Icon

Description automatically generated

**PROPOSAL TUGAS AKHIR – EF234702**

***ADAPTIVE CYCLIC LEARNING RATE* PADA ARSITEKTUR *DUAL NET TRANSFORMER* UNTUK *LONG-TERM TIME SERIES FORECASTING***

**FARRELA RANKU MAHHISA**

NRP 5025211129

Dosen Pembimbing 1

**Ir. Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D., IPM.**

NIP 198106202005011003

Dosen Pembimbing 2

**Ratih Nur Esti Anggraini, S.Kom., M.Sc., Ph.D.**

NIP 198412102014042003

**Program Studi Teknik Informatika**

Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2024

LEMBAR PENGESAHAN

***ADAPTIVE CYCLIC LEARNING RATE* PADA ARSITEKTUR DUAL NET *TRANSFORMER* UNTUK *LONG-TERM* *TIME SERIES* *FORECASTING***

**PROPOSAL TUGAS AKHIR**

Diajukan untuk memenuhi salah satu syarat

memperoleh gelar Sarjana Komputer pada

Program Studi S-1 Teknik Informatika

Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Oleh : **FARRELA RANKU MAHHISA**

NRP. 5025211129

Disetujui oleh Tim Penguji Proposal Tugas Akhir :

|  |  |  |
| --- | --- | --- |
| 1. | Ir. Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D., IPM. | Pembimbing |
| 2. | Ratih Nur Esti Anggraini, S.Kom., M.Sc., Ph.D. | Ko-pembimbing |
| 3. | Nama dan gelar penguji | Penguji |
| 4. | Nama dan gelar penguji | Penguji |
| 5. | Nama dan gelar penguji | Penguji |

**SURABAYA**

**Desember, 2024**

ABSTRAK

***ADAPTIVE CYCLIC LEARNING RATE* PADA ARSITEKTUR DUAL NET *TRANSFORMER* UNTUK *LONG-TERM* *TIME SERIES* *FORECASTING***

|  |  |  |
| --- | --- | --- |
| **Nama Mahasiswa / NRP** | **:** | **Farrela Ranku Mahhisa / 5025211129** |
| **Departemen** | **:** | **Teknik Informatika FTEIC - ITS** |
| **Dosen Pembimbing 1** | **:** | **Ir. Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D., IPM.** |
| **Dosen Pembimbing 2** | **:** | **Ratih Nur Esti Anggraini, S.Kom., M.Sc., Ph.D.** |

**Abstrak**

Data *time series* adalah kumpulan data yang diperoleh dari pengamatan berurutan dalam interval waktu tertentu, yang sering digunakan untuk menganalisis pola dan tren jangka panjang. Model *Transformer* telah menunjukkan kemampuan unggul dalam menangani ketergantungan data jangka panjang melalui mekanisme *self-attention*. Namun, tantangan besar seperti fluktuasi data dan tingkat *noise* yang tinggi masih memengaruhi akurasi model, terutama dalam memproses *long-term time series*. Untuk mengatasi tantangan ini, penelitian sebelumnya telah mengembangkan *Extended Dual network*, sebuah kerangka kerja *deep learning* dengan dua komponen utama: *fast learner* untuk mempelajari data berlabel dan *slow learner* untuk memahami data secara umum melalui *self-supervised learning*. Penelitian ini bertujuan untuk meningkatkan kinerja *Extended Dual network* dengan menerapkan metode *Adaptive Cyclic Learning Rate* (ACLR), sebuah teknik optimasi yang secara dinamis mengatur *learning rate* selama proses pelatihan. ACLR memungkinkan model untuk beradaptasi lebih baik terhadap dinamika data, mempercepat konvergensi, dan mengurangi risiko *overfitting*. Penelitian ini mencakup desain arsitektur *dual network*, penerapan ACLR, serta pengukuran evaluasi kinerjanya menggunakan *Mean Absolute Error* (MAE) dan *Mean Squared Error* (MSE) terhadap model *Transformer* lainnya, seperti *Autoformer, Informer, FEDFormer, dan Robformer*, menggunakan *dataset* publik seperti cuaca, nilai tukar uang, dan ETTh. Hasil dari penelitian ini diharapkan memberikan kontribusi signifikan dalam menciptakan model *deep learning* yang lebih akurat untuk memprediksi deret waktu jangka panjang.

**Kata kunci: Adaptif, *dual network*, *learning rate*, prediksi deret waktu, *transformer*.**

ABSTRACT

***ADAPTIVE CYCLIC LEARNING RATE ON DUAL NET TRANSFORMER ARCHICTECTURE FOR LONG-TERM TIME SERIES FORECASTING***

|  |  |  |
| --- | --- | --- |
| **Student Name / NRP** | **:** | **Farrela Ranku Mahhisa / 5025211129** |
| **Department** | **:** | **Teknik Informatika FTEIC - ITS** |
| **Advisor 1** | **:** | **Ir. Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D., IPM.** |
| **Advisor 2** | **:** | **Ratih Nur Esti Anggraini, S.Kom., M.Sc., Ph.D.** |

**Abstract**

Time series data is a collection of data obtained from consecutive observations within a certain time interval, which is often used to analyze long-term patterns and trends. The Transformer model has demonstrated superior ability in handling long-term data dependencies through its self-attention mechanism. However, major challenges such as data fluctuations and high noise levels still affect the accuracy of the model, especially in processing long-term time series. To address these challenges, previous research has developed Extended Dual network, a deep learning framework with two main components: a fast learner for learning labeled data and a slow learner for understanding data in general through self-supervised learning. This research aims to improve the performance of the Extended Dual network by applying the Adaptive Cyclic Learning Rate (ACLR) method, an optimization technique that dynamically adjusts the learning rate during the training process. ACLR allows the model to better adapt to data dynamics, accelerate convergence, and reduce the risk of overfitting. This research covers the design of the dual network architecture, the implementation of ACLR, and measure its performance evaluation using Mean Absolute Error (MAE) and Mean Squared Error (MSE) against other Transformer models, such as Autoformer, Informer, FEDFormer, and Robformer, using public datasets such as weather, money exchange rate, and ETTh. The results of this study are expected to make a significant contribution to creating a more accurate deep learning model to predict long-term time series.

***Key*words: *Adaptive, dual network, learning rate, time series forecasting, transformer.***

DAFTAR ISI

[LEMBAR PENGESAHAN i](#_Toc184747636)

[ABSTRAK ii](#_Toc184747637)

[ABSTRACT iii](#_Toc184747638)

[DAFTAR ISI iv](#_Toc184747639)

[DAFTAR GAMBAR vi](#_Toc184747640)

[DAFTAR TABEL vii](#_Toc184747641)

[DAFTAR PSEUDOCODE viii](#_Toc184747642)

[BAB 1 PENDAHULUAN 1](#_Toc184747643)

[1.1 Latar Belakang 1](#_Toc184747644)

[1.2 Rumusan Masalah 1](#_Toc184747645)

[1.3 Batasan Masalah 2](#_Toc184747646)

[1.4 Tujuan 2](#_Toc184747647)

[1.5 Manfaat 2](#_Toc184747648)

[BAB 2 TINJAUAN PUSTAKA 3](#_Toc184747649)

[2.1 Hasil Penelitian Terdahulu 3](#_Toc184747650)

[2.2 Dasar Teori 4](#_Toc184747651)

[2.2.1 Long-Term Time Series Forecasting 4](#_Toc184747652)

[2.2.2 Deep Learning 5](#_Toc184747653)

[2.2.3 *Adaptive Cyclic Learning Rate* 5](#_Toc184747654)

[2.2.4 *Transformer* 6](#_Toc184747655)

[2.2.5 *iTransformer* 8](#_Toc184747656)

[2.2.6 Dual *Network* 10](#_Toc184747657)

[2.2.7 MANTRA : *Extended* *Dual Network* 10](#_Toc184747658)

[2.2.8 Metrik Evaluasi 11](#_Toc184747659)

[BAB 3 METODOLOGI 13](#_Toc184747660)

[3.1 Analisis Dengan Penelitian Tugas Akhir Serupa 13](#_Toc184747661)

[3.2 Metode Yang Digunakan 13](#_Toc184747662)

[3.2.1 *Dataset* 14](#_Toc184747663)

[3.2.2 Preprocessing Data 16](#_Toc184747664)

[3.2.3 Pemisahan Data 16](#_Toc184747665)

[3.2.4 Pelatihan Model 17](#_Toc184747666)

[3.3 Pengembangan Model Penelitian 17](#_Toc184747667)

[3.3.1 *Fast learner* dan *Slow learner* 17](#_Toc184747668)

[3.3.2 Mekanisme *Adaptive Cyclic Learning Rate* 19](#_Toc184747669)

[3.3.3 ACLR Pada *Dual network* 20](#_Toc184747670)

[3.4 Evaluasi dan Perbandingan 22](#_Toc184747671)

[3.5 Spesifikasi Environment 23](#_Toc184747672)

[3.5.1 Perangkat Keras 23](#_Toc184747673)

[3.5.2 Perangkat Lunak 23](#_Toc184747674)

[3.6 Jadwal Pelaksanaan Penelitian 23](#_Toc184747675)

[DAFTAR PUSTAKA 24](#_Toc184747676)

DAFTAR GAMBAR

[Gambar 2.1 Arsitektur Transformer 7](#_Toc184747677)

[Gambar 2.2 Mekanisme Attention pada Transformer 8](#_Toc184747678)

[Gambar 2.3 Perbandingan arsitektur Transformer (atas) dan iTransformer (bawah) 9](#_Toc184747679)

[Gambar 2.4 Arsitektur DualNet 10](#_Toc184747680)

[Gambar 2.5 Arsitektur MANTRA 11](#_Toc184747681)

[Gambar 3.1 Alur penelitian 14](#_Toc184747682)

[Gambar 3.2 Distribusi nilai dari dataset Exchange Rate 15](#_Toc184747683)

