Imperial College London

Imperial College London

Department of Mathematics

Transformer-based Probabilistic Forecasting Model for Intraday Forex Rate Trading

Author: Changan QIAN (CID: 02202257)

A thesis submitted for the degree of

MSc in Mathematics and Finance, 2022-2023

Declaration

The work contained in this thesis is my own work unless otherwise stated.

Acknowledgements

Foremost, I would like to express my sincere gratitude to my external supervisor Albert Chan from MUFG, whose patience and supportive feedback encourages me to overcome difficulties with confidence. His very kind invitations of office visiting bring me closer to the industry, leaving some of the most memorable moments in the last a few months of my postgraduate study.

I would also like to convey my greatest appreciation to my internal supervisor Dr. Alex Tse from Imperial College. During the external thesis project, Dr. Alex is always generous to offer the professional assistance and valuable suggestions with great details. His consideration makes me feel warm and supported at all time.

In addition, I am very grateful to have Emanuele Baldazzi and Louise Caprani from the team in MUFG, who provide substantial advice and comments on my weekly progresses.

Last but not least, I want to express my deepest thanks to my parents for their encouragement and supports with no reservation. I also feel very lucky to have genuine friends like Yuchen and Bingran in this one-year study. Thank you to my partner Yixi, for all her love and company during the summer visiting.

Abstract

The main purpose of this paper is to propose a probabilistic forecasting model based on Transformer neural network for predicting the distribution of future return in financial market. We form a complete framework for intraday Foreign Exchange rate trading task, starting from data preparation to probabilistic forecasting, and finally the trading strategies. According to our empirical results, the proposed Transformer-based model achieves better performance than the LSTM-based probabilistic forecasting model regarding not only the training loss, but also several other meaningful metrics. At the end of the paper, we examine the model profitability by establishing a few strategies developed from the probabilistic predictions. Our out of sample backtesting on USDJPY and AUDJPY currency pairs reveals the great potential of the proposed Transformer-based model in generating profits.

Contents

List of Figures

List of Tables

Introduction

Foreign Exchange (FX) market is a huge and important market in financial industry, a place where people can buy and sell different currencies through currency pairs. This over-the-counter (OTC) market runs currency transactions for the whole day except weekends, with market makers constantly providing a flow of liquidity. According to Triennial Central Bank Survey 2022 from the Bank for International Settlements[1], the trading volume in OTC FX market has risen to $7.5 trillion per day in April 2022, which is 14% higher from $6.6 trillion in April 2019, keeping it the world's largest financial market. Salman Ahmed et al. pointed out that in Foreign Exchange market there are more complicated movements driven by both currency trades and regional policies, characterizing the nature of high volatility, nonlinearity, and irregularity[2]. These features make FX trading more attractive to many investors because of the potential arbitrage. As one currency pair compares the base currency to the quote currency, the holding of the quote currency generates profit when the price of the currency pair elevates, and correspondingly the loss could be caused by the decrease of price. Therefore, either for the aim of risk hedging or profit speculation, predicting the Foreign Exchange rate is always a crucial challenge.

To classify the techniques that practitioners often use in Foreign Exchange rate forecasting, Ayitey Junior et al.[3] concluded them in two streams, which are fundamental analysis and technical analysis. While fundamental analysis focuses on the economic, social, political issues and their influence on the Foreign Exchange rate, technical analysis pays more attention to the microstructure of the currency market. Technical analysis relies on historical data to predict the future movement of the currency pair, and various successful machine learning algorithms have been investigated in FX rate forecasting. Rojas et al.[4] compared the performance of Regularized Logistic Regression, Support Vector Machines (SVM), Gradient Boosting Classifier (GBC) and Neural Networks (NN) in directional FX forecasting. From their results, they suggested that SVM performed the best in the binary target while Ridge Regression in the continuous target with all algorithms outperforming a long-only benchmark.

Recent years have also witnessed an increasing number of Deep Learning models that used in predicting the rate movement according to Panda et al.[5]. Thanks to the universal approximation theorem[6], the technique of neural network promises a powerful approach to learn the complex representations over massive data. Proposed neural network architectures for Foreign Exchange rate forecasting includes Feedforward Neural Network (FNN)[7, 8], Convolutional Neural Network (CNN)[9], Recurrent Neural Network (RNN)[10, 11], Long Short-Term Memory network (LSTM)[2, 12, 13], etc. Apart from the above architectures, a most recent network called Transformer is also triggering great attention in sequence prediction community. Proposed by Vaswani et al.[14] in 2017, the emerging Deep Learning model shed its first light on Natural Language Processing (NLP), contributing significantly to recent advances in machine translation and content generation. The Transformer network is capable to capture long-term dependencies and correlations through the attention mechanism. However, it is often not easy for RNN and LSTM to handle the task. Furthermore, the extremely renowned Chat Generative Pre-Trained Transformer (ChatGPT)[15] is a typical application that reveals the power of Transformer. The incredible performance also arises the interest of applying Transformer network to time series tasks such as forecasting[16], anomaly detection[17], and classification[18].

For time series prediction, two common categories are point estimation and probabilistic forecasting [19]. Point estimation aims to predict the accurate value of the target, and the model is often trained by minimising the mean squared error between the output and the true value. Probabilistic forecasting gives the distribution estimation instead. For instance, estimating a normal density function means to predict two parameters, mu and sigma. For deep probabilistic forecasting, Salinas et al. proposed an auto-regressive model DeepAR based on LSTM neural network[20],

where parameters are learned by maximizing the log-likelihood. Due to the stochastic nature and complicated correlation among different assets inside the FX market, point estimation for price movement could be even harder. Therefore, we use probabilistic forecasting for Foreign Exchange rate prediction in this paper.

This paper proposes a Transformer-based probabilistic forecasting framework for a Foreign Exchange rate trading task. To our best knowledge, Transformer-based probabilistic forecasting for Foreign Exchange rate movements has not been investigated in other related research, which is one of the contributions of this paper. We would like to address that the proposed model is flexible to extend on various asset classes and in any frequency, and it is possible for user to conduct risk analysis on the probabilistic prediction. With the multi-head attention mechanism, the Transformer model learns to predict the forward return of the currency pair through a long sequence of historical data input. In addition, as a highlighted part, we incorporate time embedding to our feature set. Although FX market opens 24 hours a day, the trading volume might not follow the uniform distribution among different trading sessions. Since Foreign Exchange market is an OTC market, where the trading volume is not applicable, we could use the time feature as a reference of market liquidity. We evaluate our model in several performance metrics and compare the results with the DeepAR model[20] on the same data setting. We also investigate how the forecasting model performs during different trading hours, hence the intraday contribution to the daily PnL. Finally, we backtest a few strategies to examine the profitability of the proposed probabilistic forecasting model.

Chapter 1 introduces the theoretical background of this paper, including a brief review on the statistical estimation, fundamental knowledge of neural networks for time series prediction and a universal approximation theorem for distribution expression with neural network.

Chapter 2 proposes a Transformer-based probabilistic forecasting framework for FX rate trading. Starting with the section of problem description, we explain the settings of the prediction target, as well as the model's input and output. The second section illustrates the model architecture with details. In the last section of the chapter, we conduct experiments on USDJPY and AUDJPY currency pairs in several forecasting periods as the examples, and compare the predicted results with the LSTM-based model. We also discuss about features construction, training parameters, evaluation metrics and hourly performance in this section.

In chapter 3, we backtest a few trading strategies based on the model prediction, and analyse the out of sample daily PnL performance in terms of Annualised Return, Sharpe Ratio, Maximum Drawdown, etc. Additionally, an analysis of the intraday PnL contribution for each strategy will be included as well.

Chapter 1

Theoretical methodology

In this chapter, we introduce the theoretical background of probabilistic forecasting and neural network. To set the stage for probabilistic forecasting with empirical data, we first make a review on the statistical estimation. Particularly, the maximum likelihood estimation (MLE) method is emphasized as it will become the essential part of the loss function in our forecasting model. Then, the fundamental knowledge of neural network follows. Three classic neural networks for time series prediction, Recurrent Neural Network (RNN), Long Short-Term Memory network (LSTM) and Transformer network are demonstrated with great details. Finally, the universal approximation theorem gives the strong theoretical support to the methodology of this paper.

1.1 Statistical inference

Statistical inference is a method to analyse data with some existing distributions. The advantage of using a well studied distribution is that we can utilize the properties to infer plenty of valuable results. We could describe the observed data through a model with information about the average value, variance, quantiles, and many other statistics.

The reason why we need statistical inference is that even though we don't know the exact distribution of the data, we still want to use some statistical tools to characterise the patterns. Typically, the statistical tools are chosen from some probability models, and our goal is to estimate the parameters of the chosen model.

1.1.1 Parametric estimation

Given a sequence of sample data $\mathcal{X}_n = (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n)$, the basic hypothesis is that the random variables $\boldsymbol{X}_i$, for $i = 1, \ldots, n$ are independent and identically distributed (iid). Regarding the notation, for some random vector $\boldsymbol{X} \in \mathbb{R}^p$, the distribution is parameterized by β, we write f_β as the density function, F_β as the distribution and $\mathbb{E}_\beta$ as the expectation of $\boldsymbol{X}$. Specifically, we assume that F belongs to a set of parametric distributions $\mathcal{F} = (\mathcal{F}_\beta)_{\beta \in B}$, where $\beta \in B$ determines a unique distribution F_β, or in a mathematical way, $\beta \mapsto \mathbb{P}_\beta$ is injective.

To model the data, we should choose a type of distribution first. The choice may depend on some initial analysis on the data patterns. For example, we could use a normal distribution to model the data with symmetric pattern, or use a t-distribution to model the data with heavy tails. With the empirical data in hand, there are several methods to fit the parameters β. However, we mainly focus on maximum likelihood estimation (MLE) in this paper because it would be the essential part in the loss function of our probabilistic forecasting model.

1.1.2 Maximum likelihood estimation

Maximum likelihood estimation (MLE) is one of the most important and popular methods for parametric estimation. Under an assumed statistical model, the parameters are estimated by making the observed data most probable by maximizing the likelihood function.

Definition 1.1.1 (Likelihood function)**.** The likelihood function is the map $\mathcal{L}_n$ from B to $\mathbb{R}$ defined

as

$$\mathcal{L}_n(\beta) := \prod_{i=1}^{n} f_\beta\left(\boldsymbol{X}_i\right), \quad \text{for all} \quad \beta \in B,$$

and we define the log-likelihood $l_n := \log \mathcal{L}_n$.

According to the independent and identically distributed assumption, the likelihood function is nothing but the joint density function of $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n)$. Then the maximum likelihood estimator is defined as

$$\widehat{\beta}_n^{\mathrm{ML}} := \arg\max_{\beta \in B} \mathcal{L}_n(\beta).$$

The necessary condition for optimal parameters $\widehat{\beta}_n^{\mathrm{ML}}$ is the likelihood equation $\nabla l_n(\beta) = 0$.

1.1.3 Auto-regressive estimation with likelihood model

In mathematical finance, the classic Black-Scholes model assumes that in probability space $(\Omega, \mathcal{F}, \mathbb{P})$, the return X_t is driven by the stochastic process

$$X_t := \frac{\mathrm{d}S_t}{S_t} = \mu \mathrm{d}t + \sigma \mathrm{d}W_t, \tag{1.1.1}$$

where W_t is the Brownian motion in the probability space. Equation 1.1.1 assumes that the returns are independently normal distributed. However, we should note that they are not necessarily to be independent and identically distributed (iid), as some more complicated stochastic models rely on the dynamics of parameters. We will not discuss too much in this paper.

As we mentioned at the beginning of this section, the distribution of return is changing from time to time. Therefore, instead of fitting one general statistical model, the auto-regressive estimation aims to model the conditional distribution on each time step. One way is to propose a probabilistic forecasting model to output the conditional distribution's parameters from time to time. We simply introduce a generalised version here, as our proposed model will be discussed with details in the next chapter.

Denote the return series by $\{\boldsymbol{x}\}_{i,t}$, feature vector series by $\{\boldsymbol{z}\}_{i,t}$, with sample label $i = 1, \ldots, N$, and time $t = 1, \ldots, T$. We use data from time 1 to $t_0 - 1$ to predict the distribution parameters from time t_0 to T. Then, the estimators of the conditional density function is given by

$$\beta_{i,t_0:T} := \mathcal{M}_\theta\left(\boldsymbol{x}_{i,1:t_0-1}, \boldsymbol{z}_{i,1:t_0-1}\right), \tag{1.1.2}$$

where $\mathcal{M}_\theta$ is the probabilistic forecasting model parameterised by θ, showing that the distribution only depends on the historical data, or say the filtration $\mathcal{F}_{t_0}$. Once we have obtained $\beta_{i,t}$, the predicted estimator at time t, we note the conditional probability of observing $\boldsymbol{x_{i,t}}$ as $f_{\beta_{i,t}}\left(\boldsymbol{x_{i,t}}\right)$.

Similarly, the likelihood function could be written in the form of the product of conditional density functions,

$$\mathcal{L}_T(\theta) = \prod_{i=1}^{N} \prod_{t=t_0}^{T} f_{\beta_{i,t}}\left(\boldsymbol{x}_{i,t}\right) = \prod_{i=1}^{N} \prod_{t=t_0}^{T} f_{\mathcal{M}_\theta\left(\boldsymbol{x_{i,1:t_0-1}}, \boldsymbol{z_{i,1:t_0-1}}\right)}\left(\boldsymbol{x}_{i,t}\right), \tag{1.1.3}$$

and correspondingly the log-likelihood function becomes

$$l_T(\theta) = \sum_{i=1}^{N} \sum_{t=t_0}^{T} \log\left(f_{\beta_{i,t}}\left(\boldsymbol{x}_{i,t}\right)\right) = \sum_{i=1}^{N} \sum_{t=t_0}^{T} \log\left(f_{\mathcal{M}_\theta\left(\boldsymbol{x}_{i,1:t_0-1}, \boldsymbol{z_{i,1:t_0-1}}\right)}\left(\boldsymbol{x}_{i,t}\right)\right). \tag{1.1.4}$$

If we only consider one step prediction, hence $t_0 = T$. Then for simplicity of the notation, we omit t in the expression,

$$\mathcal{L}_T(\theta) = \prod_{i=1}^{N} f_{\beta_i}\left(\boldsymbol{x}_i\right) = \prod_{i=1}^{N} f_{\mathcal{M}_\theta\left(\boldsymbol{x_{i,1:t_0-1}}, \boldsymbol{z_{i,1:t_0-1}}\right)}\left(\boldsymbol{x}_i\right), \tag{1.1.5}$$

$$l_T(\theta) = \sum_{i=1}^{N} \log\left(f_{\beta_i}\left(\boldsymbol{x_i}\right)\right) = \sum_{i=1}^{N} \log\left(f_{\mathcal{M}_\theta\left(\boldsymbol{x_{i,1:t_0-1}}, \boldsymbol{z_{i,1:t_0-1}}\right)}\left(\boldsymbol{x_i}\right)\right). \tag{1.1.6}$$

Note that now we are optimising the probabilistic forecasting model's parameters θ instead of the conditional distribution's parameters, and in practice, people often use log-likelihood function to avoid extensive product calculations,

$$\widehat{\theta}_T := \arg\max_{\theta \in \Theta} l_T(\theta). \tag{1.1.7}$$

We could achieve the optimal (or sub-optimal) $\widehat{\theta}_T$ through gradient descent in deep models.

1.2 Neural networks for sequential prediction

1.2.1 Fundamentals of neural network

Deep learning is standing as a unique and powerful technique in the machine learning algorithms, making significant contribution to various areas, such as predictive forecasting, image recognition, natural language processing, content generation, etc. In this subsection, we will cover the fundamental concepts of feedforward neural network (FNN), activation functions, loss functions and backpropagation.

Feedforward neural network

The following definitions come from the deep learning lecture note by Lukas Gonon[21].

Definition 1.2.1 (Feedforward neural network)**.** Let $I, O, r \in \mathbb{N}$. A function $\boldsymbol{f} : \mathbb{R}^I \to \mathbb{R}^O$ is a feedforward neural network (FNN) with $r-1 \in \{0, 1, \ldots\}$ hidden layers, where there are $d_i \in \mathbb{N}$ units in the i-th hidden layer for any $i = 1, \ldots, r-1$, and activation functions $\boldsymbol{\sigma}_i : \mathbb{R}^{d_i} \to \mathbb{R}^{d_i}, i = 1, \ldots, r$, where $d_r := O$, if

$$\boldsymbol{f} = \boldsymbol{\sigma}_r \circ \boldsymbol{L}_r \circ \cdots \circ \boldsymbol{\sigma}_1 \circ \boldsymbol{L}_1,$$

where $\boldsymbol{L}_i : \mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i}$, for any $i = 1, \ldots, r$, is an affine function $\boldsymbol{L}_i(\boldsymbol{x}) := W^i \boldsymbol{x} + \boldsymbol{b}^i, \boldsymbol{x} \in \mathbb{R}^{d_{i-1}}$. We shall denote the class of such functions f by $\mathcal{N}_r(I, d_1, \ldots, d_{r-1}, O)$.

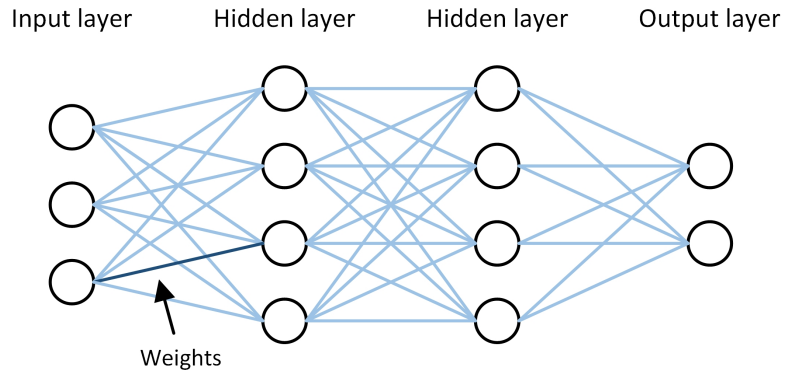


Figure 1.1: Feedforward neural network $\mathcal{N}_r(3, 4, 4, 2)$

Figure 1.1 describes the architecture of a feedforward neural network with two hidden layers, where activation functions have not been shown. The edges between two layers in the plot refer to a weights matrix, which contains a set of learnable parameters. During the training process, the parameters are changing towards a direction that minimises the loss. Now we explain the activation functions and loss functions in neural network.

Activation functions

Activation functions often play a crucial role by introducing the non-linearity to the network, enabling the model to capture complex patterns. They are designed with various motivations. Here we list the definition of activation functions that used in this paper:

- Rectified linear unit (ReLU)

$$g(x) = \max\{x, 0\}, \quad g'(x) = \begin{cases} 0, & x < 0 \\ 1, & x > 0 \end{cases}, \quad range[0, \infty).$$

- Sigmoid (Logistic, σ)

$$g(x) = \frac{1}{1 + e^{-x}}, \quad g'(x) = g(x)(1 - g(x)), \quad range(0, 1).$$

- Hyperbolic tangent (tanh)

$$g(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \quad g'(x) = 1 - g(x)^2, \quad range(-1, 1).$$

- Softmax

$$g_i(x) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}, \quad \frac{\partial g_i}{\partial x_j}(x) = \begin{cases} g_i(x)\left(1 - g_j(x)\right), & i = j \\ -g_i(x) g_j(x), & i \neq j \end{cases}, \quad range(0, 1].$$

- Softplus

$$g(x) = \log\left(1 + e^x\right), \quad g'(x) = \frac{1}{1 + e^{-x}}, \quad range(0, \infty).$$

Loss functions

In deep learning, the loss function is also known as the objective function, which is a crucial component used to quantify how good the model output is compared to the actual values (ground truth). Considering the neural network as a model with a large number of parameters that turns inputs into outputs, the goal of training the model is to optimise the parameters in the direction that minimises the loss function. The choice of the loss function depends on the type of problem being addressed by the neural network. The most frequently used loss functions are Mean Squared Error (MSE) for regression tasks and Cross Entropy loss (CE) for classification class. However, for our probabilistic forecasting model, we will use the negative log-likelihood function instead.

