

BELAJAR MEMBUAT TIME SERIES FORECASTING MENGGUNAKAN DEEP LEARNING



Disusun Oleh:

Mellania Permata Sylvie (180441100101)

Dosen Pembimbing:

Dr. Wahyudi Setiawan, S.Kom., M.Kom.

NIP. 19780926 200604 1 001

**PROGRAM STUDI SISTEM INFORMASI
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS TRUNOJOYO MADURA
2021**

LEMBAR KESEPAKATAN
PEMBUATAN BUKU TUTORIAL

BELAJAR MEMBUAT TIME SERIES FORECASTING
MENGGUNAKAN DEEP LEARNING

1. Nama Mahasiswa : Mellania Permata Sylvie
2. Program Studi : Sistem Informasi
3. Jenis : Buku Tutorial
4. Topik/Judul Buku : Belajar Membuat Time Series Forecasting Menggunakan Deep Learning
5. Outline Buku :
 - Pengenalan Machine Learning, Deep Learning dan Time Series Forecasting
 - Persiapan Praktek
 - Time Series Forecasting dengan Neural Network (NN)
 - Time Series Forecasting dengan Deep Neural Network (DNN)
 - Time Series Forecasting dengan Recurrent Neural Network (RNN)
 - Time Series Forecasting dengan LSTM
 - Time Series Forecasting dengan Real Dataset

Bangkalan, 5 Juni 2021

Pengusul,

Mahasiswa



Mellania Permata Sylvie

NIM. 180441100101

Menyetujui,

Dosen Pembimbing



Dr. Wahyudi Setiawan, S.Kom., M.Kom.

NIP. 19780926 200604 1 001

LEMBAR PENGESAHAN

Telah diperiksa dan diuji oleh Pembimbing

Pada Tanggal :

Dengan Nilai :.....

Menyetujui,

Dosen Pembimbing

Dr. Wahyudi Setiawan, S.Kom., M.Kom.

NIP. 19780926 200604 1 001

Mengetahui

Koordinator Kerja Praktek

Sri Herawati, S.Kom., M.Kom.

NIP. 19830828 200812 2 002

KATA PENGANTAR

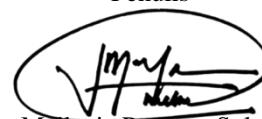
Dengan menyebut nama Allah SWT yang Maha Pengasih lagi Maha Panyayang, Penulis panjatkan puja dan puji syukur atas kehadirat-Nya, yang telah melimpahkan rahmat, hidayah, dan inayah-Nya kepada penulis, sehingga dapat menyelesaikan Buku Tutorial dengan judul “Belajar Membuat Time Series Forecasting Menggunakan Deep Learning”. Penulis juga ucapan terimakasih kepada orang tua, keluarga, dan teman-teman yang telah membantu dan memberi semangat untuk menyelesaikan buku tutorial ini. Tidak lupa ucapan terimakasih kepada Bapak Dr. Wahyudi Setiawan, S.Kom., M.Kom. yang berkenan meluangkan waktunya untuk membimbing saya dalam penyusunan buku tutorial ini, serta terimakasih kepada Ibu Sri Herawati, S.Kom., M.Kom. selaku koordinator kerja praktek yang sudah memberikan saya arahan dalam menyelesaikan buku tutorial ini.

Buku ini menjelaskan mengenai bagaimana cara membuat time series forecasting menggunakan deep learning. Sebagaimana kita tahu penggunaan machine learning banyak di sekitar kita. Oleh karena itu, penulis menyusun buku ini dengan materi yang mudah dipahami dan diperaktikkan. Sehingga diharapkan buku ini dapat membantu mereka yang baru saja mempelajari machine learning terutama mengenai Time Series Forecasting.

Penulis menyadari bahwa buku ini tak luput dari kesalahan dan masih jauh dari kata sempurna. Oleh karena itu, dengan tangan terbuka penulis menerima segala bentuk kritik dan saran terhadap buku ini, sehingga dapat menjadi bahan evaluasi dan lebih baik untuk kedepannya.