[Gambar 3.3 Plot nilai dari dataset Exchange Rate 15](#_Toc184747684)

[Gambar 3.4 Alur preprocessing data 16](#_Toc184747685)

[Gambar 3.5 Mekanisme ACLR 20](#_Toc184747686)

[Gambar 3.6 ACLR pada arsitektur Dual Net 21](#_Toc184747687)

DAFTAR TABEL

[Tabel 2.1 Hasil penelitian terdahulu 3](#_Toc184747688)

[Tabel 3.1 Komparasi dengan penelitian tugas akhir serupa 13](#_Toc184747689)

[Tabel 3.2 Dataset yang digunakan 14](#_Toc184747690)

[Tabel 3.3 Panjang waktu yang digunakan dalam penelitian 17](#_Toc184747691)

[Tabel 3.4 Rencana jadwal penelitian 23](#_Toc184747692)

DAFTAR PSEUDOCODE

[Pseudocode 3.1 Algoritma MANTRA 18](#_Toc184747693)

[Pseudocode 3.2 Algoritma ACLR pada Dual Network 22](#_Toc184747694)

*[Halaman ini sengaja dikosongkan]*

# PENDAHULUAN

## Latar Belakang

Dalam pembelajaran mesin, salah satu jenis *input* data yang dapat digunakan adalah data *time series*. Data *time series* atau deret waktu adalah kumpulan data yang diperoleh dari pengamatan berurutan berdasarkan interval waktu tertentu. Data *time series* digunakan untuk menganalisis pola dan tren yang terjadi dari waktu ke waktu. Hasil analisis *time series* dapat membantu berbagai pekerjaan manusia, misalkan seperti memprediksi harga saham, memprediksi temperatur udara atmosfer, dan lain-lain (Fulcher et al., 2013).

Memproses data *time series* yang bersifat jangka panjang (*long-term* *time series*) memerlukan model pembelajaran mesin yang mampu menangani kompleksitas dan ketergantungan antar data dalam rentang waktu yang sangat panjang. Sebagai model yang memiliki mekanisme “*self-attention*”, *Transformer* (Vaswani et al., 2017) dapat menangkap hubungan berjenis temporal yang rumit dengan ketergantungan jangka panjang antar data (Wen et al., 2022). Berbagai penelitian telah dilakukan dengan menjadikan model *Transformer* sebagai model dasar untuk mengembangkan model-model lain yang mampu menghasilkan akurasi yang lebih baik. Beberapa contohnya adalah *Informer* (H. Zhou et al., 2021), *Autoformer* (Wu et al., 2021), *FEDFormer* (T. Zhou et al., 2022), *Robformer* (Yu et al., 2024) dan model lainnya. Meskipun begitu, masih ada tantangan yang besar ketika model dihadapkan dengan *long-term* *time series*. *Time series* yang bersifat fluktuatif dan memiliki *noise* tinggi seringkali memengaruhi akurasi dari model.

Salah satu hasil pengembangan model *Transformer* dalam memproses data *time series* adalah *Extended* *Dual Network* (Ma’sum et al., 2024). *Extended* *dual network* merupakan kerangka kerja *deep learning* yang dirancang untuk memungkinkan model belajar secara berkelanjutan dari aliran data baru tanpa harus melupakan informasi sebelumnya (Ma’sum et al., 2024). Kerangka kerja ini terdiri dari dua komponen utama, yaitu *fast learner* dan *slow learner*. *Slow learner* bertugas mempelajari representasi data secara umum menggunakan metode *self*-*supervised* *learning*, sedangkan *fast learner* fokus mempelajari data baru yang sudah berlabel menggunakan metode *supervised* *learning* (Ma’sum et al., 2024).

Dengan berkembangnya model *Transformer* (Vaswani et al., 2017), diperlukan penelitian lebih lanjut untuk mengoptimasi kinerjanya. Salah satu pendekatan yang dapat digunakan adalah *Adaptive Cyclic Learning Rate* (ACLR). ACLR merupakan metode optimasi yang mengatur *learning rate* secara adaptif selama proses training. Mengingat *learning rate* memiliki peran penting dalam *neural network*, khususnya dalam menentukan kecepatan konvergensi model (Jepkoech et al., 2021), metode ini menawarkan potensi besar untuk meningkatkan performa dalam *time series* *forecasting*. Oleh karena itu, penelitian ini bertujuan untuk mengimplementasikan ACLR pada *extended* *dual network* guna mengoptimalkan prediksi dalam *long-term* *time series*.

## Rumusan Masalah

Rumusan masalah yang diangkat pada penelitian ini adalah :

1. Bagaimana membuat desain arsitektur *extended* *dual network* untuk *time series forecasting*?
2. Bagaimana menerapkan *Adaptive Cyclic Learning Rate* ke arsitektur *extended* *dual network* untuk *time series* *forecasting*?
3. Bagaimana evaluasi kinerja dari arsitektur *dual network* dengan *Adaptive Cyclic* *Learning Rate* terhadap model *Transformer* lain seperti *Autoformer, Informer, dan FEDformer* dalam *long-term* *time series* *forecasting*?.

## Batasan Masalah

Batasan masalah pada penelitian ini adalah sebagai berikut :

1. Penelitian ini menggunakan *Mean Absolute Error* dan *Mean Squared Error* sebagai metrik evaluasi dalam menentukan performa model.

2. Perbandingan performa uji coba *dataset* difokuskan hanya pada prediksi deret waktu multivariat.

3. *Dataset* yang digunakan dalam penelitian ini diambil dari sumber *dataset* publik yang sering digunakan pada penelitian terkait model *Transformer*, yakni *dataset* cuaca (weather *dataset*), nilai tukar uang (Exchange rate) dan ETTh (Electricity *Transformer* Temperature and Humidity). *Dataset* ini tidak divalidasi ulang oleh penulis, melainkan digunakan sebagaimana adanya dari sumber aslinya.

4. Arsitektur yang digunakan dalam penelitian adalah *dual network* dan terbatas pada model berupa *iTransformer*.

## Tujuan

Tujuan yang ingin dicapai pada penelitian ini adalah sebagai berikut:

1. Membangun arsitektur *dual network* dan menerapkan *Adaptive Cyclic Learning Rate* untuk optimasi arsitektur *dual network*.
2. Mengevaluasi pengaruh penerapan *Adaptive Cyclic Learning Rate* terhadap peningkatan kinerja arsitektur *dual network* dalam *long-term* *time series* *forecasting.*

## Manfaat

Manfaat pada penelitian ini adalah sebagai berikut:

1. Mendesain arsitektur *dual network* dengan mengoptimasi menggunakan *Adaptive Cyclic Learning Rate* pada *long-term* *time series* *forecasting.*
2. Membantu menciptakan model *deep learning* dengan hasil prediksi yang lebih baik.

# 

# TINJAUAN PUSTAKA

## Hasil Penelitian Terdahulu

Dalam pelaksanaan tugas akhir ini, hasil dari penelitian-penelitian sebelumnya dikumpulkan untuk dijadikan referensi dan pembanding selama proses pengujian. Informasi tersebut dirangkum dalam Tabel 2.1. Selain itu, sumber-sumber informasi juga diperoleh dari jurnal dan yang memiliki relevansi dengan topik tugas akhir ini, sehingga dapat memberikan landasan teoritis yang kokoh untuk mendukung penelitian yang dilakukan.

Tabel 2.1 Hasil penelitian terdahulu

|  |  |
| --- | --- |
| **Judul** | **Hasil Penelitian** |
| *Autoformer: Decomposition Transformers with Auto-Correlation for Long-term Series Forecasting*  (Wu et al., 2021) | *Autoformer* menggunakan *Deep Decomposition Architecture* untuk memisahkan komponen tren dan musiman untuk memahami pola jangka panjang dengan lebih baik. *Autoformer* juga mengenalkan mekanisme *Auto-Correlation,* berfungsi untuk menangkap dan mengidentifikasi pola periodik dan menggabungkan informasi dengan kompleksitas *log-linear*. Penelitian ini menunjukkan keunggulan *Autoformer* yang berhasil mencapai kinerja *state-of-the-art* dengan peningkatan relatif sebesar 38% pada enam *dataset* utama. |
| *iTransformer: Inverted Transformers Are*  *Effective For Time series Forecasting* (Y. Liu et al., 2024) | *iTransformer* dirancang menggunakan arsitektur *Transformer* yang terbalik (*inverted* *Transformer*). Perancangan arsitektur *iTransformer* memanfaatkan mekanisme *attention* *Transformer* menggunakan desain *encoder*-*only*. Setiap deret waktu di*token*isasi terlebih dahulu untuk menggambarkan karakteristik masing-masing variabel. Hasil penelitian menunjukkan bahwa dibandingkan dengan *vanilla* *Transformer* biasa, *iTransformer* mampu mereduksi metrik MSE dan MAE pada salah satu *dataset* hinga 86.1% dan 70.4% |
| *Dynamic Long-term Time-Series Forecasting via Meta Transformer Networks* (Ma’sum et al., 2024) | Penelitian ini memperkenalkan *Meta-Transformer Networks* (MANTRA) yang menggabungkan *slow learners* untuk menghasilkan representasi umum dengan *fast learners* yang cepat beradaptasi terhadap pola data yang berubah-ubah. Model ini juga menerapkan *Universal Representation Transformer* (URT), lapisan berbasis *self-attention* yang menggabungkan representasi dari beberapa *learners* untuk melakukan seleksi atau kombinasi representasi. Model MANTRA diuji menggunakan empat *dataset* dengan berbagai variasi prediksi, dan hasilnya menunjukkan peningkatan akurasi setidaknya 3% dibandingkan model biasa. |
| *AutoCyclic: Deep learning Optimizer for Time Series Data Prediction* (Arthur et al., 2024) | Penelitian ini mengembangkan metode optimasi *learning rate* bernama *AutoCyclic*. *AutoCyclic* menggunakan *Auto-Correlation* dan variansi untuk mengontrol *learning rate* secara dinamis. Hasil pengembangan *AutoCylic* menggunakan *Deep* *Transformer*, LSTM, and RNN menunjukkan bahwa pendekatan ini mengungguli metode optimasi tradisional seperti CLR dan *Adam Optimizer*. *AutoCyclic* menghasilkan skor MAPE dan MAE yang lebih rendah dibanding skenario lainnya. Meskipun begitu, *AutoCyclic* membutuhkan sumber daya komputasi yang lebih tinggi dan validasi pada *dataset* lain. |
| Pola: *Online* *Time Series* *Prediction* *by* *Adaptive* *Learning Rate*s (Zhang, 2021) | POLA (*Predicting Online by Learning Rate Adaptation*) adalah metode inovatif yang mengoptimalkan *learning rate* adaptif melalui pendekatan *meta-learning*, dengan memanfaatkan pembagian batch data menjadi *meta-training* dan *meta-validation*. Saat dites menggunakan dua *dataset* dengan tiga arsitektur RNN, LSTM, dan GRU, POLA menunjukkan hasil kompetitif dengan skor RMSE sebesar 0,5 dibandingkan *Pre-trained* and FTL RNN dengan skor 0,572 |