Training

The training process of a neural network is to minimise the empirical loss over the model parameters. The parameters are changing at a specified pace, hence the learning rate. The updating of parameters rely on "backpropagation", a scheme to calculate the gradient information and use it to adjust the parameters in the direction that minimising the empirical loss.

Backpropagation

The main idea behind backpropagation is to compute the gradients of the loss function with respect to the model's parameters during the forward pass, hence from inputs to outputs. Then, we can update these parameters in the opposite direction of the gradient during the backward pass, hence from outputs to inputs. The gradient represents the direction and magnitude of the change required to minimize the loss, and could be obtained by algorithmic differentiation, a technique used to efficiently compute the derivatives of functions with respect to their inputs.

Now we explain explicitly about the gradient computation in backpropagation. For neural network $\boldsymbol{f}_{\boldsymbol{\theta}} \in \mathcal{N}_r\left(I, d_1, \ldots, d_{r-1}, O\right)$, the neural network parameters are defined as $\boldsymbol{\theta} := \left(W^1, \ldots, W^r; b^1, \ldots, b^r\right)$, with activation functions $\boldsymbol{g}_i$ for $i = 1, \ldots, r$ between the layers. Here we use $\boldsymbol{a}^{i-1}$ and $\boldsymbol{z}^i$ to denote the input and output data flow of the ith linear layer $\boldsymbol{L}^i$, hence

$$\begin{aligned} \boldsymbol{z}^i = \left(z_1^i, \ldots, z_{d_i}^i\right) &:= \boldsymbol{L}_i\left(\boldsymbol{a}^{i-1}\right) = W^i \boldsymbol{a}^{i-1} + b^i, \\ \boldsymbol{a}^i = \left(a_1^i, \ldots, a_{d_i}^i\right) &:= \boldsymbol{g}_i\left(\boldsymbol{z}^i\right), \quad \boldsymbol{a}^0 := \boldsymbol{x}. \end{aligned}$$

Additionally, the so-called adjoint $\boldsymbol{\delta}^i = \left(\delta_1^i, \ldots, \delta_{d_i}^i\right) \in \mathbb{R}^{d_i}$ is defined as

$$\delta_j^i := \frac{\partial \ell}{\partial z_j^i}, \quad j = 1, \ldots, d_i, \quad \text{for} \quad i = 1, \ldots, r.$$

Based on the chain rule, the gradient computation in backpropagation can be expressed as

$$\boldsymbol{\delta}^r = \boldsymbol{g}_r' \left(\boldsymbol{z}^r\right) \odot \nabla_{\hat{\boldsymbol{y}}} \ell \left(\boldsymbol{a}^r, \boldsymbol{y}\right), \tag{1.2.1}$$

$$\boldsymbol{\delta}^i = \boldsymbol{g}_i' \left(\boldsymbol{z}^i\right) \odot \left(W^{i+1}\right)' \boldsymbol{\delta}^{i+1}, \quad i = 1, \ldots, r-1, \tag{1.2.2}$$

$$\frac{\partial \ell}{\partial b_j^i} = \delta_j^i, \quad i = 1, \ldots, r, \quad j = 1, \ldots, d_i, \tag{1.2.3}$$

$$\frac{\partial \ell}{\partial W_{j,k}^i} = \delta_j^i a_k^{i-1}, \quad i = 1, \ldots, r, \quad j = 1, \ldots, d_i, \quad k = 1, \ldots, d_{i-1}, \tag{1.2.4}$$

where $\odot$ refers to the component-wise Hadamard product, and we can utilise the above gradient information to update the parameters in our model.

1.2.2 Recurrent Neural Network and LSTM

Recurrent Neural Network (RNN) is a class of neural network that has been widely used in sequential modelling tasks[22], from machine translation to weather forecasting. Compared to feedforward neural network, data will not be inputted at once in RNN model, but will be processed recursively. The architecture of Recurrent Neural Network helps to handle the sequential pattern of the input series, allowing the past information persists over multiple time steps. This feature could be crucial in modelling because temporal transaction data in financial industry often presents autocorrelation. In this subsection we first explain the generalised architecture of RNN, then the introduction of Long Short-Term Memory network follows.

Recurrent Neural Network

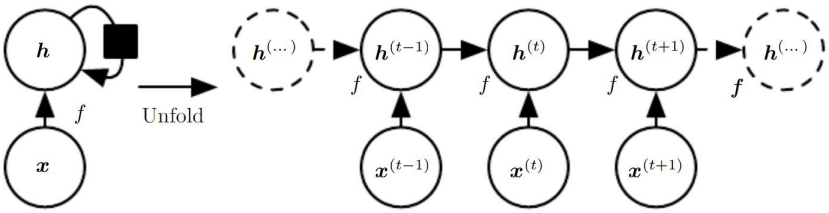


Figure 1.2: Unfolded structure of Recurrent Neural Network (RNN)[1]

The core characteristic of RNN is the ability to maintain an hidden state which acts as a summary of information from previous time steps. As shown in the unfolded structure of RNN in Figure 1.2, the internal state is updated at each loop, combining the current input and the past information. Mathematically, this can be expressed in the recursive form of

$$h^{(t)} = \nu\left(h^{(t-1)}, \boldsymbol{x}^{(t)}; \theta\right), \quad \text{for} \quad t = 1, \ldots, T, \tag{1.2.5}$$

where h represents the internal state, with $\boldsymbol{x}$ being the input and θ being the model parameters. In the book *Deep Learning* written by Goodfellow et al.[22], they present a forward pass example, in which there are two linear layers $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$, with a Hyperbolic tangent (tanh) activation function in between and a Softmax function before the output $\hat{y}^{(t)}$,

$$h^{(t)} = \tanh\left(\boldsymbol{L}_1\left(h^{(t-1)}, \boldsymbol{x}^{(t)}; \theta\right)\right), \tag{1.2.6}$$

$$\hat{y}^{(t)} = \text{softmax}\left(\boldsymbol{L}_2\left(h^{(t)}; \theta\right)\right). \tag{1.2.7}$$

Finally, we insert the model output $\hat{y}^{(t)}$ and ground truth label $y^{(t)}$ to the loss function. To get the total loss, repeat the procedure for all $\boldsymbol{x}^{(t)}$ and sum up the losses in each time step. For example,

[1]Page 370, https://www.deeplearningbook.org/contents/rnn.html

if we choose to the negative log-likelihood function as the loss function, recall equation 1.1.4 in the previous section, the total loss becomes

$$L^{\text{total}} = \sum_{t=1}^{T} L^{(t)} = -\sum_{t=1}^{T} \log p_{\mathcal{M}} \left(y^{(t)} \mid \left\{ \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)} \right\} \right), \tag{1.2.8}$$

where $p_{\mathcal{M}}$ is the conditional probability under model $\mathcal{M}$.

Long Short-Term Memory (LSTM)

The LSTM network[23] is also a Recurrent Neural Network but with a explicitly designed architecture of the repeating module. The purpose of proposing the LSTM network, as the name suggests, is to enhance both long-term and short-term memory after a long period of inputs. After updating the internal state for a number of time steps, traditional RNN often suffers from exploding and vanishing gradient[22], a phenomenon in backpropagation that the gradient is too large or too small to effectively update the model parameters. The problem was explained in depth with analysis of the backpropagated error signal in Hochreiter's paper[24]. Figure 1.3 describes how information runs through the LSTM module.

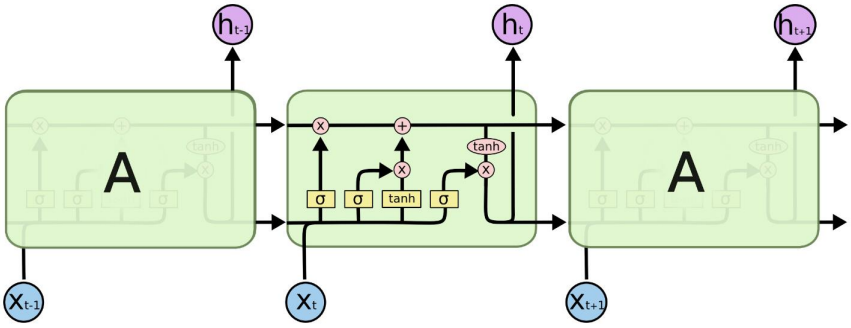


Figure 1.3: Unfolded structure of Long Short-Term Memory (LSTM)[2]

In LSTM module, the crucial components are the cell state and the gates, which determine the information flow through time. One gate often consists of a sigmoid layer and a pointwise multiplication operation. Recall that the output range of sigmoid function is $(0, 1)$, therefore, it plays as a "gate" to control the proportion that should be let through. In this way, the LSTM model learns to give way to those most useful messages when handling the long-term dependencies. Normally there are three gates in the module, from left to right, which are capable to add information to or remove information from the cell state:

- Forget gate. At time t, the LSTM module receives the cell state c_{t-1}, hidden state h_{t-1} and input $\boldsymbol{x}_t$. The forget gate gives the proportion f_t of the information that c_{t-1} should remain, using the transformed h_{t-1} and $\boldsymbol{x}_t$ as input. We use W_h^f, W_x^f and b^f to represent the linear transformation before the sigmoid function in forget gate,

$$f_t = \sigma \left(W_h^f \cdot h_{t-1} + W_x^f \cdot \boldsymbol{x}_t + b^f \right). \tag{1.2.9}$$

- Input gate. Compared to the forget gate, the input gate gives the proportion i_t of how much information in input $\boldsymbol{x}_t$ is added to the cell state. We use W_h^i, W_x^i and b^i to represent the linear transformation before the sigmoid function in input gate,

$$i_t = \sigma \left(W_h^i \cdot h_{t-1} + W_x^i \cdot \boldsymbol{x}_t + b^i \right). \tag{1.2.10}$$

- Output gate. Once we have obtained f_t and i_t, we can update the cell state to c_t. The aim of output gate is to filter out what we prefer to output rather than sending out the full information from cell state c_t,

$$o_t = \sigma \left(W_h^o \cdot h_{t-1} + W_x^o \cdot \boldsymbol{x}_t + b^o \right). \tag{1.2.11}$$

[2] https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Finally, we formularise the update of cell state in Figure 6 by

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tanh\left(W_h^c \cdot h_{t-1} + W_x^c \cdot \boldsymbol{x}_t + b^c\right), \tag{1.2.12}$$

where W_h^c, W_x^c and b^c represent the linear transformation of input $\boldsymbol{x}_t$ before being added to the state cell.

1.2.3 Multi-head attention Transformer

Transformer architecture, or the self-attention mechanism, was first introduced by Vaswani et al. in their paper "Attention is All You Need"[14]. Although the carefully designed LSTM succeeded handling the long-term dependencies, its recurrent nature prevent the LSTM network being trained effectively because the update of current state depends on the previous state and cannot be parallelised. The contribution of attention mechanism improves the computational efficiency and the ability to receive long input sequence. Attention mechanism revolutionizes Natural Language Processing, series prediction and various areas by avoiding the need for recurrent connections to process sequential data. Multi-head attention is a variation of original self-attention mechanism, and in this subsection, we start with self-mechanism first.

Phillip Lippe describes the attention mechanism in his tutorial notebook[25], "the attention mechanism describes a weighted average of (sequence) elements with the weights dynamically computed based on an input query and elements' keys". That is to say, the model with attention learns to focus more on some important elements in the input sequence. Normally there are four essential parts in the attention block:

- Score function: f_{attn}

  The score function defines a method to calculate the similarity score between the input elements. Then after computing the scores, a softmax layer process the results into the aggregation weights. One frequently used similarity metric is the dot product, which we choose in this paper.

- Queries: $Q \in \mathbb{R}^{T \times d_k}$

  A query is acting like an request to ask for the similarity scores of current element. Parameters T and d_k indicate that the input contains a sequence of d_k-dim elements at length T. The length could vary since the architecture runs in parallel.

- Keys: $K \in \mathbb{R}^{T \times d_k}$

  The keys are what the queries compare with, so they must be in the same shape. A key is the projection of the corresponding value. For example, in machine translation, the key can be the word embedding.

- Values: $V \in \mathbb{R}^{T \times d_v}$

  The values are the feature vectors we need to aggregate by the output weights.

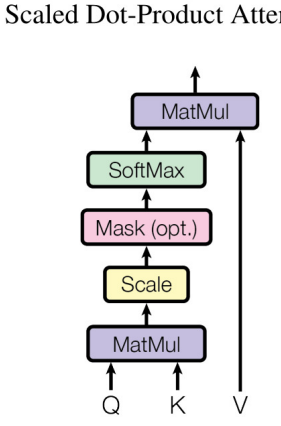


Figure 1.4: Self attention mechanism with scaled dot-product similarity[3]

Figure 1.4 illustrates the components of an self-attention block using the scaled dot product as the score function. The queries Q and keys K are first sent to the score function to calculate the dot product. Then, for sequential prediction tasks such as stock price forecasting, it is necessary to apply a mask on the scores in order to prevent the leakage of future information. Finally, the values are aggregated according to the weights calculated on the left. The computations in self-attention block can be summarised as

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{1.2.13}$$

However, aggregating by a single vector of attention weights only provides one aspect of the sequence, which might not be sufficient to express the information inside. One way is to extend the attention mechanism to multiple heads.

The multi-head attention first transforms the inputs, hence queries Q, keys K and values V, into separate parts independently. The transformations are usually parameterised by some linear layers. That is to say, the multi-head attention block could optimally preprocess the inputs through a set of learnable parameters to help extract the information from several aspects. Suppose the number of the separate parts is H, and we will finally get H aggregated outputs called heads.

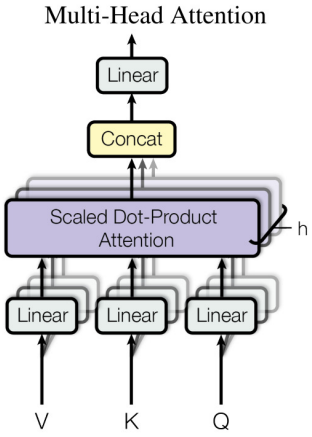


Figure 1.5: Multi-head attention mechanism with scaled dot-product similarity[4]

Figure 1.5 demonstrates the multi-head attention mechanism, which is similar to the self-attention, but with h independent copies in parallel. Starting with the initial transformations,

$$\begin{aligned} Q^i &= W_Q^i \cdot Q + b_Q^i, \\ K^i &= W_K^i \cdot K + b_K^i, \\ V^i &= W_V^i \cdot V + b_V^i, \quad \text{for} \quad i = 1, \ldots, H, \end{aligned} \tag{1.2.14}$$

the transformed parts are then sent to the score function, the same module in the self-attention block, to generate heads,

$$head^i = \text{Attention}\left(Q^i, W^i, V^i\right), \quad \text{for} \quad i = 1, \ldots, H. \tag{1.2.15}$$

In the end, concatenate all heads together as the multi-head attention, which will be the input of the output network,

$$\text{MultiheadAttention}(Q, K, V) = \text{concat}\left(head^1, \ldots, head^H\right). \tag{1.2.16}$$

In the paper "Attention is All You Need", Vaswani et al. give the full architecture of Transformer network, which is shown in Figure 9, containing an encoder and a decoder. The core structure of encoder and decoder is nothing but the multi-head attention block we have introduced above. The encoder-decoder framework is commonly used in language translation model because it is required to translate from one language to another through word embedding. However, in some series prediction tasks there is no embedding needed, a well designed decoder could be sufficient enough.

[3]The figure originates from paper "Attention is all you need".
[4]The figure originates from paper "Attention is all you need".

1.3 Universal approximation theorem for distribution

The magic power of neural network has been shown in many successful works, in an empirical way. However, people still keep curious about the theoretical commitment of the neural network approximation. Fortunately, existing papers have proven some neural network universal approximation theorems for functions under specific assumptions[6, 26, 27]. In this section, we focus on the "universal approximation theorem for expressing probability distributions" proposed by Yulong et al.[28]. They proved that, given a target distribution π and a source distribution p_z both defined on $\mathbb{R}^d$, under some assumptions there exists a deep neural network $g : \mathbb{R}^d \to \mathbb{R}$ with ReLU activation such that the push-forward $g_{\#}p_z$ is arbitrarily close to the target measure π. The detailed proof is presented on their paper[28], and we summarise the main theorem results to show the theoretical background of distribution forecasting with neural network.

Given a discrepancy measure $D(p, \pi)$ which evaluates the difference between two probabilistic measures p and π, the approximation task can be formulated as

$$\inf_{g \in \mathcal{G}_{\text{NN}}} D\left(g_{\#}p_z, \pi\right), \tag{1.3.1}$$

where $D(p, \pi)$ is typically defined in the form of integral probability metric (IPM) with $\mathcal{F}_D$ being a certain class of witness functions,

$$D(p, \pi) = d_{\mathcal{F}_D}(p, \pi) := \sup_{f \in \mathcal{F}_D} \left|\mathbf{E}_{X\sim p} f(X) - \mathbf{E}_{X\sim \pi} f(X)\right|. \tag{1.3.2}$$

In the paper, Yulong et al. discuss the universal approximation theorem on three discrepancy measures:

- Wasserstein Distance
$$\mathcal{W}_1(p, \pi) = \inf_{\gamma \in \Gamma(p,\pi)} \int |x - y| \gamma(dxdy),$$
where the witness class is chosen as the class of 1-Lipschitz functions $\mathcal{F}_D := \left\{f : \mathbb{R}^d \to \mathbb{R} : \text{Lip}(f) \leq 1\right\}$.

- Maximum Mean Discrepancy (MMD)
$$\text{MMD}(p, \pi) = \sup_{\|f\|_{\mathcal{H}_k} \leq 1} \left|\mathbf{E}_{X\sim p} f(X) - \mathbf{E}_{X\sim \pi} f(X)\right|,$$
where the witness class is chosen as the unit ball of a reproducing kernel Hilbert space (RKHS) $\mathcal{F}_D := \{f \in \mathcal{H}_k : \|f\|_{\mathcal{H}_k} \leq 1\}$.

- Kernelized Stein Discrepancy (KSD)
$$\text{KSD}(p, \pi) = \sup_{\|f\|_{\mathcal{H}_k} \leq 1} \mathbf{E}_{X\sim p} \left[\mathcal{T}_\pi f(X)\right],$$
where the witness class is chosen as $\mathcal{F}_D := \{\mathcal{T}_\pi f : f \in \mathcal{H}_k \text{ and } \|f\|_{\mathcal{H}_k} \leq 1\}$, for $\mathcal{T}_\pi$ being the Stein-operator defined by $\mathcal{T}_\pi f := \nabla \log \pi \cdot f + \nabla \cdot f$.

Theorem 1.3.1 (Universal approximation theorem for distribution)**.** *Denote π and p_z as the target and source distributions respectively, and assume that p_z is absolutely continuous with respect to the Lebesgue measure. Then under the specific assumptions on π and kernel function k for each discrepancy measure, for any given approximation error ϵ, there exists a positive integer n, and a fully connected and feed-forward deep neural network $u = \text{FNN}\left(\left\{W^\ell, b^\ell\right\}_{\ell=1}^{L+1}\right)$ of depth $L = \lceil \log_2 n \rceil$ and width $N = 2^L = 2^{\lceil \log_2 n \rceil}$, with d inputs and a single output and with ReLU activation such that $d_{\mathcal{F}_D}\left(u_{\#}p_z, \pi\right) \leq \epsilon$.*

The upper bound of n is given explicitly on each discrepancy measure according to the specific assumptions on target distribution π and kernel function k. These theoretical results are shown in the appendix A.1 of this paper.

Although the assumptions are sometimes too strict to achieve in practice, the theorem at least informs us the potential of learning distribution expression using neural network. In other words, if the historical features could sufficiently determine the distribution of the forward return, then it is possible for a finite neural network to forecast under some assumptions.

Chapter 2

Probabilistic forecasting for intraday FX rate movements

This chapter focuses on predicting the distribution of future movements in Foreign Exchange rate. In the world's largest financial market, the price of a currency pair could change dramatically every minute. Therefore, it is worth modelling the intraday movements in a more scientific way so that investors are able to monitor the risk and the potential arbitrage closely. We formalise the prediction problem in the first section, including an analytical investigation on some empirical data. Then the second section introduces our approaches toward the problem. We start with the LSTM-based model, which is a variation from the DeepAR model proposed by Salinas et al.[20], and continue to our Transformer-based model, which is one of the contributions of this paper. In the last section, we set up experiments to evaluate the predictive power of the models. Several accuracy metrics are also defined to compare the performances between the LSTM-based model and the Transformer-based model.

