Modeling Stock Trading At Closing with Minimal Transformer

CS7180 Final Project

Bronte Sihan Li, Cole Crescas

01

INTRODUCTION

We introduce the topic and background

02

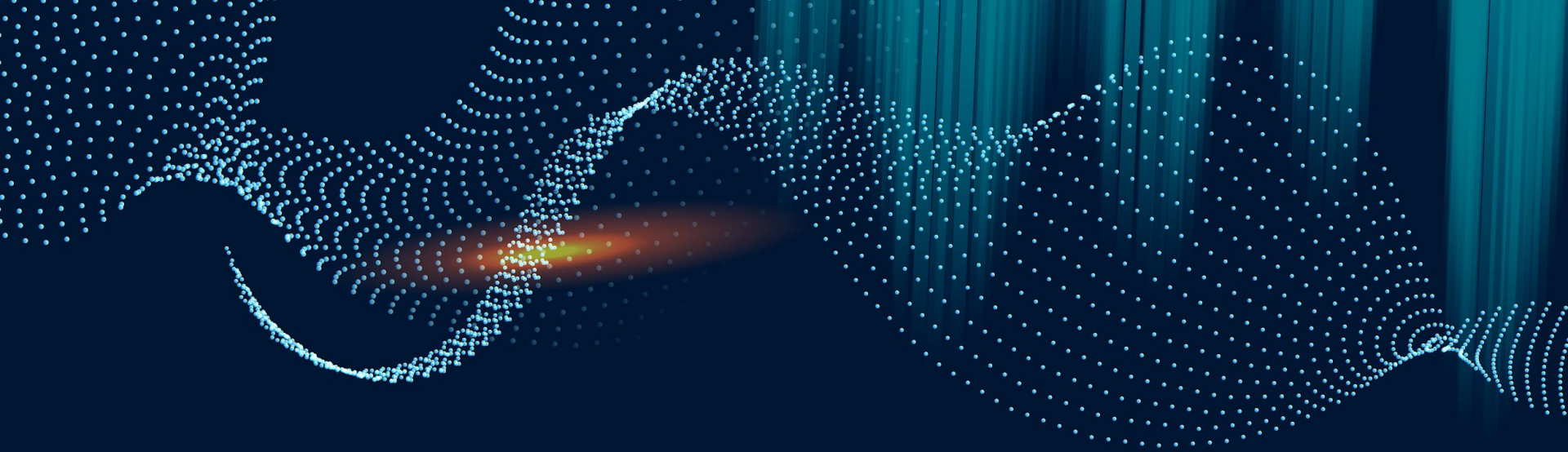
METHODS

We discuss the methods used

03

RESULTS

We present our results and discussion



01 Introduction

Stock prediction and time series data

Stock prediction problem

- “Stock exchanges are fast-paced, high-stakes environments where every second counts. The intensity escalates as the trading day approaches its end, peaking in the critical final ten minutes. These moments, often characterised by heightened volatility and rapid price fluctuations, play a pivotal role in shaping the global economic narrative for the day.”

- The challenge was to predict stock prices in the closing 10 minutes of trading using mean absolute error as the evaluation criteria

Terminology

- **seconds_in_bucket** - The number of seconds elapsed since the beginning of the day's closing auction, always starting from 0.
- **imbalance_size** - The amount unmatched at the current reference price (in USD)
- **[bid/ask]_price** - Price of the most competitive buy/sell level in the non-auction book.
- **[bid/ask]_size** - The dollar notional amount on the most competitive buy/sell level in the non-auction book.
- **wap** - The weighted average price in the non-auction book