## Dasar Teori

### Long-Term Time Series Forecasting

Deret waktu (*time series*) adalah kumpulan observasi kuantitatif yang disusun berdasarkan urutan kronologis (Kirchgässner et al., 2012). Konsep ini pertama kali diperkenalkan oleh Jan Tinbergen pada tahun 1939 melalui model ekonometrik untuk Amerika Serikat, yang menjadi dasar penelitian empiris di bidang ini (Kirchgässner et al., 2012). Misalkan titik waktu saat pengukuran adalah *T*, dan pengamatan pada waktu tertentu *T* adalah maka himpunan semua pengamatan ini kita sebut sebagai *time series*, yang dapat ditulis sebagai (Parzen, 1961).

*Time series* *forecasting* atau peramalan deret waktu adalah teknik untuk memprediksi nilai di masa depan berdasarkan data historis yang terurut menurut waktu. Tujuan utama dari peramalan deret waktu adalah untuk memahami pola dan tren yang ada dalam data. Peramalan deret waktu juga berfungsi mengidentifikasi faktor-faktor yang dapat mempengaruhi prediksi di masa depan. Misalkan kita memiliki data deret waktu seperti , maka prediksi nilai di masa depan adalah . Angka *h* menunjukkan seberapa jauh ke depan prediksi ingin dilakukan, yang disebut sebagai *forecasting horizon*. Prediksi nilai , yang dibuat berdasarkan informasi hingga waktu *N*, dituliskan sebagai. Dengan kata lain, kita menggunakan data sebelumnya untuk memperkirakan nilai pada langkah-langkah waktu berikutnya hingga *h* (Chatfield, 2000).

Peramalan deret waktu dengan nilai *h* yang relatif besar biasa disebut dengan *Long-Term* *Time Series* *Forecasting*. Peramalan ini bertujuan memprediksi data di masa depan untuk jangka waktu yang lebih panjang dibandingkan metode biasa. Misalkan dengan *dataset* *time series* dari hari pertama hingga hari ke-, , kita harus memprediksi nilai ke depan selama kurang lebih 2 tahun, , dengan 720. Tantangan *long-term* *time series* *forecasting* lebih besar karena semakin jauh waktu yang diprediksi, semakin sulit pola data ditebak, terutama jika ada perubahan tren atau gangguan yang tak terduga. Oleh karena itu, diperlukan model yang mampu mengenali pola jangka panjang dan hubungan antar data dengan baik.

Berbagai metode telah dikembangkan untuk melakukan *time series forecasting*, yang secara umum dapat dibagi menjadi metode statistik, pembelajaran mesin, dan *deep learning*. Metode statistik seperti ARIMA (*Auto Regressive Integrated Moving Average*) (Shumway & Stoffer, 2017) menjadi salah satu pendekatan klasik yang sering digunakan karena kemampuannya menangkap pola stasioner dalam data. Metode pembelajaran mesin dan *deep learning* menjadi lebih umum digunakan karena dapat menangani pola kompleks dan non-linear. Pembelajaran mesin menjadi dasar untuk beberapa metode seperti *Support Vector Regression* (SVR) (Awad, 2015), *Random Forest (Breiman, 2001)*, dan *Gradient Boosting (Friedman, 2001)*. Di sisi lain, metode berbasis *deep learning* seperti RNN (*Recurrent Neural networks*) (Medsker et al., 2001), LSTM (*Long Short-Term Memory*) (Hochreiter, 1997), dan GRU (*Gated Recurrent Unit*) (Cho et al., 2014) juga dikembangkan untuk menyelesaikan permasalah *long-term* *time series*. Model terbaru seperti *Transformers (Vaswani et al., 2017)* menunjukkan hasil yang cukup baik dalam menangkap hubungan jangka panjang tanpa ketergantungan pada data secara berurutan (Wen et al., 2022).

### Deep Learning

*Deep learning* merupakan cabang kecerdasan buatan yang menjadi terobosan signifikan dalam dunia *machine learning*. Konsep *deep learning* berkembang dari pemahaman mendalam tentang bagaimana sistem saraf biologis manusia memproses informasi. Hal ini menginspirasi untuk menciptakan model komputasi yang mampu belajar dan beradaptasi secara kompleks (Rosenblatt, 1958).

*Perceptron*, *neural network* sederhana, memiliki fungsi untuk mengklasifikasikan sebuah *input* menjadi sebuah output (Du et al., 2022) seperti yang dijelaskan pada Persamaan 2.1. *Perceptron* menggunakan fungsi aktivasi untuk menghitung jumlah setiap *input* pada iterasi ke- yang dikalikan dengan bobot , dengan bias , untuk menghasilkan output .

|  |  |
| --- | --- |
|  | (2.1) |

Dimana :

* **:** Output
* **:** Fungsi Aktivasi
* **:** Bias
* **:** *Input*
* **:** Bobot
* **:** Jumlah total fitur

### *Adaptive Cyclic Learning Rate*

*Adaptive Cyclic Learning Rate* (ACLR) adalah pendekatan inovatif yang menggabungkan prinsip *learning rate* adaptif dengan *learning rate* siklikal untuk meningkatkan pelatihan model pembelajaran mesin, khususnya dalam *time series* *forecasting*. Konsep *cyclical* *learning rate*s pertama kali diperkenalkan untuk membantu *neural network* dari *local minima* selama pelatihan (Smith, 2017). Penelitian menunjukkan bahwa pelatihan dengan *cyclical* *learning rate* dapat meningkatkan akurasi klasifikasi sambil mengurangi kebutuhan tuning sehingga menghasilkan hasil yang lebih baik dalam iterasi yang lebih sedikit (Smith, 2017).

Dalam *time series* *forecasting*, Zhang memperkenalkan POLA (*Predicting Online by Learning Rate Adaptation*), yang secara otomatis mengatur *learning rate* pada RNN untuk menyesuaikan pola deret waktu yang berubah (Zhang, 2021). Metode POLA ini menunjukkan pentingnya adaptive *learning rate* dalam skenario *online* *machine learning*, di mana model harus terus memperbarui parameter berdasarkan data baru. Metode yang diusulkan Guo, *adaptive gradient learning* untuk RNN, mampu melakukan *time series* *forecasting* di tengah anomali data dan *change points* (Guo et al., 2016).

Penelitian lain menemukan bahwa *cyclical* *learning rate*s dengan metode *Exponential Moving Average* (EMA) dapat meningkatkan kinerja model *deep learning* secara signifikan pada segmentasi gambar tumor otak (Fajar et al., 2023). *Learning rate* siklik ini membantu model mengeksplorasi lebih banyak kemungkinan parameter dan mencapai hasil konvergensi yang lebih optimal. Seperti yang terlihat pada Persamaan 2.2, EMA pada waktu ke- dihitung dengan memperhatikan nilai EMA sebelumnya, dan nilai *input* terbaru . Nilai ​ pada waktu ke- dihitung dengan mengalikan nilai sebelumnya dengan faktor dan kemudian menambahkan hasil perkalian antara nilai *input* dengan faktor peluruhan ​. Faktor peluruhan ini menentukan seberapa besar pengaruh nilai *input* terbaru dalam perhitungan EMA.

|  |  |
| --- | --- |
|  | (2.2) |
| Dimana :   * : *learning rate* pada waktu ke- * : nilai *smoothing*, dengan rentang * : EMA pada waktu ke- |  |

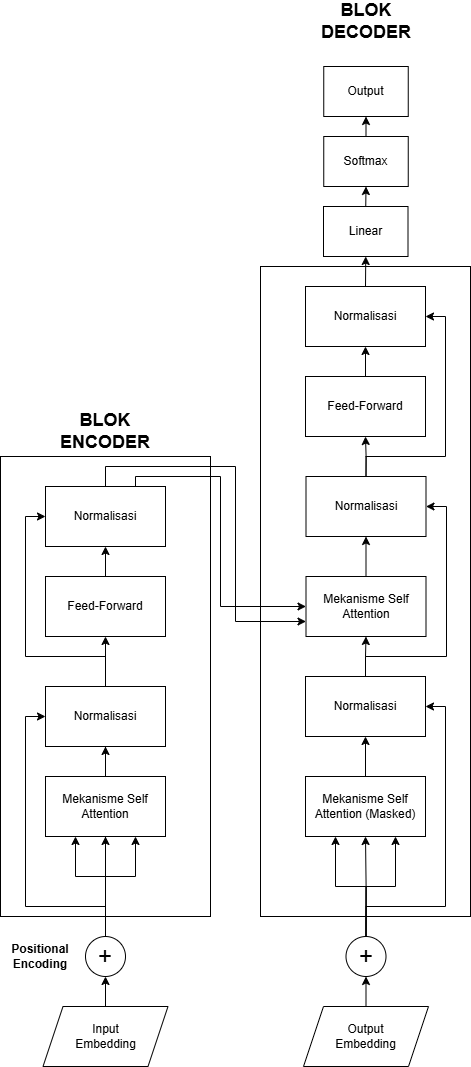
### *Transformer*

Model *Transformer* pertama kali diperkenalkan dalam jurnal *“Attention is All You Need”* (Vaswani et al., 2017), adalah sebuah arsitektur *neural network* yang dirancang untuk memproses data berurutan. *Transformer* memanfaatkan dua komponen utama, yaitu *encoder* dan *decoder*.