Akhir kata, penulis berharap buku tutorial ini bisa memberikan manfaat kepada pembaca.

Bangkalan, 5 Juni 2021

Penulis

Mellania Permata Sylvie

DAFTAR ISI

LEMBAR KESEPAKATAN.....	ii
LEMBAR PENGESAHAN	iv
KATA PENGANTAR	v
DAFTAR ISI	vii
DAFTAR GAMBAR.....	x
BAB 1 PENGENALAN	
1.1 Pengenalan Machine Learning	1
1.1.1. Pengertian.....	1
1.1.2. Jenis-Jenis	2
1.1.3. Contoh Implementasi	3
1.2 Pengenalan Deep Learning.....	5
1.2.1. Pengertian	5
1.2.2. Jenis-Jenis	5
1.3 Pengenalan Time Series Forecasting	7
1.3.1. Pengertian.....	7
1.3.2. Pola Data.....	9
1.3.3. Contoh	11
BAB 2 PERSIAPAN PRAKTEK	
2.1 Alat dan Bahan	12
2.2 Petunjuk Penggunaan Google Colab	12
2.3 Instalasi Library.....	19
BAB 3 TIME SERIES FORECASTING DENGAN NEURAL NETWORK	
3.1 Menyiapkan Fitur dan Label.....	23
3.1.1. Pengertian.....	23
3.1.2. Membagi Feature dan Label.....	24

3.2 Menyiapkan Windowed Dataset.....	30
3.3 Percobaan dengan Single Layer Neural Network.....	32
 BAB 4 TIME SERIES FORECASTING DENGAN DEEP NEURAL NETWORK	
4.1. Konsep Deep Neural Network.....	46
4.2. Percobaan dengan DNN	49
4.3. Memperbaiki Model DNN	54
4.4. Kode Keseluruhan DNN.....	61
 BAB 5 TIME SERIES FORECASTING DENGAN RECURRENT NEURAL NETWORK	
5.1. Konsep RNN	69
5.1.1. Pengertian RNN	69
5.1.2. Cara Kerja RNN	70
5.2. Percobaan dengan RNN	71
5.3. Memperbaiki Model RNN.....	78
5.4. Kode Keseluruhan RNN	82
 BAB 6 TIME SERIES FORECASTING DENGAN LSTM	
6.1. Konsep LSTM	88
6.2. Percobaan dengan LSTM	89
6.3. Memperbaiki Model LSTM	95
6.4. Kode Keseluruhan LSTM.....	98
 BAB 7 TIME SERIES FORECASTING DENGAN REAL DATASET	
7.1. Dataset	104
7.1.1. Pengertian.....	104
7.1.2. Sumber Dataset	104
7.1.3. Sunspots Dataset	106
7.2. Membuat Model	107

7.3. Perbaikan Model.....	118
7.4. Kode Keseluruhan	125
CATATAN	134
DAFTAR PUSTAKA.....	135
PROFIL PENULIS	138

DAFTAR GAMBAR

Gambar 1.1. Contoh Multivariate Time Series	8
Gambar 1.2. Contoh Grafik Pola Trend.....	9
Gambar 1.3. Contoh Grafik Pola Seasonality	9
Gambar 1.4. Contoh Grafik Pola Autocorrelations.....	10
Gambar 1.5. Contoh Grafik Pola Noise	10
Gambar 2.1. Tampilan Search Link Google Colab.....	13
Gambar 2.2. Tampilan Pencarian Google Colab	13
Gambar 2.3. Tampilan Daftar Notebook	14
Gambar 2.4. Tampilan Beranda Google Colab.....	15
Gambar 2.5. Tampilan Pilihan Notebook Baru.....	15
Gambar 2.6. Tampilan Pilihan Buka Notebook	15
Gambar 2.7. Tampilan Pilihan Upload Notebook.....	16
Gambar 2.8. Tampilan Memberi Nama Notebook	16
Gambar 2.9. Tampilan Menambahkan Kode	16
Gambar 2.10. Tampilan Menjalankan Kode	17
Gambar 2.11. Tampilan Menambahkan Teks	17
Gambar 2.12. Tampilan Mengatur Prosesor	18
Gambar 2.13. Tampilan Menghubungkan G-Drive	18
Gambar 2.14. Tampilan Mengupload File	19
Gambar 2.15. Tampilan Menyimpan Notebook	19
Gambar 2.16. Tampilan Logo Tensorflow	20
Gambar 2.17. Tampilan Kode Tensorflow	20
Gambar 2.18. Tampilan Logo NumPy.....	21
Gambar 2.19. Tampilan Kode NumPy	21
Gambar 2.20. Tampilan Logo Matplotlib	22
Gambar 2.21. Tampilan Kode Matplotlib.....	22

