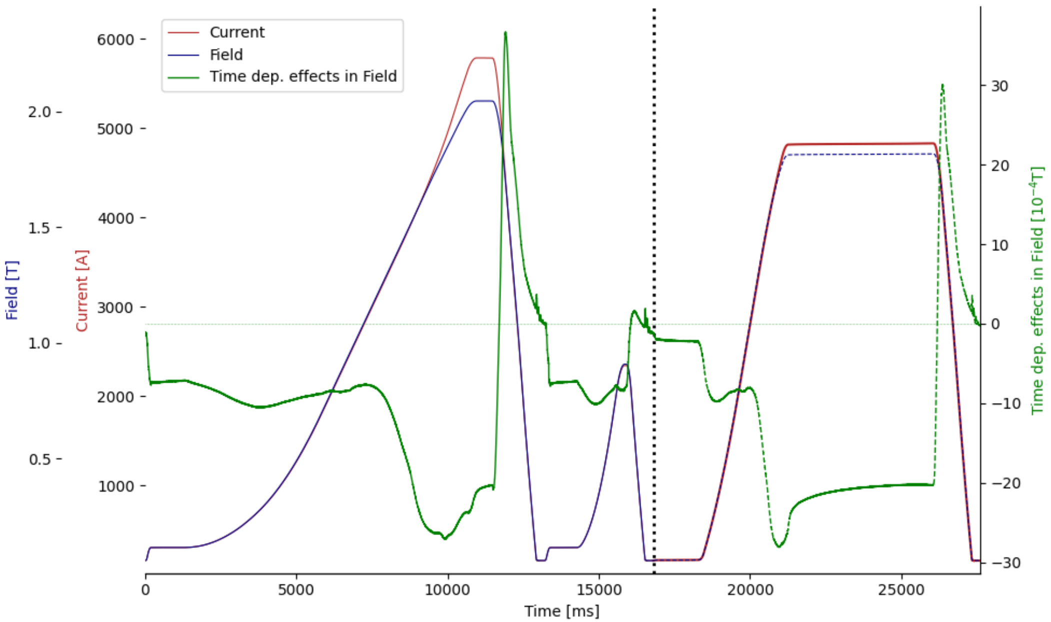
- Example: predicting future B, as univariate time series
- This does not make any sense since we know that future B depends on I

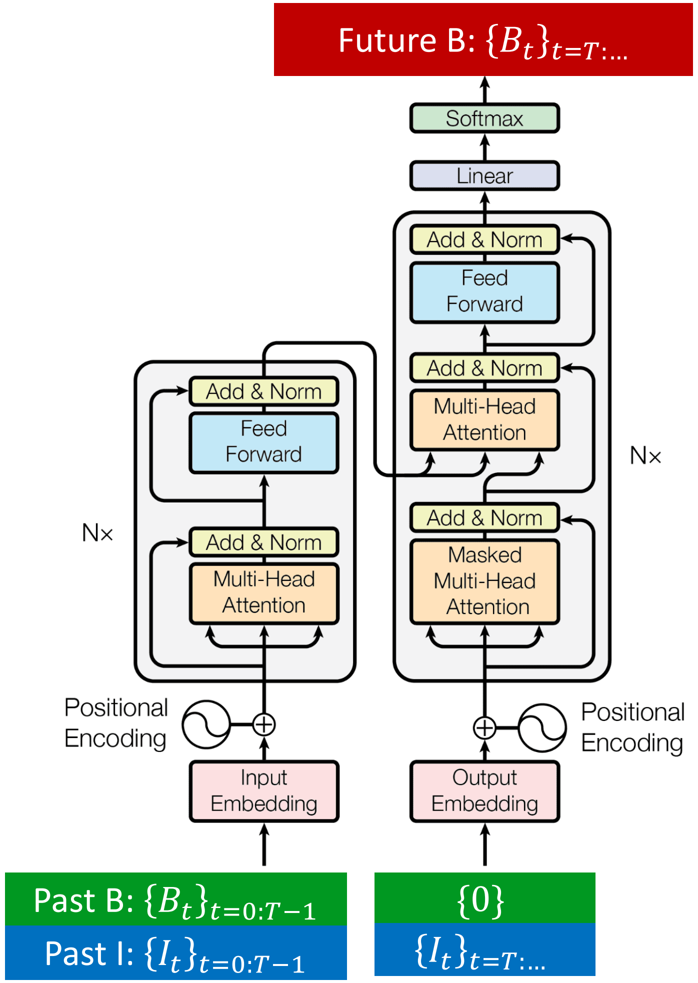


Sequence-to-Sequence neural models

Enter the Transformer

- Example: predicting future B, conditioned on past and future I
  - Multivariate time series “forecasting”

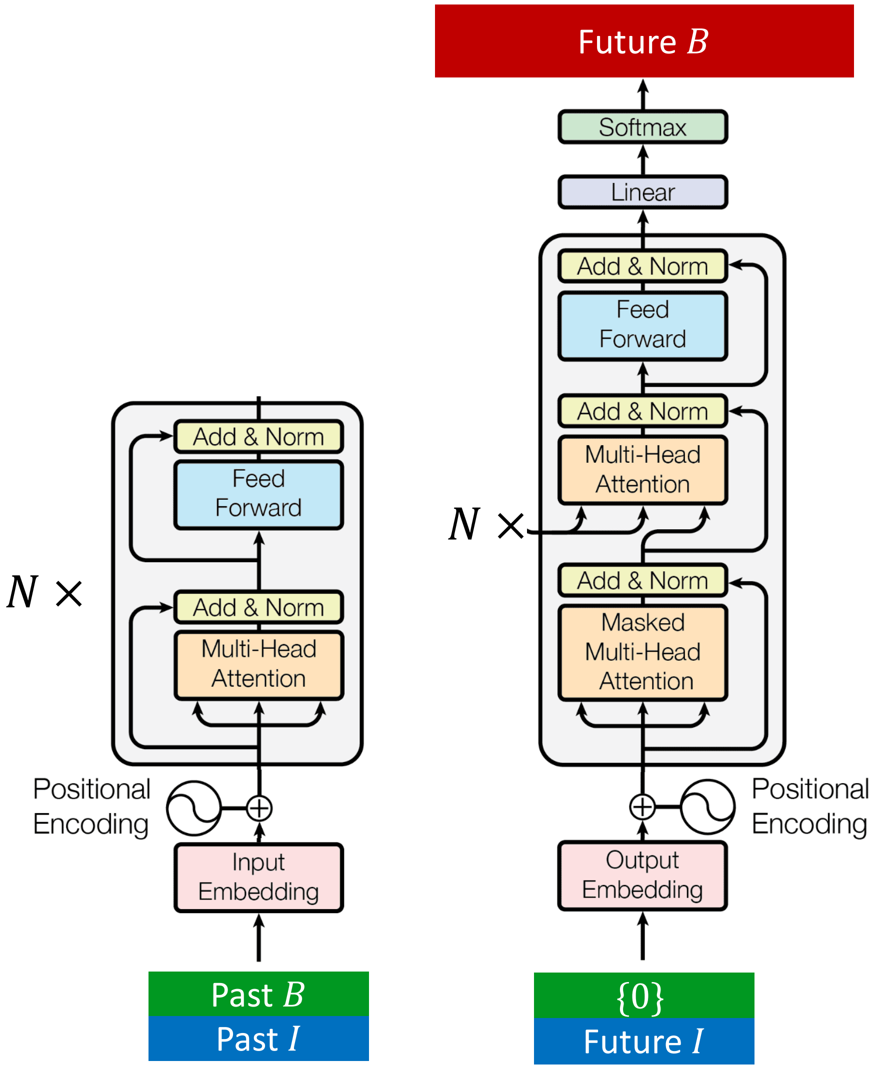




Deconstructing the Transformer

Encoder-Decoder models

Process an input sequence (e.g. a sentence) and extract meaningful, context-aware representation



Generate an output sequence based on the context provided by the encoder, autoregressively