$$WAP = \frac{BidPrice * AskSize + AskPrice * BidSize}{BidSize + AskSize}$$

Project timeline

Stage 1

Data exploration and methods design

Stage 2

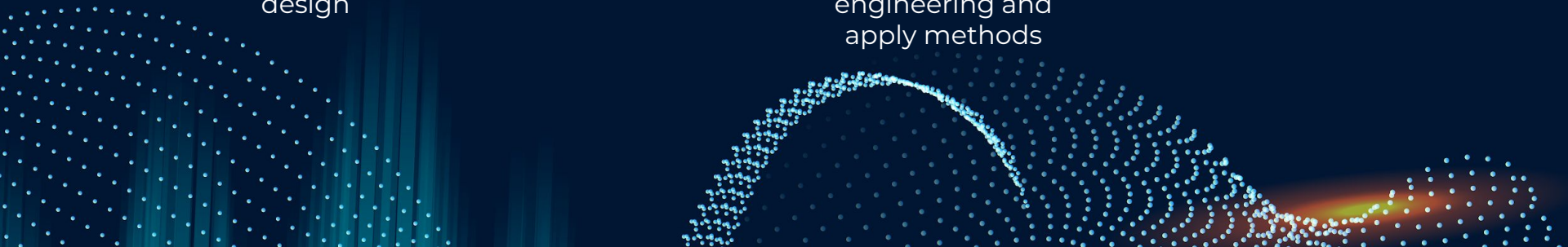
Getting a baseline performance

Stage 3

Additional feature engineering and apply methods

Stage 4

Hyperparameter tuning and compile results



Previous work

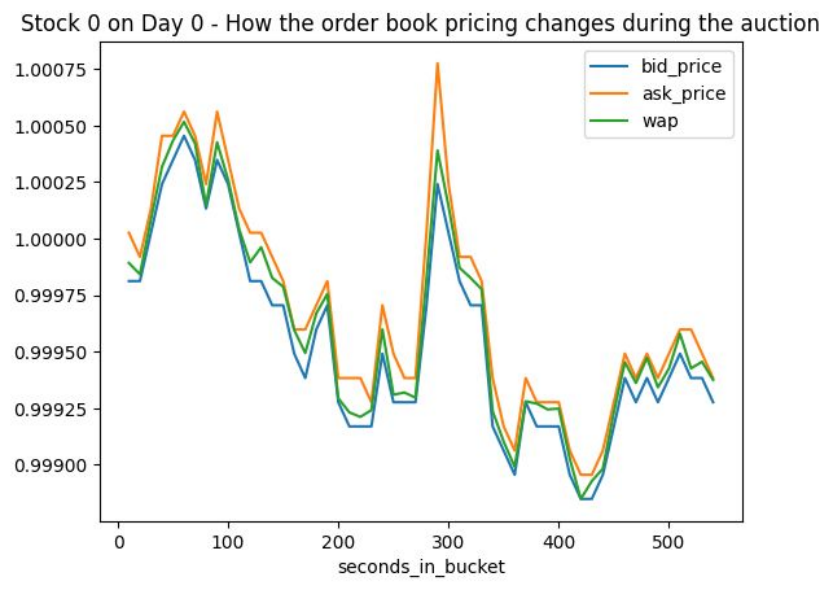
- Daiya et al. 2021 proposed a **multi-modal framework** that leverages financial indicators as well as news to improve prediction accuracy
- Yoo et al. 2021 focuses on finding **correlation between stocks** and designed a multi-level context building approach
- Hu et al. 2021 took a similar approach comparing methods Support Vector Regression (SVR) and Long Short-Term Memory (LSTM) and Temporal Fusion Transformer (TFT) where **TFT demonstrated superior results**
- Simhayev et al. 2023 **proved that simple transformer models could outperform** more complex models and linear models on time series datasets

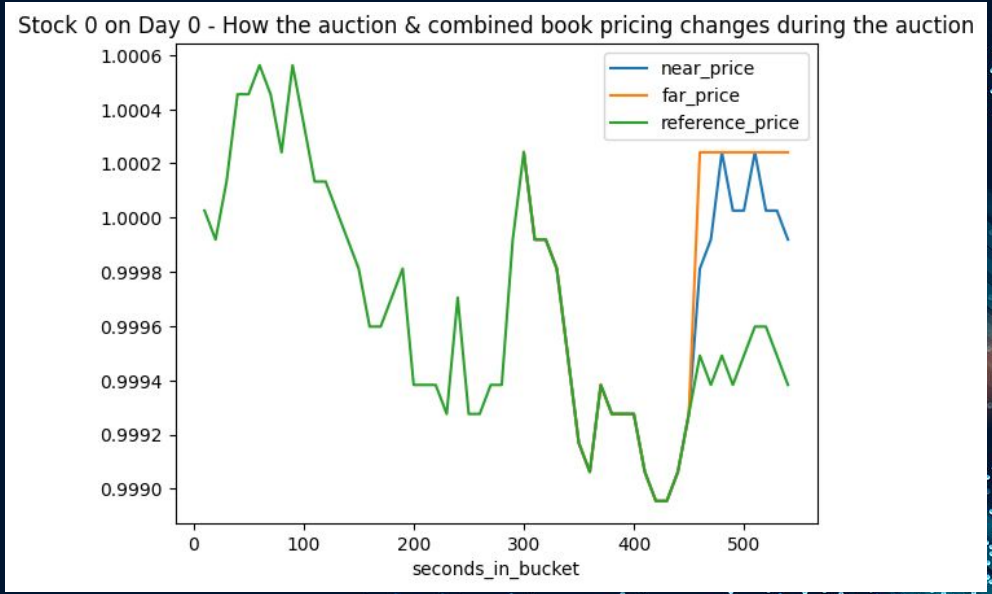
Stock prediction dataset

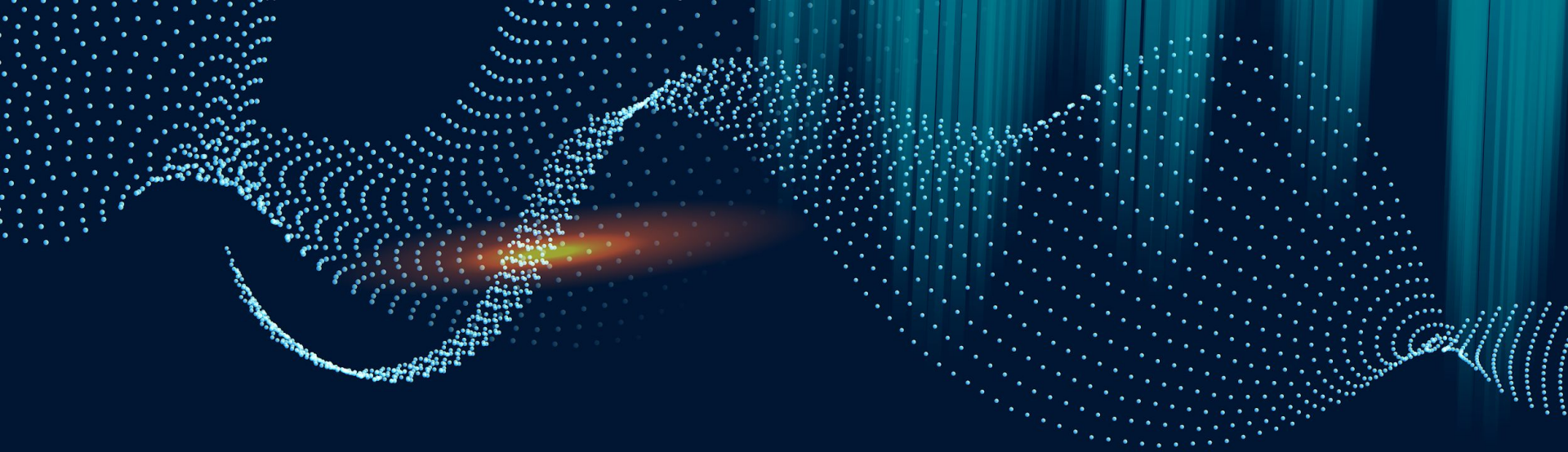
- The synthetic index is a custom weighted index of Nasdaq-listed stocks constructed by Optiver for this competition.
- The unit of the target is basis points, which is a common unit of measurement in financial markets. A 1 basis point price move is equivalent to a 0.01% price move.
- Where t is the time at the current observation, we can define the target:

$$Target = \left(\frac{StockWAP_{t+60}}{StockWAP_t} - \frac{IndexWAP_{t+60}}{IndexWAP_t}\right) * 10000$$

Sample chart for one stock







02 Methods

Classical machine learning vs. deep learning

Feature Engineering

We focus on producing a target time series for each stock

Stock ID
Date ID
Imbalance size
Reference price
Far price
Near price
Matched size
Weighted average price
...