Gambar 3.1. Ilustrasi Feature dan Label	23
Gambar 3.2. Contoh Feature dan Label	24
Gambar 3.3. Kode Mengimpor Library	25
Gambar 3.4. Kode Memeriksa Versi Tensorflow	25
Gambar 3.5. Kode Membuat Dataset Sederhana	26
Gambar 3.6. Kode dan Hasil Menambahkan dataset.window	27
Gambar 3.7. Kode Menambahkan drop_remainder	28
Gambar 3.8. Kode Memasukkan Nilai ke NumPy.....	28
Gambar 3.9. Kode Membagi Data	29
Gambar 3.10. Kode Mengacak Data.....	30
Gambar 3.11. Kode Membuat Windowed Dataset	31
Gambar 3.12. Kode Impor Library dan Periksa Versi Tensorflow	32
Gambar 3.13. Kode Membuat Bentuk Data Series	33
Gambar 3.14. Kode Membagi Dataset.....	34
Gambar 3.15. Kode Mengatur Window, Batch, Shuffle Buffer	34
Gambar 3.16. Kode Membuat Windowed Dataset	34
Gambar 3.17. Kode Membuat Model Neural Network.....	35
Gambar 3.18. Hasil Model Summary	35
Gambar 3.19. Kode Compile dan Training Model	36
Gambar 3.20. Hasil Prediksi	36
Gambar 3.21. Kode Visualisasi Prediksi	37
Gambar 3.22. Hasil Visualisasi Prediksi.....	37
Gambar 3.23. Rumus MAE	38
Gambar 3.24. Kode dan Hasil Output MAE.....	38
Gambar 3.25. Kode Keseluruhan Bab 3	39
Gambar 4.1. Ilustrasi Proses DNN.....	47
Gambar 4.2. Ilustrasi Layer DNN.....	48
Gambar 4.3. Kode Impor Library dan Cek Versi Tensorflow	49

Gambar 4.4. Kode Membuat Fungsi Data Series	50
Gambar 4.5. Kode Membagi Dataset	51
Gambar 4.6. Kode Visualisasi Data Series	51
Gambar 4.7. Hasil Visualisasi Data Series	51
Gambar 4.8. Kode Membuat Windowed Dataset	52
Gambar 4.9. Kode Membuat Model DNN	52
Gambar 4.10. Hasil Summary Model DNN	52
Gambar 4.11. Kode Mentraining Model	53
Gambar 4.12. Kode Visualisasi Hasil Prediksi	53
Gambar 4.13. Visualisasi Hasil Prediksi	54
Gambar 4.14. Kode dan Hasil MAE	54
Gambar 4.15. Kode Model Sebelumnya	55
Gambar 4.16. Kode Menambahkan Callback dan Training Model	55
Gambar 4.17. Kode Visualisasi Kecepatan Epoch	56
Gambar 4.18. Hasil Visualiasi Kecepatan Epoch	56
Gambar 4.19. Kode Membuat Model Baru.....	57
Gambar 4.20. Kode Visualisasi Hasil Loss	57
Gambar 4.21. Visualisasi Hasil Loss	58
Gambar 4.22. Kode Visualisasi Hasil Loss Detail.....	58
Gambar 4.23. Visualisasi Hasil Loss Detail	59
Gambar 4.24. Kode Visualisasi Hasil Prediksi	59
Gambar 4.25. Visualisasi Hasil Prediksi	60
Gambar 4.26. Kode dan Hasil MAE	60
Gambar 4.27. Kode Keseluruhan Bab 4	61
Gambar 5.1. Ilustrasi Konsep RNN	70
Gambar 5.2. Cara Kerja RNN.....	71
Gambar 5.3. Kode Impor Library dan Cek Versi Tensorflow	72
Gambar 5.4. Kode Membuat Fungsi Data Series	72

