

# Real-Time Crypto Trading System

Rich (Chen Feng) Tsai

Github repo: <https://github.com/ChenFengTsai/Crypto-real-time-trading-system>

## Tools Used



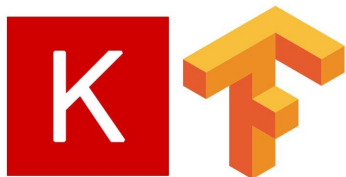
asyncio



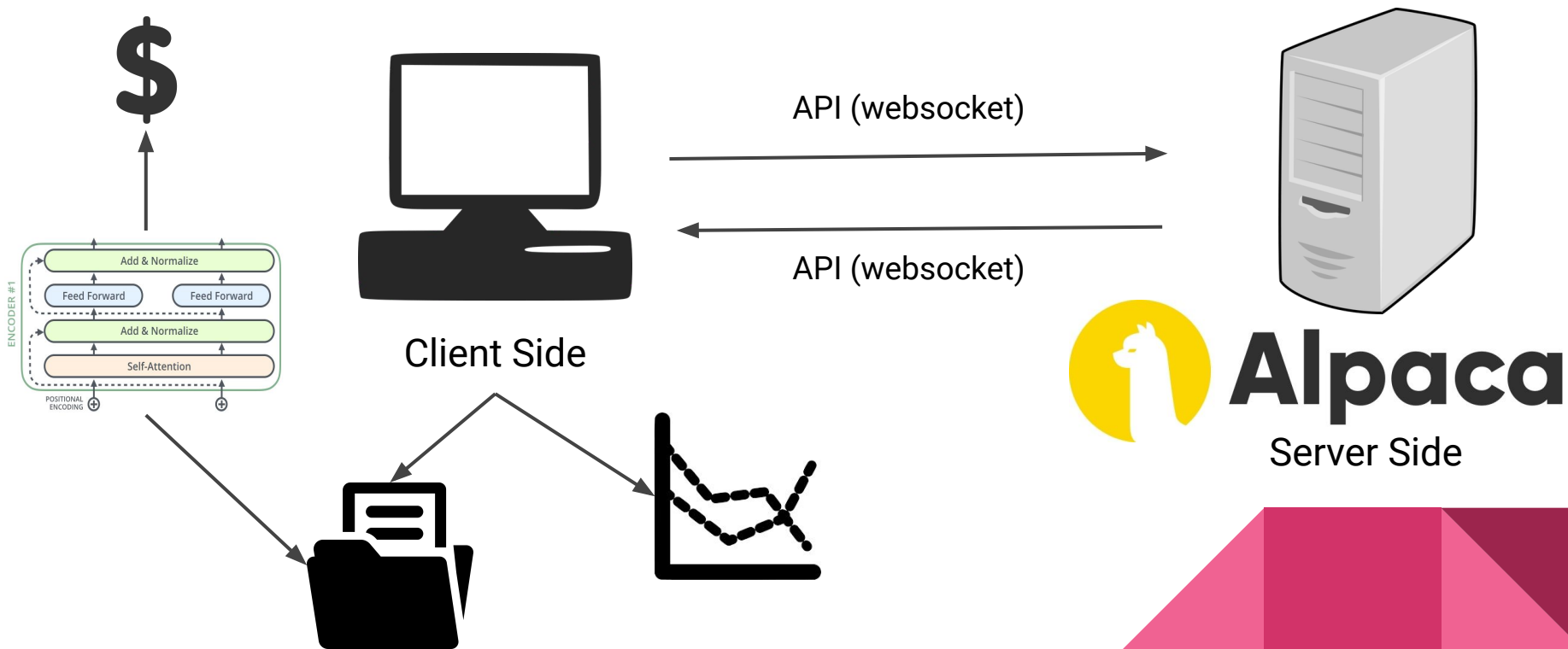
# Alpaca

- `alpaca_trade_api`
- `alpaca.data.live`

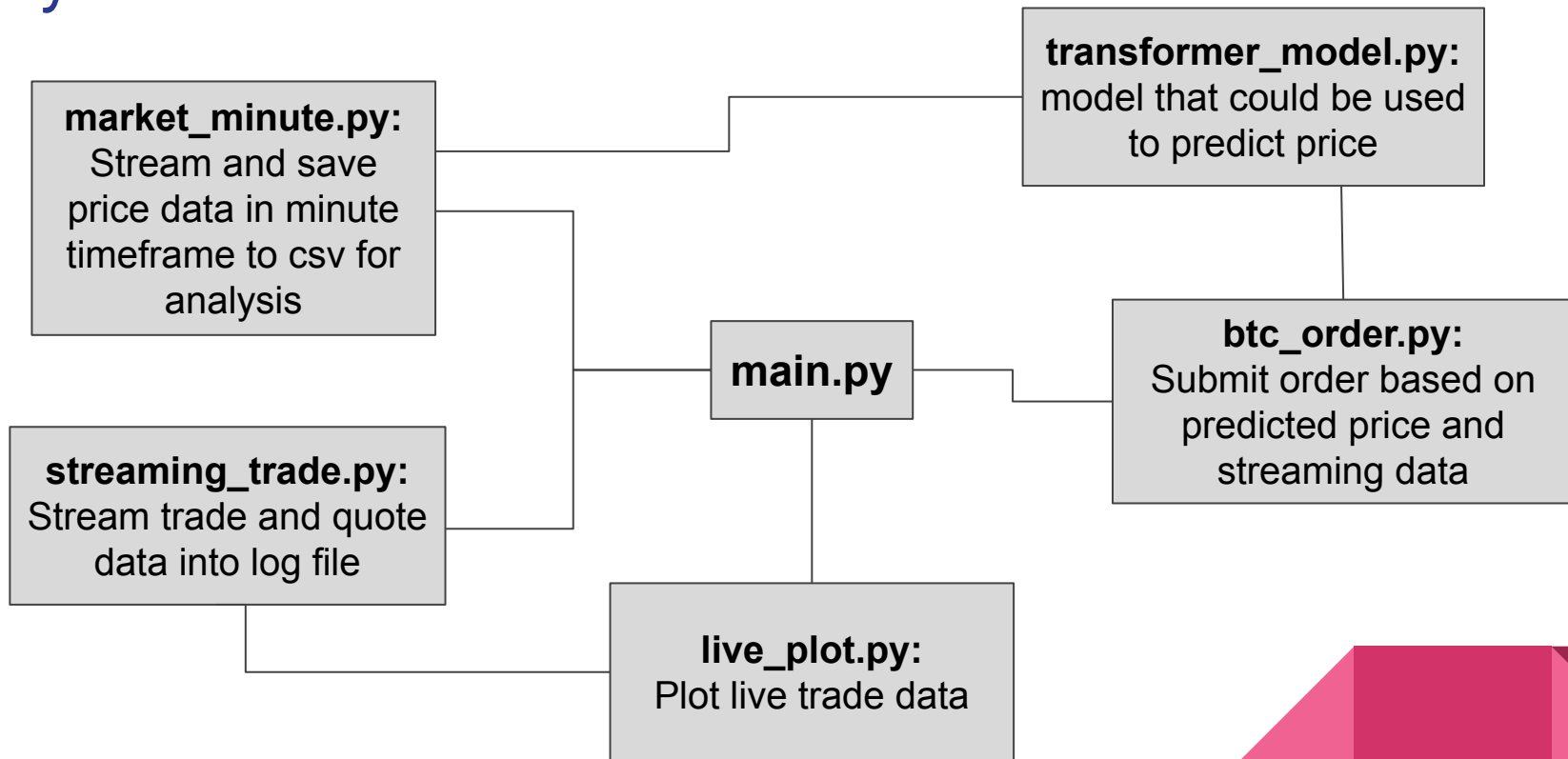
# matplotlib



# Client Server Diagram



# System Architecture



# Structure

## **main.py:**

Trigger streaming, plotting, loading data

## **market\_minute.py:**

Extract data in minute timeframe and save them under the price\_bar\_minute folder

## **streaming\_trade.py:**

Extract trade, quotes data in real-time manner and save them under the logging folder