Encoder-only examples

- BERT
  arXiv:1810.04805

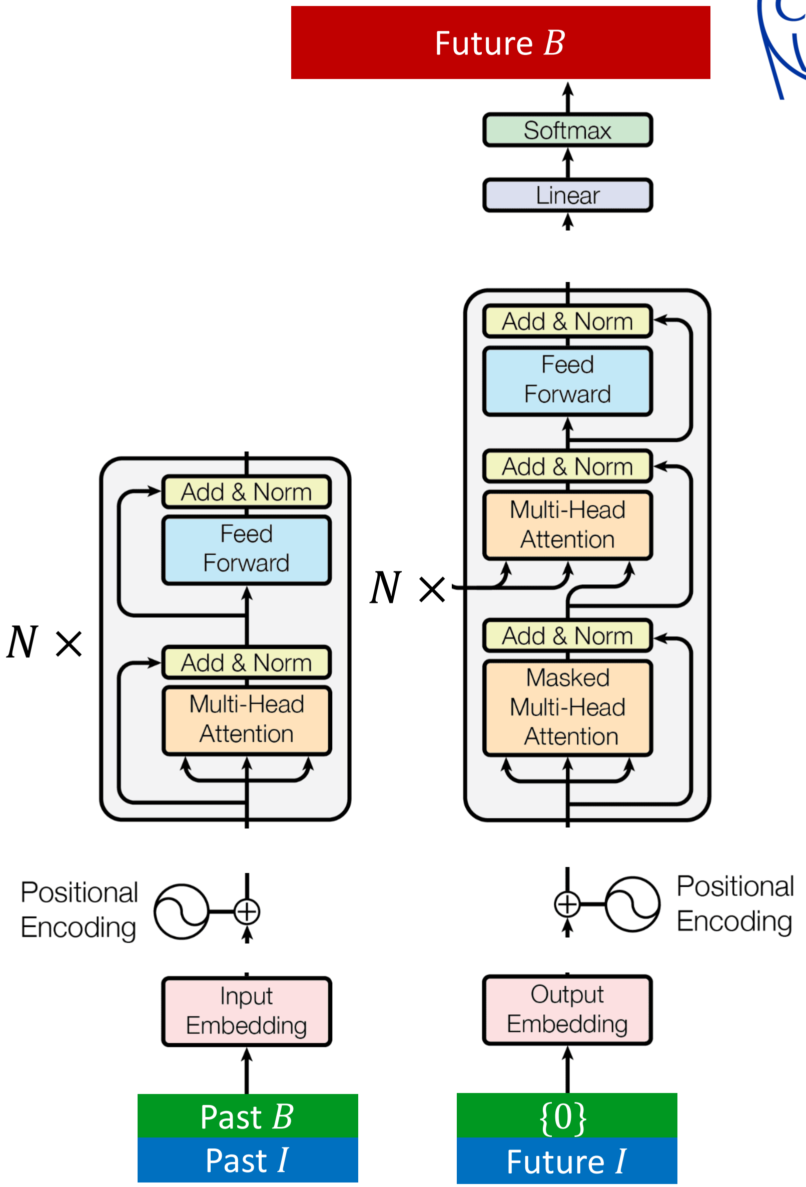
Decoder-only examples

- GPT
  arXiv:2005.14165

Deconstructing the Transformer

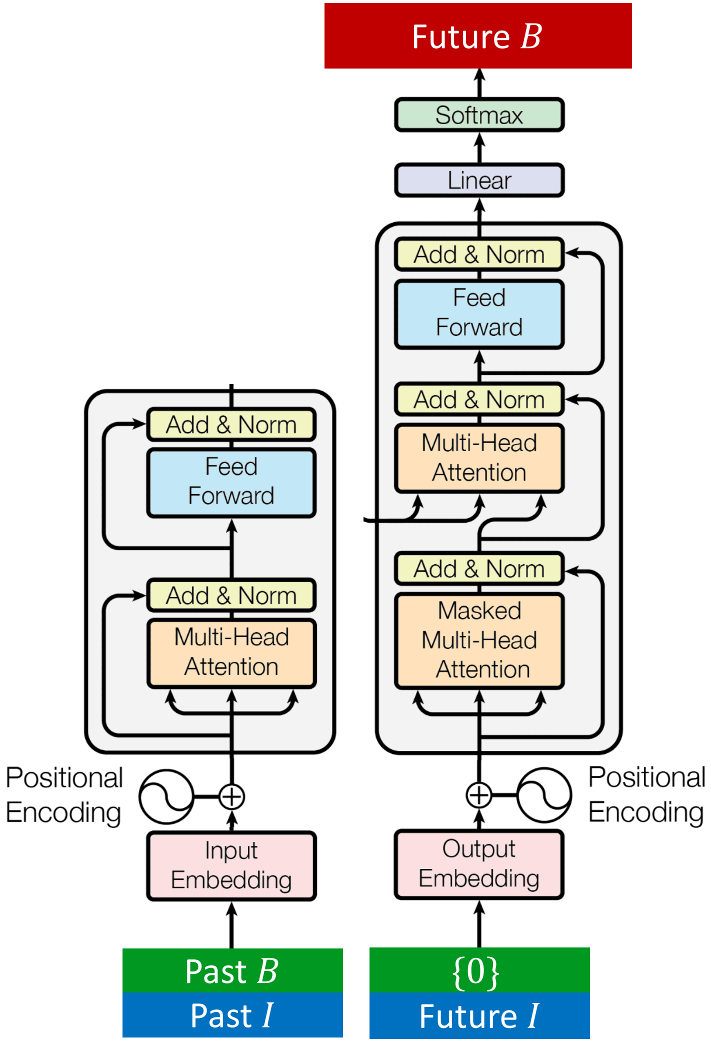
Let us break down the transformer piece-by-piece

Understanding the Encoder makes understanding the Decoder easy



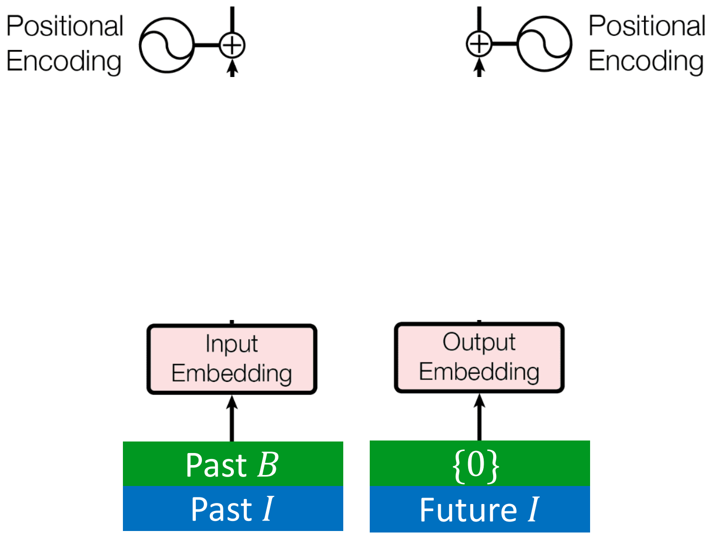
Deconstructing the Transformer

Embedding and positional encoding



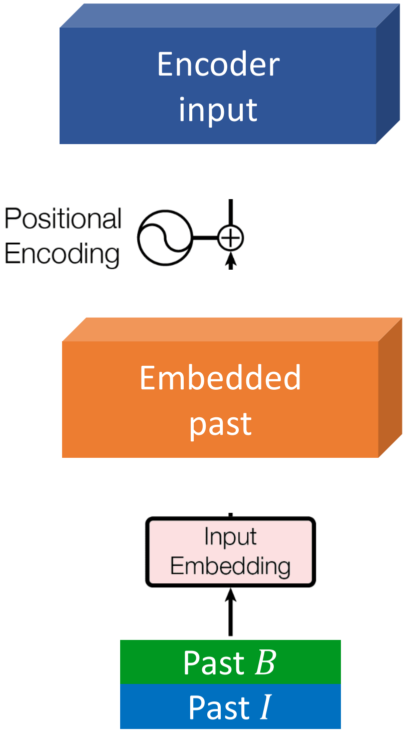
Deconstructing the Transformer
Embedding and positional encoding

The same steps are applied for inputs and outputs, just with different data



Deconstructing the Transformer

Embedding and positional encoding

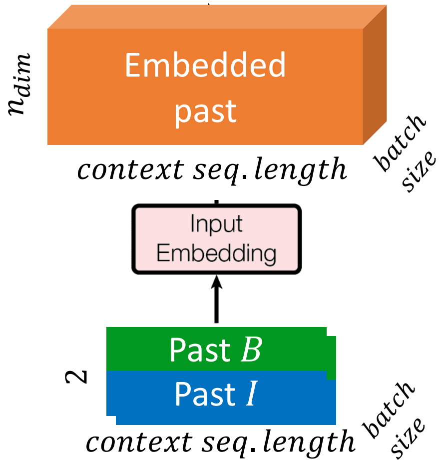


Deconstructing the Transformer

Embedding and positional encoding

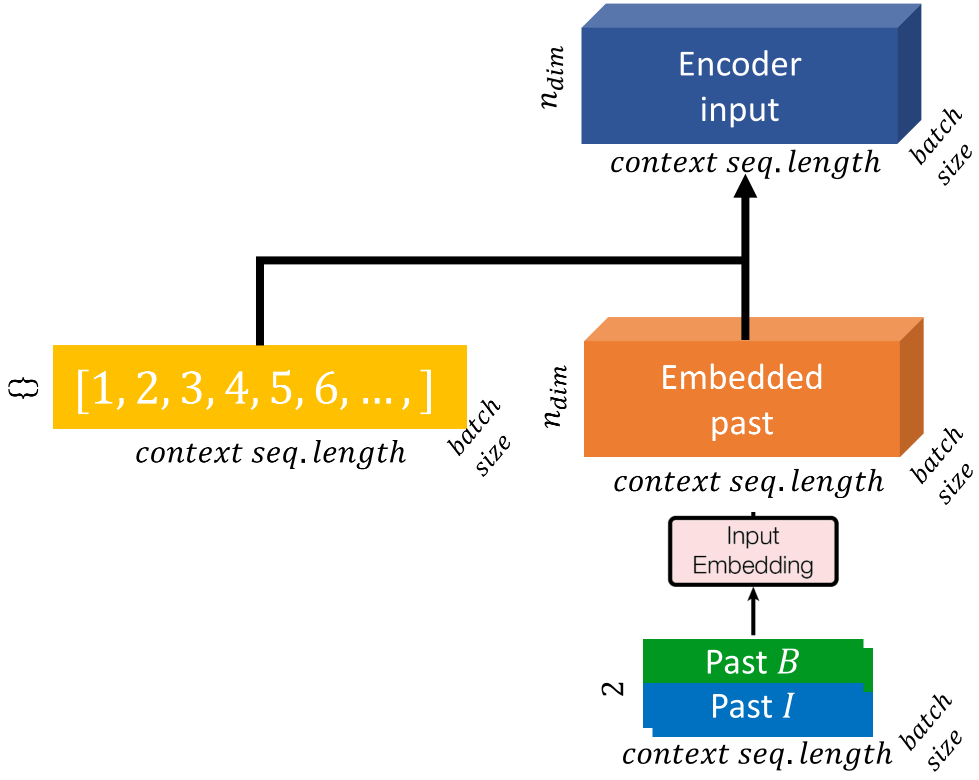
Embedding

- Inputs must be transformed from input space to embedded feature space
  - Typically using linear layers or even RNNs
  - In NLP this can be word embeddings like word2vec or BERT