*Transformer* memperkenalkan mekanisme *self-attention* sebagai inti dari proses pembelajaran representasi data. Mekanisme ini memungkinkan model untuk menangkap hubungan atau keterkaitan antar elemen dalam sebuah deret data secara langsung, tanpa memperhatikan posisi sekuensial seperti yang dilakukan oleh arsitektur sebelumnya, seperti RNN (Medsker et al., 2001)dan LSTM (Hochreiter, 1997). Hal ini membuat *Transformer* lebih efisien dalam menangani data berurutan dengan panjang yang besar karena tidak tergantung pada perhitungan secara berurutan. Dapat dilihat pada Gambar 2.1, dua komponen *Transformer*, *encoder* dan *decoder* bekerja secara berpasangan untuk mempelajari representasi data dan menghasilkan prediksi (Vaswani et al., 2017).

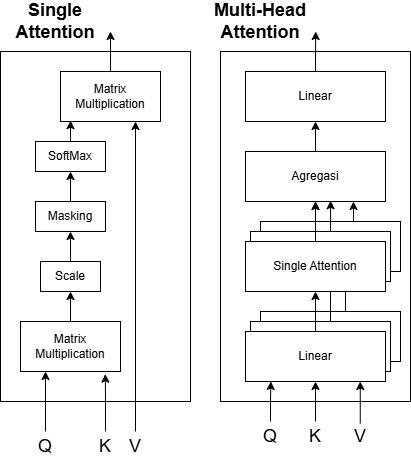
*Encoder* bertugas memproses deret *input* yang telah diubah menjadi representasi vektor melalui proses *embedding*. *Encoder* terdiri 6 lapisan, di mana setiap lapisan memiliki 2 sublapisan. Lapisan pertama adalah mekanisme *multi-head* *attention* yang berfungsi menangkap hubungan antar elemen dalam deret. Setelah itu, hasilnya diteruskan ke lapisan kedua, yakni blok *feed-forward* untuk mempelajari fitur representasi data. Proses normalisasi, lapisan Add dan *Layer* Norm, diterapkan pada output setelah setiap dari masing-masing dua sub*layer* untuk menjaga stabilitas.

*Decoder* menerima *input* berupa vektor hasil *encoder* untuk digenerate menjadi , yakni pergeseran dari . *Decoder* juga terdiri dari 6 lapisan identik, tetapi dilengkapi dengan sublapisan *masked* *multi-head* *attention*. *Masking* berfungsi untuk memastikan bahwa *decoder* hanya memperhitungkan elemen sebelumnya dalam deret saat memprediksi elemen berikutnya. Selanjutnya, *decoder* menggunakan informasi dari *encoder* melalui mekanisme cross-*attention*, di mana *query* berasal dari *decoder* sementara *key* dan *value* berasal dari *encoder*.



Gambar 2.1 Arsitektur Transformer

Seperti terlihat pada Gambar 2.2 dan Persamaan 2.3, mekanisme inti dari *Transformer* adalah *attention*, yang membuat model dapat memproses hubungan antar elemen dalam deret dengan cara memetakan sebuah *query* (Q) ke pasangan *key* (K) dan *value* (V), di mana *query*, *key*, dan *value* adalah vektor yang merepresentasikan elemen data. Bobot (*weight*) yang digunakan untuk setiap *value* dihitung berdasarkan kesesuaian (*similarity*) antara *query* dan *key* yang sama.



Gambar 2.2 Mekanisme Attention pada Transformer

|  |  |
| --- | --- |
|  | (2.3) |

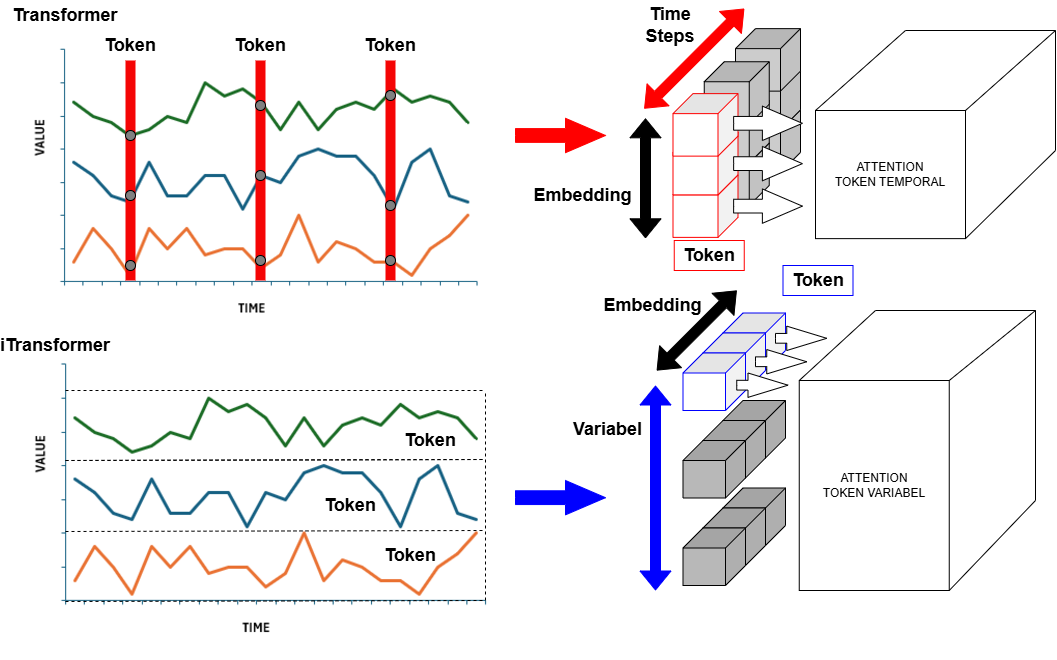
Dimana :

* : *Query*
* : *Key*
* : *Value*
* : Dimensi dari *Key*

### *iTransformer*

Model *Inverted* *Transformer* (*iTransformer*) pertama kali dikemukan oleh Liu, dimana model ini dirancang untuk menangani tugas prediksi deret waktu multivariat dengan pendekatan yang lebih adaptif dan efisien dibandingkan model *Transformer* standar (Y. Liu et al., 2024). Model ini memproses data dengan terlebih dahulu mengubah deret waktu menjadi representasi vektor melalui fungsi *embedding*. Representasi ini kemudian diolah melalui beberapa lapisan *Transformer* block (TrmBlock) yang terdiri dari mekanisme *self-attention* dan jaringan *feed-forward*. Hasil akhir dari lapisan *Transformer* ini diproyeksikan untuk menghasilkan prediksi masa depan . Tidak seperti *Transformer* biasanya, *iTransformer* tidak menggunakan *positional* *embedding* karena urutan data sudah secara alami ditangkap oleh struktur neuron *feed-forward* yang digunakan.

Pada Gambar 2.3, dapat dilihat *iTransformer* menggunakan pendekatan yang berbeda dengan tidak menggabungkan semua variabel di satu titik waktu menjadi satu *token* seperti *Transformer* (Vaswani et al., 2017). Sebagai gantinya, model ini mengubah masing-masing deret waktu dari setiap variabel menjadi *token* terpisah yang disebut "*variate* *token*" atau “*token* variabel”. Pendekatan ini membantu model untuk lebih memahami pola secara menyeluruh pada tiap variabel dan memanfaatkan mekanisme *attention* dengan lebih baik untuk menangkap hubungan antar variabel. Selain itu, jaringan *feed-forward* pada *iTransformer* dirancang untuk mengenali pola-pola yang dapat digunakan secara umum dari data deret waktu dan membuat prediksi yang lebih akurat (Y. Liu et al., 2024).



Gambar 2.3 Perbandingan arsitektur Transformer (atas) dan iTransformer (bawah)

Dalam memprediksi setiap deret waktu dari variabel diubah menjadi representasi *token* awal menggunakan fungsi *embedding* sesuai dengan Persamaan 2.4.

|  |  |
| --- | --- |
|  | (2.4) |

Pada Persamaan 2.5, dapat dilihat bahwa *token* hasil *embedding* diproses secara iteratif melalui lapisan *Transformer* block, di mana merupakan keluaran dari lapisan ke-

|  |  |
| --- | --- |
|  | (2.5) |

Selanjutnya pada Persamaan 2.6, setelah melalui semua lapisan, *token* terakhir diproyeksikan untuk menghasilkan prediksi .

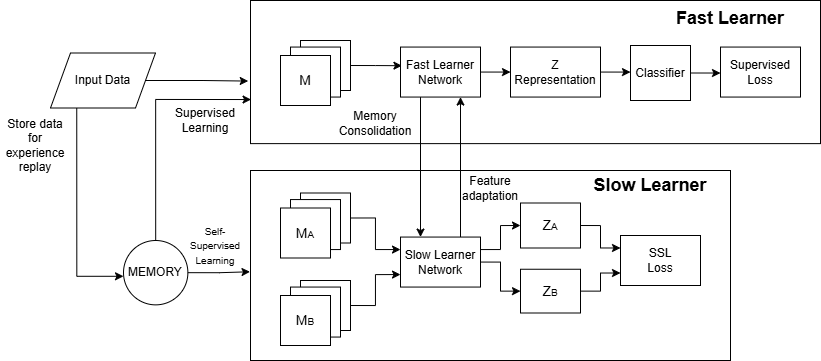
|  |  |
| --- | --- |
|  | (2.6) |

Pada *iTransformer*, *token* merepresentasikan variabel dengan dimensi , yang dihasilkan melalui proses *embedding* dan *projection* yang dilakukan oleh *Multi-Layer Perceptron* (MLP). *Embedding* mengubah data deret waktu menjadi representasi berdimensi , sementara *projection* mengubahnya menjadi prediksi dengan dimensi . *Token*-*token* ini saling berinteraksi menggunakan mekanisme *self-attention* untuk memahami hubungan antar variabel, kemudian diproses melalui blok *feed-forward*.