## **live\_plot.py:**

Plot the trade data in live

## **transformer\_model.py:**

Layers of the BERT + Time embeddings model

## **btc\_order.py:**

Submit order based on predicted price by transformer

## **model folder:**

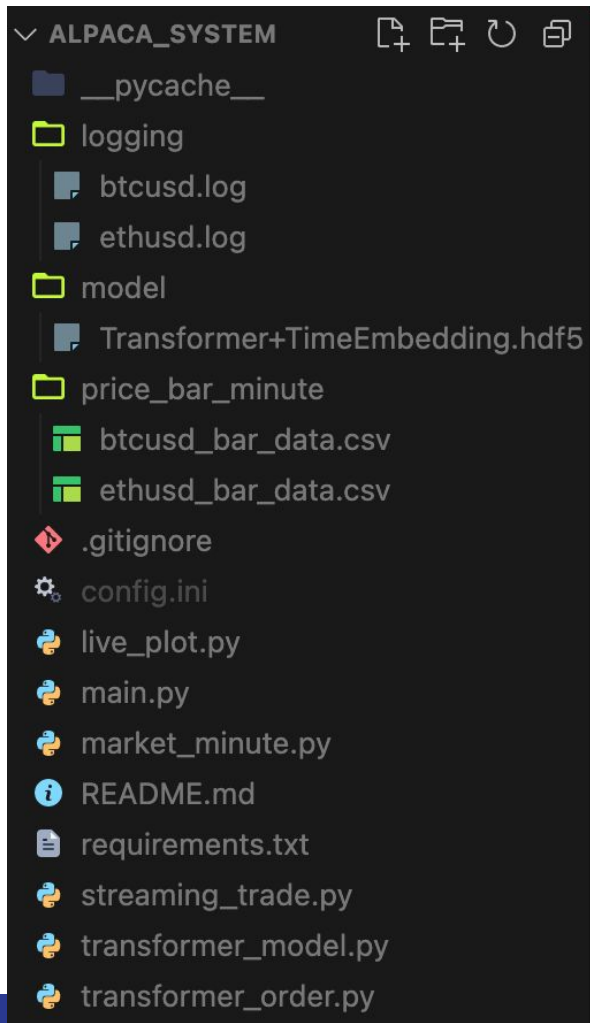
The final model is under this folder

## **requirements.txt:**

Required packages is listed in this file

## **config.ini:**

Configuration info is listed in this file



# Requirement

## **requirements.txt:**

To run the code, you need to install the packages in the requirements.txt file, which includes alpaca API, tensorflow, asyncio, etc

## **config.ini:**

To initialize connection with alpaca API, you need to have API key and secret key, and save them into the config.ini file.



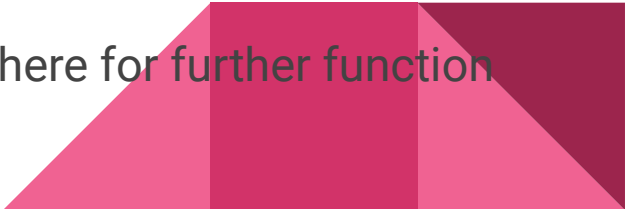
# main.py

From the main.py file, you can decide which file to run by adding argument in the command line, like

```
python3 main.py --symbols BTC/USD ETH/USD --func stream
```

- In the “symbols” argument, you can put several crypto for further operation.
- In the “func” argument, you can put one of the options: ['stream', 'collector', 'plot'] to conduct respective operation.
  - stream → streaming\_trade.py
  - collector → market\_minute.py
  - plot → live\_plot.py

In the main.py, I initialize the needed client or subscribers here for further function operation.



# market\_minute.py

After running `python3 main.py --symbols BTC/USD ETH/USD --func collector` the code will start listen to BTC and ETH through the `CryptoDataStream` package in alpaca API and start streaming bar data with close, open, high, low, and volume in a “minute” timeframe into the csv file. We can further utilize these files for modeling.