Deconstructing the Transformer

Embedding and positional encoding



Positional encoding

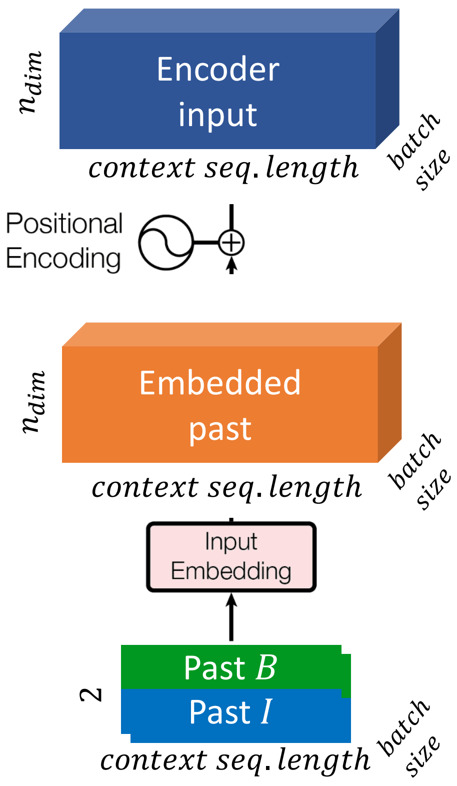
- The transformer does not know the order of the inputs
  - Add positional information with positional encodings
  - Usually with a sin + cos encoding
  - Add the position information point-wise to the embeddings

Deconstructing the Transformer

Embedding and positional encoding

Embedding

- Inputs must be transformed from input space to embedded feature space
  - Typically using linear layers or even RNNs
  - In NLP this can be word embeddings like word2vec or BERT

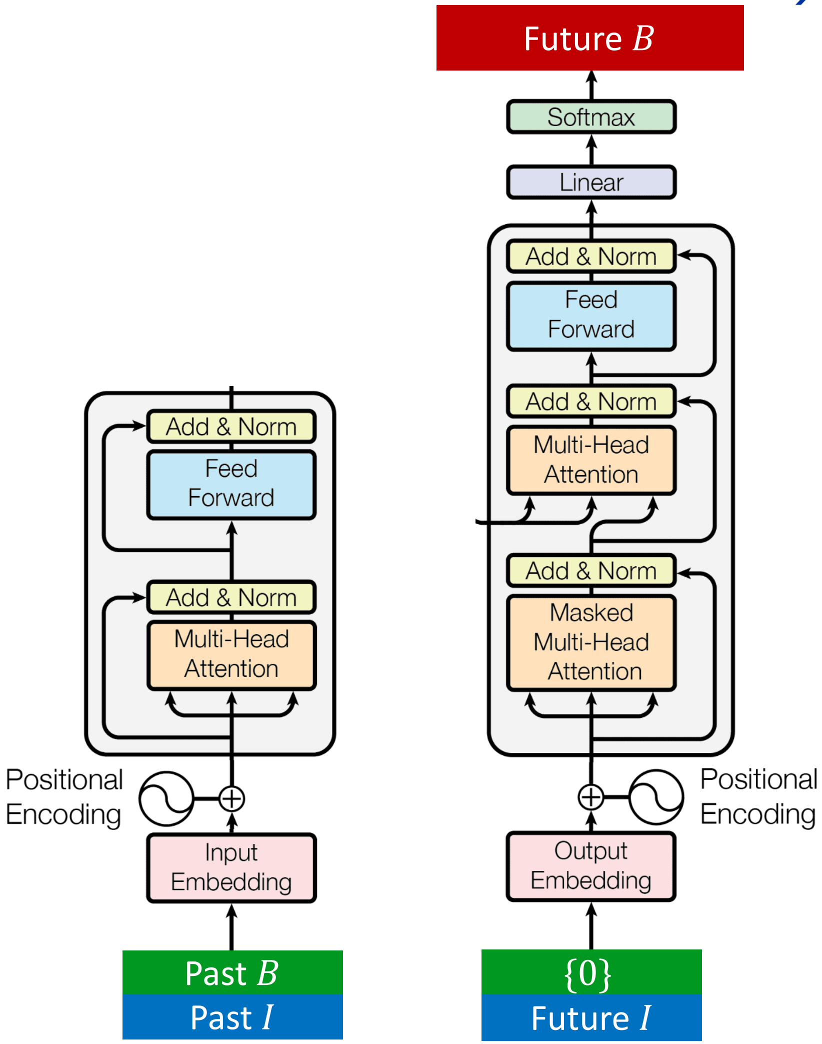


Positional encoding

- The transformer does not know the order of the inputs
  - Add positional information with positional encodings
  - Usually with a sin + cos function
  - Add the position information point-wise to the embeddings

Deconstructing the Transformer

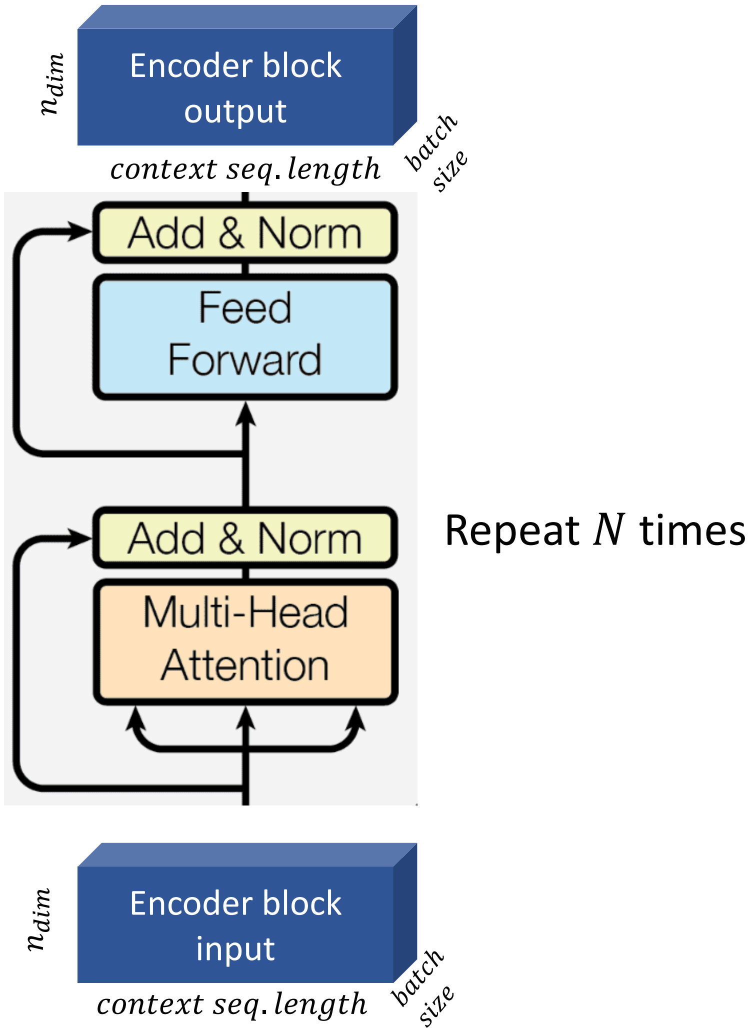
The Encoder Layer



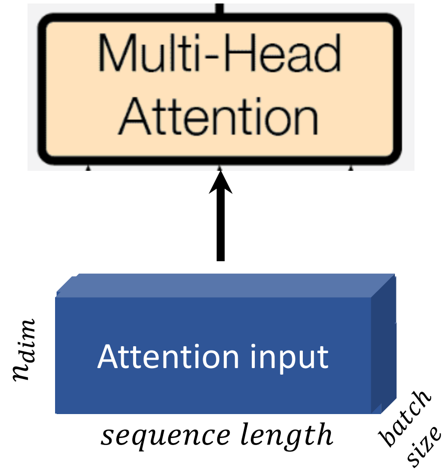
Deconstructing the Transformer

The Encoder Layer

- The block learns context-aware representations from attention
- Repeated layers allow transformer to learn complex patterns
- Residual connections allow information to propagate