2.1 Problem description

Foreign Exchange market runs 24 hours a day except weekends, with four major trading sessions, Sydney (9p.m. to 6a.m.), Tokyo (0a.m. to 9a.m.), London (7a.m. to 4p.m.) and New York (1p.m. to 10 p.m.), which are shown in Figure 2.1 with UTC time format. However, it is not realistic for a human trader to monitor the market movements 24 hours a day, and Przemysław et. al[29] also note that the market liquidity varies from time to time. They conclude this by an analysing the spread of currency pairs in different hourly intervals, which means the trading performance could be affected by the time factor. Therefore, people are trying to develop predictive models for optimal trading. The purpose of predicting the intraday movements of a target currency pair is to first analyse the quantitative patterns in the trading flow, and then derive an automated trading strategy from the predictions. To formalise the prediction problem, we begin with some brief investigation on the data.

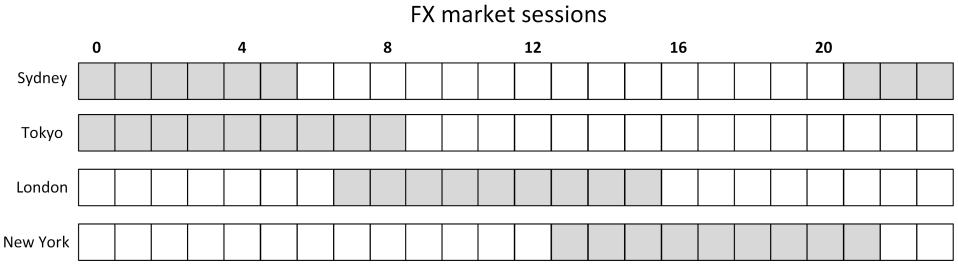
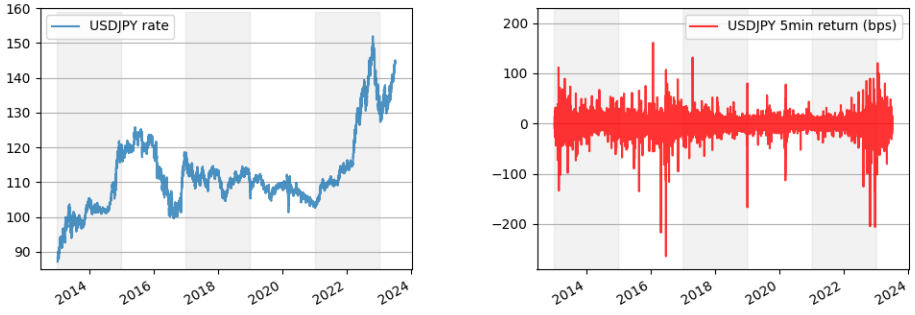


Figure 2.1: Four major FX trading sessions (UTC)

We use $\{\boldsymbol{s}\}_{t=0...N}$ to represent the Foreign Exchange rate series and $\{\boldsymbol{x}\}_{t=1...N}$ to be the return series, defined by $x_{t+1} := \frac{s_{t+1}-s_t}{s_t}$. Figure 2.2(a) shows the historical 5min rate chart of currency pair USDJPY, which we use as an example in this paper, in recent 10 years. The value of the



(a) 5min Foreign Exchange rate

(b) 5min return in basis points (bps)

Figure 2.2: Foreign Exchange rate and return of USDJPY from 2013 to 2023

rate means how much one should pay in Japanese Yen to trade for one United States Dollar. The plot indicates that Foreign Exchange rate could change dramatically over different periods of time, like the soaring trend in 2015 and 2022 or the relative less volatile period from 2017 to 2021. The movements are driven by either the effects of currency policies, global economy, market microstructure, etc. From Figure 2.2(b), the 5min return chart (in basis points), we also observe that the large movements are clustering together in those volatile periods. This phenomenon is called "volatility clustering", which was first observed by Mandelbrot (1963), that "large changes tend to be followed by large changes, of either sign, and small changes tend to be followed by small changes."[30] Specifically, if we split the window by every 2 years, we can calculate statistical results accordingly in each period, such as the expected value, variance, skewness, kurtosis, positive ratio and negative ratio. Given the definition as below:

- Expected value (mean) $\mu := \mathbb{E}[X]$.
- Variance: $\sigma^2 := \mathbb{E}\left[(X-\mu)^2\right]$.
- Skewness $\tilde{\mu}_3 := \mathbb{E}\left[\left(\frac{X-\mu}{\sigma}\right)^3\right]$.
- Kurtosis $\kappa := \mathbb{E}\left[\left(\frac{X-\mu}{\sigma}\right)^4\right] - 3$.
- Positive ratio $p_+ := \mathbb{E}\left[\mathbb{1}_{\{X\geq 0\}}\right]$.
- Negative ratio $p_- := 1 - p_+$.

These numerical results in each period are shown in Table 2.1, describing the statistical characteristics of return distributions. From the table we can see the statistical characteristic of returns varies a lot in different periods, indicating that the distribution of return is also changing from time to time.

	Mean	**Variance**	**Skewness**	**Kurtosis**	**Positive**	**Negative**
2013 - 2015	0.022728	14.489453	-0.009868	1.105282	0.516817	0.483183
2015 - 2017	-0.00079	16.164099	0.002713	-0.199531	0.51121	0.48879
2017 - 2019	-0.003848	8.56508	-0.001225	-0.178429	0.514193	0.485807
2019 - 2021	-0.003781	7.412371	0.012091	-0.839723	0.516593	0.483407
2021 - 2023	0.016616	10.735019	-0.075978	4.797739	0.524627	0.475373
2023 Jan - Jun	0.033741	21.775505	0.014367	1.784595	0.515993	0.484007

Table 2.1: Periodic statistical results of USDJPY 5min return (in bps) from 2013 to 2023

The magnitude of mean value reflects the overall trend in that period, whereas the variance shows how volatile the movements are. Skewness and kurtosis could be used to examine the

symmetry and tail property. In most periods, the skewness values are close to zero, while the kurtosis values have seen some significantly high periods in 2013-2015, 2021-2023 and the first half of 2023. These results indicate that the 5min returns are likely to have a symmetric and heavy-tailed pattern in a great amount of time. Therefore, it could be beneficial to model not only the absolute value of return, but also the statistical characteristics through distribution forecasting. In this paper, we hope to build a forecasting model with statistical inference to predict the return distribution dynamically.

Without loss of generality, we continue to express the forward return in $x_{t+1} := \frac{s_{t+1}-s_t}{s_t}$, which means the change of price from t to $t+1$. Recall that in statistical inference, we need to assume a statistical model initially, and suppose it is parameterised by β_{t+1} at time t. In this way, we can define $f_{\beta_{t+1}}(\cdot)$ as the density function of forward return x_{t+1}.

Assumption 2.1.1. The distribution parameters $\boldsymbol{\beta}_{t+1}$ of the forward return x_{t+1} is $\mathcal{F}_t$ measurable.

The basic assumption of the forecasting model is that the distribution of the asset's future movements is only dependant on the historical data. The proposed neural network models in this paper simply take a fixed length lookback window of the return and features series as the input at time t to predict the distribution parameters β_{t+1} of the forward return x_{t+1}.

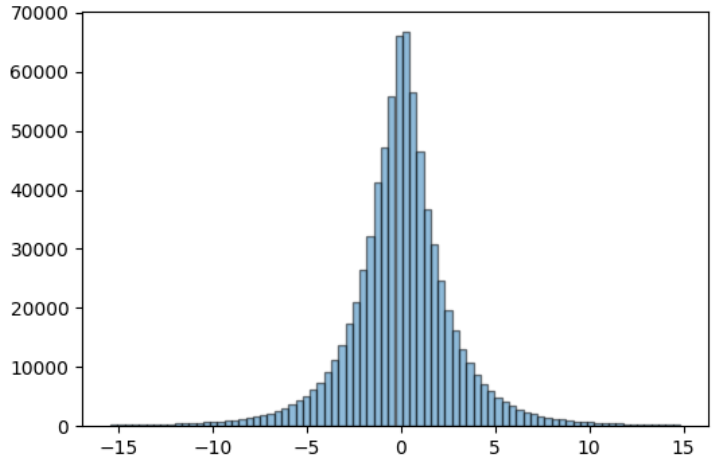


Figure 2.3: USDJPY 5min return histogram from Jan 2013 to Jun 2023 (bps)

Assumption 2.1.2. The forward return x_{t+1} is modelled by normal distribution $\mathcal{N}\left(\mu_{t+1}, \sigma_{t+1}^2\right)$, hence $\boldsymbol{\beta}_{t+1} = (\mu_{t+1}, \sigma_{t+1})$.

To support our hypothesis, we plot a histogram of the 5min return from Jan 2013 to Jun 2023 in Figure 2.3. Even the normal distribution might not be the best probabilistic model to fit the forward return because it often fails to characterise the heavy-tailed property, which is commonly observed in financial data, we still choose to use it for simplicity in parameters and at least, the symmetric pattern we discovered in Table 2.1 could be properly expressed. Figure 2.4 describes an example of the forecasting model that uses historical data to predict the parametric estimation of the forward return at time t. Each row on the left refers to one input feature vector with length T, where T indicates the lookback window size. The output of the model is simply defined as the estimated parameters according to which statistical distribution you are predicting, for example, the normal distribution in our assumption. Next we give the generalised definition of the probabilistic forecasting model used in our paper.

Definition 2.1.3 (Probabilistic forecasting model)**.** Let $\{\boldsymbol{x}\}_{t=1\ldots N}$ be the return series and $\{\boldsymbol{z}\}_{t=1\ldots N}$ be the features series. Given length T, for $T < t_0 < N$, probabilistic forecasting model $\mathcal{M}$ at time t_0 takes the fixed size lookback window of $(\boldsymbol{x}_{t_0-T+1}, \ldots, \boldsymbol{x}_{t_0})$ and $(\boldsymbol{z}_{t_0-T+1}, \ldots, \boldsymbol{z}_{t_0})$ as the input, and outputs the estimated distribution parameters $\hat{\beta}_{t_0+1}$ of the $\boldsymbol{x}_{t_0+1}$.

Finally, we formalise the optimisation problem of the probabilistic forecasting model $\mathcal{M}_\theta$, with the optimal parameters denoted by $\hat{\theta}$,

Figure 2.4: Working flow of the probabilistic forecasting model

$$\widehat{\theta} := \arg\min_{\theta \in \Theta} L\left(\boldsymbol{x}_{T+1:N}; \hat{\beta}_{T+1:N}\right), \tag{2.1.1}$$

where the estimated normal distribution parameters is produced by

$$\hat{\beta}_{t+1} = (\hat{\mu}_{t+1}, \hat{\sigma}_{t+1}) := \mathcal{M}_\theta\left(\boldsymbol{x}_{t-T+1:t}, \boldsymbol{z}_{t-T+1:t}\right), \tag{2.1.2}$$

and the loss function L is defined as the negative log-likelihood

$$\begin{aligned} L\left(\hat{\beta}_{T+1:N}; \boldsymbol{x}_{T+1:N}\right) :&= -l\left(\hat{\beta}_{T+1:N}; \boldsymbol{x}_{T+1:N}\right) \\ &= -\sum_{t=T}^{N-1} \log\left(f_{\hat{\beta}_{t+1}}\left(\boldsymbol{x}_{t+1}\right)\right), \\ &= -\sum_{t=T}^{N-1} \log\left(\frac{1}{\hat{\sigma}_{t+1}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\boldsymbol{x}_{t+1}-\hat{\mu}_{t+1}}{\hat{\sigma}_{t+1}}\right)^2}\right). \end{aligned} \tag{2.1.3}$$

2.2 Model framework

In section 2.1, we have formalised the optimisation problem in equation 2.1.1, showing the optimal parameters $\hat{\theta}$ in forecasting model $\mathcal{M}$ are achieved by minimising the negative log-likelihood loss function. Recently, the deep models are increasingly used in a wide range of areas for its powerful potential in representing complex patterns over massive data. Several successful architectures of neural network have been developed for time series tasks, such as the Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Transformer and other typical models. In this paper, we consider to use LSTM and Transformer these two advanced architectures as the probabilistic forecasting model. Specifically, the LSTM-based model is a variation from the DeepAR model proposed by Salinas et al., which was used to make predictions on three non-financial public datasets, "parts", "electricity", and "traffic"[20]. The main contribution of this paper is to propose the Transformer-based probabilistic forecasting model for Foreign Exchange rate movements prediction, and compare the results with the LSTM-based model over several parameters settings.

2.2.1 Long Short-Term Memory (LSTM) model

Recall that Long Short-Term Memory (LSTM) network is a type of recurrent neural network (RNN) architecture designed to handle the vanishing gradient problem and capture long-range dependencies in sequences. Distinguished applications have been seen in the fields of language modelling, machine translation, time series prediction, etc. Financial sequence is sometimes autocorrelative, which means that the information from several periods ago would be beneficial for predicting the future movements. Therefore, we first introduce the LSTM-based model in this subsection and set it as a baseline neural network model.

We start with 2.1.2, which describes that the model $\mathcal{M}_\theta$ taking the historical data from a fixed length lookback window and outputing the estimated parameters for the distribution forecasting of the forward return. In LSTM-based model, which we introduce in the first chapter, the inputs are fed recursively to the neural network. Then at time t, the model outputs the cell state c_t and hidden state h_t, and sends them again to the neural network as part of the input at time $t+1$.

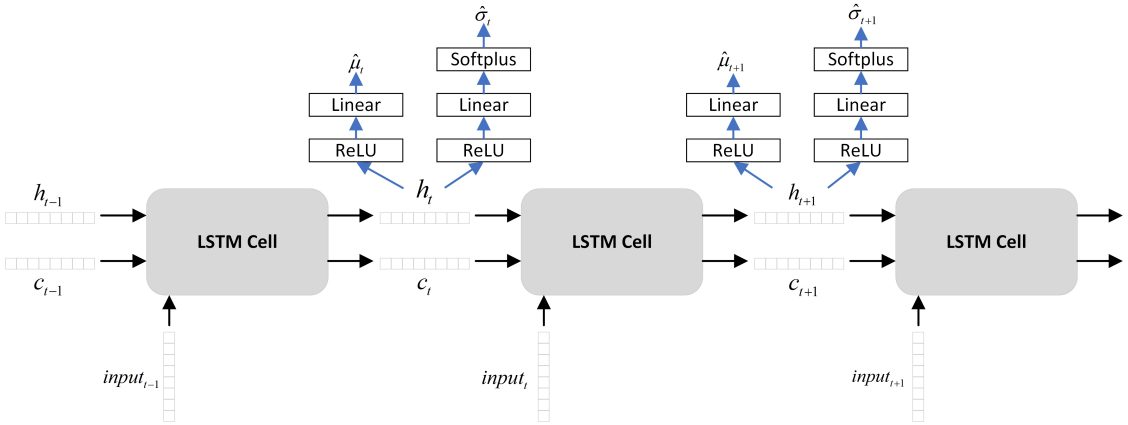


Figure 2.5: Unfolded forward computation structure of LSTM-based probabilistic forecasting model for normal distribution prediction

The unfolded forward computation structure of the LSTM-based probabilistic forecasting model is shown in Figure 2.5. We should note that there is only one LSTM cell block in the training and the unfolded LSTM cell blocks are only for intuitive explanation. Each input sample is defined as a T-length lookback window of the historical features, such as price and other technical indicators. However, in recurrent structure, they are not fed into the model at once, but step by step. The forward computation at time t relies on previous results. We notice that the model takes three streams of input at time t, which are the input feature vector $input_t$, cell state c_t and the hidden state h_t. The dimension of the feature vector is dependant on the features number, while the dimension of h_t and c_t should be the same and be set as a hyperparameter. In this paper, we set the dimension to be 64. At the very beginning, the cell state and hidden state are not defined because there is no previous output from the LSTM block. Therefore, by convention, we initialise c_0 and h_0 with zero vector.

The computation insides the LSTM cell block goes the same as the computation from equation 1.2.9 to equation 1.2.12. At time t, the model use the three streams of input to give the output of a new cell state c_{t+1} and a new hidden state h_{t+1}. Both of them are passing forward, but only the hidden state h_t is used to produce the output. For probabilistic forecasting task, the output is defined as the parametric estimation, hence the estimated parameters $\hat{\mu}_{t+1}$ and $\hat{\sigma}_{t+1}$ in our assumption 2.1.2. The network architectures for predicting $\hat{\mu}_{t+1}$ and $\hat{\sigma}_{t+1}$ from h_{t+1} are simple and independent, and both of them contain a ReLU activation layer and a linear layer. An additional Softplus layer is placed before the output of $\hat{\sigma}_{t+1}$ for shifting the sigma value to positive according to the definition in normal density function. Finally, after sending the whole sample step by step to the model, we obtain a sequence of predicted parameters $(\hat{\mu}_{t+1}, \hat{\sigma}_{t+1})$ with length T. Substituting the predicted values to the loss function in 2.1.3, we can calculate the conditional probability $f_{\hat{\beta}_{t+1}}$ and the loss of the sample. The model could be updated through the gradient computed from the batch loss by some optimisation algorithms like stochastic gradient descent (SGD).

In summary, LSTM have significantly advanced the field of sequence modeling and analysis by addressing the limitations of long-term dependency handling in traditional RNNs. However, the increased complexity and computational demands call for a balance between the model's potential and practical feasibility. As the architecture of neural network continues to evolve, it's crucial to explore some new variations, such as Transformer-based models, that offer competitive computational efficiency than LSTMs.

2.2.2 Multi-head attention Transformer model

Transformer network has created a new era of sequence modeling and natural language processing, revolutionizing how we approach tasks like machine translation, content generation, series prediction, and more. With the attention mechanism and parallel processing capabilities, Transformers have quickly risen to prominence and demonstrated unparalleled performance in various domains. However, to our best knowledge, this is the first time of Transformer-based probabilistic forecasting model being used to predict the distribution of intraday movements in Foreign Exchange market.

In this subsection, we propose the multi-head attention Transformer-based model as an advanced probabilistic forecasting model in this paper.

Similar to the LSTM-based model, the Transformer-based remains to be a sequence to sequence model, which is, from a sequential input $(input_{t-T+1}, \ldots, input_t)$ to sequential outputs $(\hat{\mu}_{t-T+2}, \ldots, \hat{\mu}_{t+1})$ and $(\hat{\sigma}_{t-T+2}, \ldots, \hat{\sigma}_{t+1})$. We should address that although both of them are sequence to sequence model, the execution manners are different. Recall that LSTM models receive inputs recursively, and at time t, the LSTM cell uses the previous output of hidden state h_t and cell state c_t as part of the input, then forward them to produce the output at next stage. Like any RNN model, this kind of recurrent nature obstructs the parallelism. In comparison, as an advantage, the Transformer-based models can execute simultaneously on all the temporal input elements, which greatly improves the learning efficiency.