If you would like to look at the infromation of the minute data about btc, click in `btcsud_bar_data.csv` file under the `price_bar_minute` folder.  
For example,

Symbol ▼	Timestamp	Open ▼	High ▼	Low ▼	Close ▼	Volume ▼	Trade_count	Vwap ▼
BTC/USD	2023-05-09 2	27734.43	27734.43	27724.87	27734.43	0.03	9	27733.95
BTC/USD	2023-05-09 2	27734.43	27747.39	27734.43	27747.39	0.2	16	27736.3
BTC/USD	2023-05-10 0	27687.54	27687.54	27670.11	27671.82	0.18	14	27679.3
BTC/USD	2023-05-10 0	27679.74	27683.9	27671.75	27671.75	0.06	6	27673.83
BTC/USD	2023-05-10 0	27682.32	27687.54	27671.75	27678.16	0.07	14	27678.55
BTC/USD	2023-05-10 0	27678.17	27687.54	27676.62	27679.96	0.11	24	27678.71
BTC/USD	2023-05-10 0	27687.54	27693.94	27675.52	27675.52	0.1	12	27683.61
BTC/USD	2023-05-10 0	27683.25	27685.14	27670.12	27672.78	0.13	19	27676.32



# streaming\_trade.py

After running `python3 main.py --symbols BTC/USD ETH/USD --func stream` the code will start listen to BTC and ETH through the `Stream` package in alpaca API and start streaming trade and quote data to the log file. We can further utilize these files for analysis in a granular manner.

If you would like to look at the logs about btc, click in `btcusd.log` file under the `logging` folder. For example,

```
2023-05-09 16:17:22,843 - INFO - quote {'T': 'q', 'S': 'BTC/USD', 'bp': 27720.0, 'bs': 0.01036, 'ap': 27731.36, 'as': 0.03171, 't': Timestamp(seconds=1683667042, nanoseconds=781915298)}
```

```
2023-05-09 16:17:21,923 - INFO - trade {'T': 't', 'S': 'BTC/USD', 'p': 27724.75, 's': 0.00364, 't': Timestamp(seconds=1683667041, nanoseconds=921000000), 'i': 16586143, 'tks': 'S'}
```

# live\_plot.py

After running `python3 main.py --symbols BTC/USD ETH/USD --func plot` the code will start listen to BTC and ETH through the **Stream** package in alpaca API and start streaming trade data and plot them in live. In this file, I inherit properties in AlpacaStream class object in streaming\_trade to get the real time log data.



# transformer\_model.py

## Includes:


The basic layers of the transformer model used to predict the price (BERT structure + Time embeddings)

No training process in this project, only the structure.

The inputs for the model to make predictions are the percentage changes of the bar data with **“close”**, **“open”**, **“high”**, **“low”** and **“volume”** in a hourly timeframe.

The predicted output will be the percentage changes of the price for the next hour. After getting the percentage changes, we can transform it back to the “price” scale.


There is also a load\_model() function here for us to load the model back in the transformer\_order.py file.




# transformer\_order.py

After running `python3 main.py --func run_order --script btc_order.py` we will submit order based on the predicted price on transformer model in the hourly timeframe. Since the transformer model I built is only training on BTC data, I can now only make prediction for BTC and submit BTC order.

Steps in the code:

1. Load model
  2. Get next hour predicted price
  3. **Place a market buy order:** Do not have a position and the current price < predicted price
  4. **Place a market sell order:** Do have a position and the current price > predicted price
  5. Execute once an hour
- 

# Future Works

1. Design notification system to alert me of price drop or price surge
  2. Train more models for different crypto not just btc
  3. Use more diverse models like moving average model which does not require training
  4. Design more complex strategies, not just one order per hour
  5. Include news sentiment analysis into our models
  6. Combine several functionality together so they can be run simultaneously
  7. Design data pipeline to save log or data information into database for future analysis
- 

# Main Accomplishments

1. Extract real-time price information and save them in file and plot it in live
  2. Extract different timeframes of price information for future analysis
  3. Utilize Transformer model to make prediction of next hour price
  4. Automatically submit orders based on the prediction of the model
  5. Utilize command line in main.py file to run different functions and different crypto symbols
- 