Gambar 5.5. Kode Membagi Dataset	74
Gambar 5.6. Kode Memvisualisasikan Data Series	74
Gambar 5.7. Visualisasi Data Series.....	74
Gambar 5.8. Kode Membuat Windowed Dataset	75
Gambar 5.9. Kode Membuat Model RNN.....	75
Gambar 5.10. Kode Mentraining Model RNN	76
Gambar 5.11. Kode Visualsasi Prediksi Model RNN.....	77
Gambar 5.12. Visualsasi Prediksi Model RNN	77
Gambar 5.13. Kode dan Hasil MAE	78
Gambar 5.14. Kode Mencari Nilai Optimum	78
Gambar 5.15. Hasil Nilai Optimum.....	79
Gambar 5.16. Kode Membuat Perbaikan Model RNN	80
Gambar 5.17. Kode Mentraining Model RNN	80
Gambar 5.18. Kode Visualisasi Hasil Prediksi	80
Gambar 5.19. Visualisasi Hasil Prediksi	81
Gambar 5.20. Kode dan Hasil MAE.....	81
Gambar 5.21. Kode Keseluruhan Bab 5	82
Gambar 6.1. Ilustrasi LSTM	89
Gambar 6.2. Kode Impor Library dan Cek Versi Tensorflow	89
Gambar 6.3. Kode Membuat Fungsi Data Series.....	90
Gambar 6.4. Kode Membagi Dataset.....	91
Gambar 6.5. Kode Membuat Windowed Dataset	91
Gambar 6.6. Kode Membuat Model LSTM.....	92
Gambar 6.7. Kode Mentraining Model.....	93
Gambar 6.8. Kode Visualisasi Hasil Prediksi	93
Gambar 6.9. Visualisasi Hasil Prediksi.....	94
Gambar 6.10. Kode dan Hasil MAE	94
Gambar 6.11. Kode Mencari Nilai Optimum	95

Gambar 6.12. Hasil Nilai Optimum	95
Gambar 6.13. Kode Membuat Model LSTM Baru	96
Gambar 6.14. Kode Mentraining Model LSTM Baru	96
Gambar 6.15. Kode Visualisasi Hasil Prediksi	97
Gambar 6.16. Visualisasi Hasil Prediksi.....	97
Gambar 6.17. Kode dan Hasil MAE	97
Gambar 6.18. Kode Keseluruhan Bab 6	98
Gambar 7.1. Logo Kaggle	105
Gambar 7.2. Logo UCI Machine Learning	105
Gambar 7.3. Logo Google Dataset Search	106
Gambar 7.4. Logo Portal Satu Data Indonesia.....	106
Gambar 7.5. Tampilan Dataset Sunspots Kaggle	106
Gambar 7.6. Kode Impor Library dan Cek Versi Tensorflow	108
Gambar 7.7. Kode Fungsi Grafik Data Series	108
Gambar 7.8. Kode Menghubungkan GDrive	109
Gambar 7.9. Tampilan Kode Autorisasi	109
Gambar 7.10. Tampilan Memilih Akun	110
Gambar 7.11. Tampilan Link Kode Autorisasi	110
Gambar 7.12. Tampilan Mengisi Kode Autorisasi	111
Gambar 7.13. Kode Mengakses Dataset	111
Gambar 7.14. Tampilan Info Dataset	111
Gambar 7.15. Kode dan Isi Dataset	112
Gambar 7.16. Kode Membuka Dataset	112
Gambar 7.17. Visualisasi Dataset Sunspots	113
Gambar 7.18. Kode Membagi Dataset	114
Gambar 7.19. Kode Membuat Windowed Dataset	114
Gambar 7.20. Kode Fungsi Prediksi Model	114
Gambar 7.21. Kode Membuat Model	115