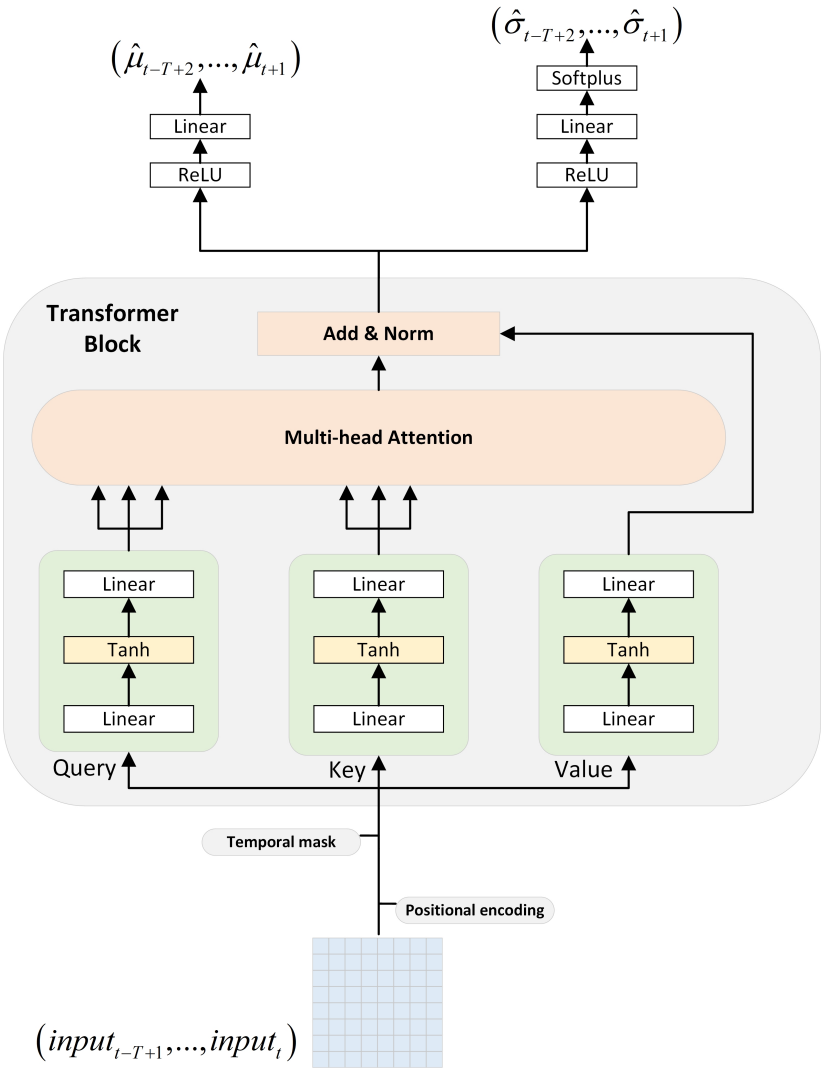


Figure 2.6: Architecture of multi-head attention Transformer-based probabilistic forecasting model for normal distribution prediction

However, one advantage of recurrent model is the capability to retain the sequential information in the input series. This capability could be essential because in most temporal sequences, the elements are not totally independent, but somehow dependent on the previous status. If we do not involve the sequential or temporal information in the feature set, then according to the architecture of Transformer, the elements are treated independently, causing the loss of causal information. In the paper "Attention is All You Need" by Vaswani et al., they mention that one way to make use of the order of the sequence is to inject some information about the relative or absolute position of the elements in the sequence. They propose to add "positional encodings" to the input embeddings

by sine and cosine functions of different frequencies,

$$PE_{(pos,2i)} = \sin\left(pos/10000^{2i/d_{\text{model}}}\right), \tag{2.2.1}$$

$$PE_{(pos,2i+1)} = \cos\left(pos/10000^{2i/d_{\text{model}}}\right), \tag{2.2.2}$$

where d_{model} is the embedding dimension, pos is the position and i is the dimension. In this paper, we use "time encoding", which is also defined as sine and cosine functions, but in two dimensions, to realise the similar thing. We will give the formal definition of "time encoding" feature in the following section.

Before sending the input to the framework, we should add a temporal mask to prevent information leakage. Considering the sequence to sequence prediction model, what we would expect is that the predictions $\hat{\mu}_{t_0+1}$ and $\hat{\sigma}_{t_0+1}$ are not using any future information beyond time t_0, hence the predictions are only using $(input_{t-T+1}, \ldots, input_{t_0})$, where t is the current time and t_0 is some time in the lookback window.

In chapter 1, we introduce the multi-head attention mechanism starting with the initial transformation 1.2.14, which is a linear layer with a weights matrix W and a bias b, and this form originates from the "Attention is All You Need" paper. However, we discovered that only using a simple linear layer may cause the gradient explosion in training. The reason behind is that we are using the negative log-likelihood loss 2.1.3 instead of mean squared error (MSE) or some other common loss functions. Therefore, according to a random initialisation of the neural network's parameters, the ultimate outputs $\hat{\mu}$ and $\hat{\sigma}$ is largely affected by the scale of input vector if there is no scaling process in between. Then, there is a chance that the likelihood being close to zero, causing extremely large value in the loss function. To this end, we introduce the "Tanh" activation function to the initial transformation before the multi-head attention part. Recall that the "Tanh" function maps the input to the range $[-1, 1]$, which limits the scale of the value passing through. In our experiments, adding a "Tanh" layer successfully avoid the gradient explosion caused by the sensitivity of negative log-likelihood loss function.

After the initial transformation, queries Q and keys K are sent to the multi-head attention block to compute the similarity scores. The scores are then used to aggregate the values V in the "Add & Norm" part shown in Figure 2.6, where the heads are concatenated together. Finally, the aggregated results are sent to two different networks for $\hat{\mu}$ and $\hat{\sigma}$ predictions, the same procedure we have introduced in the LSTM-based model.

2.3 Empirical results

After setting up the architecture of probabilistic forecasting model, this section focuses on the application to some empirical datasets. Starting with the subsection "Data preparation", we first describes how we use the raw 1min spot price data to generate structured feature data, and followed by the parameters setting in the training of LSTM-based model and Transformer-based model. At last, we evaluate how good one model works by comparing the training loss and some accuracy metrics on the predicted results.

2.3.1 Data preparation

The empirical data used in this paper is downloaded from histdata.com [31] in 1min frequency. The raw data downloaded from the website only contains four prices, "open", "high", "low" and "close". Since Foreign Exchange market opens 24 hours a day except weekends, the 1min trading periods are consistent in most of the time. In this case, we simply ignore the "open" column because it should be identical to the "close" price of last period if the periods are consistent. Therefore, we only use "high", "low" and "close" values in this paper.

Aggregate to lower frequency

The raw data comes in 1min frequency, but in this paper, we choose to use 5min frequency instead to control the input window length. For example, if we set the input as the information of past 3 hours, then for 1min frequency, we need 180 intervals, however, the number can be decreased to 36 if we use 5min frequency instead. For neural network models, larger input means there are

more parameters and computations to conduct, not only making the training process slow but also causing the overfitting problem in some cases. In other words, a longer lookback period can be inserted to the model at the same cost if we aggregate the data into a lower frequency. In fact, one can flexibly aggregate the data in any frequency beyond 1min, the highest frequency that provided by raw data.

The aggregation processes of the original data "high", "low" and "close" price from 1min to 5min frequency are defined as

$$high_t^{5\text{min}} := \max\left(high_{t-4}^{1\text{min}}, \ldots, high_t^{1\text{min}}\right), \tag{2.3.1}$$

$$low_t^{5\text{min}} := \min\left(low_{t-4}^{1\text{min}}, \ldots, low_t^{1\text{min}}\right), \tag{2.3.2}$$

$$close_t^{5\text{min}} := close_t^{1\text{min}}. \tag{2.3.3}$$

Equations 2.3.1 to 2.3.3 show that we can use vector $\left(high_t^{5\text{min}}, low_t^{5\text{min}}, close_t^{5\text{min}}\right)$ to express the interval $(t-5, t]$. Therefore, we could split one hour into 12 pieces, and report the quotes every 5 minutes. We will construct our feature set on the 5min quotes.

Features construction

Traditional technical indicators in financial trading are calculated from historical prices and volumes, such as the moving average, MACD, VWAP and other so called "momentum factors" and "inverse factors". Unfortunately, the historical volume data is not applicable for Foreign Exchange market because of the nature of an OTC market. Przemysław et. al[29] note that the market liquidity varies in a single day, explained by an analysis on the spread of currency pairs in different hourly intervals. Inspired by that, we introduce the "high minus low spread" as the value of "high" price minus "low" price, a feature that reflects the liquidity. Figure 2.7 gives the hourly average of the "high minus low spread" for USDJPY currency pair from 2013 to 2023. It is clear that the largest spread is discovered in 1p.m. to 3p.m., when the London session overlaps with the New York session. While the smallest spread is discovered at around 9p.m., when the New York market is going to close.

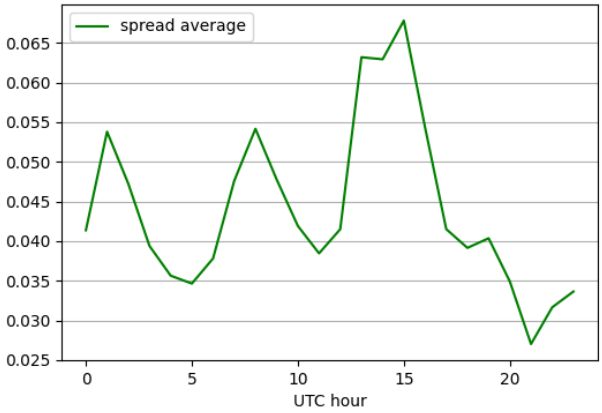


Figure 2.7: Hourly average of "high minus low spread" for USDJPY from 2013 to 2023

Sometimes it would also be beneficial to include some cross-asset information such as the data of relative rates and indices, but this destroys the self-contained property in constructing the feature set. For simplicity, we only consider features derived from the historical data of our target asset, hence the USDJPY currency pair as an example in this paper.

We conclude the constructed features for our probabilistic forecasting model in four classes:

- Z-score features

  The Z-score, also known as the standard score, is a statistical measure that quantifies how far a particular data point is from the mean of a dataset in terms of standard deviations. In financial area, people often calculate the Z-score according to a historical window so that there is no information leakage in calculation. That is, to standardise the original value and

	Description	Definition	Class
feature_1	high price	$Z = \frac{x - \mu_x}{\sigma_x}$, computed in z-score window	Z-score
feature_2	low price		
feature_3	close price		
feature_4	high minus low spread		
feature_5	EMA (close, span=5)		
feature_6	EMA (close, span=20)		
feature_7	EMA (close, span=30)		
feature_8	EMA (close, span=60)		
feature_9	high price	$R_t = \frac{x_t - x_{t-1}}{x_{t-1}}$	Return
feature_10	low price		
feature_11	close price		
feature_12	high price	$Std = \sqrt{\mathbb{E}\left[(x - \mu_x)^2\right]}$, computed in z-score window	Volatility
feature_13	low price		
feature_14	close price		
feature_15	dimension 1 (sine)	$T_1 = \sin\left(\frac{hour \cdot 60 + minute}{24 \cdot 60} \cdot 2\pi\right)$	Time
feature_16	dimension 2 (cosine)	$T_2 = \cos\left(\frac{hour \cdot 60 + minute}{24 \cdot 60} \cdot 2\pi\right)$	

Table 2.2: Definition of the forecasting model input features

compare the data points from different scales. In this paper, we mainly apply the computation of Z-score on price related features, such as the spot price, spread, exponential moving average (EMA), etc. The exponential moving average is a statistical calculation that helps smooth out fluctuations and noise in time-series data, which is commonly used in finance, defined as

$$EMA_t = \alpha \cdot x_t + (1 - \alpha) \cdot EMA_{t-1},$$

where α is the decay factor that controls how the importance of data decays through time. There are a few ways to define the decay factor, and we choose to use

$$\alpha = 2/(span + 1), \quad \text{for} \quad span \geq 1,$$

where *span* can be explained as how many periods you are looking at since the importance over the *span* periods could be ignore. The widely used technical indicator MACD is also defined as the subtraction of two EMA with different decay factors. Therefore, we hope by providing the original version of EMA, the neural network could acquire information beyond MACD. The Z-score features gives the model information of the level or the location that current price stays at. Furthermore, we won't need the batch normalisation before sending those features to the neural network, and the length of the historical window can be a hyperparameter set by the user.

- Return features

  In the feature set, the return features are simply defined as the recent movement in "high", "low" and "close" prices. Volatility clustering is a common phenomenon in finance, which describes that a large movement is likely to be followed by another large movement. At least in predicting the magnitude of next 5min return, it should be somehow informative. The return features are sent directly to the model rather than being normalised because the distribution of financial returns is often heavy-tailed, which means that in the normalisation or computation of Z-score, some large movements would become extreme values which are harmful in training.

- Volatility features

  Similar to the Z-score class, the calculation of the volatility features is also based on a historical window. It is simply taken as the variance of the return, the second central moment

in a specific period. Since we are predicting the distribution parameters, specifically, we assume to use the normal distribution as the underlying statistical model, where the parameters are (μ, σ), the first and second central moment of the future return. Considering the auto-correlation and volatility clustering, it is worth involving any statistical information.

- Time features

  Finally, we encode the intraday time stamp in two dimensions using the sine and cosine functions, and the encoding can be seen as a method of positional encoding in Transformer architecture 2.6. Recall that even though Foreign Exchange market opens 24 hours a day, the trading volume can be different from time to time. For example, London session often witnesses the most active and volatile trading activities compared to other sessions. Then, a simple idea is that, we might expect a larger σ in our parameters estimation during the active trading periods, and time features give that information. In addition, the usage of sine and cosine functions retains the cyclical nature of the time, for example, 11p.m. is close to 0a.m. in this encoding system. For intraday time representation, we transform the *hour* and *minute* to a two-dimension feature (T_1, T_2) by

$$T_1 = \sin\left(\frac{hour \cdot 60 + minute}{24 \cdot 60} \cdot 2\pi\right),$$

$$T_2 = \cos\left(\frac{hour \cdot 60 + minute}{24 \cdot 60} \cdot 2\pi\right).$$

Finally, we give the explicit definition of the 16 features derived from the historical price data in Table 2.2.

2.3.2 Parameters setting

This paper focuses on forming the whole processing flow of the proposed framework, from data preparation, model forecasting to strategy backtesting, using the asset of USDJPY currency pair as an example.

Training/test dataset

We select the historical price data from 2013-01-01 to 2023-06-30 as the whole sample, and split the dataset into a training set and a test set. The training set starts from 2013-01-01 and ends at 2020-12-31, while the test set starts from 2021-01-01 and ends at 2023-06-30. According to Figure 2.2, the training set covers both highly volatile and slightly volatile periods, and the test set begins with a low volatility and also experiences the high volatility recently.

Features construction

There are several window sizes we did not specify in previous section. One is the length of the sequential input to the probabilistic forecasting model, which we call the *lookback_window*. Another one is used to calculate the "Z-score" class and "Volatility" class features, which we call the *historical_window*. We summarise them in the following Table 2.3.

Parameters	Value
lookback_window	36×5min (3 hours)
historical_window	$5 \times 288 \times 5$min (5×24 hours)

Table 2.3: Parameters in features construction

Model architectures

Although we have shown the detailed architecture of the LSTM-based model and the Transformer-based mdoel in Figure 2.5 and Figure 2.6, there are still several parameters about the neural network we need to specify. In LSTM-based model, we should define the $LSTM\_layers$, which is the number of stacked LSTM layer, and the *hidden_dimension*, which is the number of neurons

in representing the hidden state and cell state. In multi-head attention Transformer-based model, we should define the $Transformer\_layers$, which is the number of stacked Transformer layer, and the $heads\_num$, which is the number of heads that used in computing the multi-head attention. These parameters are specified in Table 2.4.

Parameters	Value
$LSTM\_layers$	3
$hidden\_dimension$	64
$Transformer\_layers$	3
$heads\_num$	4

Table 2.4: Parameters in model architectures

Forecasting period

In our negative loglikelihood loss function defined in Equation 2.1.3,

$$L\left(\hat{\beta}_{T+1:N}; \boldsymbol{x}_{T+1:N}\right) = -\sum_{t=T}^{N-1} \log\left(\frac{1}{\hat{\sigma}_{t+1}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\boldsymbol{x}_{t+1}-\hat{\mu}_{t+1}}{\hat{\sigma}_{t+1}}\right)^2}\right),$$

where N is the length of the lookback window and label x_{t+1} is defined as the forward return of a specific forecasting period. We investigate the model performance on predicting the distribution of the forward return over the $forecasting\_period$ parameters listed in Table 2.5.

Parameters	Value
$forecasting\_period$	$5min, 15min, 30min, 60min$

Table 2.5: Parameters of forecasting period

Training process

Finally, we specify the training parameters such as the learning rate, batch size, training epochs as well as the dropout probability in the output networks. The dropout layers are added after the linear layers in the output networks. We conclude them in Table 2.6.

Parameters	Value
$learning\_rate$	0.001
$batch\_size$	64
$training\_epochs$	5
$dropout\_prob$	0.1

Table 2.6: Parameters in training process

2.3.3 Results analysis

In this subsection, we train the LSTM-based and Transformer-based probabilistic forecasting models on USDJPY currency pair over 5min, 15min, 30min and 60min forecasting periods and AUDJPY currency pair over 5min. Then utilise some metrics such as negative loglikelihood loss, directional accuracy, root mean squared error (RMSE) and 95% covered ratio, which are useful and intuitive to evaluate how good the probabilistic forecasting is. We will compare the Transformer-based model and the LSTM-based model under the same data and parameters setting. By setting up a few thresholds for labels, we discover that the models do better in predicting the larger movements. Furthermore, we also investigate the average performance of the models in an hourly analysis, evaluating how well the models work in different intraday hours.

Unlike the point estimation model, whose output prediction could be defined straightly as the target return, the probabilistic forecasting model produce somehow ambiguous output that represents the estimated parameters of a selected statistical distribution. Therefore, our motivation is that, by introducing some handcrafted accuracy and error metrics, we could better interpret the predicted results. Now we first explain the definition of each proposed metric.

Metrics definition

- Negative loglikelihood loss (NLL)

  Recall that we have formalised the optimisation problem in Section 2.1, in which we train the model to minimise the negative loglikelihood loss. This could be the most straight forward way to score the model because this error metric is what we use for training. The likelihood function indicates how likely to observe the given sample data under a statistical assumption. For example, the statistical assumption in this paper is to use the normal distribution, where the parameters are powered by our forecasting model. The likelihood value closer to 1 means the more accurate the model predicts in term of distribution. Then similarly, the negative loglikelihood loss (NLL) is taking the opposite value of the logarithmic likelihood,

$$\begin{aligned} \text{NLL} : &= \frac{1}{M} L\left(\hat{\beta}_{1:M}; \boldsymbol{x}_{1:M}\right) \\ &= -\frac{1}{M} \sum_{t=0}^{M-1} \log\left(\frac{1}{\hat{\sigma}_{t+1}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\boldsymbol{x}_{t+1}-\hat{\mu}_{t+1}}{\hat{\sigma}_{t+1}}\right)^2}\right), \end{aligned} \tag{2.3.4}$$

  where M represents the size of test dataset. The opposite and logarithm transformation projects the $[0, 1]$ range of the likelihood to the $[0, \infty)$ range of the NLL. NLL indicates the model should be better if the value is closer to 0.

- Directional accuracy (DA)

  This is a handcrafted metric proposed in this paper. For each time step, the probabilistic forecasting model transforms the input features to two parameters $(\hat{\mu}, \hat{\sigma})$, while directional accuracy (DA) only considers the correctness of $\hat{\mu}$ in term of its sign comparing to the ground truth label. We summarise this as

$$\text{DA} := \mathbb{E}\left[\mathbb{1}_{\hat{\mu} \cdot label \geq 0}\right], \tag{2.3.5}$$

  and the condition that $\hat{\mu} \cdot label \geq 0$ means the $\hat{\mu}$ has the same sign as the label, hence being correct in predicting the direction of the future movement.

- 95% covered ratio (95% CR)

  This is also a handcrafted metric proposed in this paper. This time, we consider both $\hat{\mu}$ and $\hat{\sigma}$, but not in term of direction. We choose to analyse what is the ratio of the predicted $\hat{\mu}$ that falls into the range $[\hat{\mu}-2\hat{\sigma}, \hat{\mu}+2\hat{\sigma}]$. In normal distribution $\mathcal{N}(\mu, \sigma)$, the range $[\mu-2\sigma, \mu+2\sigma]$ accounts for around 95% probability, and that is the reason we call this metric "95% covered ratio". Mathematically, the 95% CR is defined as

$$95\%\ \text{CR} := \mathbb{E}\left[\mathbb{1}_{\hat{\mu}-2\hat{\sigma} \leq label \leq \hat{\mu}+2\hat{\sigma}}\right]. \tag{2.3.6}$$

  Although this metric might not be as intuitive as the directional accuracy, it provides a more comprehensive aspect from both $\hat{\mu}$ and $\hat{\sigma}$. We could understand the 95% CR in a way that, if the value is closer to 95%, then possibly the model fits better.