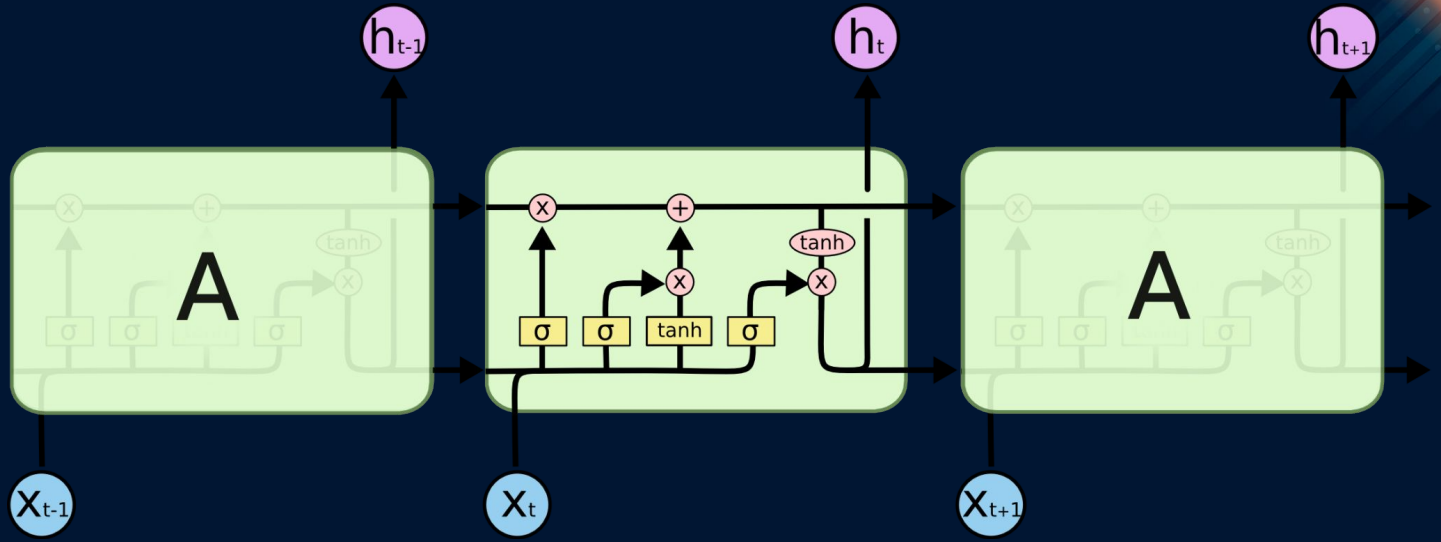
	stock_id	seconds_in_bucket	imbalance_size	imbalance_buy_sell_flag	reference_price	matched_size	far_price	near_price	bid_price	bid_size	...
0	0.0	0.0	3180602.8	1.0	0.999812	13380277.0	0.000000	1.000000	0.999812	60651.50	...
191	0.0	10.0	1299772.8	1.0	1.000026	15261107.0	0.000000	1.000000	0.999812	13996.50	...
382	0.0	20.0	1299772.8	1.0	0.999919	15261107.0	0.000000	1.000000	0.999812	4665.50	...
573	0.0	30.0	1299772.8	1.0	1.000133	15261107.0	0.000000	1.000000	1.000026	55998.00	...
764	0.0	40.0	1218204.4	1.0	1.000455	15342675.0	0.000000	1.000000	1.000241	14655.95	...
...	...	...	...	...	...	...	...	...	...	...	...
104420	0.0	500.0	0.0	0.0	0.998999	13711310.0	0.998999	0.998999	0.998795	29397.00	...
104612	0.0	510.0	0.0	0.0	0.998999	13711310.0	0.998999	0.998999	0.998897	19600.00	...
104804	0.0	520.0	0.0	0.0	0.998999	13711310.0	0.998999	0.998999	0.998897	11760.00	...
104996	0.0	530.0	0.0	0.0	0.998999	13711310.0	0.998999	0.998999	0.998897	10192.00	...
105188	0.0	540.0	0.0	0.0	0.998999	13711310.0	0.998999	0.998999	0.998897	29596.00	...

	stock_id	date_id	seconds_in_bucket	target
0	0	0	0	-3.029704
1	0	0	10	0.389814
2	0	0	20	4.220009
3	0	0	30	5.450249
4	0	0	40	3.169775
...	...	...	...	...
5290995	199	480	500	-7.209778
5290996	199	480	510	-9.750128
5290997	199	480	520	3.629923
5290998	199	480	530	4.760027
5290999	199	480	540	-6.530285

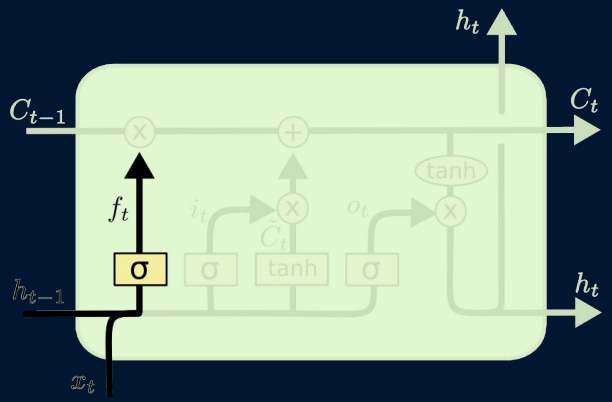
	0	1	2	3	4
55	2.909899	-9.719729	1.569986	18.889904	3.089905
56	-1.959801	-15.720129	2.889633	2.989769	6.519556
57	-7.609725	-8.620024	4.969835	0.300407	6.459951
58	-8.879900	4.229546	2.260208	-1.839995	3.620386
59	-8.339882	-1.479983	1.319647	-1.559854	-0.489950