Gambar 7.22. Kode dan Summary Model	116
Gambar 7.23. Kode Mentraining Model.....	116
Gambar 7.24. Kode Visualisasi Hasil Prediksi	117
Gambar 7.25. Visualisasi Hasil Prediksi.....	117
Gambar 7.26. Kode dan Hasil MAE	118
Gambar 7.27. Kode Mencari Nilai Optimum	118
Gambar 7.28. Hasil Grafik Nilai Optimum	118
Gambar 7.29. Kode Membuat Model Baru.....	119
Gambar 7.30. Kode Mentraining Model.....	120
Gambar 7.31. Kode Visualisasi Hasil Prediksi	120
Gambar 7.32. Visualisasi Hasil Prediksi	120
Gambar 7.33. Kode dan Hasil MAE	121
Gambar 7.34. Kode Visualisasi Training Loss	121
Gambar 7.35. Hasil Visualisasi Training Loss	122
Gambar 7.36. Kode Zoom Visualisasi Training Loss	122
Gambar 7.37. Hasil Zoom Visualisasi Training Loss	123
Gambar 7.38. Kode Melihat Hasil Prediksi	123
Gambar 7.39. Hasil Prediksi	124
Gambar 7.40. Kode Keseluruhan Bab 7	125

BAB 1

PENGENALAN

1.1. Pengenalan Machine Learning

Pada era yang canggih ini, istilah machine learning sering kita dengar. Machine learning banyak digunakan dalam keseharian kita. Machine learning telah memberikan pengaruh yang sangat besar dalam kehidupan manusia saat ini. Setelah ini, kita akan membahas mengenai pengertian machine learning.

1.1.1. Pengertian Machine Learning

Machine learning merupakan perpaduan berbagai macam disiplin ilmu pengetahuan yang kompleks. Machine learning dikembangkan berdasarkan disiplin ilmu seperti statistika, matematika dan data mining sehingga mesin dapat belajar dengan menganalisa data tanpa perlu di program ulang atau diperintah.

Istilah machine learning pertama kali dikenalkan oleh Arthur Samuel pada tahun 1959 melalui jurnalnya yang berjudul “Some Studies in Machine Learning Using the Game of Checkers”[1].

Dengan begitu, orang pertama yang memberikan definisi tentang machine learning adalah Arthur Samuel, beliau memberikan definisi bahwa :

“Machine learning is field of study that gives computers the ability to learn without being explicitly programmed”[2].

Berdasarkan definisi di atas, dapat kita ketahui bahwa Machine learning adalah bidang studi yang memberikan komputer kemampuan untuk belajar tanpa diprogram secara eksplisit.

1.1.2. Jenis-Jenis Machine Learning

Machine learning terbagi menjadi beberapa jenis. Berdasarkan karakteristik data dan jenis supervisi yang didapatkan oleh program selama pelatihan, machine learning dibagi ke dalam 4 kategori, diantaranya yaitu :

1) Supervised Learning

Supervised learning adalah kategori machine learning yang menyertakan solusi yang diinginkan (yang disebut label) dalam proses pembelajarannya[3]. Dataset yang digunakan telah memiliki label dan algoritma kemudian mempelajari pola dari pasangan data dan label tersebut. Algoritma supervised learning mudah dipahami dan performa akurasinya pun mudah diukur.

Salah satu contoh dari supervised learning yaitu mesin/robot yang belajar menjawab pertanyaan sesuai dengan jawaban yang telah disediakan manusia (Chatbot).

2) Unsupervised Learning

Unsupervised learning merupakan kategori machine learning yang melakukan proses belajar sendiri untuk melabeli atau mengelompokkan data[3]. Jadi dataset yang digunakan tidak memiliki label. Contoh dari Unsupervised learning yaitu robot/mesin yang berusaha belajar menjawab pertanyaan secara mandiri tanpa ada jawaban yang disediakan manusia.