- Root mean squared error (RMSE)

  Root mean squared error is one of the most commonly used metrics in a wide area, quantifying the average distance between the model outputs and the ground truth labels. However, the tricky point is that, we only have one sample, the given label, for each predicted normal distribution $\mathcal{N}(\hat{\mu}, \hat{\sigma})$ so that this is also a metric that barely looks at $\hat{\mu}$. To apply the RMSE metric, we have to assume that the label represents the "mean" value for its true underlying distribution. Then we can finally define the RMSE in Equation xx,

$$\text{RMSE} := \sqrt{\frac{1}{M} \sum_{i=1}^{M} (label_i - \hat{\mu}_i)^2}, \tag{2.3.7}$$

  where M is the size of the test dataset.

USDJPY - 5min forecasting period

From this part, we train the LSTM-based and Transformer-based probabilistic forecasting model over different forecasting periods, and evaluate the out of sample (on test dataset from 2021 to 2023) model performance through the metrics of negative loglikelihood loss (NLL), directional accuracy (DA), 95% covered ratio (95% CR) and root mean squared error (RMSE). Specifically, we set a series of thresholds for labels and run the same metrics analysis on the selected samples. Finally, we analyse the hourly performance to see if there is any advantageous period for intraday trading.

We start with 5min forecasting period, which means the model uses the lookback window of constructed features to predict the distribution of next 5min return. Given the ground truth labels, we plot the true 5min returns as well as the predicted normal distribution parameterised by $(\hat{\mu}, \hat{\sigma})$ in Figure 2.8 (for LSTM-based model) and Figure 2.9 (for Transformer based model). The way we illustrate the distribution is to shadow the 95% confidence range $[\hat{\mu} - 2\hat{\sigma}, \hat{\mu} + 2\hat{\sigma}]$ in blue, which intuitively describes how the distribution is fitted to the ground truth labels.

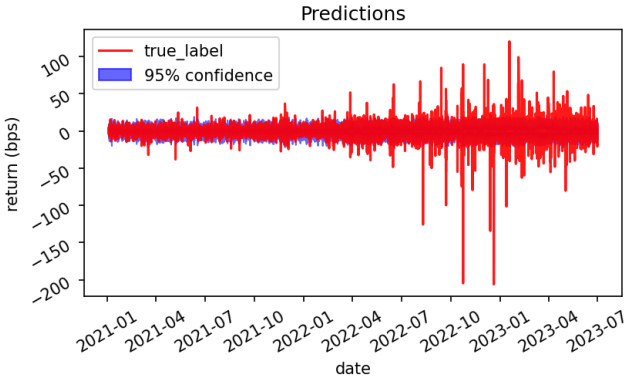


Figure 2.8: 95% confidence range for 5min USDJPY return distributions by LSTM-based model

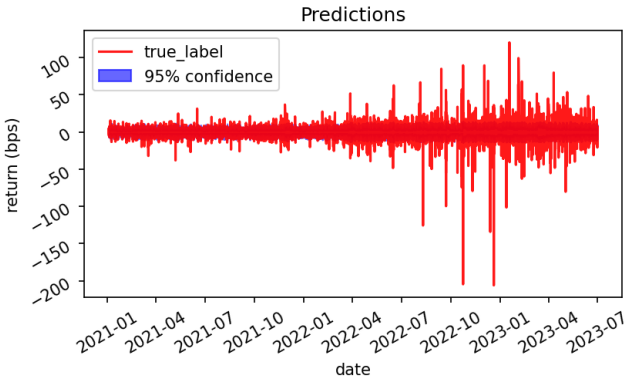


Figure 2.9: 95% confidence range for 5min USDJPY return distributions by Transformer-based model

From Figure 2.8 and Figure 2.9 we discover that, LSTM-based model is producing larger 95% confidence intervals, or we could interpret as larger $\hat{\sigma}$, with the evidence that the blue area in LSTM-based figure is greater than that in Transformer-based figure. While the common pattern is that for those extreme movements happened in 2022 and 2023, both model cannot cover them in their 95% confidence ranges. So far, it is still unclear to tell which model works better without quantifying the performance.

The motivation of this paper is to first set up a probabilistic forecasting model, and then use the predictions to form a trading strategy. Generally speaking, the large movements, in both directions, are profitable if we have a good predictive model, but equivalently, there are more risk exposures for making wrong decisions. Therefore, we come up with the idea of setting a series of

Label thresholds	Proportion	NLL	DA	95% CR	RMSE
0.0	1.0	2.462	0.5	0.922	3.558
0.5	0.777	2.747	0.516	0.9	4.03
1.0	0.593	3.082	0.517	0.869	4.594
1.5	0.453	3.439	0.518	0.832	5.208
2.0	0.352	3.788	0.519	0.793	5.829
2.5	0.273	4.164	0.519	0.751	6.508
3.0	0.213	4.553	0.52	0.71	7.215
3.5	0.169	4.949	0.519	0.67	7.926
4.0	0.136	5.353	0.519	0.634	8.657
4.5	0.109	5.804	0.52	0.595	9.43

Table 2.7: Evaluation metrics for USDJPY 5min return distributions by LSTM-based model

Label thresholds	Proportion	NLL	DA	95% CR	RMSE
0.0	1.0	2.404	0.505	0.946	3.548
0.5	0.777	2.635	0.519	0.930	4.021
1.0	0.593	2.915	0.519	0.909	4.586
1.5	0.453	3.225	0.518	0.880	5.202
2.0	0.352	3.538	0.514	0.846	5.824
2.5	0.273	3.883	0.508	0.803	6.504
3.0	0.213	4.249	0.506	0.755	7.213
3.5	0.169	4.627	0.503	0.707	7.924
4.0	0.136	5.031	0.498	0.658	8.656
4.5	0.109	5.476	0.494	0.607	9.429

Table 2.8: Evaluation metrics for USDJPY 5min return distributions by Transformer-based model

thresholds for the ground truth labels, and analyse the performance on the selected samples whose label magnitude exceeds the thresholds.

The ten thresholds are defined in the unit of basis point, ranging from 0 to 4.5. For example, to select the samples with threshold at 2.5, we should filter out all the samples with the absolute labels greater than or equal to 2.5. We also calculate the proportion that the selected samples are taking in the whole test dataset. Four proposed metrics NLL, DA, 95% CR and RMSE are then applied accordingly. We summarise these results for LSTM-based model and Transformer-based model in Table 2.7 and Table 2.8.

Note that we start with threshold at zero, which means no limitation on the labels, and of course the proportion of the whole dataset equals to 1. The proportion drops when the threshold increases so that we could focus on the tail performance on both sides.

For negative loglikelihood loss (NLL) metric, the metric that be straightly used as the loss function in our forecasting models, Transformer-based model achieves better performance than LSTM-based model in all threshold settings. However, both LSTM and Transformer models see a worsening trend when the threshold increases. The same pattern is also found in metrics of 95% CR and RMSE. One possible reason is the impact of extreme values. Since we are using the "mean" value $\hat{mu}$ and "variance" $\hat{\sigma}$ to evaluate the probabilistic prediction performance, we assume that the samples are more likely to get closer to $\hat{mu}$ according to the property of normal distribution. However, we are now setting a specific threshold that filters out those "higher" returns, and in many cases they are somehow "extreme" to their underlying distribution and causes the bias. This kind of "bias" could significantly affect any metric evaluating the distance between predicted $\hat{\mu}$ and the true return label such as root mean squared error (RMSE).

Overall, the Transformer-based model outperforms the LSTM-based model in metrics of NLL and 95% CR, and presents a slight advantage in RMSE, while the results in directional accuracy (DA) metric looks quite interesting. Transformer-based model starts with better DA in lower threshold settings, but the accuracy drops when the threshold rises. In comparison, the directional accuracy of the LSTM-based model keeps getting better with the increasing thresholds. We could interpret the results in a sense that the Transformer-based model generally performs better in forecasting the distribution, while the LSTM-model looks to have a potential in predicting the direction of larger movements.

Furthermore, as a highlighted part, we also conduct an analysis on time effect. The trading activities in different sessions could vary a lot according to our previous discussion on Figure 2.7. Recall that the feature set involves the time encoding, from which the model could possibly learns something about the time effect. We are curious to figure out how the models would perform on each one hour period.

Before getting into the same metrics evaluation for hourly periods, we would like to first investigate the statistical characteristic of each period. The results are shown in Table 2.9.

Hour	Mean	Variance	Skew	Kurtosis	Positive	Negative
0	-0.08	16.88	1.272	-70.94	0.512	0.488
1	0.085	13.263	-0.006	0.507	0.52	0.48
2	0.006	12.143	0.033	1.137	0.512	0.488
3	0.008	14.624	-0.171	9.168	0.527	0.473
4	0.018	7.146	0.014	0.579	0.53	0.47
5	0.042	5.922	-0.007	0.55	0.529	0.471
6	0.035	7.679	-0.009	0.574	0.526	0.474
7	0.021	12.494	-0.022	0.37	0.527	0.473
8	-0.023	14.631	0.005	-0.124	0.514	0.486
9	-0.01	15.232	0.029	-0.996	0.515	0.485
10	0.044	9.147	-0.004	0.54	0.518	0.482
11	0.019	8.539	-0.009	0.394	0.516	0.484
12	0.048	11.65	0.054	2.379	0.52	0.48
13	-0.007	40.919	0.016	-0.549	0.515	0.485
14	-0.027	26.891	0.022	-0.554	0.509	0.491
15	0.038	22.479	-0.036	0.61	0.514	0.486
16	0.03	13.805	-0.048	1.187	0.519	0.481
17	0.008	8.877	-0.01	0.558	0.521	0.479
18	0.037	7.967	0.025	1.241	0.531	0.469
19	-0.022	8.229	0.041	-1.202	0.515	0.485
20	0.059	4.797	-0.01	1.344	0.528	0.472
21	-0.097	4.176	-0.153	-8.187	0.517	0.483
22	0.206	7.545	1.498	58.896	0.593	0.407
23	0.036	6.955	0.031	0.729	0.53	0.47

Table 2.9: Hourly statistical results of USDJPY 5min return (in bps) from 2021 to 2023 (UTC)

Here we want to show that the performances on each period are not consistent. From the empirical analysis in Table 2.9, hour 22 has the largest "mean" value while hour 13 has the largest "variance", reflecting that the trading activities in the market keep changing in a day. Recall that in UTC time zone, hour 13 is the overlapping part of London session and New York session, in which a large number of trades happen. Hour 22 often witnesses a low liquidity because of the closure of both the London market and New York market. Therefore, we choose to plot the evaluation metrics for hour 13 and hour 22 in Figure 2.10 and Figure 2.11 as an example.

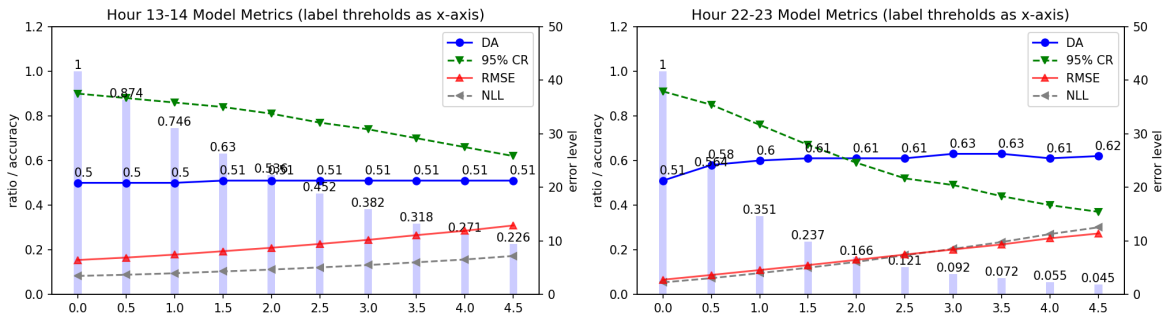


Figure 2.10: Evaluation metrics for hour 13 and hour 22 for 5min USDJPY return distributions by LSTM-based model

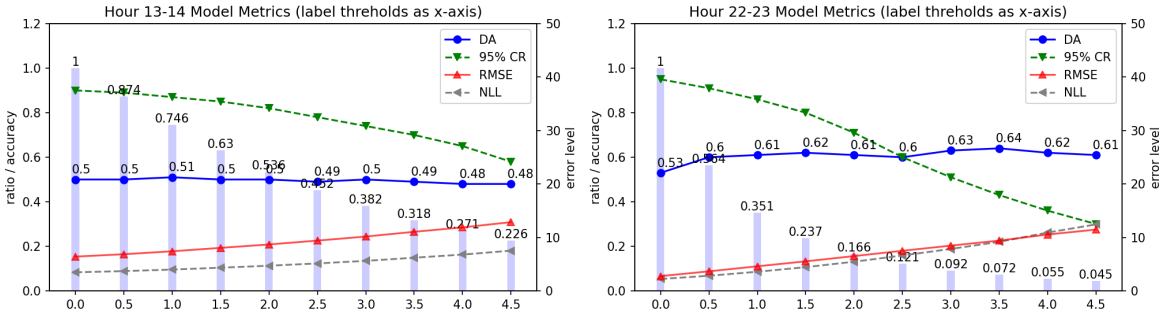


Figure 2.11: Evaluation metrics for hour 13 and hour 22 for 5min USDJPY return distributions by Transformer-based model

From the plots, it is not difficult to see the directional accuracy (DA) soars up in hour 22, even if other metrics remain getting worse as the threshold shifts. The blue bars in the figures indicate the proportion of the samples selected by the thresholds. It should be convincing to see there are more larger movements happened in hour 13 than hour 22 because in Table 2.9 the variance in hour 13 is 40.919, while the value is only 7.545 in hour 22. The motivation of showing the two figures is to address the fact that the probabilistic models could have some advantageous hours, which might be beneficial for automatic trading.

So far we have finished the discussion about the performance of LSTM-based model and Transformer-based model on USDJPY with 5min forecasting period. Conservatively speaking, the Transformer-based model performs no worse than LSTM-based model, and especially in some metrics such as negative loglikelihood loss (NLL) and 95% confidence ratio (95% CR), it shows even better performance. To conclude, considering the computing efficiency, the Transformer-based model could be qualified as an alternative probabilistic forecasting model other than the LSTM-based model in this case.

AUDJPY - 5min forecasting period

In the previous part, we have discussed about the performance of the probabilistic forecasting models on USDJPY currency pair. As a supplement of the experiments, and to exhibit the flexibility of the proposed models, we consider to train the AUDJPY currency pair based on the same methodology in USDJPY with 5min forecasting period. Similarly, we present the results in the order of 95% confidence range, evaluation metrics, hourly statistical results and some hour sample analysis.

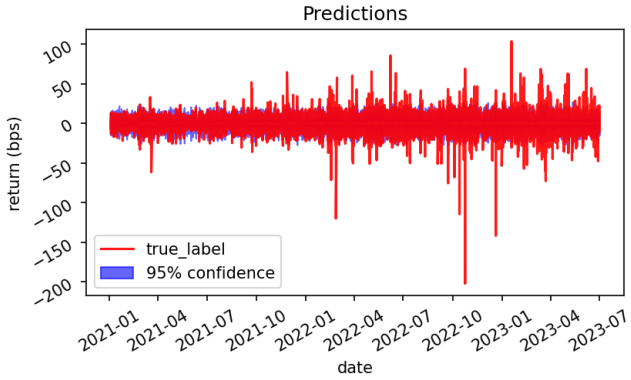


Figure 2.12: 95% confidence range for 5min AUDJPY return distributions by LSTM-based model

We plot the 95% confidence range in blue and the true labels in red. Figure 2.12 shows the results of probabilistic forecasting on AUDJPY currency pair in out of sample period predicted by LSTM-based model, while Figure 2.13 refers to the Transformer-based model. Both models are trained for 5 epochs with the parameters setting described in subsection 2.3.2. The blue 95%

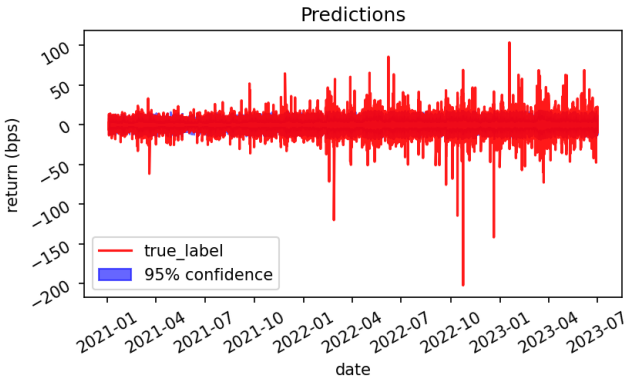


Figure 2.13: 95% confidence range for 5min AUDJPY return distributions by Transformer-based model

confidence shadow area in Figure 2.12 seems still larger than that in Figure 2.13. According to the previous analysis of USDJPY currency pair, this could be caused by the over estimation of the volatility parameter $\hat{\sigma}$ in the predicted normal distribution. We still need more numerical results to quantify this.

The evaluation metrics for LSTM-based model and Transformer-based model are computed and illustrated in Table 2.10 and Table 2.11 respectively.

Label thresholds	Proportion	NLL	DA	95% CR	RMSE
0.0	1.0	2.819	0.505	0.925	4.461
0.5	0.858	2.961	0.515	0.912	4.811
1.0	0.731	3.121	0.517	0.897	5.2
1.5	0.619	3.297	0.518	0.879	5.62
2.0	0.522	3.487	0.519	0.857	6.073
2.5	0.435	3.698	0.519	0.83	6.572
3.0	0.363	3.918	0.52	0.8	7.091
3.5	0.302	4.151	0.519	0.766	7.631
4.0	0.252	4.393	0.517	0.729	8.187
4.5	0.21	4.642	0.516	0.69	8.764

Table 2.10: Evaluation metrics for AUDJPY 5min return distributions by LSTM-based model

Label thresholds	Proportion	NLL	DA	95% CR	RMSE
0.0	1.0	2.766	0.508	0.946	4.449
0.5	0.858	2.883	0.518	0.937	4.802
1.0	0.731	3.017	0.52	0.926	5.192
1.5	0.619	3.167	0.521	0.912	5.615
2.0	0.522	3.33	0.521	0.896	6.069
2.5	0.435	3.512	0.52	0.875	6.569
3.0	0.363	3.705	0.52	0.85	7.09
3.5	0.302	3.909	0.519	0.821	7.631
4.0	0.252	4.123	0.517	0.787	8.187
4.5	0.21	4.349	0.515	0.749	8.764

Table 2.11: Evaluation metrics for AUDJPY 5min return distributions by Transformer-based model

Metrics negative loglikelihood loss (NLL) and root mean squared error (RMSE) are loss indicators, which means "smaller is better". Directional accuracy (DA) and 95% confidence ratio are two handcrafted metrics, and the higher score in DA means the higher chance to win the directional bet, while a good 95% CR score should get close to 95%. According to the criteria above, we find that the Transformer-based model outperforms the LSTM-based model in most cases, except for the DA metric with label threshold 4.5. Even though in term of RMSE metric, two models are close to each other, Transformer-based model still keeps slight advantage in small thresholds.

Now we continue to investigate the statistical characteristic of AUDJPY currency pair on an hourly basis in Table 2.12.