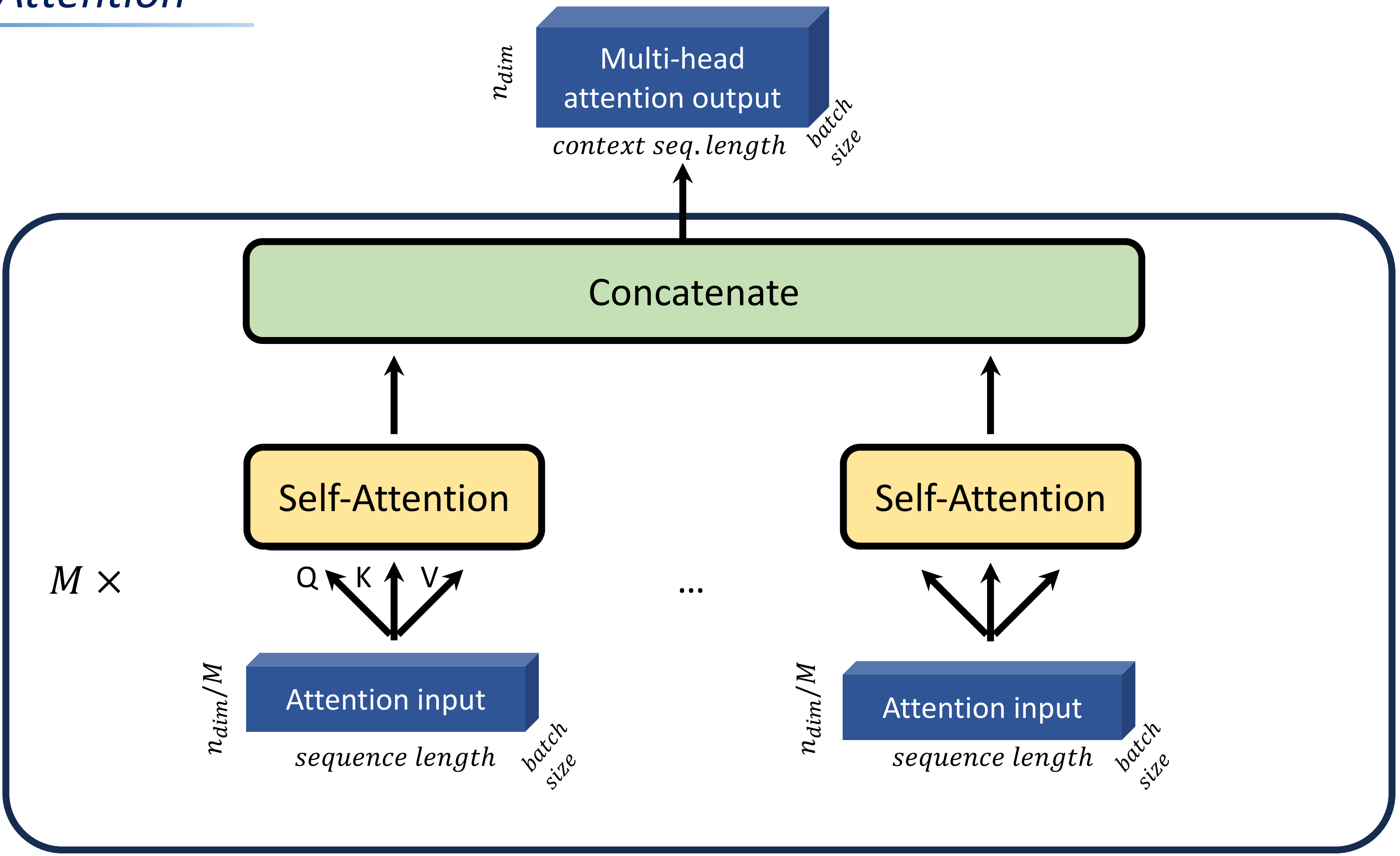


Deconstructing the Transformer
Self-Attention



Deconstructing the Transformer

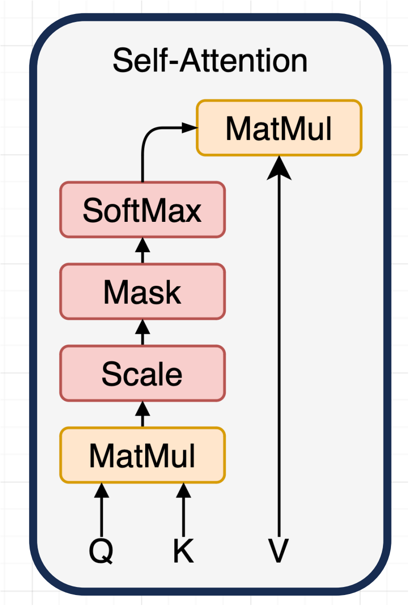
Multi-Head Self-Attention

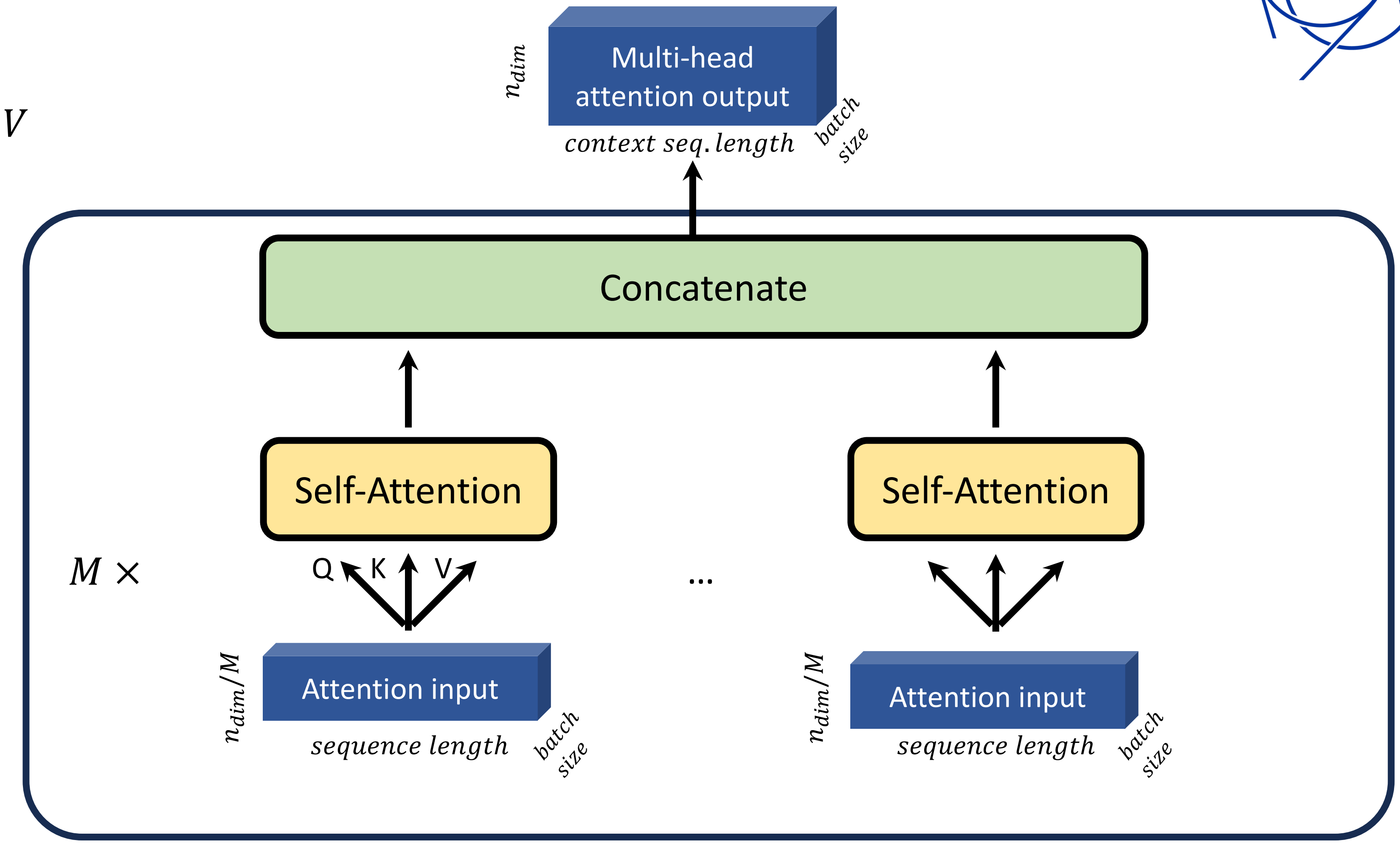


Deconstructing the Transformer

Multi-Head Self-Attention

Self Attention = $SoftMax(\frac{Q\,K^T}{\sqrt{n_{dim}/M}})V$

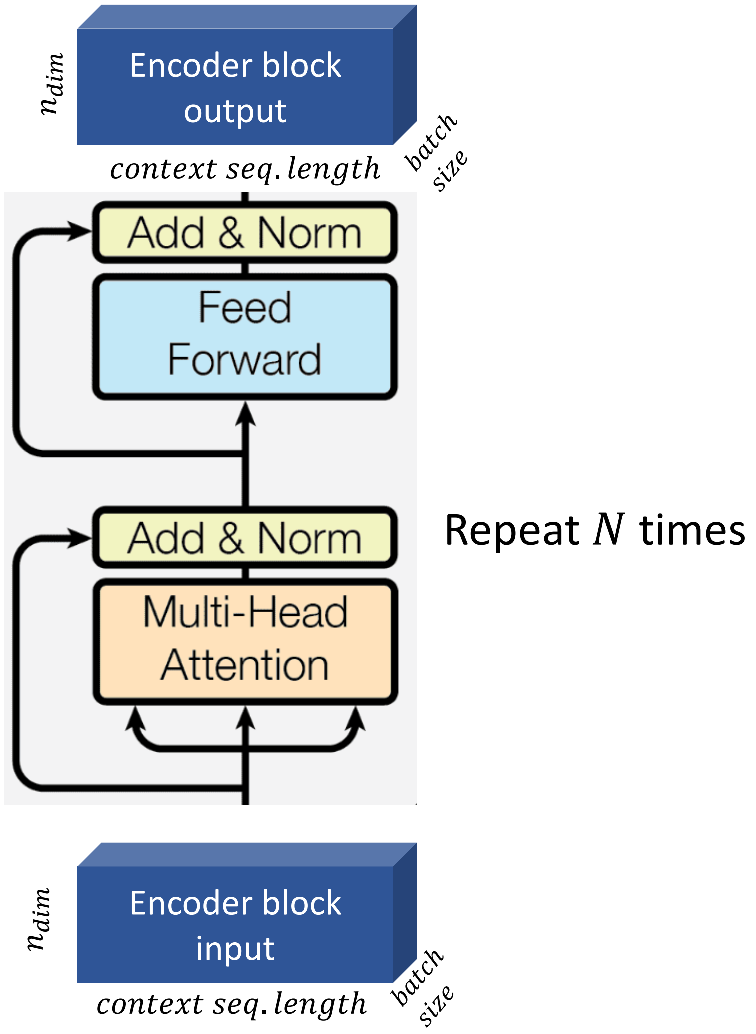