3) Semi-Supervised Learning

Semi-supervised learning merupakan gabungan dari supervised learning dan unsupervised learning. Dataset untuk pelatihan sebagian memiliki label dan

sebagian tidak[3]. Salah satu teknologi yang menggunakan Semi-supervised learning adalah Google Photos.

4) Reinforcement Learning

Reinforcement Learning merupakan model yang belajar menggunakan sistem reward dan penalti. Reinforcement learning adalah teknik yang mempelajari bagaimana membuat keputusan terbaik, secara berurutan, untuk memaksimalkan ukuran sukses kehidupan nyata[4]. Entitas pembuat keputusan belajar melalui proses trial dan eror. Salah satu contoh dari Reinforcement learning adalah program AlphaGo.

1.1.3. Contoh Implementasi

Di era teknologi yang berkembang pesat ini, sebenarnya teknologi machine learning sudah banyak digunakan dalam kehidupan sehari-hari. Namun, mungkin ada beberapa orang yang tidak menyadarinya. Oleh karena itu, berikut ini merupakan beberapa contoh implementasi teknologi machine learning di sekitar kita.

a. Bidang Computer Vision

Computer Vision merupakan penggunaan teknologi machine learning untuk mengelola data berupa gambar atau bisa disebut dengan deteksi gambar. Computer Vision adalah ilmu yang menggunakan image processing untuk membuat keputusan berdasarkan citra yang didapat dari sensor[5]. Computer vision merupakan penggunaan machine learning yang paling umum. Contoh dari computer

vision yaitu deteksi objek dan wajah, tulisan, bahkan deteksi plat mobil.

b. Bidang Virtual Personal Assistants

Kalian pasti sudah kenal dengan Google assistant milik Google, Siri milik Apple, dan Alexa milik Amazon. Teknologi tersebut merupakan beberapa contoh penggunaan machine learning di bidang virtual personal assistants. Jadi, mereka dibuat untuk membantu manusia selayaknya mereka adalah seorang asisten.

c. Bidang Search Engine

Search engine merupakan salah satu penerapan machine learning yang populer. Namun, banyak orang yang kurang menyadarinya. Contohnya jika kita browsing menggunakan Google, maka teknologi machine learning akan menerapkan metode klasifikasi untuk memunculkan hasil pencarian kita.

d. Bidang Kedokteran

Penerapan machine learning dalam bidang kedokteran bisa berupa deteksi penyakit pasien. Deteksi penyakit ini bisa melalui deteksi gambar. Teknologi machine learning akan mempelajari data-data yang berhubungan dengan dengan gejala yang ditunjukkan. Misalnya deteksi penyakit diabetes atau sekarang yang banyak digunakan yaitu deteksi penyakit Covid-19.

e. Bidang Recommendation System

Recommendation system juga merupakan penerapan machine learning yang sering kita temui dalam kehidupan

sehari-hari. Contohnya yaitu rekomendasi video pada Youtube, rekomendasi dalam media sosial seperti di Instagram, Facebook, bahkan video di fyp TikTok. Semua itu merupakan contoh dari penerapan machine learning.

1.2. Pengenalan Deep Learning

Sepertinya tak banyak dari kita yang familiar dengan Deep Learning. Setelah ini kita akan membahas mengenai deep learning.

1.2.1. Pengertian Deep Learning

Deep learning merupakan cabang dari machine learning, dimana deep learning ini memiliki algoritma jaringan saraf tiruan yang dapat belajar dan beradaptasi terhadap sejumlah besar data[6]. Algoritma jaringan saraf tiruan pada deep learning terinspirasi dari struktur otak manusia. Deep learning ini memiliki beberapa jenis, kita akan bahas pada sub bab setelah ini.