Hour	Mean	Variance	Skew	Kurtosis	Positive	Negative
0	-0.206	23.345	2.02	-87.633	0.497	0.503
1	0.072	23.582	-0.008	0.287	0.518	0.482
2	0.036	24.79	0.044	1.087	0.509	0.491
3	0.016	22.223	-0.078	2.221	0.525	0.475
4	0.023	14.488	-0.013	0.568	0.511	0.489
5	0.058	15.946	0.16	3.534	0.519	0.481
6	0.056	15.097	0.005	0.282	0.52	0.48
7	0.015	20.445	-0.006	0.079	0.515	0.485
8	-0.026	28.067	0.001	-0.076	0.505	0.495
9	-0.075	23.149	0.051	-1.485	0.515	0.485
10	0.028	15.953	-0.0	0.14	0.511	0.489
11	-0.053	15.1	0.011	-0.416	0.502	0.498
12	0.042	17.331	0.011	0.371	0.516	0.484
13	0.002	30.048	-0.003	0.066	0.519	0.481
14	0.016	33.55	-0.003	0.11	0.51	0.49
15	-0.056	37.087	0.019	-0.238	0.509	0.491
16	0.034	21.661	-0.002	0.211	0.519	0.481
17	0.012	14.362	0.009	0.202	0.512	0.488
18	0.024	13.381	0.001	0.455	0.517	0.483
19	0.026	14.209	-0.017	0.861	0.524	0.476
20	0.049	10.243	0.024	0.447	0.518	0.482
21	-0.341	8.86	1.075	-14.637	0.492	0.508
22	0.331	19.078	-0.721	38.771	0.583	0.417
23	0.194	12.253	0.155	3.067	0.536	0.464

Table 2.12: Hourly statistical results of AUDJPY 5min return (in bps) from 2021 to 2023 (UTC)

We note that the three largest variance values are discovered in hour 13, 14 and 15, before the closure of the London session. This time, we select two hours accounting for the most volatile and least volatile periods respectively, hence hour 15 with variance at 37.087 and hour 21 with variance at 8.86, plotted in Figure 2.14 and 2.15.

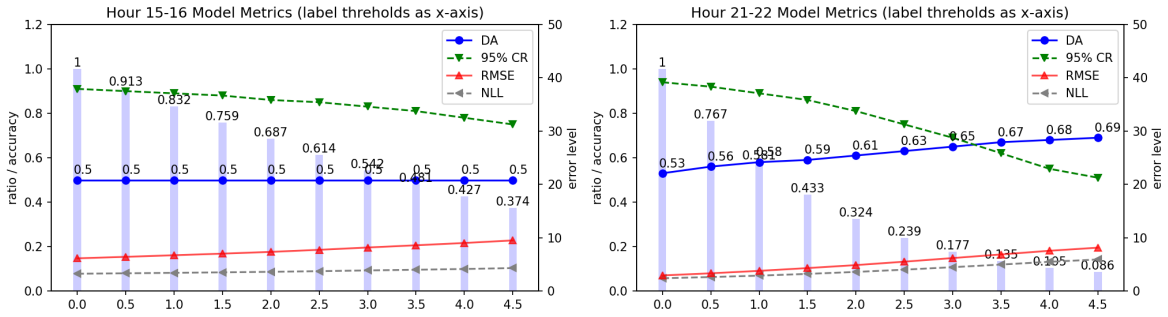


Figure 2.14: Evaluation metrics for hour 15 and hour 21 for 5min AUDJPY return distributions by LSTM-based model

In hour 15, there are more large movements since the proportion of labels whose magnitude exceeds 4.5 is staying high as 37.4%, while in hour 21 the proportion drops to 8.6%. LSTM-based model shows no advantage at predicting the direction in hour 13, but it achieves higher directional accuracy (DA) than Transformer-based model in hour 21. In contrast, the Transformer-based model slightly outperforms the LSTM-based model in directional accuracy in hour 13. These results indicate that we should not neglect the "time effect" in Foreign Exchange market, and the two probabilistic forecasting models could perform differently even in the same hour period.

To conclude, the two empirical experiments on currency pair USDJPY and AUDJPY have

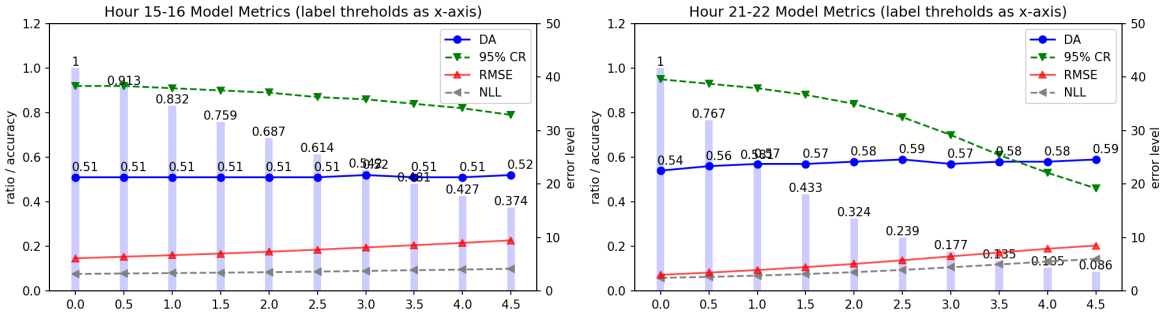


Figure 2.15: Evaluation metrics for hour 15 and hour 21 for 5min AUDJPY return distributions by Transformer-based model

shown the potential of the multi-head attention Transformer-based probabilistic forecasting model. Particularly, we also examine the model performances of the 15min, 30min and 60min forecasting period. The numerical results are attached in the appendix of this paper.

At least, in terms of metrics such as negative loglikelihood loss (NLL), 95% confidence ratio (95% CR) and root mean squared error (RMSE), Transformer-based model outperforms the LSTM-based model after 5 epochs of training. Furthermore, considering that the training cost of Transformer network is far lower than LSTM network because of the parallelism in matrix computation, we suggest that the proposed multi-head attention Transformer-based model should be qualified as an alternative framework for time series probabilistic forecasting tasks.

Chapter 3

Trading strategies

In this chapter, we develop a few daily strategies utilising the predictions from our Transformer-based probabilistic forecasting model. Specifically, we use the sign of predicted $\hat{\mu}$ as the trading direction, or in other words, we take long position if the output is positive and short position if negative. The predicted volatility $\hat{\sigma}$ is used to optimise the position by maximising a utility function. At the end of the day, the daily strategies clear the position and calculate the daily PnL. We settle the daily PnL in the base currency JPY first, then transform to the quote currency according to the instantaneous rate, and backtest the strategies in terms of Annualised Return, Sharpe Ratio and Maximum Drawdown in the out of sample period from 2021 to 2023. In comparison, we set "buy and hold" strategy and "20-period moving average" strategy as the benchmarks. These proposed and benchmark strategies are introduced in the first section of this chapter.

3.1 Strategies establishment

This section introduces the proposed and benchmark strategies of the paper. From data preparation, feature construction to model prediction, we have fulfilled the probabilistic forecasting framework. We may now focus on building some automated trading strategies using the predicted results. For simplicity, we only consider spot trading, that is, we purchase or sale the currency for immediate delivery. Options are not included.

3.1.1 Proposed strategies

For consistency with the previous variable definitions, we continue to use $(\hat{\mu}_{t+1}, \hat{\sigma}_{t+1})$ for the output parameters from the Transformer-based probabilistic forecasting model at time t. To comply with the risk exposure, we set the following assumption.

Assumption 3.1.1. The maximal exposure is set to be 10 million quote currency for intraday trading.

Recall that the amount in the quote currency, for example the USD in USDJPY, can be seen as the the share that one holds the currency pair. Therefore, we can interpret the constraint as the maximal holdings in a day. Here are two simple strategies we use to examine the prediction power in trading.

- Directional trading with no scaling

  Specifically, the directional trading strategy takes the sign of the "mean" parameter $\hat{\mu}$ in the predicted normal distribution. If we assume that the model predicts the true underlying distribution for the forward return, then $\hat{\mu}$ indicates how would be the movement like on average. For each time step, we have

$$\text{Position}_t = \begin{cases} \text{10 m}, & \hat{\mu}_{t+1} \geq 0, \\ \text{-10 m}, & \hat{\mu}_{t+1} < 0. \end{cases} \tag{3.1.1}$$

- Directional trading with mean-variance scaling

  Simple directional trading is too naive to make full use of the predicted results. With $\hat{\mu}_{t+1}$ and $\hat{\sigma}_{t+1}$ in hand, we come up with the idea of optimising the position using the classic "mean-variance" framework.

  Our goal is to find out the optimal proportion of the portfolio that should be invested in the quote currency, noted by π_t. Without loss of generality, at time t, we use v_t to express the unit value of portfolio and let s_t be the foreign exchange rate of the currency pair. Then we could formalise the one step PnL as

$$v_{t+1} - v_t = \pi_t \cdot x_{t+1}, \tag{3.1.2}$$

  where x_{t+1} is the forward return. Fortunately, we have already obtained the predicted normal distribution $\mathcal{N}(\hat{\mu}_{t+1}, \hat{\sigma}_{t+1})$ for x_{t+1}. We optimise the conditional expectation of the exponential utility function by

$$\sup_{\pi_t \in \Pi} \quad \mathbb{E}_t \left[U\left(v_{t+1}\right) \right], \tag{3.1.3}$$

  where

$$U\left(v_{t+1}\right) := -e^{-\gamma v_{t+1}}, \tag{3.1.4}$$

  and γ is the risk aversion coefficient. Then we replace v_{t+1} in equation 3.1.3,

$$\begin{aligned} & \sup_{\pi_t \in \Pi} \quad \mathbb{E}_t \left[U\left(v_{t+1}\right) \right] \\ = & \sup_{\pi_t \in \Pi} \quad \mathbb{E}_t \left[-e^{-\gamma v_t - \gamma \pi_t \cdot x_{t+1}} \right] \\ = & \sup_{\pi_t \in \Pi} \quad -e^{-\gamma v_t} \mathbb{E}_t \left[e^{-\gamma \pi_t \cdot x_{t+1}} \right] \\ = & \sup_{\pi_t \in \Pi} \quad -e^{-\gamma v_t} \cdot e^{-\gamma \pi_t \hat{\mu}_{t+1} + \gamma^2 \pi_t^2 \hat{\sigma}_{t+1}^2 / 2}, \end{aligned} \tag{3.1.5}$$

  the optimisation problem equals to the minimisation

$$\min_{\pi_t \in \Pi} \quad -\gamma \pi_t \hat{\mu}_{t+1} + \frac{\gamma^2 \sigma^2}{2} \pi_t^2, \tag{3.1.6}$$

  where the optimal proportion

$$\pi_t^* := \frac{\hat{\mu}_{t+1}}{\hat{\sigma}_{t+1}^2 \gamma}. \tag{3.1.7}$$

  Finally, the directional trading strategy with mean-variance scaling can be summarised as

$$\text{Position}_t = \begin{cases} \min\left(\pi_t^*, 1\right) \cdot \ 10 \text{ m}, & \pi_t^* \geq 0, \\ \max\left(\pi_t^*, -1\right) \cdot \ 10 \text{ m}, & \pi_t^* < 0. \end{cases} \tag{3.1.8}$$

  Note that for the constraint on the position exposure in 3.1.1, proportion π_t^* should be truncated to $[-1, 1]$.

3.1.2 Benchmark strategies

The "buy and hold" and "20-period moving average" are two basic strategies we use as the benchmarks in this paper.

- Buy and hold

  "Buy and hold" strategy is a long-only strategy. Investor enters with position in the beginning of the day, and do nothing until the end of the day. The daily PnL of the strategy only depends on the initial price and the last price of the day. The definition of the "buy and hold" strategy could be defined as equation 3.1.9.

$$\text{Position}_t = \ 10 \text{ m}. \tag{3.1.9}$$

- 20-period moving average

  The idea of "moving average" strategy is to distinguish bullish and bearish market, and investor goes long in bullish and goes short in bearish. The assumption is that, if the price or rate s_t stays above the moving average, we identify the market as bullish and vice versa. For 20-period moving average, we could summarise as

$$\text{Position}_t = \begin{cases} \text{10 m}, & s_t \geq ma_{20}, \\ \text{-10 m}, & s_t < ma_{20}. \end{cases} \tag{3.1.10}$$

3.2 Backtesting on empirical dataset

We backtest the strategies on the empirical dataset of USDJPY and AUDJPY currency pairs. The forecasting period is set to be 5min so that each strategy might produce a new position for the next 5min holding period. In this section, we first run the evaluation process for proposed strategies and benchmark strategies on the whole sample from 2013 to 2023 to see the overall performance of the model. This is also an intuitive way to see the how severe the overfitting problem is. After that, we keep a close eye on the out of sample performance. As we close the position in the end of the day, it is natural to aggregate all the intraday PnLs to a daily one. Several classic metrics are introduced in the second part of this section to quantify the strategy performance in terms of the daily PnL series. Finally, we still do some investigation on the intraday contribution to the total PnL, in other words, the "time effect".

3.2.1 PnL evaluation

At the beginning, we specify four proposed strategies and two benchmark strategies with their alias in Table 3.2.1.

Strategy	Alias
Directional trading with no scaling	Strategy 1
Directional trading with scaling $\gamma = 0.01$	Strategy 2
Directional trading with scaling $\gamma = 0.05$	Strategy 3
Directional trading with scaling $\gamma = 0.1$	Strategy 4
Buy and hold	Benchmark 1
20-period moving average	Benchmark 2

Table 3.1: Specified trading strategies and their alias

We first evaluate the proposed strategies and benchmark strategies on the whole sample data from 2013 to 2023, where the red shadow area refers to the out of sample period from 2021 to 2023. The daily PnLs are settled in base currency JPY at the end of the day.

USDJPY - 5min forecasting period

In Figure 3.1, strategy 1 and strategy 2 outperform other strategies, achieving around 3 billion PnL in JPY with a maximal exposure at 10 million USD during trading. Strategy 2 to strategy 4 are directional trading strategies with different risk aversion parameter γ, controlling the scaling magnitude. Recall that in 3.1.7, the optimal proportion π^* is also determined by γ, whereas with the truncating process in 3.1.8, the directional trading strategy with scaling might keep close to the no scaling version if the risk aversion γ is small enough. That is why we see strategy 2 with $\gamma = 0.01$ is closer to strategy 1 than strategy 3 and strategy 4 with $\gamma = 0.05$ and $\gamma = 0.1$.

Although the scaling process sacrifices part of the profits, it is still beneficial because the PnL curve gets smoother after scaling. For example, strategy 3 and strategy 4 are smoother than strategy 1 and strategy 2. A smoother curve often indicates a less volatile pattern, which is crucial in risk management.

If we look at the out of sample period from 2021 to 2023 in red zone, there is no significant discount on the performance comparing to the in sample period from 2013 to 2020. From this we could infer that our Transformer-based probabilistic forecasting model with 5-epoch training does not suffer too much from the overfitting problem.

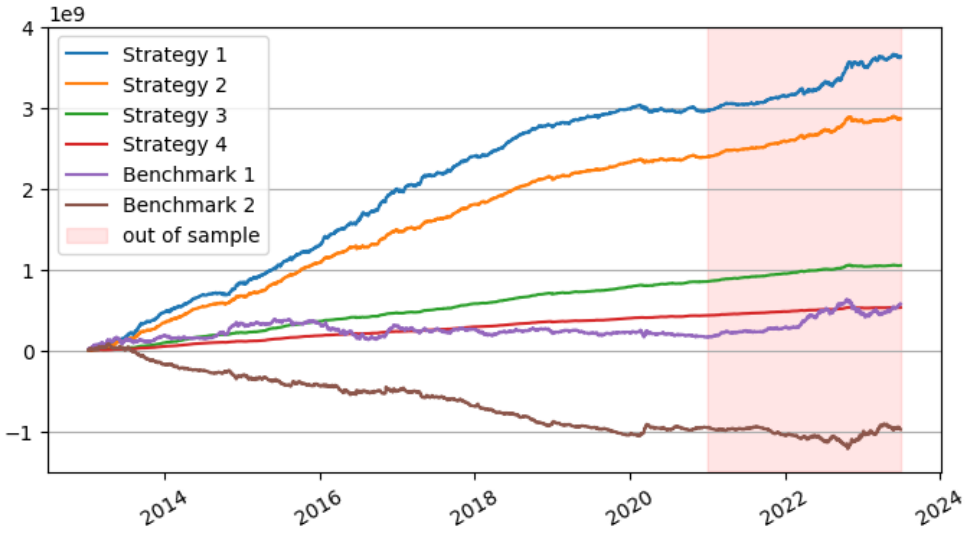


Figure 3.1: Whole sample strategies comparison in cumulative daily PnL (in base currency JPY) for USDJPY from 2013 to 2023

AUDJPY - 5min forecasting period

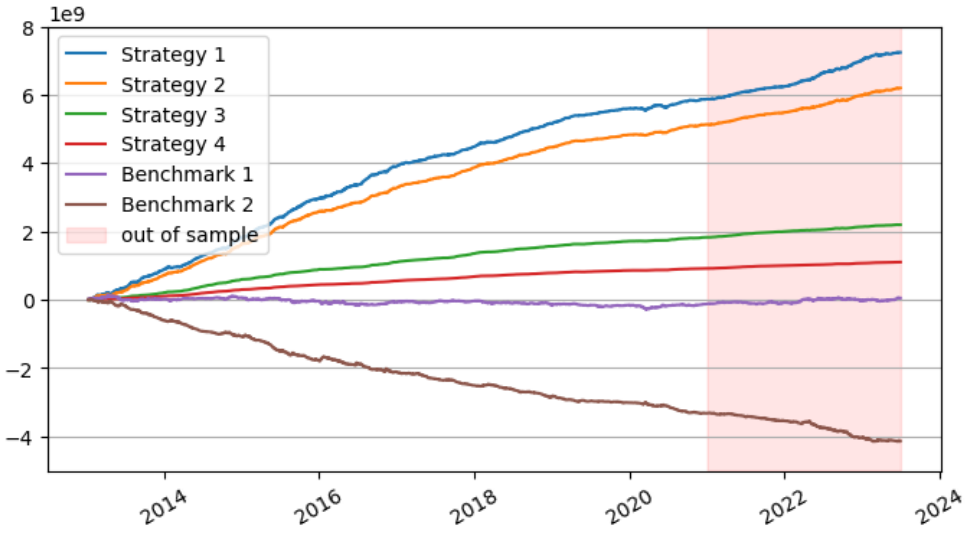


Figure 3.2: Whole sample strategies comparison in cumulative daily PnL (in base currency JPY) for AUDJPY from 2013 to 2023

For the whole sample PnL evaluation of AUDJPY in Figure 3.2, all of the proposed strategies perform better than the benchmark strategies. Similarly, we can see the scaling process sacrifice part of the profits for smoother pattern in PnL. We do not observe significant overfitting problem in the out of sample period as well.

One interesting pattern in benchmark 2, hence the 20-period moving average strategy, is that, the cumulative PnL goes down monotonically, which means we are keep making the wrong decisions. This reflects that possibly, there could be a mean reverting pattern that prevents the trend-following strategies making profits. However, if we trade oppositely, maybe we could exploit the unique pattern.

3.2.2 Out of sample performance

Figure 3.1 and Figure 3.2 have described intuitive performance of our proposed strategies which depend on the probabilistic predictions. Now, we focus on quantifying the out of sample performance using three classic PnL metrics.

Definition 3.2.1 (Annualised Return)**.** Annualised Return is sometimes recognised as the compound annual growth rate , a measure of average annual growth rate over a period of time. The metric enables us to compare two strategies with different period length. The Annualised Return for horizon T is calculated by

$$AR := \left(\frac{v_T}{v_1}\right)^{T/252} - 1,$$

where v_T and v_1 represent the ending value and initial value of the portfolio.

Definition 3.2.2 (Sharpe Ratio)**.** The Sharpe Ratio is also recognised as the risk-adjusted return, which considers both the return and volatility. The formula of annualised Sharpe Ratio is written as

$$SR := \frac{AR - r_f}{\sigma_{\text{annual}}},$$

where we assume the risk free return $r_f = 0$. The standard deviation of Annualised Return is defined as

$$\sigma_{\text{annual}} := \sqrt{252} \cdot \sigma_{\text{daily}}.$$

Definition 3.2.3 (Maximum Drawdown)**.** Maximum drawdown is a measure of risk that indicates the largest percentage decline from portfolio peak value to a trough value over a specific time period, defined as

$$MDD := \frac{v_{peak} - v_{trough}}{v_{peak}}.$$

Assumption 3.2.4 (PnL settlement)**.** The daily PnL is settled in base currency, hence JPY for both USDJPY and AUDJPY currency pairs.

Assumption 3.2.5 (Return calculation)**.** The maximal exposure is defined in the quote currency, which should be set as the initial capital. In return calculation, we transform the daily PnL from base currency to quote currency (USD in USDJPY and AUD in AUDJPY) using the instantaneous rate at the end of the day, and backtest the percentage movements in quote currency PnL.