LSTM

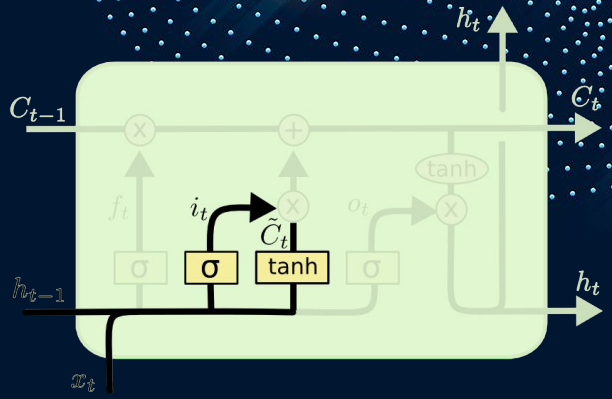
Introduced by Hochreiter & Schmidhuber (1997) to learn long term dependencies

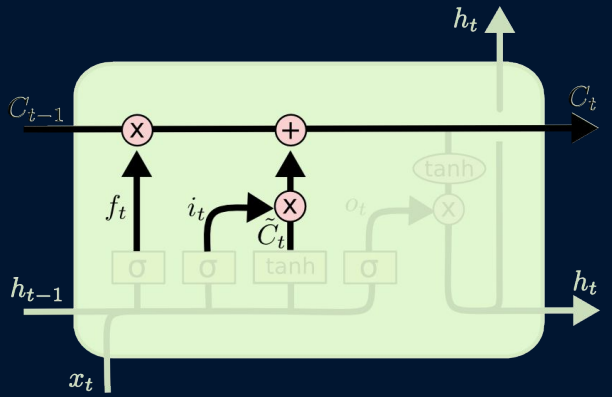


forget gate layer

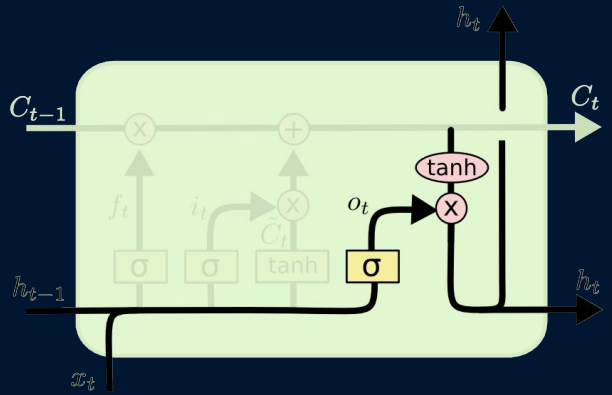


input gate layer



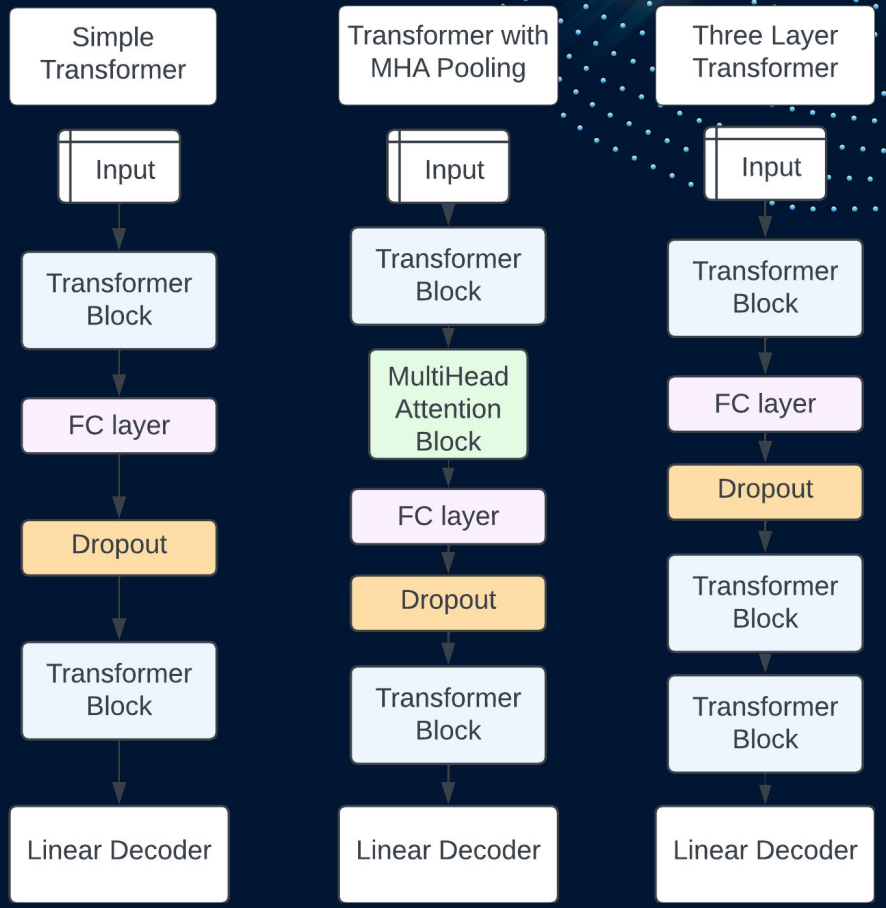


update cell state



Decide on the output

Simple Transformer Iterations



Temporal Fusion Transformer

- Attention Mechanism for interpretability:

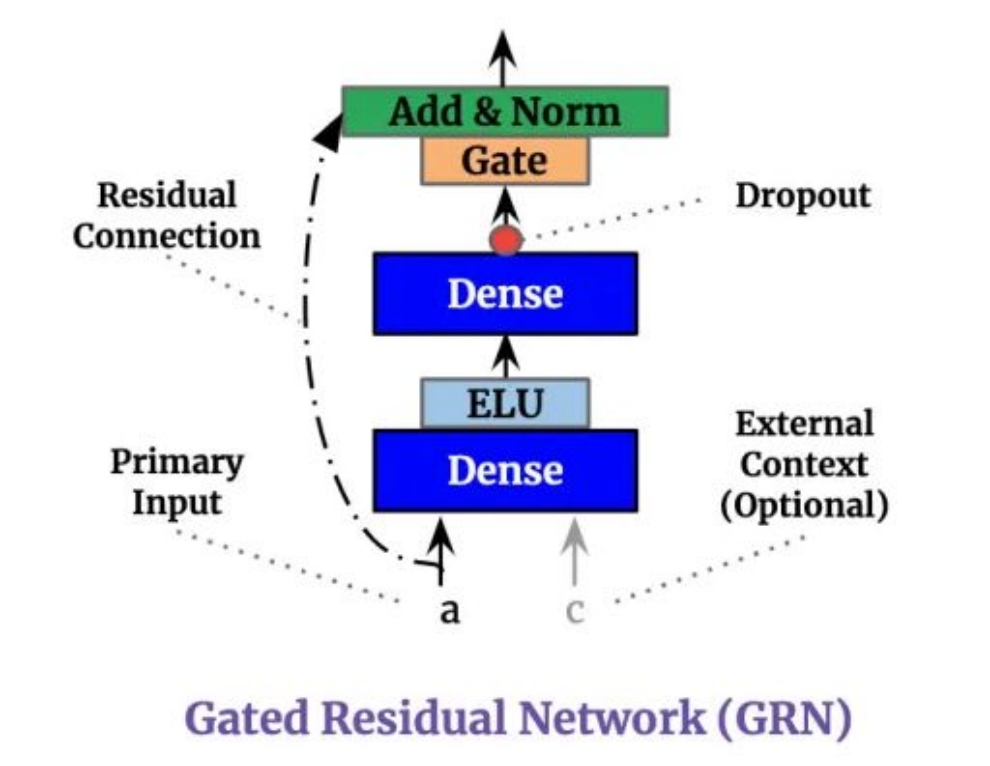
$$InterpretableMultiHead(Q,K,V) = \frac{1}{h}\sum_{i=1}^{n} head_i W_H$$

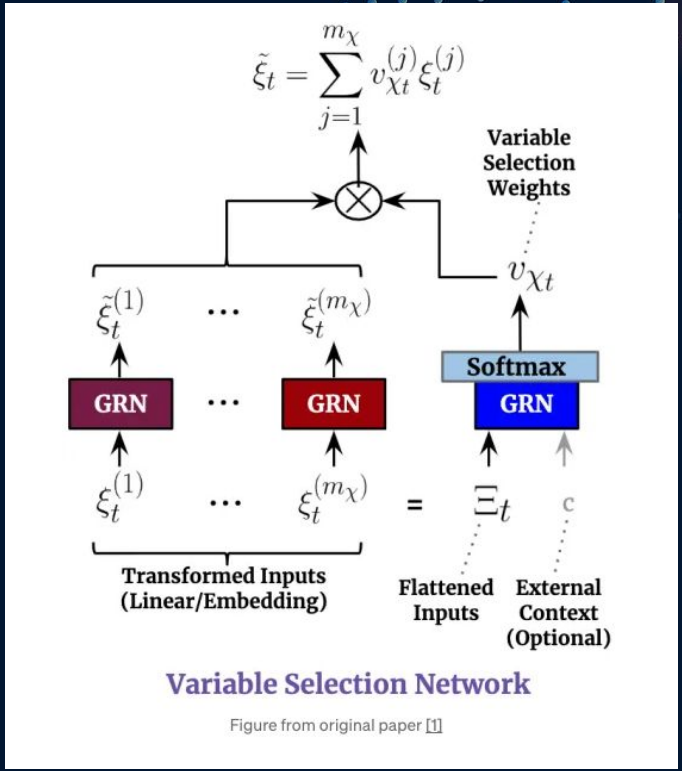
$$\text{where } head_i = Attention(QW_Q^{(i)}, KW_K^{(i)}, VW_V)$$

- Sequence to Sequence
- Variable Selection
- Quantile regression:

$$QL(y,\hat{y},q) = q\max(0, y-\hat{y}) + (1-q)\max(0,\hat{y}-y)$$

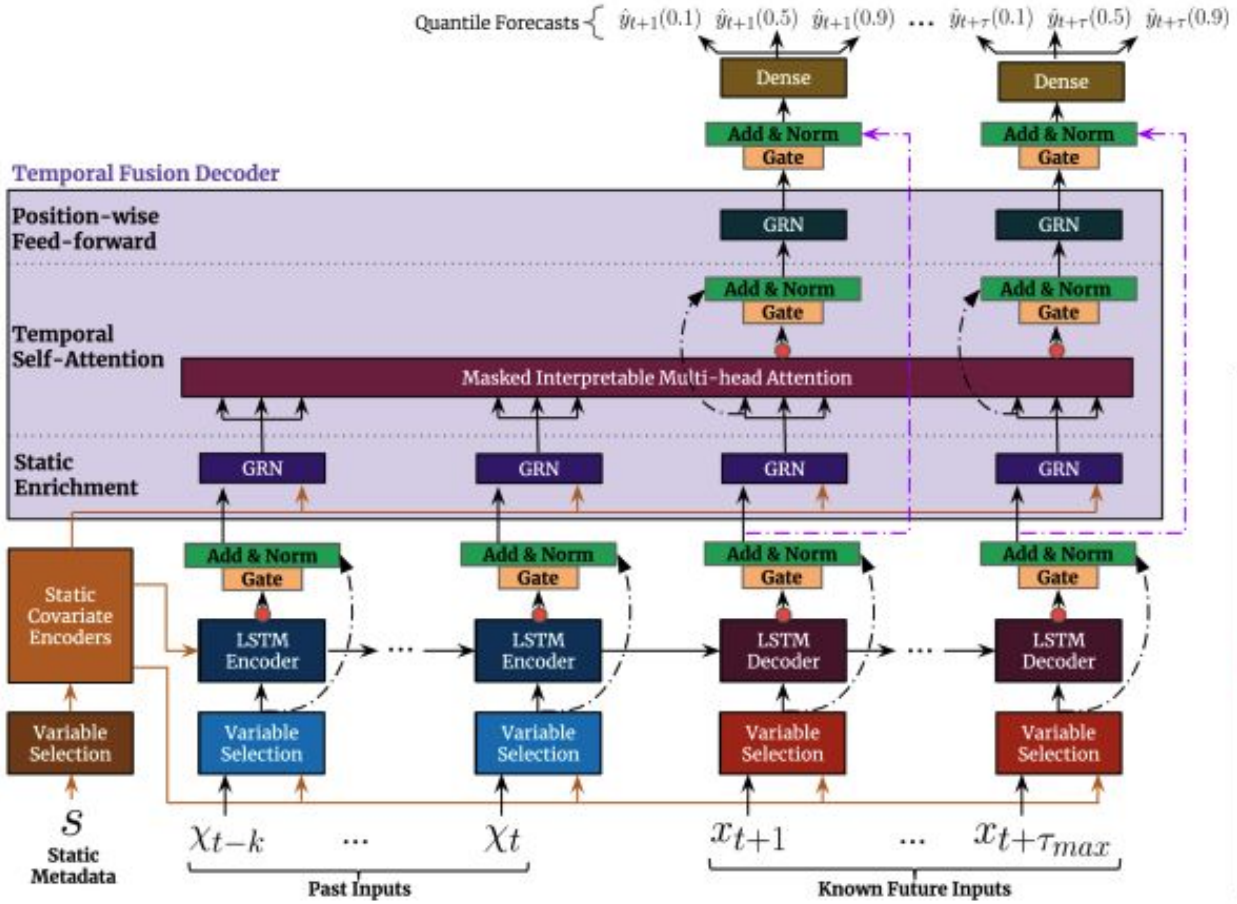
Temporal Fusion Transformer

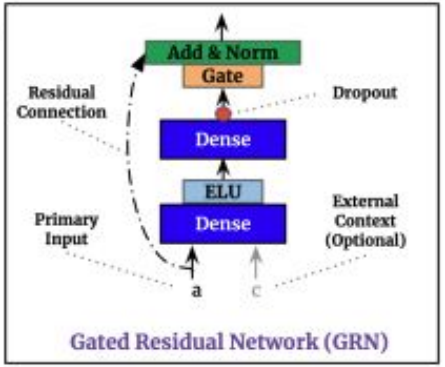


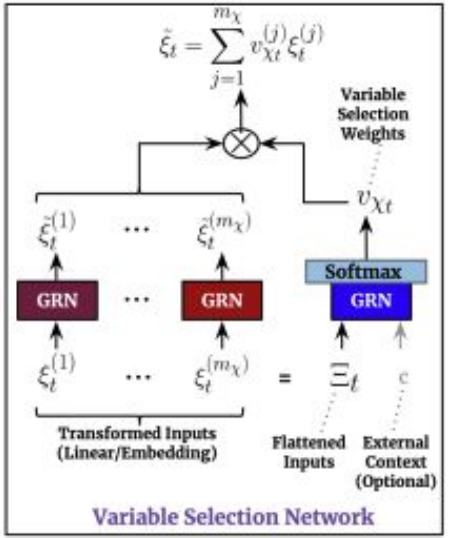


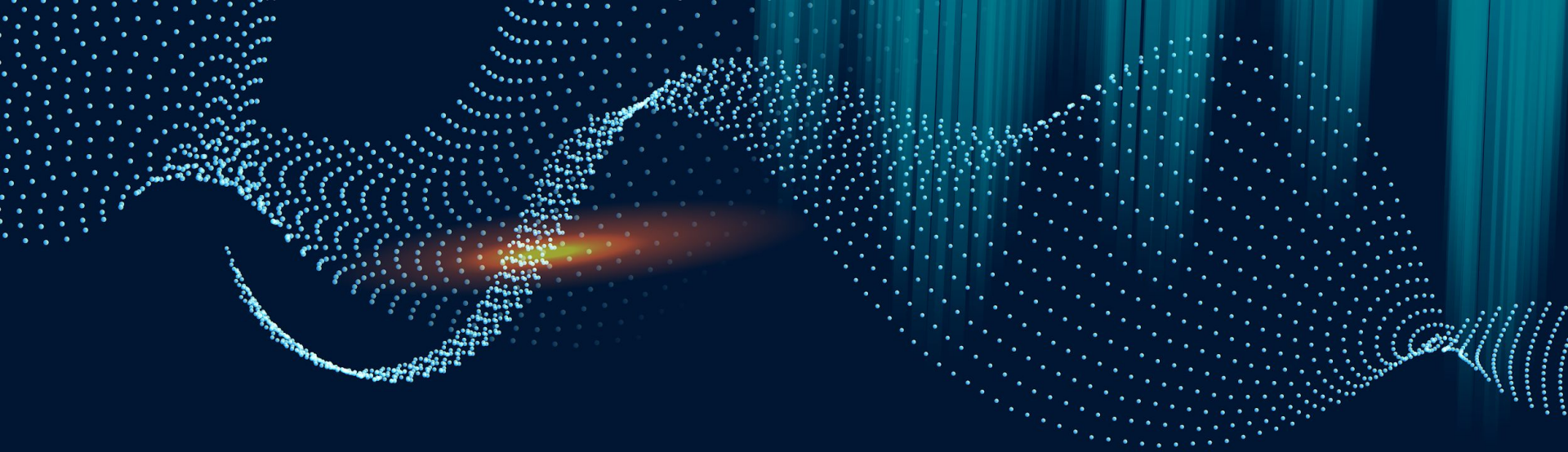
*ELU = Exponential Linear Unit activation function

Temporal Fusion Transformer









03 Results

Experiments and discussion

Transformer parameter tuning

Iteration	Layers	Pooling?	n_heads	MAE	Dropout %	Activation Function
1	2	No	8	5.415	50%	ReLu
2	3	Yes	8	5.83	50%	ReLu
3	2	No	8	5.425	50%	ReLu
4	2	No	4	5.425	50%	ReLu
5	2	No	8	5.192	25%	ReLu
6	2	No	8	5.19	10%	GeLu
7	2	No	8	5.116	10%	ReLu