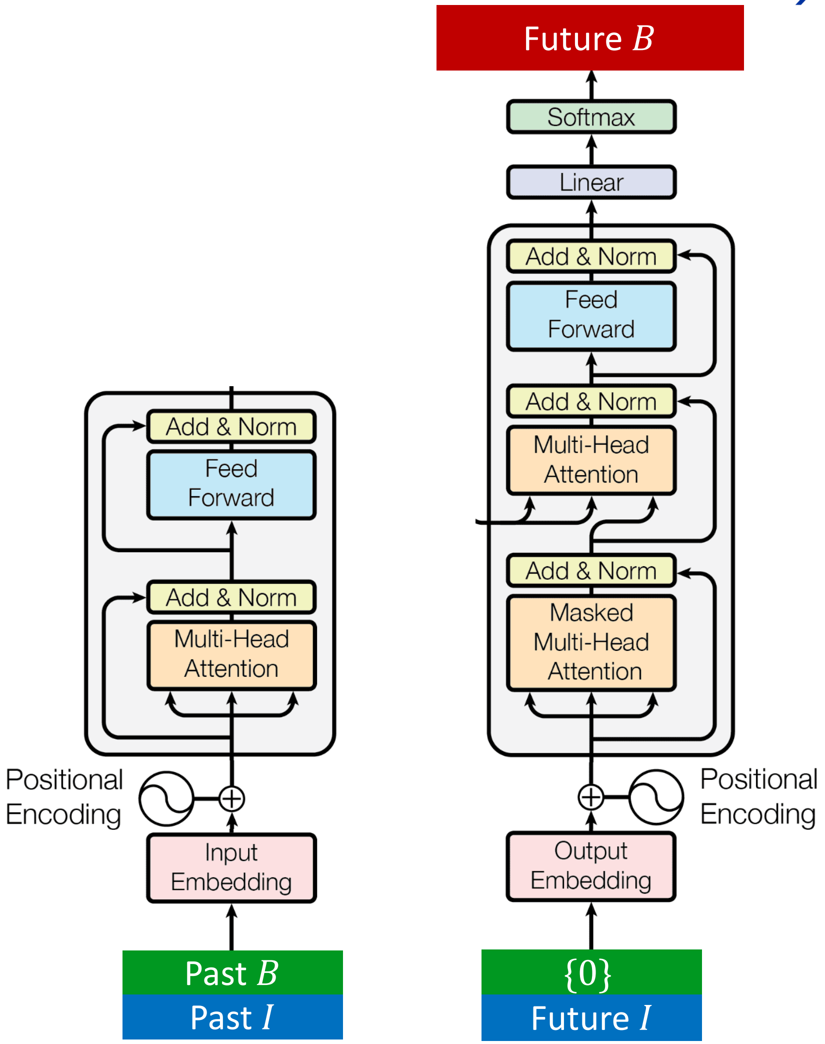
Deconstructing the Transformer

The Encoder Layer

- The block learns context-aware representations from attention
- Repeated layers allow transformer to learn complex patterns
- Residual connections allow information to propagate



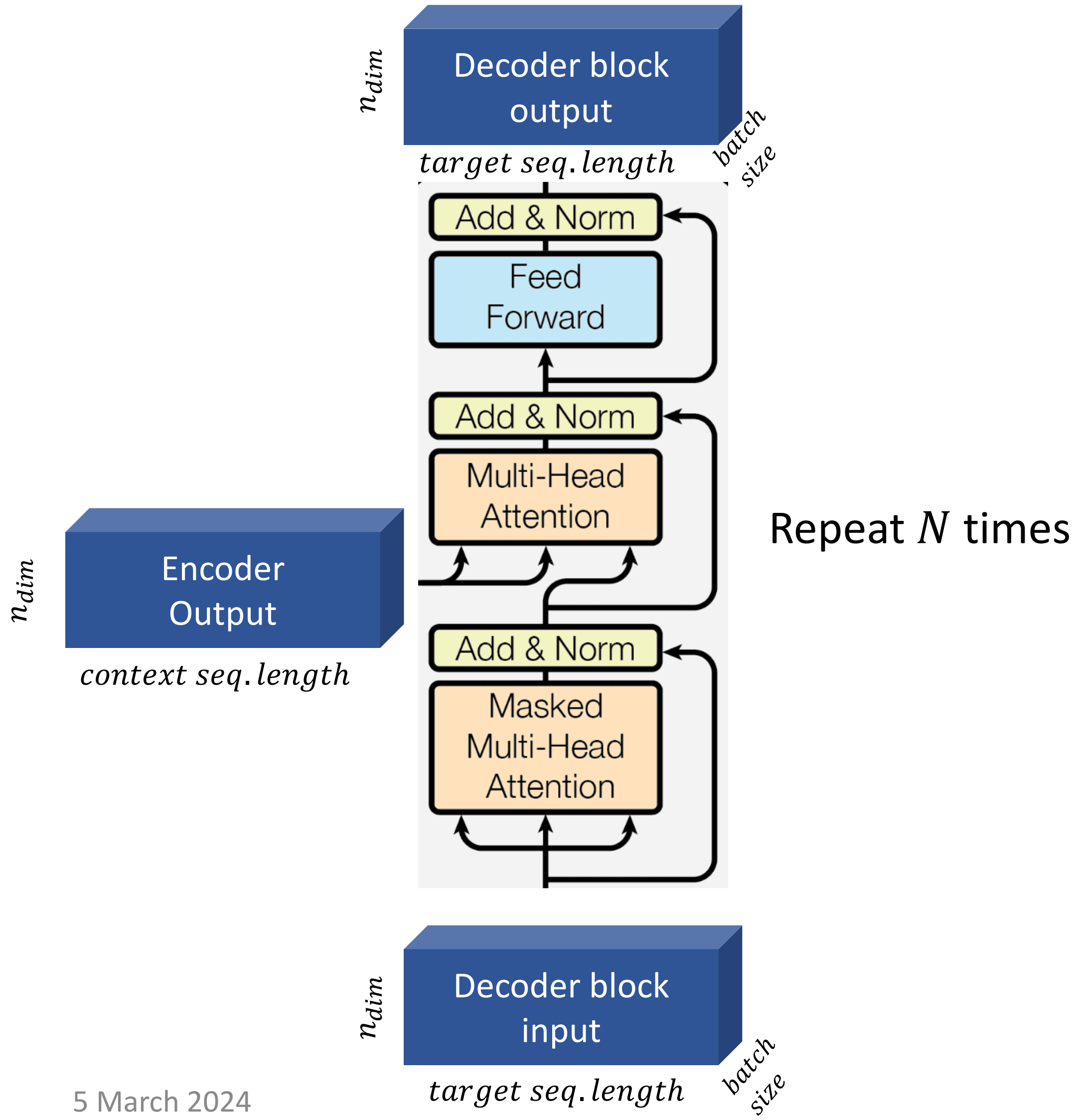
Deconstructing the Transformer
The Decoder Layer