USDJPY - 5min forecasting period

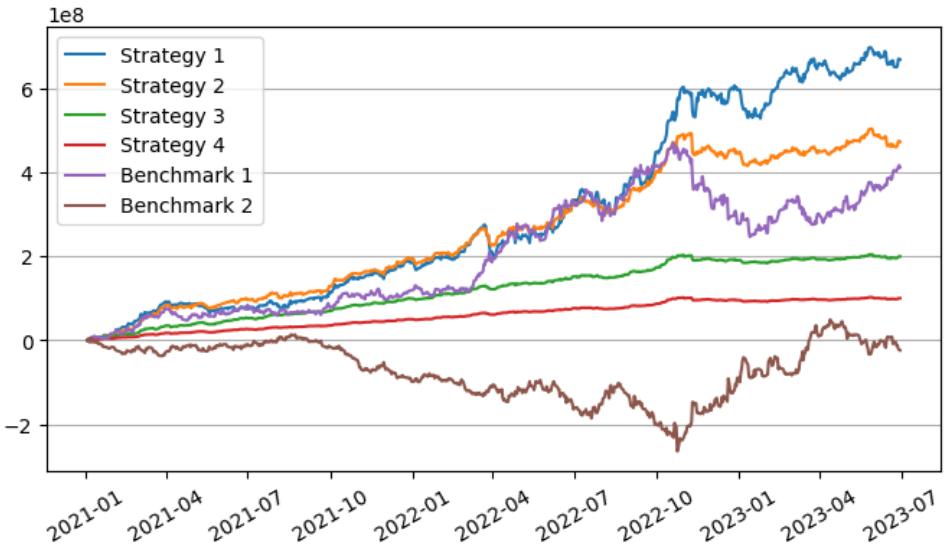


Figure 3.3: Out of sample strategies comparison in cumulative daily PnL (in base currency JPY) for USDJPY from 2021 to 2023

Figure 3.3 shows the out of sample cunulative daily PnL for USDJPY. This time, only strategy 1 and strategy 2 beat benchmark 1, the "buy and hold" strategy. One of the reason could be that, a distinct uprising trend in the USDJPY Foreign Exchange rate appears in the out of sample period, making those "long-only" strategies profitable. If we look at the two benchmark strategies, we note the cumulative PnL performances are negative correlated, indicating the "mean-reverting" pattern in short period market microstructure.

Strategy	Annualised Return	Sharpe Ratio	Maximum Drawdown
Strat 1	0.131399	2.199855	0.051151
Strat 2	0.096136	2.373096	0.031606
Strat 3	0.042868	3.156164	0.012098
Strat 4	0.021954	3.079365	0.006485
Benchmark 1	0.084754	1.457981	0.091643
Benchmark 2	-0.005411	-0.058717	0.187636

Table 3.2: Out of sample cumulative PnL metrics for USDJPY from 2021 to 2023

We summarise the numerical metric results for USDJPY in Table 3.2. Strategy 1 has the highest Annualised Return at around 13.1%, while strategy 3 obtains the best annualised Sharpe Ratio at around 3.16. With the mean-variance scaling process, strategy 2 to strategy 4 outperform other strategies in both Sharpe Ratio and Maximum Drawdown. Overall the proposed strategies certainly achieve better performance than the benchmark strategies.

AUDJPY - 5min forecasting period

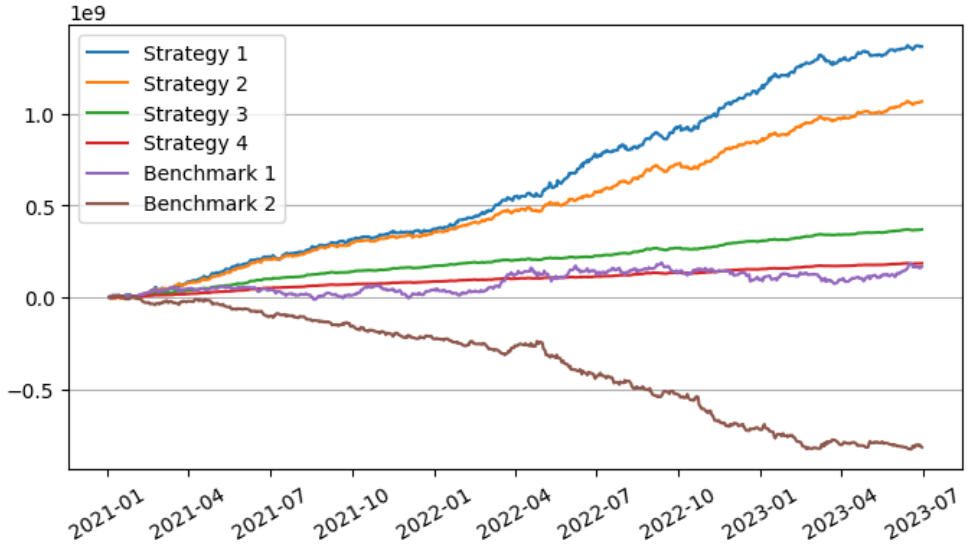


Figure 3.4: Out of sample strategies comparison in cumulative daily PnL (in base currency JPY) for AUDJPY from 2021 to 2023

Figure 3.4 shows the out of sample cunulative daily PnL for AUDJPY. Strategy 1 to strategy 3 achieve higher cumulative PnL than the benchmark strategies. Although the accumulated PnL of strategy 4 does not appear significant advantage in comparison to benchmark 1, the smoother pattern in strategy 4 promises that the strategy is less risky than the other.

The numerical metric results for AUDJPY are summarised in Table 3.3. Similarly, strategy 1 has the highest Annualised Return at around 33.1%, while strategy 2 obtains the best Sharp Ratio by sacrificing part of the profits for the less volatile curve. Strategy 4 shows significant success in controlling the Maximum Drawdown to a level below 1%, far lower than 91.8% in benchmark 2.

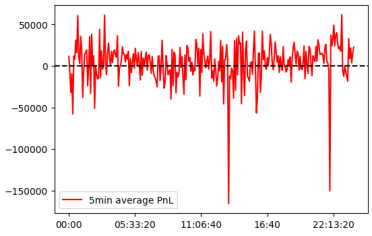
In conclusion, the proposed strategies again outperform the benchmark strategies in all PnL metrics, showing the potential of the Transformer-based probabilistic forecasting model.

Strategy	Annualised Return	Sharpe Ratio	Maximum Drawdown
Strat 1	0.330511	4.283139	0.050744
Strat 2	0.272532	4.42483	0.030464
Strat 3	0.110629	4.036591	0.017053
Strat 4	0.05846	3.843874	0.009452
Benchmark 1	0.053018	0.622707	0.089726
Benchmark 2	-0.458353	-0.983776	0.917751

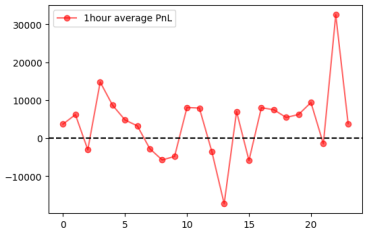
Table 3.3: Out of sample cumulative PnL metrics for AUDJPY from 2021 to 2023

3.2.3 Intraday PnL contribution

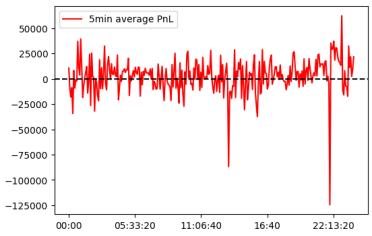
We have conducted hourly analysis of the probabilistic forecasting models in Chapter 2, from which we discover that the directional accuracy (DA) could vary a lot in different intraday periods. The motivation of this subsection is to analyse how the inconsistency would affect the intraday PnL contribution. We plot the intraday average PnL contribution in 5min and 1 hour frequencies for each proposed strategy in Figure 3.5 for USDJPY and Figure 3.6 for AUDJPY.



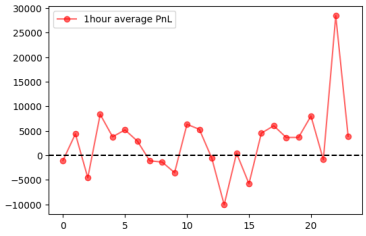
(a) Strat 1: 5min average



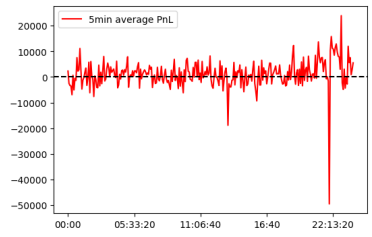
(b) Strat 1: 1 hour average



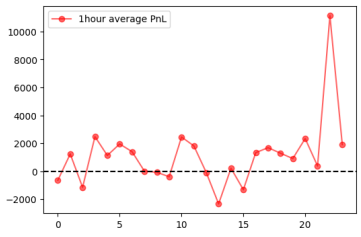
(c) Strat 2: 5min average



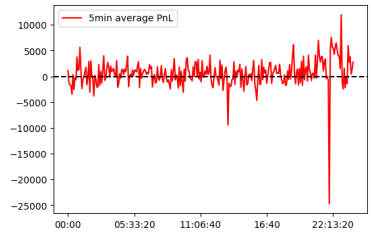
(d) Strat 2: 1 hour average



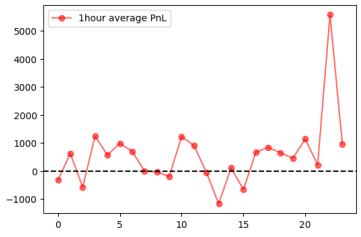
(e) Strat 3: 5min average



(f) Strat 3: 1 hour average

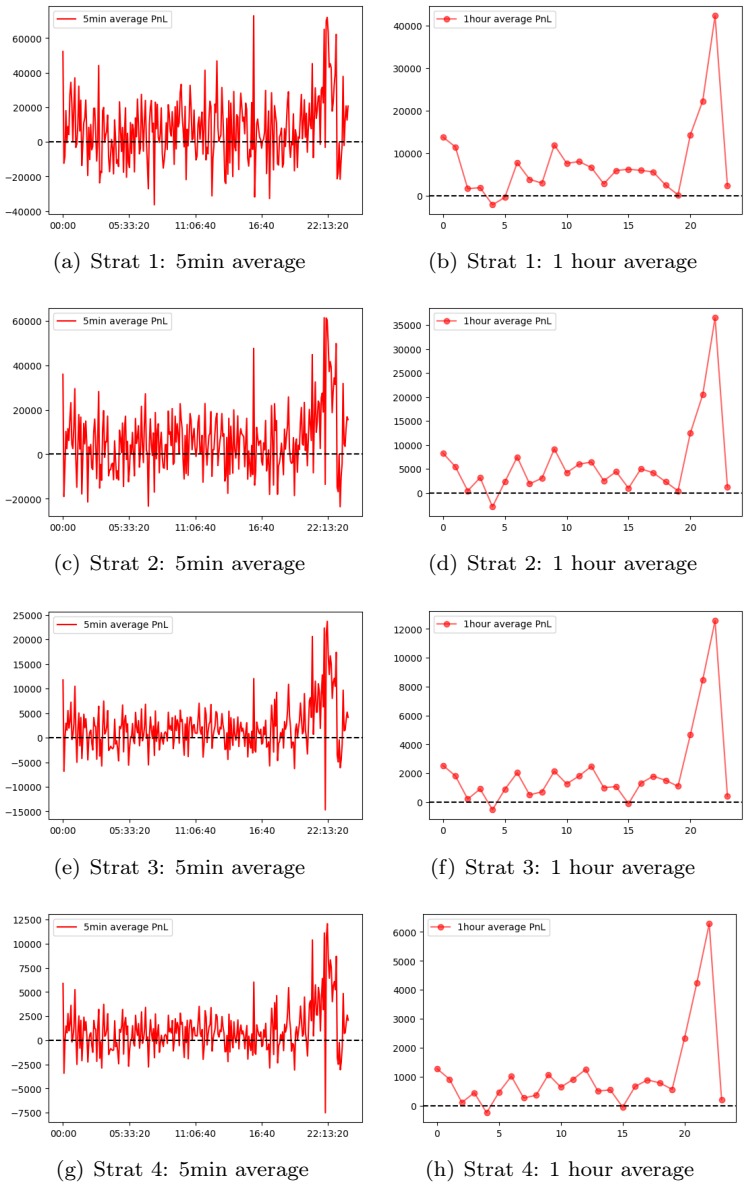


(g) Strat 4: 5min average



(h) Strat 4: 1 hour average

Figure 3.5: Intraday PnL contribution of for USDJPY in out of sample period



(a) Strat 1: 5min average

(b) Strat 1: 1 hour average

(c) Strat 2: 5min average

(d) Strat 2: 1 hour average

(e) Strat 3: 5min average

(f) Strat 3: 1 hour average

(g) Strat 4: 5min average

(h) Strat 4: 1 hour average

Figure 3.6: Intraday PnL contribution of for AUDJPY in out of sample period

Figure 3.5 shows that periods 3-7, 10-12 and 16-0 are more profitable for our proposed strategies in USDJPY trading. Recall that we investigate the accuracy metrics of the probabilistic forecasting model in hour 22, where the model achieves directional accuracy (DA) over 60%, causing the extreme and unrealistic profit in the above analysis. However, this arbitrage could be less practical for the trading desk because in that period the market is rather illiquid, making the trading riskier than other other periods. In general, even we exclude the unrealistic trading hour, the strategies are still capable to generate profits consistently. Furthermore, we find that strategy 2 to strategy 4 scale down the magnitude of the PnL, which explains why we see their cumulative PnLs are smoother.

Figure 3.6 for AUDJPY looks even more optimistic than Figure 3.5 for USDJPY, with a distinct shift from the "zero line" to the positive side. We discover that in 1 hour average contribution analysis, over 20 hour periods are generating the profits in a day. Surprisingly, the mean-variance scaling process also shifts the average PnL in hour 5 and 19 from negative to positive.

To conclude, these intraday PnL contribution results support the potential of the proposed multi-head attention Transformer-based probabilistic forecasting model generating profits even under some simple directional strategies. Furthermore, the "positive shift" in Figure 3.5 and Figure 3.6 reveals that the model has certainly learned something about the "underlying distribution".

Conclusion and Future Work

In this paper, the main contribution is to propose a multi-head attention Transformer-based probabilistic forecasting model for intraday Foreign Exchange rate trading task, and form a complete framework from probabilistic forecasting to trading strategies. Our numerical results suggest that the proposed Transformer-based model outperforms the LSTM-based model, which is developed from the DeepAR model by Salinas et al., not only in the negative loglikelihood training loss but also in some other accuracy and error metrics. Furthermore, the paralleled computations in Transformer neural network allow the model to be trained more efficiently than the LSTM-based model. In the end, to examine the profitability of the model, we establish a few simple directional trading strategies that utilises the predicted distribution from the forecasting model. Backtesting on USDJPY and AUDJPY currency pairs exhibits its great potential in making profits.

At the beginning of this paper, we introduce the concepts of probabilistic forecasting and fundamental neural network architectures for sequential prediction. Unlike the regression or classification tasks, the probabilistic models are trained to minimise the negative loglikelihood loss over a certain distribution class. For simplicity, we select normal distribution as the underlying statistical model. Our model is aimed to produce the best estimation on the parameters that characterising the distribution of future return, based on a lookback window of price features. We construct 16 features in classes of "z-score", "return", "volatility" and "time". Specifically, the "time" features also play a role as the "positional encoding" in Transformer network, providing the sequential information in the input series. The Transformer-based model and the LSTM-based model are trained on the same dataset and both for 5 epochs, with the training period from 2013 to 2020 and testing period from 2021 to 2023. We choose USDJPY and AUDJPY currency pairs as individual assets, setting the forecasting period to be 5, 15, 30 and 60 min for USDJPY and 5 min for AUDJPY. The probabilistic forecasting models are evaluated with metrics of negative loglikelihood loss (NLL), directional accuracy (DA), 95% covered ratio (95% CR) and root mean squared error (RMSE). We also introduce a series of label thresholds to evaluate the model performance on the selected samples with return magnitude exceeding a specific threshold. According to the numerical results, the Transformer-based model performs better than the LSTM-based model in NLL, DA and RMSE for both currency pairs. Time effect is also studied in this paper by analysing the average performance on each 1-hour period in a day and the results show that some periods are more advantageous for the models to predict. Finally, we propose four directional trading strategies developed from the model output, hence the predicted distribution of the forward return. Compared with two benchmark strategies "buy and hold" and "20-period moving average", the proposed strategies exhibit distinct performance in Annualised Return, Sharpe Ratio and Maximum Drawdown for both USDJPY and AUDJPY with 5 min forecasting period. In addition, we analyse the intraday PnL contribution and discover the hourly average PnLs are shifted slightly to the positive side, indicating that the model could have learned some patterns about the return distribution.

Future areas of research could include variations in Transformer architectures, underlying assets, forecasting periods and trading strategies. In this paper, we choose a 4-head attention Transformer network as the model, and adding up the heads number and enlarging the features set could possibly further improve the performance. Due to page constraints, we only investigate the application on USDJPY and AUDJPY currency pairs, and we will apply the model to more assets in the future to better support the conclusion. In some papers, people set a longer forecasting period than us, which could also be worth looking into. In our view, the volatility estimation from our proposed model could be beneficial to those volatility strategies including hedging, options trading, etc. Therefore, the investigation of trading strategies is another area we will explore.

Appendix A

Theoretical supplements

A.1 Universal approximation theorem for distribution

The following theoretical assumptions and results originate from the paper by Yulong et al.[28].

Assumption A.1.1 (Kernel assumption 1)**.** The kernel k is integrally strictly positive definite: for all finite non-zero signed Borel measures μ defined on $\mathbb{R}^d$,

$$\iint_{\mathbb{R}^d} k(x,y)d\mu(x)d\mu(y) > 0. \tag{A.1.1}$$

Assumption A.1.2 (Kernel assumption 2)**.** There exists a constant $K_0 > 0$ such that

$$\sup_{x\in\mathbb{R}^d} |k(x,x)| \leq K_0. \tag{A.1.2}$$

Assumption A.1.3 (Kernel assumption 3)**.** The kernel function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is twice differentiable and there exists a constant $K_1 > 0$ such that

$$\max_{m+n\leq 1} \sup_{x,y} \left\|\nabla_x^m \nabla_y^n k(x,y)\right\| \leq K_1 \text{ and } \sup_{x,y} |\mathrm{Tr}\left(\nabla_x \nabla_y k(x,y)\right)| \leq K_1(1+d). \tag{A.1.3}$$

Definition A.1.4 (L-Lipschitz)**.** $s_\pi(x)$ is globally Lipschitz in $\mathbb{R}^d$, if there exists a constant $\tilde{L} > 0$ such that $|s_\pi(x) - s_\pi(y)| \leq \tilde{L}|x-y|$ for all $x, y \in \mathbb{R}^d$. As a result, there exists $L > 0$ such that

$$|s_\pi(x)| \leq L(1+|x|) \quad \text{for all } x \in \mathbb{R}^d. \tag{A.1.4}$$

Definition A.1.5 (sub-Gaussian)**.** The probability measure π is sub-Gaussian, if there exist $m = (m_1, \ldots, m_d) \in \mathbb{R}^d$ and $v > 0$ such that

$$\mathbf{E}_{X\sim\pi}\left[\exp\left(\alpha^T(X-m)\right)\right] \leq \exp\left(|\alpha|^2 v^2/2\right) \quad \text{for all } \alpha \in \mathbb{R}^d \tag{A.1.5}$$

Furthermore, assume that $\max_i |m_i| \leq m^*$ for some $m^* > 0$.

A.1.1 Upper bound of complexity parameter n

The upper bound of complexity parameter n in theorem 1.3.1 depends on the the choice of the discrepancy measure, along with some restrictions of the target distribution π and kernel function k. The specific assumptions and the theoretical results for each measure are shown below:

1. Wasserstein Distance

   If π satisfies that $M_3 = \mathbf{E}_{X\sim\pi}|X|^3 < \infty$, n satisfies

   $$n \leq \begin{cases} \frac{C}{\varepsilon^2}, & d = 1 \\ \frac{C\log^2(\varepsilon)}{\varepsilon^2}, & d = 2 \\ \frac{C^d}{\varepsilon^d}, & d \geq 3 \end{cases}, \tag{A.1.6}$$

   where constant C depends on M_3.