### Dual *Network*

*Dual network* atau *DualNet* adalah arsitektur *neural network* yang dirancang untuk menghadapi tantangan *continual* *learning*, yaitu kemampuan model untuk mempelajari tugas-tugas baru secara berurutan tanpa kehilangan informasi dari tugas sebelumnya (Pham et al., 2021). Pertama kali dikemukakan pada 2021, pendekatan utama yang digunakan dalam *DualNet* adalah *neural network* dengan strategi pembelajaran fast dan slow. *Fast learner* *network* bertugas untuk mempelajari tugas-tugas baru dengan cepat, sehingga mampu beradaptasi dengan informasi baru dalam waktu singkat. Sementara *slow learner* *network* dilatih secara perlahan dan bertahap. *Slow learner* berfungsi sebagai penyimpan memori jangka panjang untuk menjaga pengetahuan dari tugas-tugas sebelumnya agar tidak terlupakan.

Proses belajar *DualNet* terjadi dalam dua fase yang berjalan secara bersamaan, yakni fase *Self*-*Supervised* *Learning* (SSL) dan fase *Supervised* *Learning*. Pada fase SSL, *slow learner* menggunakan data tanpa label dari memori episodik (*episodic memory*) untuk mengoptimalkan tujuan pembelajaran SSL. Episodic memory berfungsi untuk menyimpan *subset* data yang dipelajari dan akan diselingi saat mempelajari data baru (Lopez-Paz & Ranzato, 2017). Pada Gambar 2.4, ketika data berlabel baru diterima, *Fast learner* menggunakan representasi yang dihasilkan *Slow learner* untuk menyesuaikan diri dan mempelajari data tersebut. Kesalahan yang muncul selama proses ini akan diperbaiki melalui *backpropagation* ke kedua modul (Slow dan *Fast learner*).

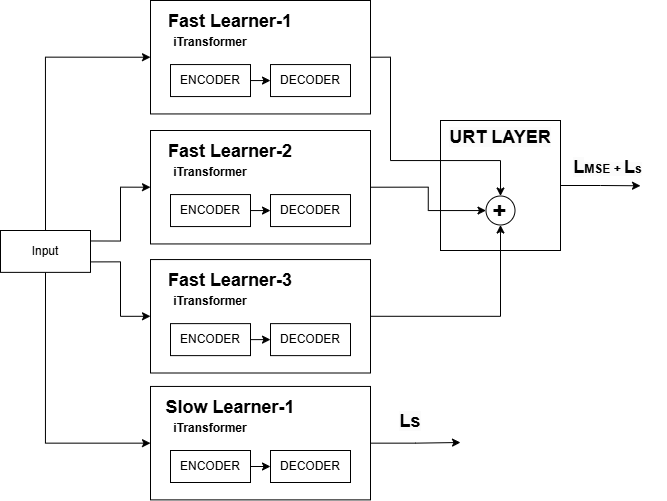


Gambar 2.4 Arsitektur DualNet

### MANTRA : *Extended* *Dual Network*

Meta-*Transformer* *Network* (MANTRA) adalah versi pengembangan lebih lanjut dari *DualNet* (Ma’sum et al., 2024). MANTRA menggunakan gabungan dari satu buah jaringan *slow learner* dan tiga buah jaringan *fast learner* (Ma’sum et al., 2024). Pada MANTRA, jaringan *slow learner* memiliki tujuan untuk mengenalkan *noise* pada jaringan parameter *fast learner*. Dapat dilihat pada Gambar 2.5, MANTRA menggunakan *Universal Representation* *Transformer* (URT) (L. Liu et al., 2020) untuk mengatasi perubahan data yang tidak menentu. URT membantu menciptakan pola yang sesuai untuk setiap tugas dengan menggunakan sedikit pengaturan, sehingga proses konvergen model menjadi lebih cepat (L. Liu et al., 2020).

MANTRA juga menggunakan *controlled reconstruction strategy* untuk *slow learner* dibandingkan menggunakan strategi *Barlow Twins* (Zbontar et al., 2021), karena strategi ini memerlukan lebih sedikit komputasi dan tidak bergantung pada augmentasi data. Pembelajaran *slow learner* dilakukan dengan meminimalkan fungsi *loss* *masked* menggunakan data deret waktu tanpa label. Fungsi *loss* keseluruhan menggabungkan dengan *loss* *unmasked* menggunakan parameter pengatur . Selain itu, strategi regularisasi juga diterapkan untuk mencegah model terlalu terfokus pada *subset* data yang sama di setiap *epoch*, sehingga mengurangi risiko *overfitting*.



Gambar 2.5 Arsitektur MANTRA

### Metrik Evaluasi

Metrik evaluasi digunakan untuk menilai seberapa baik model prediksi bekerja setelah proses pelatihan, terutama dalam konteks prediksi deret waktu. Metrik ini membantu mengukur tingkat kesalahan antara nilai aktual dan hasil prediksi model. Dengan menggunakan metrik evaluasi kita dapat membandingkan performa berbagai model (Saigal S & Mehrotra D, 2012). Dalam penelitian ini, digunakan dua metrik evaluasi yang umum, yaitu *Mean Absolute Error* (MAE) dan *Mean Squared Error* (MSE). MAE mengukur rata-rata perbedaan absolut antara nilai aktual dan prediksi sesuai dengan Persamaan 2.7, sehingga mudah dipahami dan memberikan gambaran langsung tentang seberapa besar rata-rata kesalahan model dalam satuan data asli. MSE, di sisi lain, menghitung rata-rata kuadrat dari kesalahan seperti pada Persamaan 2.8 sehingga lebih sensitif terhadap kesalahan besar yang penting untuk meminimalkan dampak prediksi yang meleset jauh pada jangka panjang. Dalam prediksi deret waktu jangka panjang, penggunaan MAE dan MSE memberikan gambaran lengkap tentang kinerja model, di mana MAE menunjukkan rata-rata kesalahan secara keseluruhan, sementara MSE membantu mendeteksi dan mengatasi kesalahan besar yang dapat memengaruhi analisis dan keputusan.

|  |  |
| --- | --- |
|  | (2.7) |

Dimana sebagai nilai aktual, nilai prediksi, dan jumlah observasi.

|  |  |
| --- | --- |
|  | (2.8) |

Dimana dan memiliki definisi yang sama seperti pada MAE.

# METODOLOGI

## Analisis Dengan Penelitian Tugas Akhir Serupa

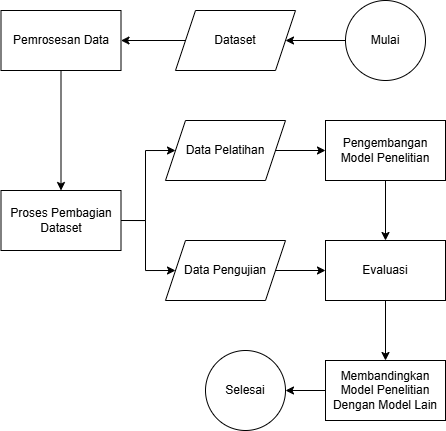
Penelitian tugas akhir ini mengadopsi arsitektur dual network dari MANTRA (Ma’sum et al., 2024). Pada MANTRA, terdapat banyak potensi untuk mengoptimasi modelnya seperti jenis model yang digunakan pada *learner*, penggunaan *learning rate* yang adaptif, dan aspek lainnya. Oleh karena itu, terdapat penelitian tugas akhir lain yang juga sama-sama mengadopsi arsitektur yang sama tetapi dengan fokus yang berbeda. Uraian komparasi dengan penelitian tugas akhir yang serupa dapat dilihat pada Tabel 3.1.

Tabel 3.1 Komparasi dengan penelitian tugas akhir serupa

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nama** | **Judul** | **Penjelasan Singkat** | **Persamaan** | **Perbedaan** |
| Farrela Ranku Mahhisa | Adaptive Cyclic Learning Rate Pada Arsitektur Dual Net Transformer Untuk Long-Term Time Series Forecasting | Mengoptimasi arsitektur *dual network* yang diadopsi dari MANTRA yang menggunakan blok *iTransformer* pada kedua *learner*, *fast learner* dan *slow learner,* dengan menerapkan *Adaptive Cyclic Learning Rate* untuk mengatur *learning rate* secara dinamis berdasarkan progres pelatihan dalam rangka meningkatkan kinerja model dalam prediksi deret waktu jangka panjang. | Mengadopsi arsitektur *dual network* dari MANTRA dan menggunakan *iTransformer* pada blok *slow learner* | Berfokus pada penggunaan ACLR pada setiap blok *learner.* |
| Andika Laksana Putra | Peramalan Deret Waktu Jangka Panjang Menggunakan Arsitektur Dualnet Transformer dengan Pendekatan Fast Learner (FRnet) dan Slow Learner (iTransformer) | Mengembangkan model ensemble deep learning dengan arsitektur DualNet pada pendekatan Fast Learner menggunakan model Frequency-based Rotation Network (FRNet) dan Slow Learner menggunakan iTransformer. FRNet berperan dalam membaca pola temporal dan menangkap dependensi global pada domain frekuensi. iTransformer bekerja secara self-supervised learning untuk membantu fast learners memberikan representasi data secara berhala sehingga dapat beradaptasi terhadap noise yang tinggi. | Berfokus pada penggunaan FRNet pada blok *fast learner.* |

## Metode Yang Digunakan

Seperti yang dijelaskan pada Gambar 3.1, penelitian dimulai dengan pengumpulan *dataset*, yang kemudian melalui tahap pemrosesan data untuk memastikan data siap digunakan. Selanjutnya, *dataset* dibagi menjadi dua bagian: data pelatihan dan data pengujian. Data pelatihan digunakan untuk mengembangkan model penelitian, sementara data pengujian digunakan untuk mengevaluasi performa model yang telah dibangun. Setelah model dievaluasi, hasilnya dibandingkan dengan model lain untuk menentukan keunggulan atau kelemahan model yang dikembangkan.