Deconstructing the Transformer

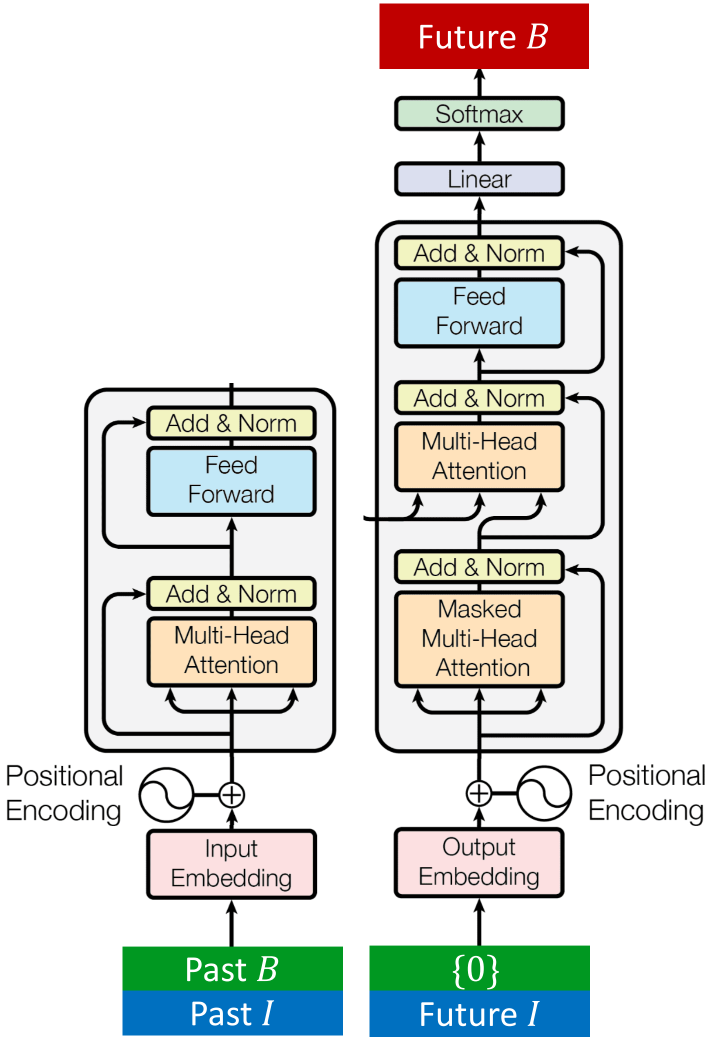
The Decoder Layer

- Masked attention hides the future during decoding
- Encoder output make the Query and Key in multi-head attention
- Residual connections allow information to propagate



Deconstructing the Transformer

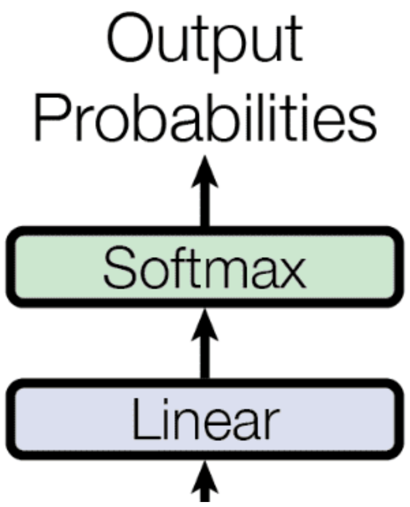
Output projection



Deconstructing the Transformer

Output projection

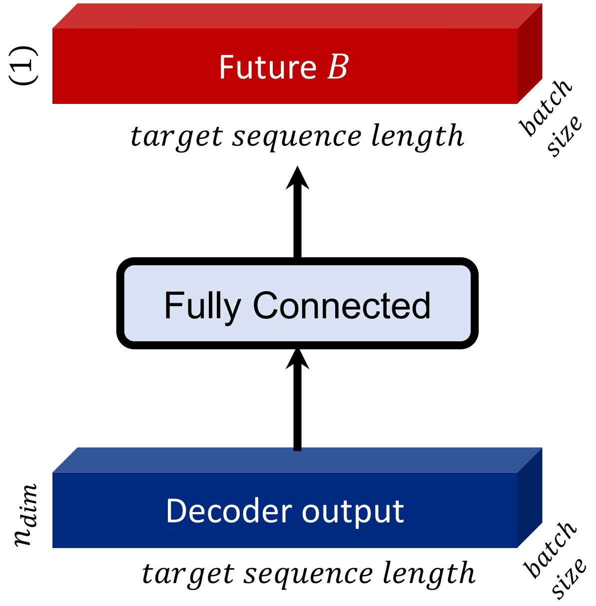
- In NLP, we predict the next token (classification)



Deconstructing the Transformer

Output projection

- In NLP, we predict the next token (classification)
- In time series, we want the future value (regression)
- Use feed-forward layers to project embeddings to 1 (for point estimate), or many (for other loss functions)

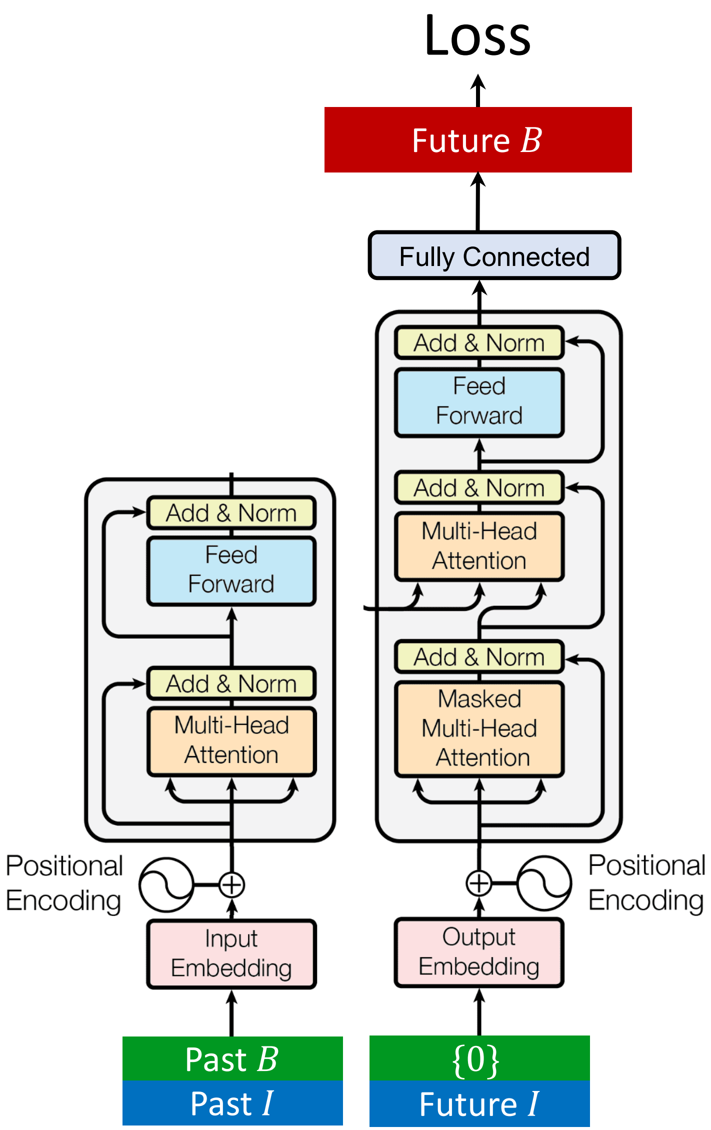


Deconstructing the Transformer
Loss function

Classic loss functions

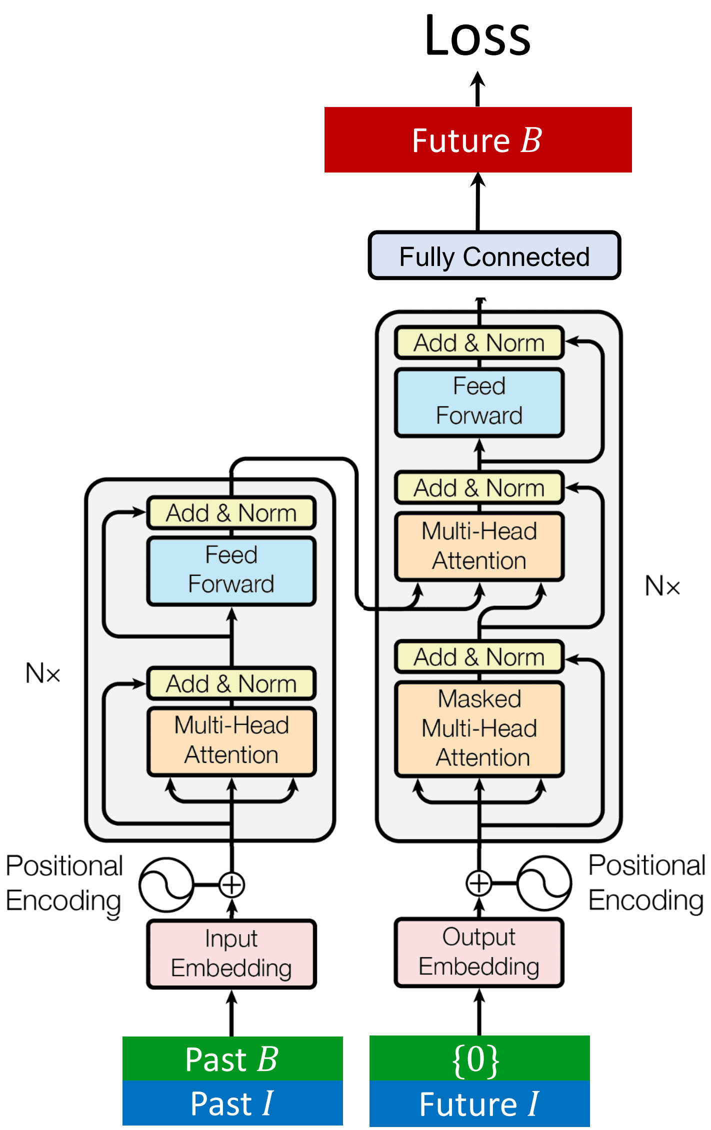
- For point-estimates: MSE, MAE
- Quantile loss to learn prediction interval
- NLL to learn probability distribution

- Choice depends on task and data
  - Does the data have a lot of inherent variance or outliers?