2. Maximum Mean Discrepancy (MMD)

   If kernel k satisfies assumption A.1.2, then

$$n \leq \frac{C}{\varepsilon^2}, \tag{A.1.7}$$

   with a constant C depending only on the constant K_0 in assumption A.1.2.

3. Kernelized Stein Discrepancy (KSD)

   If k satisfies assumption A.1.3 with constant K_1 and if π is L-Lipschitz and sub-Gaussian with parameters L, m, n, then

$$n \leq \frac{Cd}{\varepsilon^2}, \tag{A.1.8}$$

   where the constant C depends only on L, K_1, m^*, v, but not on d.

Appendix B

Empirical supplements

B.1 Probabilistic forecasting model

B.1.1 USDJPY - 15min forecasting period

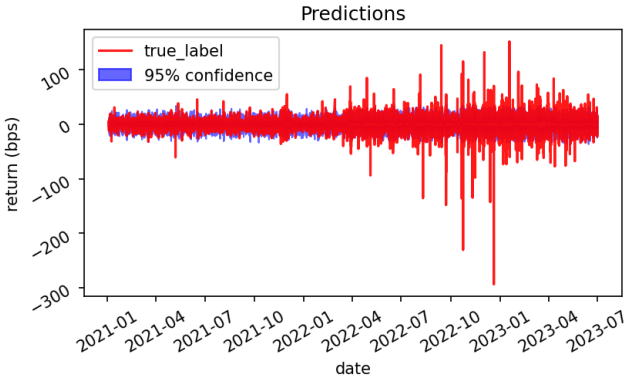


Figure B.1: 95% confidence range for 15min USDJPY return distributions by LSTM-based model

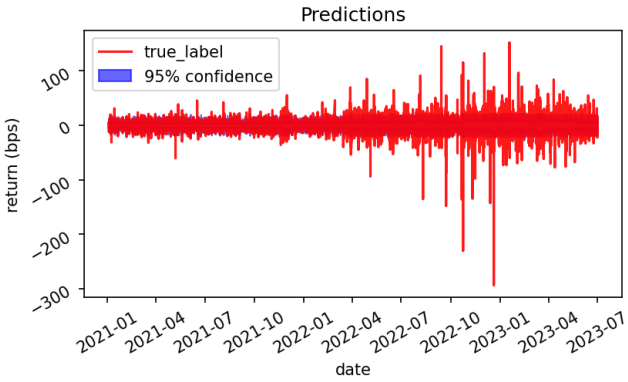


Figure B.2: 95% confidence range for 15min USDJPY return distributions by Transformer-based model

Label thresholds	Proportion	NLL	DA	95% CR	RMSE
0.0	1.0	3.013	0.51	0.925	6.224
0.5	0.872	3.159	0.518	0.914	6.655
1.0	0.752	3.328	0.519	0.9	7.151
1.5	0.645	3.514	0.52	0.884	7.698
2.0	0.554	3.703	0.521	0.865	8.265
2.5	0.474	3.905	0.521	0.844	8.881
3.0	0.405	4.114	0.522	0.821	9.529
3.5	0.349	4.322	0.52	0.798	10.179
4.0	0.301	4.537	0.517	0.774	10.851
4.5	0.26	4.755	0.517	0.75	11.539

Table B.1: Evaluation metrics for USDJPY 15min return distributions by LSTM-based model

Label thrsholds	Proportion	NLL	DA	95% CR	RMSE
0.0	1.0	2.956	0.503	0.949	6.131
0.5	0.872	3.073	0.511	0.941	6.564
1.0	0.752	3.21	0.511	0.932	7.062
1.5	0.645	3.363	0.512	0.921	7.611
2.0	0.554	3.523	0.512	0.908	8.179
2.5	0.474	3.695	0.512	0.892	8.795
3.0	0.405	3.876	0.511	0.874	9.443
3.5	0.349	4.058	0.511	0.854	10.093
4.0	0.301	4.246	0.508	0.831	10.763
4.5	0.26	4.441	0.507	0.806	11.45

Table B.2: Evaluation metrics for USDJPY 15min return distributions by Transformer-based model

Hour	Mean	Variance	Skew	Kurtosis	Positive	Negative
0	-0.148	49.446	1.256	-41.895	0.508	0.492
1	0.185	37.489	0.019	0.69	0.514	0.486
2	-0.039	49.483	0.154	-11.105	0.493	0.507
3	0.099	33.077	-1.694	87.974	0.526	0.474
4	0.091	21.105	0.126	3.286	0.531	0.469
5	0.062	17.62	-0.054	0.7	0.53	0.47
6	0.163	23.194	-0.004	1.873	0.533	0.467
7	0.019	39.666	-0.013	0.138	0.523	0.477
8	-0.087	44.215	0.089	-1.306	0.508	0.492
9	0.035	43.168	-0.11	2.563	0.517	0.483
10	0.099	26.188	-0.042	1.006	0.53	0.47
11	0.063	25.411	0.068	1.596	0.509	0.491
12	0.105	38.502	-0.051	1.788	0.523	0.477
13	-0.02	131.099	0.018	-0.735	0.516	0.484
14	-0.033	80.663	0.016	-0.305	0.506	0.494
15	0.124	62.807	-0.107	2.195	0.514	0.486
16	0.054	40.709	-0.211	3.709	0.515	0.485
17	-0.001	25.042	0.0	-0.045	0.52	0.48
18	0.165	24.404	0.124	5.343	0.535	0.465
19	-0.075	24.716	0.294	-4.697	0.513	0.487
20	0.184	13.765	0.1	6.411	0.527	0.473
21	-0.272	14.271	-0.67	-28.778	0.484	0.516
22	0.606	15.816	1.536	29.956	0.608	0.392
23	0.043	19.196	0.008	0.485	0.521	0.479

Table B.3: Hourly statistical results of USDJPY 15min return (in bps) from 2021 to 2023 (UTC)

B.1.2 USDJPY - 30min forecasting period

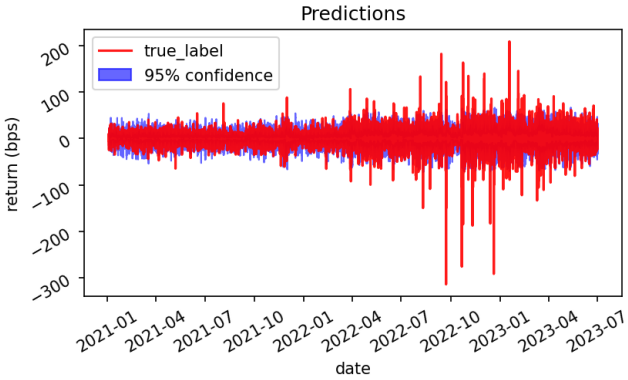


Figure B.3: 95% confidence range for 60min USDJPY return distributions by LSTM-based model

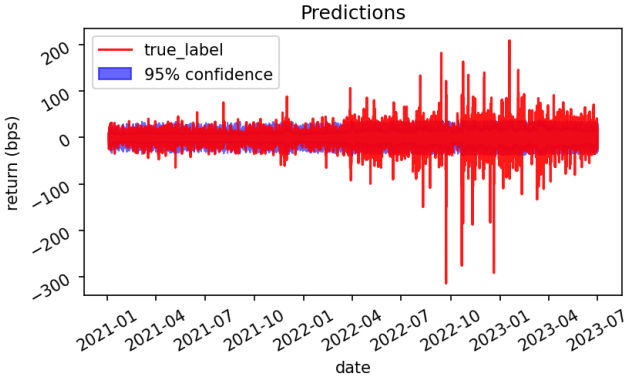


Figure B.4: 95% confidence range for 60min USDJPY return distributions by Transformer-based model

Label thresholds	Proportion	NLL	DA	95% CR	RMSE
0.0	1.0	3.426	0.508	0.916	8.874
0.5	0.909	3.53	0.513	0.908	9.295
1.0	0.821	3.648	0.513	0.898	9.762
1.5	0.739	3.778	0.514	0.887	10.274
2.0	0.666	3.91	0.514	0.875	10.796
2.5	0.595	4.057	0.513	0.861	11.378
3.0	0.534	4.203	0.512	0.846	11.968
3.5	0.479	4.353	0.511	0.83	12.574
4.0	0.43	4.508	0.511	0.813	13.203
4.5	0.386	4.668	0.511	0.796	13.855

Table B.4: Evaluation metrics for USDJPY 30min return distributions by LSTM-based model

Label thresholds	Proportion	NLL	DA	95% CR	RMSE
0.0	1.0	3.311	0.514	0.948	8.702
0.5	0.909	3.389	0.519	0.943	9.127
1.0	0.821	3.478	0.519	0.937	9.597
1.5	0.739	3.577	0.52	0.93	10.112
2.0	0.666	3.678	0.52	0.922	10.637
2.5	0.595	3.792	0.519	0.913	11.222
3.0	0.534	3.907	0.519	0.903	11.813
3.5	0.479	4.024	0.518	0.892	12.421
4.0	0.43	4.145	0.517	0.88	13.053
4.5	0.386	4.272	0.517	0.866	13.705

Table B.5: Evaluation metrics for USDJPY 30min return distributions by Transformer-based model

Hour	Mean	Variance	Skew	Kurtosis	Positive	Negative
0	-0.187	95.307	0.554	-14.853	0.515	0.485
1	0.219	74.333	0.022	0.915	0.509	0.491
2	-0.039	113.146	0.264	-10.335	0.498	0.502
3	0.226	53.535	-1.301	51.624	0.523	0.477
4	0.163	39.176	0.167	4.143	0.537	0.463
5	0.15	33.963	-0.136	1.086	0.534	0.466
6	0.356	48.61	-0.113	2.242	0.536	0.464
7	-0.044	84.841	0.023	-0.265	0.513	0.487
8	-0.178	98.5	0.528	-9.839	0.515	0.485
9	0.137	83.459	-0.547	11.079	0.527	0.473
10	0.14	47.777	-0.014	1.23	0.52	0.48
11	0.191	53.412	0.138	2.69	0.516	0.484
12	0.151	85.632	-0.123	1.646	0.532	0.468
13	0.041	281.874	-0.03	0.9	0.519	0.481
14	-0.073	167.897	0.053	-0.628	0.514	0.486
15	0.183	123.674	-0.265	3.772	0.514	0.486
16	0.186	64.656	-0.339	5.003	0.53	0.47
17	-0.066	45.812	0.058	-0.929	0.512	0.488
18	0.393	49.536	0.205	7.883	0.533	0.467
19	-0.152	46.361	0.582	-6.96	0.518	0.482
20	0.367	25.361	0.206	10.01	0.527	0.473
21	-0.443	27.248	-0.832	-23.868	0.447	0.553
22	1.077	32.819	1.388	22.381	0.632	0.368
23	0.051	37.46	-0.004	0.293	0.524	0.476

Table B.6: Hourly statistical results of USDJPY 30min return (in bps) from 2021 to 2023 (UTC)

B.1.3 USDJPY - 60min forecasting period

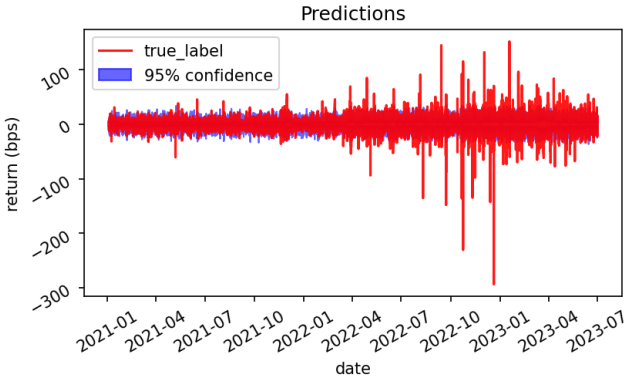


Figure B.5: 95% confidence range for 15min USDJPY return distributions by LSTM-based model

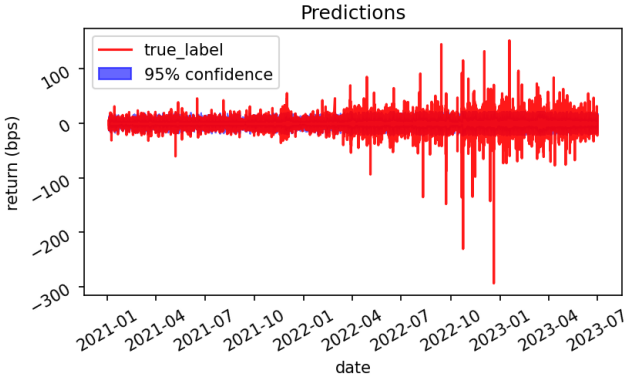


Figure B.6: 95% confidence range for 15min USDJPY return distributions by Transformer-based model

Label thresholds	Proportion	NLL	DA	95% CR	RMSE
0.0	1.0	3.808	0.507	0.906	12.647
0.5	0.936	3.881	0.51	0.899	13.058
1.0	0.873	3.964	0.511	0.892	13.514
1.5	0.81	4.056	0.512	0.884	14.012
2.0	0.753	4.149	0.512	0.875	14.514
2.5	0.698	4.248	0.513	0.865	15.048
3.0	0.647	4.35	0.513	0.855	15.595
3.5	0.6	4.455	0.512	0.844	16.158
4.0	0.557	4.561	0.512	0.832	16.734
4.5	0.515	4.675	0.511	0.819	17.347

Table B.7: Evaluation metrics for USDJPY 60min return distributions by LSTM-based model

Label thresholds	Proportion	NLL	DA	95% CR	RMSE
0.0	1.0	3.69	0.496	0.949	12.413
0.5	0.936	3.74	0.498	0.946	12.827
1.0	0.873	3.796	0.498	0.942	13.286
1.5	0.81	3.859	0.498	0.937	13.787
2.0	0.753	3.923	0.498	0.933	14.292
2.5	0.698	3.992	0.498	0.927	14.828
3.0	0.647	4.064	0.497	0.922	15.379
3.5	0.6	4.138	0.497	0.916	15.945
4.0	0.557	4.213	0.498	0.909	16.523
4.5	0.515	4.294	0.499	0.902	17.137

Table B.8: Evaluation metrics for USDJPY 60min return distributions by Transformer-based model

Hour	Mean	Variance	Skew	Kurtosis	Positive	Negative
0	0.046	160.425	-0.001	0.433	0.522	0.478
1	0.104	150.632	0.073	1.139	0.496	0.504
2	-0.004	221.967	0.026	-0.709	0.511	0.489
3	0.489	100.146	-1.592	49.324	0.535	0.465
4	0.236	68.757	0.006	3.281	0.542	0.458
5	0.393	68.604	-0.398	3.132	0.541	0.459
6	0.478	113.204	-0.36	3.016	0.536	0.464
7	-0.194	186.977	0.094	-1.034	0.508	0.492
8	-0.152	226.753	0.787	-13.599	0.509	0.491
9	0.42	140.99	-1.134	20.854	0.533	0.467
10	0.049	94.154	-0.038	0.526	0.508	0.492
11	0.427	133.025	0.112	3.711	0.528	0.472
12	0.189	350.325	-0.214	3.87	0.528	0.472
13	-0.019	449.05	0.01	-0.299	0.519	0.481
14	0.242	332.374	-0.147	1.556	0.515	0.485
15	0.281	242.22	-0.905	13.192	0.514	0.486
16	0.364	116.717	-1.111	18.704	0.543	0.457
17	-0.14	91.148	0.058	-1.716	0.51	0.49
18	0.738	98.674	0.45	18.686	0.537	0.463
19	-0.138	74.962	0.45	-4.003	0.534	0.466
20	0.822	47.52	-0.22	24.693	0.564	0.436
21	-0.596	50.597	-1.099	-21.859	0.406	0.594
22	1.577	74.021	1.477	19.818	0.615	0.385
23	0.093	96.001	-0.18	3.221	0.53	0.47

Table B.9: Hourly statistical results of USDJPY 60min return (in bps) from 2021 to 2023 (UTC)

Bibliography

[1] Bank for International Settlements. Otc foreign exchange turnover in april 2022. *BIS Triennial Central Bank Survey 2022*, 2022.

[2] Salman Ahmed, Saeed-Ul Hassan, Naif Radi Aljohani, and Raheel Nawaz. Flf-lstm: A novel prediction system using forex loss function. *Applied Soft Computing*, 97:106780, 2020.

[3] Michael Ayitey Junior, Peter Appiahene, Obed Appiah, and Christopher Ninfaakang Bombie. Forex market forecasting using machine learning: Systematic literature review and meta-analysis. *Journal of Big Data*, 10(1):9, 2023.

[4] Christian González Rojas and Molly Herman. Foreign exchange forecasting via machine learning. *Obtenido de http://cs229. stanford. edu/proj2018/report/76. pdf*, 2018.

[5] Manaswinee Madhumita Panda, Surya Narayan Panda, and Prasant Kumar Pattnaik. Exchange rate prediction using ann and deep learning methodologies: A systematic review. In *2020 Indo–Taiwan 2nd International Conference on Computing, Analytics and Networks (Indo-Taiwan ICAN)*, pages 86–90. IEEE, 2020.

[6] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[7] Amit R Nagpure. Prediction of multi-currency exchange rates using deep learning. *International Journal of Innovative Technology and Exploring Engineering*, 8(6):316–322, 2019.

[8] Svitlana Galeshchuk and Sumitra Mukherjee. Deep networks for predicting direction of change in foreign exchange rates. *Intelligent Systems in Accounting, Finance and Management*, 24(4):100–110, 2017.

[9] Yiqi Zhao and Matloob Khushi. Wavelet denoised-resnet cnn and lightgbm method to predict forex rate of change. In *2020 International Conference on Data Mining Workshops (ICDMW)*, pages 385–391. IEEE, 2020.

[10] Pedro Escudero, Willian Alcocer, and Jenny Paredes. Recurrent neural networks and arima models for euro/dollar exchange rate forecasting. *Applied Sciences*, 11(12):5658, 2021.

[11] Alexander Jakob Dautel, Wolfgang Karl Härdle, Stefan Lessmann, and Hsin-Vonn Seow. Forex exchange rate forecasting using deep recurrent neural networks. *Digital Finance*, 2:69–96, 2020.

[12] Ching-I Lee, Chia-Hui Chang, and Feng-Nan Hwang. Currency exchange rate prediction with long short-term memory networks based on attention and news sentiment analysis. In *2019 International Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pages 1–6. IEEE, 2019.

[13] Md Saiful Islam and Emam Hossain. Foreign exchange currency rate prediction using a gru-lstm hybrid network. *Soft Computing Letters*, 3:100009, 2021.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[15] OpenAI. ChatGPT - Release Notes. "https://help.openai.com/en/articles/6825453-chatgpt-release-notes", 2023.

[16] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32, 2019.

[17] Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Anomaly transformer: Time series anomaly detection with association discrepancy. *arXiv preprint arXiv:2110.02642*, 2021.

[18] Chao-Han Huck Yang, Yun-Yun Tsai, and Pin-Yu Chen. Voice2series: Reprogramming acoustic models for time series classification. In *International conference on machine learning*, pages 11808–11819. PMLR, 2021.

[19] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021.

[20] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

[21] Lukas Gonon. Deep learning lecture note. page 18, 2022.

[22] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[23] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[24] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[25] Phillip Lippe. UvA Deep Learning Tutorials. `https://uvadlc-notebooks.readthedocs.io/en/latest/`, 2022.

[26] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[27] Ioannis Karatzas and Steven E. Shreve. *Brownian motion and stochastic calculus*, volume 113. Springer Science & Business Media, 1991.

[28] Yulong Lu and Jianfeng Lu. A universal approximation theorem of deep neural networks for expressing probability distributions. *Advances in neural information processing systems*, 33:3094–3105, 2020.

[29] Przemysław Grądzki and Piotr Wójcik. Is attention all you need for intraday forex trading? *Expert Systems*, page e13317, 2023.

[30] BENOIT MANDELBROT. The variation of certain speculative prices. *The Journal of Business*, 36(4):394–419, 1963.

[31] HistData.com. `https://www.histdata.com/`.