Temporal Fusion Transformer tuning

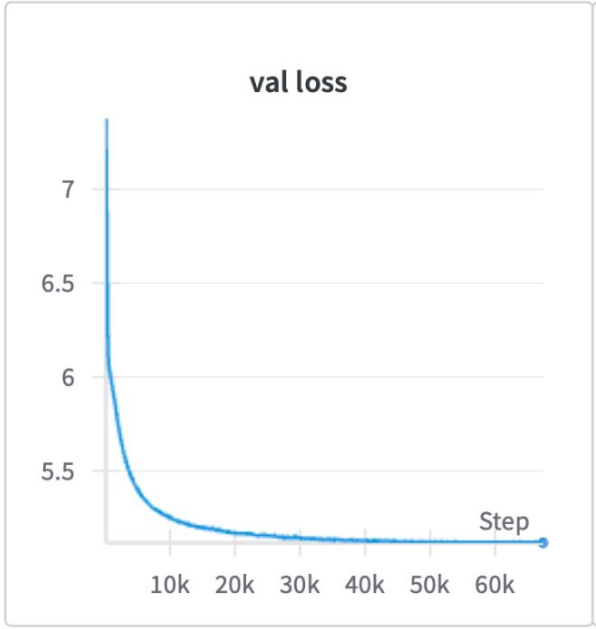
Hidden size	Learning Rate	n_heads	MAE	Dropout %	Epochs
8	.041	2	5.411	10%	11
64	.037	8	5.361	10%	12

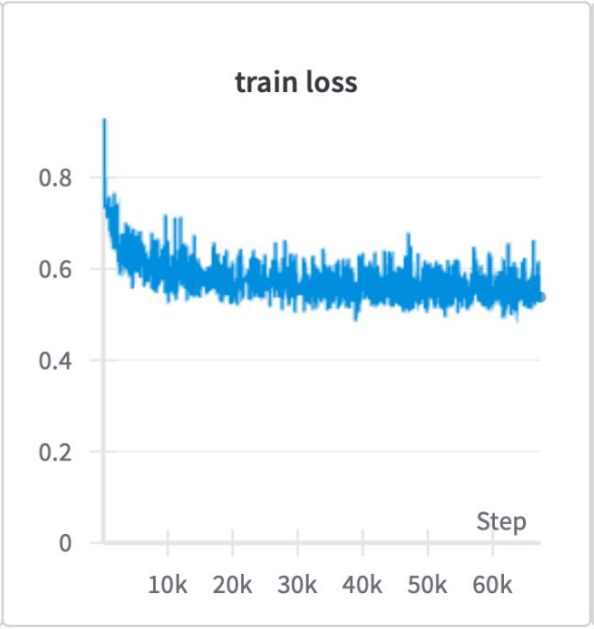
* Hidden size is used for continuous variables
* ELU is the default activation function
* We used a learning rate optimizer function from pytorch.Tuner
* Early stopping is used

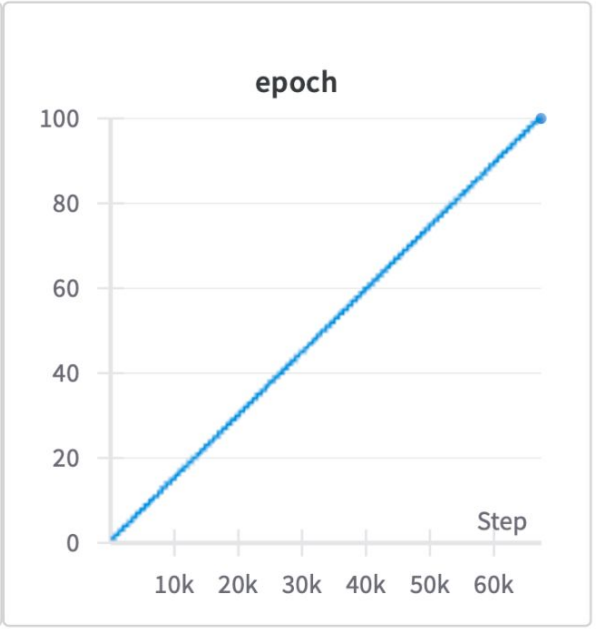
Machine Learning vs Deep Learning

Model	Validation MAE	Link
XGB	6.27	NA
LightGBM	6.202	NA
LSTM w/Kalman	5.424	WANDB
LSTM	5.519	WANDB
Transformer w/Kalman	5.425	WANDB
Temporal Fusion Transformer	5.361	Notebook in repo
Best Transformer Model	5.116	WANDB

Best Transformer Model curves







Model Metadata

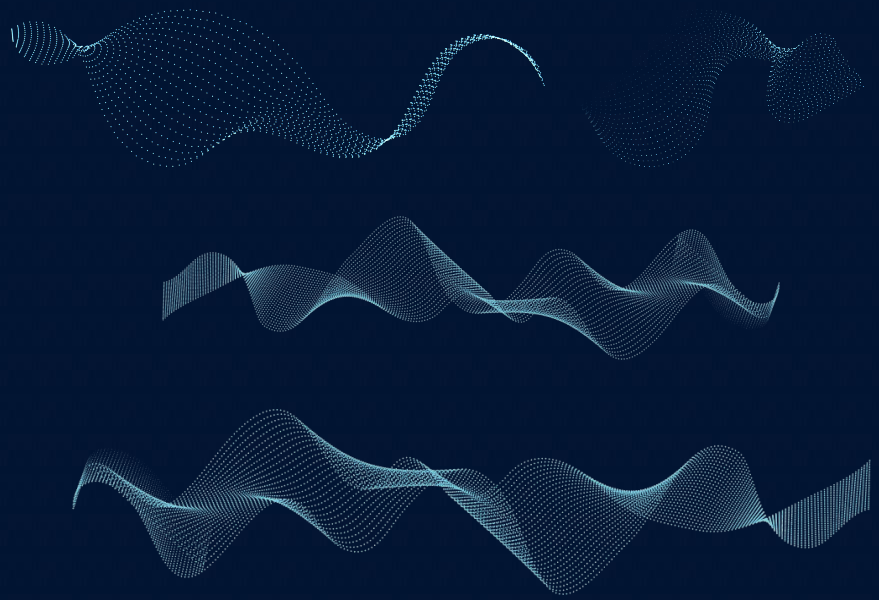
Model	Params	Throughput (samples per second)
LSTM	0.1M	16185
Simple Transformer	6.6M	25417
Transformer with MHA pooling	6.6M	18194
Three layer Transformer	9.8M	22827