Deconstructing the Transformer

Putting it all together



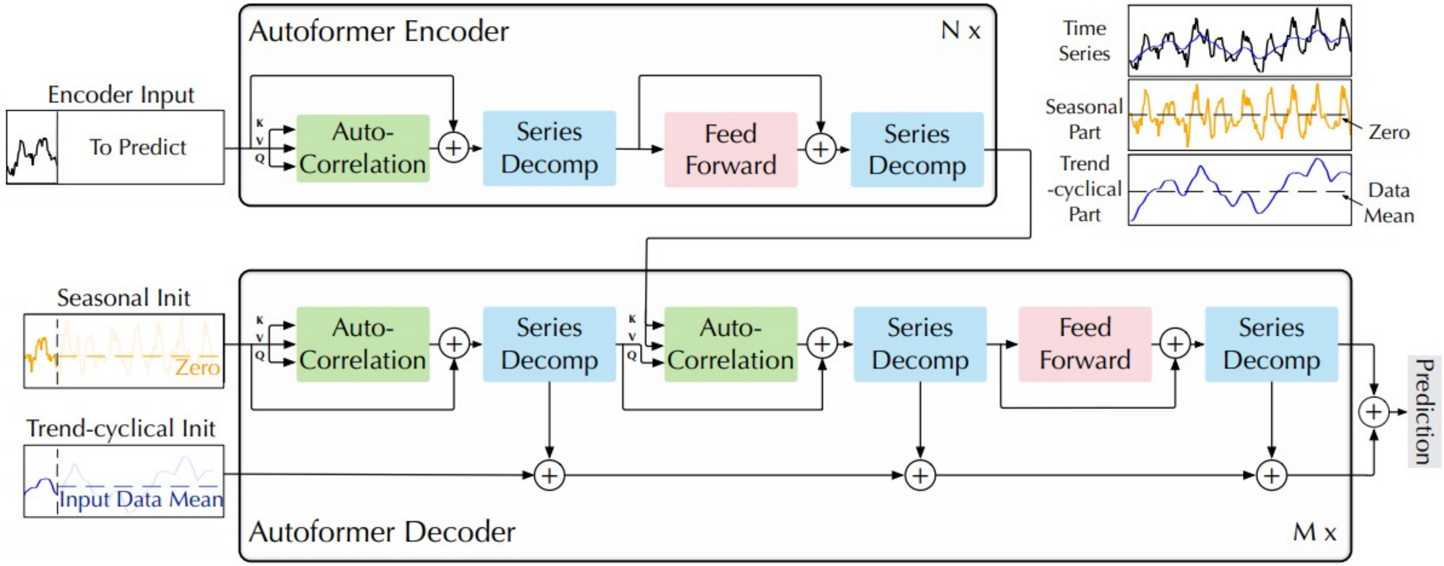
Hands-On

Let’s get our hands dirty in Python

Advanced Transformer Architectures for Time Series

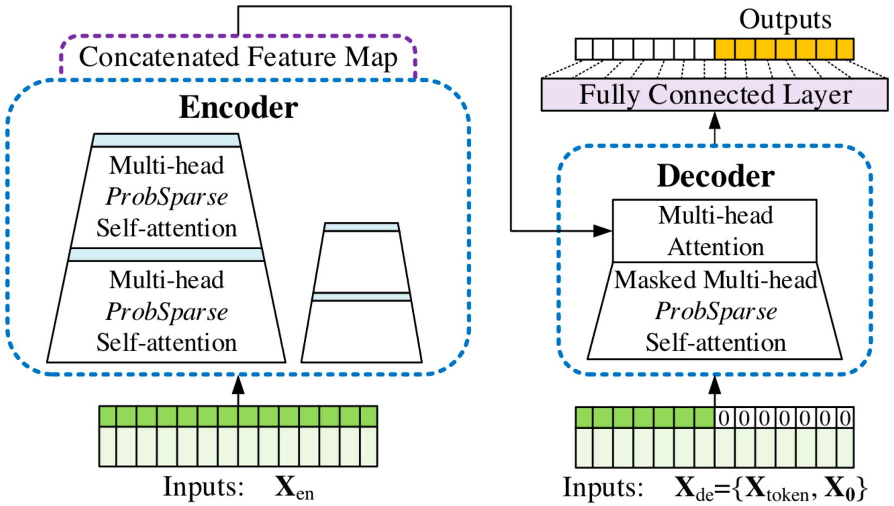
Autoformer and Informer

Autoformer – decomposing the time series components



arXiv:2106.13008

Informer – for long term time series forecasting

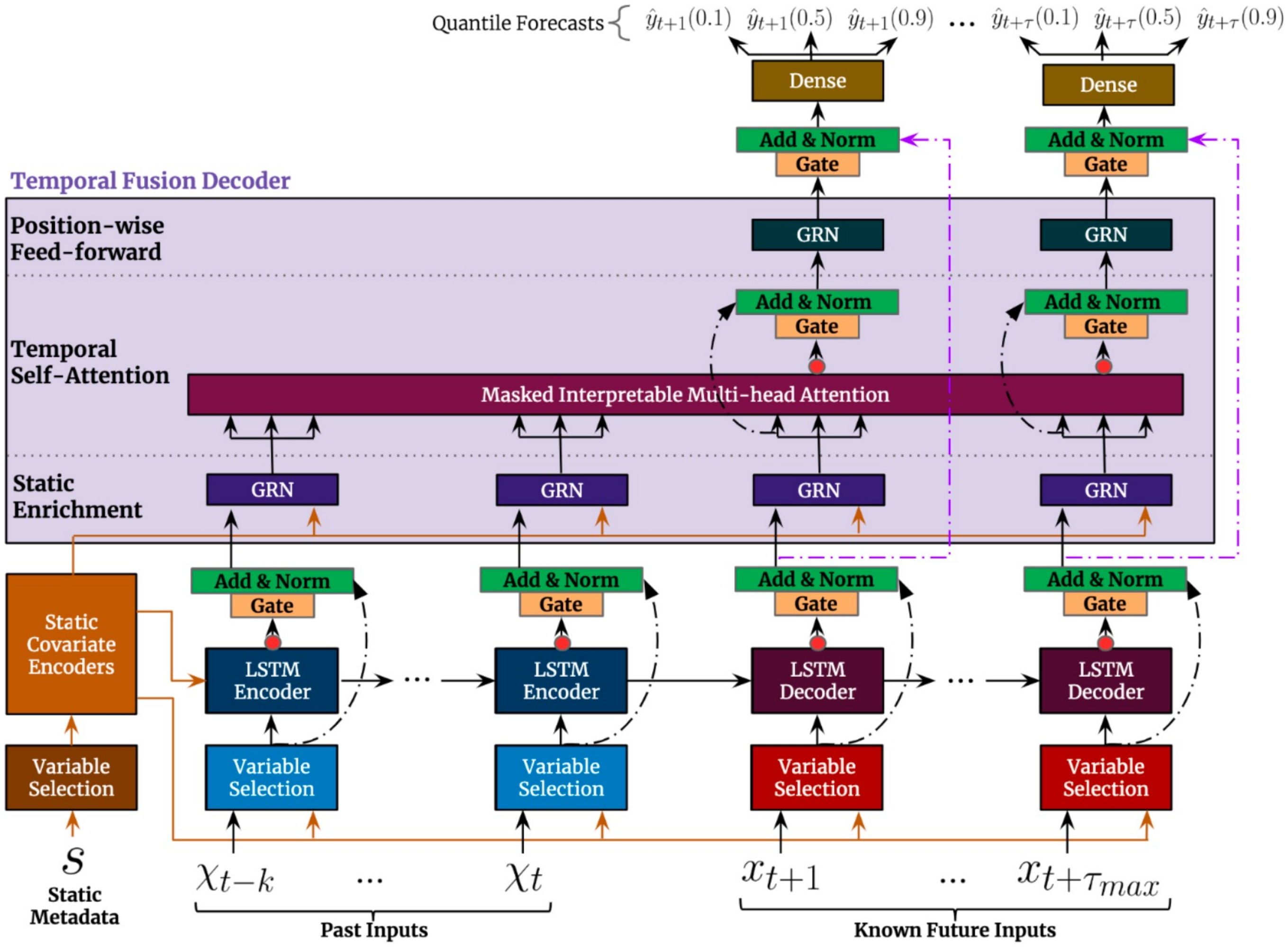


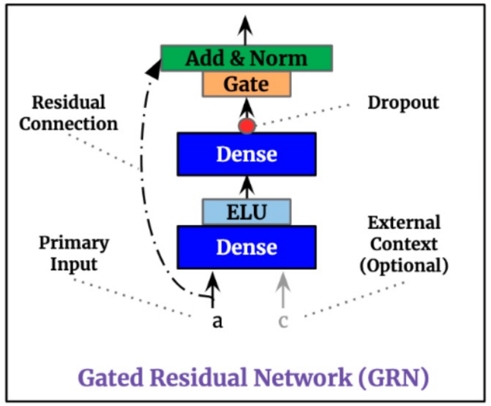
arXiv:2012.07436

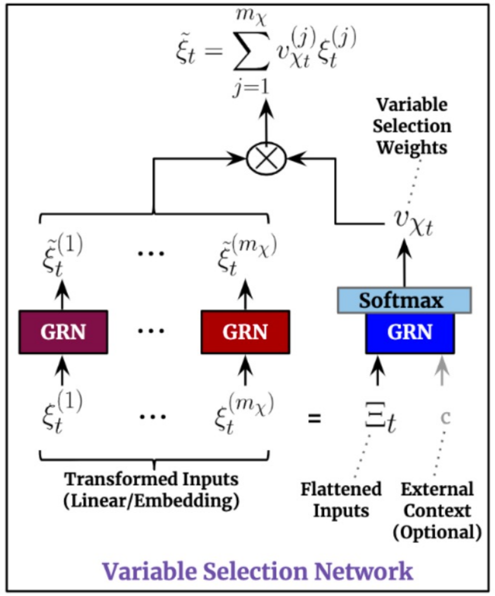
… and many more

Advanced Transformer Architectures

Temporal Fusion Transformer



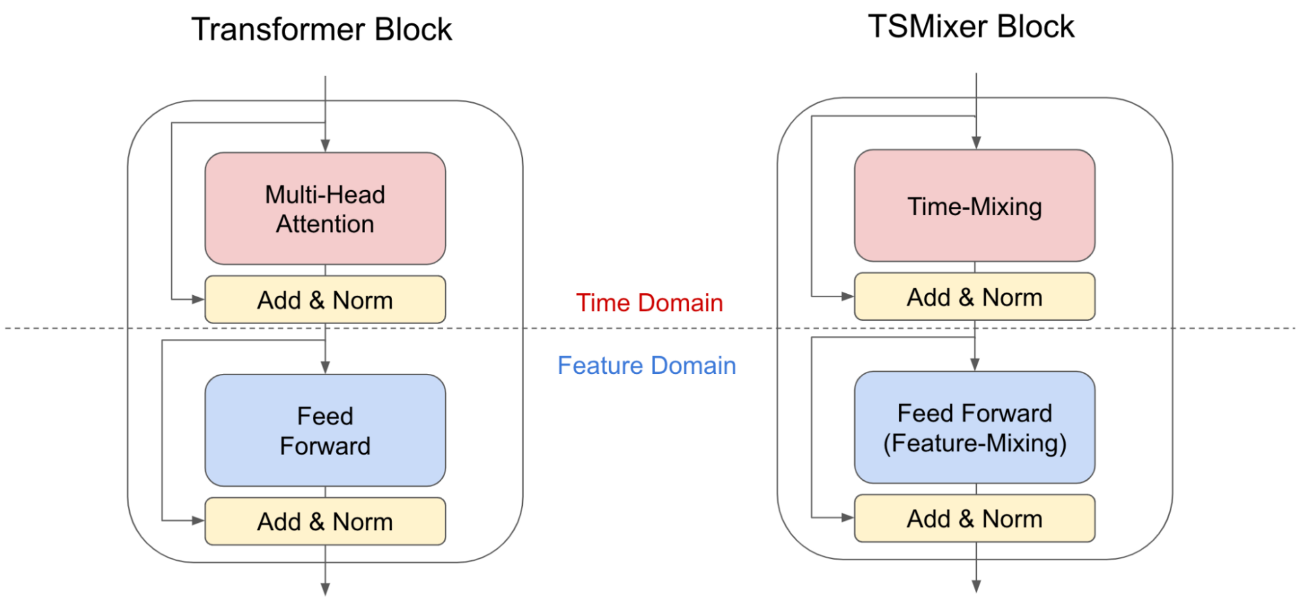


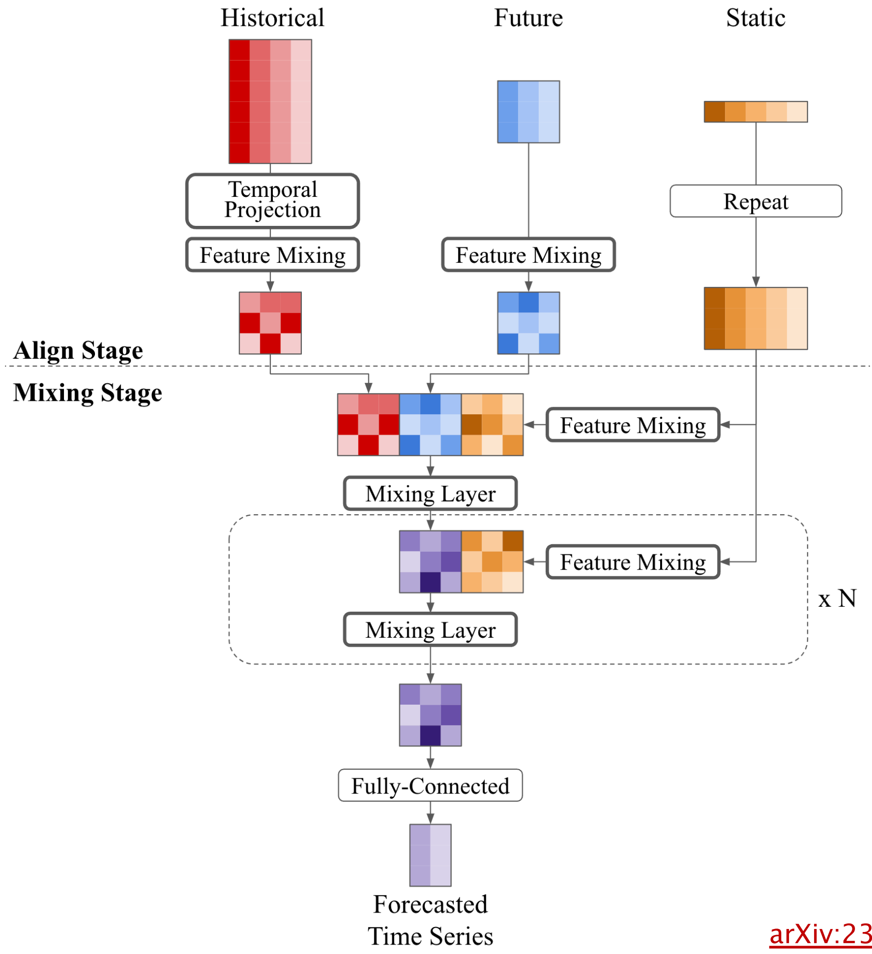


arXiv:1912.09363

Transformer Alternatives for Time Series