Gambar 3.1 Alur penelitian

### *Dataset*

Dalam penelitian ini, digunakan *dataset* publik yang sering dipakai dalam *long-term* *time series* *forecasting*. *Dataset* ini merupakan *dataset* yang diambil dari kasus dunia nyata, yakni data cuaca (Wu et al., 2021), data nilai tukar mata uang (Lai et al., 2018), dan data penggunaan tenaga listrik (H. Zhou et al., 2021). Untuk lebih jelas, detail mengenai *dataset* dapat dilihat pada Tabel 3.2.

Tabel 3.2 Dataset yang digunakan

|  |  |  |
| --- | --- | --- |
| ***Dataset*** | **Jumlah fitur** | **Penjelasan** |
| *Weather* | 21 | Data cuaca yang dicatat setiap 10 menit sepanjang tahun 2020. *Dataset* ini mencakup 21 indikator meteorologi, seperti suhu udara, kelembapan, dan variabel cuaca lainnya |
| *Exchange rate* | 8 | Nilai tukar harian dari delapan negara yang berbeda dalam rentang waktu dari tahun 1990 hingga 2016. |
| *ETTh* | 7 | *Dataset* ini mencatat data yang dikumpulkan dari transformator listrik, termasuk beban listrik dan suhu oli. Data ini dicatat setiap 15 menit |

Sebagai langkah awal analisis, dilakukan eksplorasi data awal (*Exploratory Data Analysis*) untuk memahami karakteristik *dataset*. EDA dilakukan secara khusus pada dataset nilai tukar mata uang (*Exchange rate*). Pada Gambar 3.2, hasil distribusi nilai tukar dari delapan negara menunjukkan pola yang bervariasi, memberikan indikasi adanya dinamika yang unik pada masing-masing negara. Dapat dilihat pada Gambar 3.3, dataset *Exchange rate* memberikan gambaran temporal data yang mencerminkan pergerakan nilai tukar dalam rentang waktu tertentu.

A group of blue and black bars

Description automatically generated

Gambar 3.2 Distribusi nilai dari dataset Exchange Rate

A graph of a graph

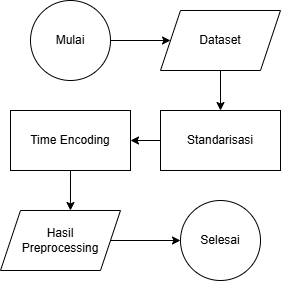
Description automatically generated with medium confidence

Gambar 3.3 Plot nilai dari dataset Exchange Rate

Dalam penelitian ini, masing-masing *dataset* dimodelkan secara terpisah. Hal ini dilakukan untuk menjaga keunikan karakteristik masing-masing *dataset*, seperti perbedaan domain dan skala waktu pengambilan data. Dengan pemodelan terpisah, dapat dihasilkan model yang lebih spesifik dan akurat sesuai dengan pola dari setiap dataset. Pendekatan ini juga mempermudah analisis performa prediksi untuk tiap *dataset* secara individu. Berdasarkan distribusi pada gambar, tampak bahwa rentang nilai pada beberapa kolom berbeda secara signifikan. Misalnya, kolom 5 memiliki rentang nilai yang jauh lebih kecil dibandingkan dengan kolom lainnya. Selanjutnya, kolom 1 hingga 4 memiliki distribusi yang lebih terpusat, tetapi tetap menunjukkan variasi yang berbeda. Beberapa kolom lainnya memiliki pola yang cukup unik dan tidak seragam. Hal ini menunjukkan bahwa skala fitur tidak konsisten di seluruh dataset. Dalam kasus seperti ini, standarisasi perlu dilakukan untuk menyelaraskan skala data sehingga model tidak memberikan bobot lebih besar pada fitur dengan rentang nilai yang lebih besar.

### Preprocessing Data

Alur lengkap pemrosesan *dataset* sebelum memasuki pelatihan model dapat dilihat pada Gambar 3.4. Tahapan ini bertujuan untuk mempersiapkan data sebelum memasuki tahap pelatihan model yang akan dilatih. Tahap pertama adalah standarisasi data yang dilakukan untuk menyeragamkan skala setiap fitur dalam *dataset* sehingga dapat menghindari bias terhadap fitur dengan nilai skala yang lebih besar. Selanjutnya, data akan melalui tahap *time encoding*, yaitu pemisahan atribut waktu, seperti tanggal atau jam, dari variabel lainnya pada setiap baris data.



Gambar 3.4 Alur preprocessing data

### Pemisahan Data

Setelah tahap praproses selesai, ketiga *dataset* yang digunakan dalam penelitian ini akan dibagi menjadi tiga bagian: data pelatihan, data validasi, dan data pengujian. Data pelatihan digunakan untuk melatih model, data validasi digunakan untuk menyesuaikan parameter model selama pelatihan, dan data pengujian digunakan untuk mengevaluasi kinerja akhir model yang telah dilatih. Pada penelitian ini, data dibagi dengan rasio 70% untuk data pelatihan, 10% untuk data validasi, dan 20% untuk data pengujian.

### Pelatihan Model

Pelatihan model dilakukan untuk menciptakan model yang dapat memprediksi nilai deret waktu ke depan. Model akan menerima 2 argumen, yakni *sequence length* dan *prediction* *length*. *Sequence length* adalah panjang waktu dari *dataset* yang akan dijadikan *input* untuk model dapat belajar. *prediction* *length* adalah panjang waktu yang akan diprediksi oleh model. Untuk panjang waktu yang akan dipakai pada penelitian ini dapat dilihat pada Tabel 3.3.

Tabel 3.3 Panjang waktu yang digunakan dalam penelitian

|  |  |
| --- | --- |
| ***Sequence Length*** | ***Prediction Length*** |
| 96 | 96 |
| 192 |
| 336 |
| 720 |

## Pengembangan Model Penelitian

### *Fast learner* dan *Slow learner*

Penelitian ini mengembangkan model dengan arsitektur *extended* *dual network* dari MANTRA yang terdiri dari tiga komponen *fast learner*s dan satu komponen *slow learner*. Keempat komponen ini digabung untuk mengatasi tantangan dalam prediksi deret waktu jangka panjang yang bersifat dinamis dan kompleks. Tiga *fast learner*s dalam model ini diadaptasi dari arsitektur MANTRA. Berdasarkan analisis sensitivitas yang dilakukan oleh pengembang MANTRA, tiga *fast learner*s dianggap jumlah yang paling optimal untuk mendukung performa model (Ma’sum et al., 2024). Komponen *slow learner*, yang juga diadaptasi dari MANTRA, hanya terdiri dari satu unit. Komponen ini berperan penting dalam meningkatkan ketahanan (*robustness*) model dengan memberikan representasi data secara berkala kepada *fast learner*s, sehingga membantu model beradaptasi pada data dengan tingkat *noise* yang tinggi. Dalam penelitian ini, arsitektur *fast learner* dan *slow learner* menggunakan *iTransformer* (Y. Liu et al., 2024), yang memiliki desain *encoder*-*only* sederhana dan ringan secara komputasi. Algoritma Mantra dapat dilihat pada Pseudocode 3.1.

|  |
| --- |
| **Algoritma MANTRA**  ***Input*:**  - *Fast learner* , *Slow learner* , lapisan URT , jumlah *epoch* , jumlah batch *dataset* pelatihan  **Output:**  - Parameter Jaringan yang diperbarui  #Langkah 1: Latih *Fast learner* dan *Slow learner*  for : do :  for do :  - = sampel minibatch dari *dataset* pelatihan  - Hitung *loss*  - Perbarui parameter pembelajar cepat  - Hitung dan hasilkan  - Hitung *loss* berdasarkan persamaan (3.1)  - Perbarui parameter pembelajar lambat ) berdasarkan  - Perbarui parameter pembelajar cepat berdasarkan +  end for  end for  Bekukan Parameter *Fast learner*  #Langkah 2: Latih URT  for : do :  for do :  - sampel minibatch dari *dataset* pelatihan  - Hitung *loss*  - Perbarui parameter  end for  end for |
|  |

Pseudocode 3.1 Algoritma MANTRA

*Controlled Reconstruction Strategy* yang diadopsi dari MANTRA adalah metode untuk menghitung *loss* function pada *slow learner* (Ma’sum et al., 2024). Tujuan dari metode ini adalah untuk meningkatkan ketahanan terhadap data dengan tingkat *noise* yang tinggi. Strategi rekonstruksi ini nantinya akan diterapkan dalam arsitektur model *iTransformer*, yang berfungsi sebagai *slow learner*. Sebelum data dimasukkan ke dalam *slow learner*, data tersebut akan melalui proses *masking*, yang mengubah beberapa nilai pada *timestep* tertentu menjadi nol untuk menambahkan *noise*. Proses ini bertujuan untuk memperbarui parameter jaringan pada *fast learner*. *Masking* dilakukan secara acak pada beberapa *timestep*. Setelah proses *masking*, data yang sudah di*mask* akan diteruskan ke model *iTransformer* untuk dilakukan prediksi. Hasil prediksi tersebut kemudian akan dihitung nilai *loss*-nya. Nilai *loss* yang dihitung adalah kombinasi dari *loss* yang diperoleh dari prediksi data yang telah di*mask* dan yang tidak di*mask*. Rumus perhitungan *loss* *masked* dan *loss* *unmasked* dapat dilihat pada Persamaan 3.1 dan Persamaan 3.2

|  |  |
| --- | --- |
|  | (3.1) |
|  | (3.2) |

Dimana:

|  |  |
| --- | --- |
|  | = Nilai *loss* dari data *mask* |
|  | = Nilai *loss* dari data *unmasked* |
|  | = Tensor mask biner yang bernilai satu atau nol |
|  | = Fungsi prediksi model *slow learner* |
|  | = Data *input* yang tidak di-*mask* |
|  | = Data *input* yang belum di-*mask* |
|  | = Dimensi data *masked* |
|  | = Total sampel *dataset* |