0.03 millisecond

Average Inference time of our
Simple Transformer

On a Nvidia A6000 GPU

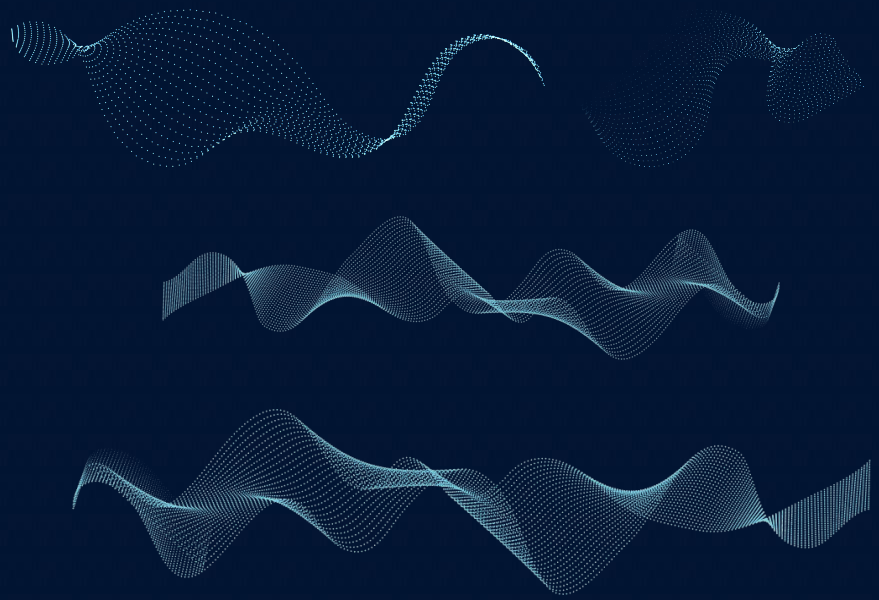
Discussion



Thoughts:

- Simple is sometimes better as proven from this project
- If we were using a more complex dataset with more features or a larger time range a complex model might perform better
- Our results corroborated findings from related work
- Often simple transformers work best for time series prediction

Discussion



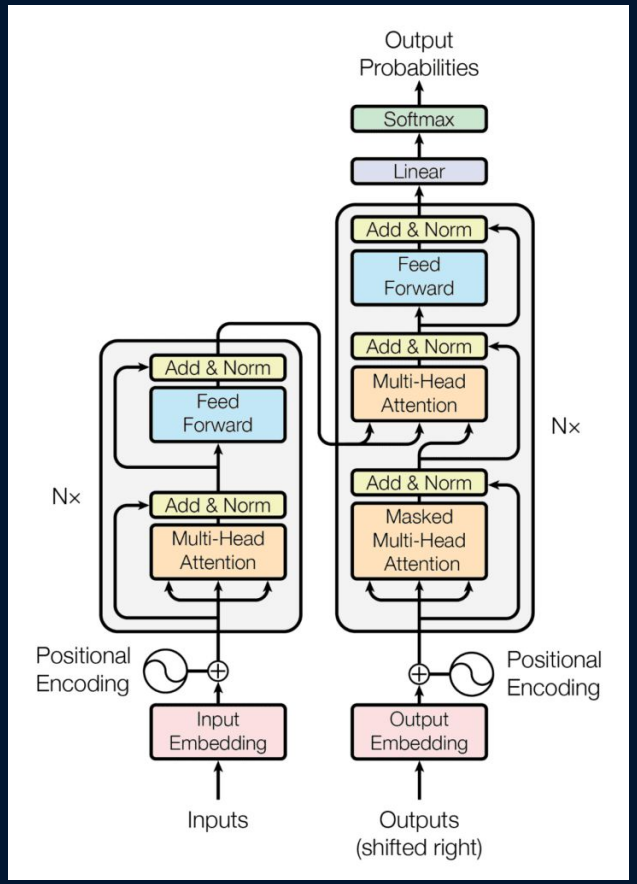
Potential applications

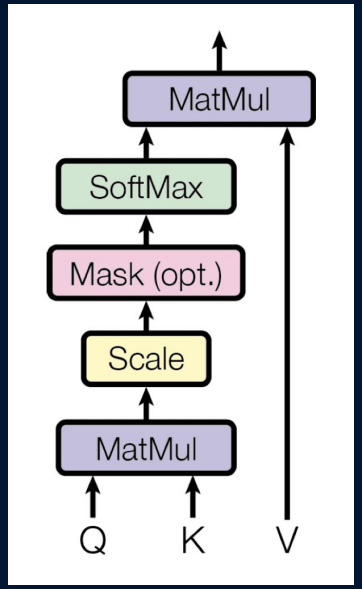
- Real-time stock movement prediction
- Scaling up to larger number of stocks and longer time series
- Aid in trading and auction book reconciliation automation

Limitations

- Trade-off between performance and model complexity
- Data may not always be available in real-time
- No evaluation on larger data sets or incorporation of sentiment data
- Analysis on accuracy per stock would be helpful

Appendix





Transformer Block

Kalman Filter

A Kalman filter is an **optimal estimator** - ie infers parameters of interest from indirect, inaccurate and uncertain observations.

It is **recursive** so that new measurements can be processed as they arrive.

The inputs are noisy and sometimes inaccurate measurements. The outputs are **less noisy** and sometimes more accurate estimates