TSMixer – An All MLP Architecture for Time Series Forecasting

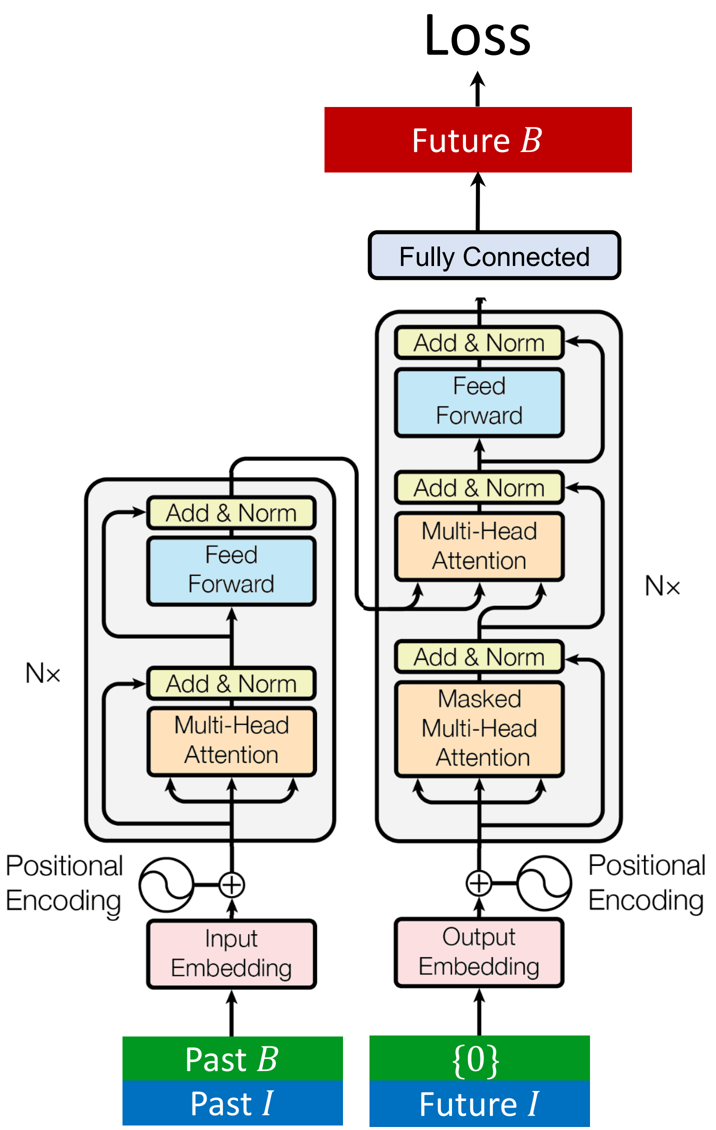




arXiv:2303.06053

Conclusions

- Multivariate time series forecasting is a challenging task
  - Long prediction horizons and high accuracy is especially challenging
- Transformers represent the state-of-the-art of sequence modeling
  - Dominant in language tasks, but not for time series
- We have seen today
  - What goes in to a transformer …
  - … and what comes out on the other side …
  - … and how to do it in Python
- Have fun playing around with transformers!



Appendix

PyTorch Libraries with Transformer implementations for Time Series

- Huggingface transformers
- GluonTS
- Dart
- Neuralforecast
- Pytorch-forecasting