*Loss* yang diperoleh dari data yang di*mask* maupun dari data yang tidak di*mask* dihitung menggunakan metrik evaluasi Mean Square Error (MSE). Kedua nilai *loss* ini kemudian dinormalisasi untuk menghasilkan *loss* dari *slow learner* dengan menggunakan *trade-off* λ. Proses perhitungan ini bertujuan untuk merekonstruksi nilai *loss* baru dengan mempertimbangkan komposisi dalam agregasi kedua nilai *loss* tersebut. Persamaan untuk menghitung nilai *loss* baru dijelaskan dalam Persamaan 3.3

|  |  |
| --- | --- |
|  | (3.3) |

Dimana:

|  |  |
| --- | --- |
|  | = Nilai *loss* dari data *mask.* |
|  | = Nilai *loss* dari data *unmasked.* |

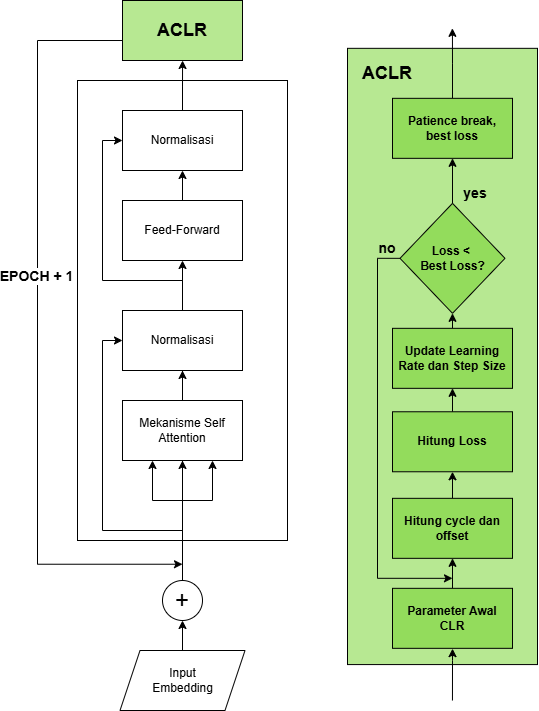
### Mekanisme *Adaptive Cyclic Learning Rate*

Konsep dasar dari ACLR adalah mengoptimalkan proses pelatihan model dengan mengatur *learning rate* secara dinamis berdasarkan progress pelatihan dan jumlah *epoch*. Tidak seperti metode *learning rate* tetap atau konvensional, ACLR menggunakan variasi periodik *learning rate* antara nilai minimum dan nilai maksimum untuk membantu model keluar dari jebakan *local minima* dan mempercepat proses konvergensi ke solusi optimal. Variasi ini dirancang mengikuti pola sinusoidal atau gelombang segitiga untuk memastikan transisi yang baik antara nilai minimum dan maksimum. Dengan pendekatan ini, pada awal pelatihan, *learning rate* yang tinggi memungkinkan model untuk belajar dengan cepat, sementara pada tahap akhir, *learning rate* yang rendah digunakan untuk menyempurnakan hasil pelatihan. Secara matematis, *learning rate* pada *epoch* ke- dihitung menggunakan Persamaan 3.4.

|  |  |
| --- | --- |
|  | (3.4) |

Dimana adalah nilai minimum, adalah nilai maksimum, dan adalah jumlah total *epoch*.

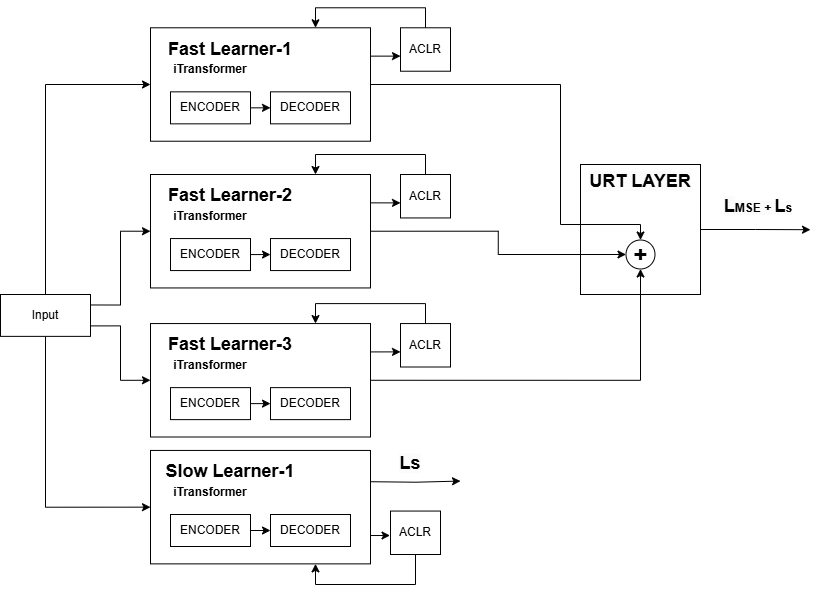
Pada awal pelatihan, *learning rate* dimulai dari nilai yang tinggi untuk mempercepat proses eksplorasi parameter, sehingga model dapat menghindari *local minima*. Secara bertahap, *learning rate* menurun untuk menyempurnakan parameter selama tahap akhir pelatihan. ACLR juga memanfaatkan perubahan siklus berdasarkan performa model, dengan memonitor *loss* secara berkala. Jika *loss* menunjukkan peningkatan yang signifikan, step size dikurangi untuk menjaga kestabilan. Sebaliknya, jika *loss* stagnan atau menurun dengan perlahan, langkah siklus diperbesar untuk meningkatkan kecepatan adaptasi, sehingga model tetap responsif terhadap perubahan pola data. Selain itu, ACLR dilengkapi dengan mekanisme *early stopping* yang akan melakukan penghentian pelatihan ketika *loss* tidak menunjukkan perbaikan dalam sejumlah iterasi tertentu. Early stopping berfungsi untuk mencegah risiko *overfitting*. Fitur ini bekerja bersama *patience counter* yang berfungsi sebagai pengukur toleransi terhadap nilai *loss*. Saat *loss* tidak membaik, nilai *patience counter* menurun, dan pelatihan dihentikan jika nilai tersebut mencapai nol. Sebaliknya, jika *loss* membaik, *patience counter* akan diatur ulang. Mekanisme ACLR dapat dilihat pada Gambar 3.5.



Gambar 3.5 Mekanisme ACLR

### ACLR Pada *Dual network*

ACLR akan digunakan pada setiap blok *learner* dalam arsitektur *dual network*. ACLR dirancang untuk menyesuaikan proses belajar sesuai dengan peran masing-masing blok. Pada *fast learner*, ACLR membantu agar pembelajaran lebih cepat beradaptasi dengan perubahan pola data, menggunakan siklus *learning rate* yang lebih aktif untuk menangkap informasi baru dengan cepat. Sementara itu, pada *slow learner*, ACLR diterapkan dengan pola siklus yang lebih tenang dan stabil untuk fokus pada pola umum dan jangka panjang. Penyesuaian *learning rate* dilakukan secara terpisah pada masing-masing learner, berdasarkan hasil performa seperti nilai *loss* yang didapatkan selama pelatihan. Pendekatan ini memastikan *fast learner* bisa merespon perubahan data dengan cepat, sedangkan *slow learner* tetap konsisten meski ada *noise* atau gangguan dalam data. Implementasi ACLR pada arsitektur *dual network* dapat dilihat pada Gambar 3.6.



Gambar 3.6 ACLR pada arsitektur Dual Net

Dalam implementasi ACLR ke arsitektur *dual network*, pada setiap *epoch*, data diambil dalam bentuk batch dari *dataset* pelatihan, lalu *fast learner* menghitung *loss* menggunakan *Mean Squared Error* (MSE). Berdasarkan nilai *loss* ini, *learning rate* pada *fast learner* disesuaikan menggunakan ACLR, sehingga dapat mempercepat proses pembelajaran pada fase awal dan memperhalus penyempurnaan model pada fase akhir. Proses serupa dilakukan pada *slow learner*, di mana *learning rate* disesuaikan berdasarkan *loss* spesifiknya untuk meningkatkan adaptabilitas pembelajaran. Setelah parameter keduanya diperbarui, *fast learner* juga menggabungkan *loss* dari *slow learner* untuk memperbaiki hasil pembelajaran secara keseluruhan. Setelah pelatihan selesai, parameter *fast learner* dibekukan, dan model melanjutkan pelatihan pada URT. Algoritma implementasi ACLR pada *dual network* dapat dilihat pada Pseudocode 3.2

|  |
| --- |
| **Algoritma ACLR Pada Arsitektur *Dual Network***  for *epoch* in range():  for batch in range():  data\_batch = get\_batch(*dataset*)  *loss*\_mse = compute\_mse\_*loss*(fast\_learner, data\_batch)  # ACLR Pada *Fast learner*  lr\_fast = adjust\_*learning*\_rate(*loss*\_mse, lr\_min, lr\_max, cycle, step\_size)  update\_parameters(fast\_learner, lr\_fast)  *loss*\_s = compute\_slow\_learner\_*loss*(slow\_learner, data\_batch)  # ACLR Pada *Slow learner*  lr\_slow = adjust\_*learning*\_rate(*loss*\_s, lr\_min, lr\_max, cycle, step\_size)  update\_parameters(slow\_learner, lr\_slow)  update\_parameters(fast\_learner, lr\_fast, *loss*\_mse + *loss*\_s)  freeze\_parameters(fast\_learner)  for *epoch* in range():  for batch in range():    data\_batch = get\_batch(*dataset*)  *loss*\_urt = compute\_mse\_*loss*(urt, data\_batch)  update\_parameters(urt, *learning*\_rate, *loss*\_urt) |

Pseudocode 3.2 Algoritma ACLR pada Dual Network

## Evaluasi dan Perbandingan

Model penelitian yang akan dikembangkan untuk prediksi deret waktu jangka panjang dengan model uji coba lainnya akan dilakukan evaluasi menggunakan metrik MAE dan MSE. Metode perbandingan dilakukan dengan membandingkan model ini terhadap beberapa model Transformer terkini, seperti Autoformer (Wu et al., 2021), Informer (H. Zhou et al., 2021), dan FEDformer (T. Zhou et al., 2022). Selain itu, model yang akan dikembangkan juga akan dibandingkan dengan model berarsitektur dual network tanpa implementasi ACLR

Autoformer menggunakan mekanisme *auto-correlation* untuk menggantikan *self-attention* tradisional, sehingga model menangkap pola berulang secara lebih efisien tanpa memerlukan kompleksitas komputasi tinggi (Wu et al., 2021). Pendekatan ini mengurangi *noise* dalam data dan lebih fokus pada komponen frekuensi yang relevan, menjadikannya unggul dalam menangkap pola musiman dan tren. Informer, di sisi lain, mengusung mekanisme *probSparse self-attention* yang memprioritaskan informasi penting dengan mengabaikan interaksi yang kurang signifikan (H. Zhou et al., 2021). Hal ini meningkatkan efisiensi pemrosesan untuk dataset berskala besar dan berdimensi tinggi. Sementara itu, FEDformer (Frequency Enhanced Decomposed Transformer) mengintegrasikan transformasi frekuensi untuk membagi data menjadi komponen *low-frequency* dan *high-frequency* (T. Zhou et al., 2022). Strategi ini akan membuat model menangkap pola makro dan mikro secara lebih efektif dan memberikan performa yang tinggi dalam memprediksi data yang sangat fluktuatif. Perbandingan kinerja dilakukan untuk mengevaluasi bagaimana pendekatan adaptif ACLR pada dual network dapat melampaui keunggulan masing-masing model Transformer tersebut dalam tugas long-term time series forecasting.

## Spesifikasi Environment

Untuk mendukung uji coba performa model penelitian, digunakan beberapa peralatan yang dapat dilihat sebagai berikut:

### Perangkat Keras

1. 12th Gen Intel(R) Core(TM) i9-12900K (24 CPUs), ~3.2GHz

2. Random Access Memory (RAM) 64 GB

3. GeForce® RTX 3080 Ti with 8GB

4. Solid State Drive (SSD) 1 TB

5. Mouse dan keyboard

### Perangkat Lunak

1. Sistem Operasi Windows 11

2. Visual Studio Code

3. Python beserta library pendukung penelitian

## Jadwal Pelaksanaan Penelitian

Jadwal pelaksanaan penelitian yang akan dilakukan oleh penulis dapat dilihat dengan detail pada Tabel 3.4

Tabel 3.4 Rencana jadwal penelitian

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No | Kegiatan | Minggu ke | | | | | | | | | | | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 1 | Studi pustaka |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 | Eksplorasi *dataset* |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 | Eksplorasi model |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 | Mengembangkan model |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 | Pelaporan kemajuan |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 6 | Eksperimen pada model |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 7 | Penyusunan laporan |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 8 | Laporan disetujui |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 9 | Pengumpulan laporan |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

DAFTAR PUSTAKA

Arthur, C., Yudistira, N., & Dewi, C. (2024). AutoCyclic: Deep Learning Optimizer for Time Series Data Prediction. *IEEE Access*, *12*, 14014–14026. https://doi.org/10.1109/ACCESS.2024.3356553

Awad, M. , K. R. (2015). *Efficient Learning Machines*.

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Chatfield, C. (2000). *TIME-SERIES FORECASTING*.

Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). *On the Properties of Neural Machine Translation: Encoder–Decoder Approaches*.

Du, K.-L., Leung, C.-S., Mow, W. H., & Swamy, M. N. S. (2022). Perceptron: Learning, Generalization, Model Selection, Fault Tolerance, and Role in the Deep Learning Era. *Mathematics*, *10*(24). https://doi.org/10.3390/math10244730

Fajar, A., Sarno, R., Fatichah, C., Susilo, R. I., & Pangestu, G. (2023). Cyclical Learning Rate Optimization on Deep Learning Model for Brain Tumor Segmentation. *IEEE Access*, *11*, 119802–119810. https://doi.org/10.1109/ACCESS.2023.3326475

Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, *29*(5), 1189–1232. http://www.jstor.org/stable/2699986

Fulcher, B. D., Little, M. A., & Jones, N. S. (2013). Highly comparative time-series analysis: The empirical structure of time series and their methods. *Journal of the Royal Society Interface*, *10*(83). https://doi.org/10.1098/rsif.2013.0048

Guo, T., Xu, Z., Yao, X., Chen, H., Aberer, K., & Funaya, K. (2016). Robust online time series prediction with recurrent neural networks. *Proceedings - 3rd IEEE International Conference on Data Science and Advanced Analytics, DSAA 2016*, 816–825. https://doi.org/10.1109/DSAA.2016.92

Hochreiter, S. (1997). Long Short-term Memory. *Neural Computation MIT-Press*.

Jepkoech, J., Mugo, D. M., Kenduiywo, B. K., & Too, E. C. (2021). The Effect of Adaptive Learning Rate on the Accuracy of Neural Networks. In *IJACSA) International Journal of Advanced Computer Science and Applications* (Vol. 12, Issue 8). www.ijacsa.thesai.org

Kirchgässner, G., Wolters, J., & Hassler, U. (2012). *Introduction to modern time series analysis*. Springer Science & Business Media.

Lai, G., Chang, W.-C., Yang, Y., & Liu, H. (2018). Modeling long-and short-term temporal patterns with deep neural networks. *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 95–104.

Liu, L., Hamilton, W., Long, G., Jiang, J., & Larochelle, H. (2020). *A Universal Representation Transformer Layer for Few-Shot Image Classification*.

Liu, Y., Hu, T., Zhang, H., Wu, H., Wang, S., Ma, L., & Long, M. (2024). *iTransformer: Inverted Transformers Are Effective for Time Series Forecasting*.

Lopez-Paz, D., & Ranzato, M. A. (2017). Gradient Episodic Memory for Continual Learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 30). Curran Associates, Inc. https://proceedings.neurips.cc/paper\_files/paper/2017/file/f87522788a2be2d171666752f97ddebb-Paper.pdf

Ma’sum, M. A., Sarkar, M. D. R., Pratama, M., Ramasamy, S., Anavatti, S., Liu, L., Habibullah, H., & Kowalczyk, R. (2024). Dynamic Long-Term Time-Series Forecasting via Meta Transformer Networks. *IEEE Transactions on Artificial Intelligence*, *5*(8), 4258–4268. https://doi.org/10.1109/TAI.2024.3365775

Medsker, L. R., Jain, L., & others. (2001). Recurrent neural networks. *Design and Applications*, *5*(64–67), 2.

Parzen, E. (1961). *An Approach To Time Series Analysis*.

Pham, Q., Liu, C., & Hoi, S. C. H. (2021). *DualNet: Continual Learning, Fast and Slow*. https://github.com/phquang/DualNet.

Rosenblatt, F. (1958). THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN 1. In *Psychological Review* (Vol. 65, Issue 6).

Saigal S, & Mehrotra D. (2012). *PERFORMANCE COMPARISON OF TIME SERIES DATA USING PREDICTIVE DATA MINING TECHNIQUES Advances in Information Mining*. *4*(1), 57–66. http://www.bioinfo.in/contents.php?id=32

Shumway, R. H., & Stoffer, D. S. (2017). *Springer Texts in Statistics Time Series Analysis and Its Applications*. http://www.springer.com/series/417

Smith, L. N. (2017). Cyclical learning rates for training neural networks. *Proceedings - 2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017*, 464–472. https://doi.org/10.1109/WACV.2017.58

Vaswani, A., Brain, G., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). *Attention Is All You Need*.

Wen, Q., Zhou, T., Zhang, C., Chen, W., Ma, Z., Yan, J., & Sun, L. (2022). *Transformers in Time Series: A Survey*. http://arxiv.org/abs/2202.07125

Wu, H., Xu, J., Wang, J., & Long, M. (2021). Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, & J. W. Vaughan (Eds.), *Advances in Neural Information Processing Systems* (Vol. 34, pp. 22419–22430). Curran Associates, Inc. https://proceedings.neurips.cc/paper\_files/paper/2021/file/bcc0d400288793e8bdcd7c19a8ac0c2b-Paper.pdf

Yu, Y., Ma, R., & Ma, Z. (2024). Robformer: A robust decomposition transformer for long-term time series forecasting. *Pattern Recognition*, *153*. https://doi.org/10.1016/j.patcog.2024.110552

Zbontar, J., Jing, L., Misra, I., Lecun, Y., & Deny, S. (2021). *Barlow Twins: Self-Supervised Learning via Redundancy Reduction*. https://github.com/facebookresearch/barlowtwins

Zhang, W. (2021). Pola: Online Time Series Prediction by Adaptive Learning Rates. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, *2021-June*, 3375–3379. https://doi.org/10.1109/ICASSP39728.2021.9414906

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., & Zhang, W. (2021). *Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting*. www.aaai.org

Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., & Jin, R. (2022). *FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting*. https://github.com/MAZiqing/FEDformer.