1.2.2. Jenis-Jenis Deep Learning

Deep Learning sudah dikembangkan ke berbagai model atau arsitektur yang berbeda-beda. Beikut merupakan beberapa jenis-jenis dari model algoritma deep learning.

a. Deep Neural Network (DNN)

Algoritma DNN (Deep Neural Networks) adalah salah satu algoritma berbasis jaringan saraf yang dapat digunakan untuk pengambilan keputusan[7].

b. Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN) merupakan jenis arsitektur jaringan saraf tiruan yang pemrosesannya dipanggil berulang-ulang untuk memproses masukan yang biasanya adalah data sekuensial[6]. RNN masuk dalam kategori deep learning karena data diproses melalui banyak lapis (layer). RNN ini banyak digunakan untuk Natural Language Processing (NLP), pengenalan suara, bahkan seperti yang akan kita buat yaitu untuk time series forecasting.

c. Convolutional Neural Network (CNN)

Convolutional Neural Network merupakan salah satu jenis algoritma Deep Learning yang dapat menerima input berupa gambar, menentukan aspek atau obyek apa saja dalam sebuah gambar yang bisa digunakan mesin untuk belajar mengenali gambar, dan membedakan antara satu gambar dengan yang lainnya[6]. CNN terdiri dari neuron yang memiliki weight, bias dan activation function. Convolutional layer juga terdiri dari neuron yang tersusun sedemikian rupa sehingga membentuk sebuah filter dengan panjang dan tinggi (pixels).

d. Artificial Neural Network (ANN)

Artificial Neural Network (ANN) merupakan set algoritma yang berkerja seperti jaringan saraf otak manusia, dimana neuron saling terhubung satu dengan lainnya, bekerja untuk memproses informasi[7]. Tujuan utama dari ANN adalah menjadikan komputer memiliki kemampuan cognitif seperti otak manusia, memiliki kemampuan problem solving dan dapat melakukan proses pembelajaran.

1.3. Pengenalan Time Series Forecasting

Pada bagian ini kita akan membahas mengenai pengenalan topik utama dari buku ini nih. Jadi kita akan membahas mengenai time series. Baik langsung saja kita bahas di sub bab berikut.

1.3.1. Pengertian Time Series

Menurut Hanke & Wichers, Time series merupakan himpunan observasi data terurut dalam waktu[8]. Metode time series adalah metode peramalan dengan menggunakan analisa pola hubungan antara variabel yang akan diperkirakan dengan variabel waktu. Jadi, time series ini merupakan sebuah urutan nilai yang teratur dimana biasanya memiliki jarak yang sama dari waktu ke waktu.



Gambar 1.1. Contoh Multivariate Time Series

Gambar 1.1. merupakan contoh dari grafik Multivariate Time Series. Grafik tersebut merupakan grafik data tingkat kelahiran dan kematian di Jepang. Grafik tersebut termasuk ke dalam time series data karena data tersebut berada dalam rentang waktu tertentu.

Forecasting merupakan suatu bidang ilmu untuk memperkirakan atau memprediksi kejadian di masa yang akan datang[9]. Hal ini dapat dilakukan dengan mengambil data historis sebelumnya dan memproyeksikannya ke masa yang akan datang dengan cara membentuk model matematis atau prediksi intuisi bersifat subyektif yang disesuaikan dengan pertimbangan yang baik.

Time Series Forecasting adalah sebuah bagian dari machine learning yang berfokus pada atribut waktu. Time series forecasting berfokus pada analisis rentetan data yang berurutan (Sequential) terhadap waktu, lalu memprediksi data-data yang akan datang berdasarkan data sebelumnya[10].

1.3.2. Pola Data Time Series

Dalam time series, khususnya time series forecasting memerlukan untuk memperhatikan tipe atau pola data. Secara umum, terdapat empat macam pola data time series. Berikut merupakan pola-pola datanya.

a. Trend

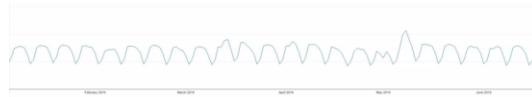
Trend merupakan suatu pola time series dimana time series tersebut memiliki arah spesifik tertentu yang dituju[11]. Terkadang polanya linier meningkat atau menurun dari waktu ke waktu. Berikut merupakan contoh grafik dari trend.



Gambar 1.2. Contoh Grafik Pola Trend

b. Seasonality

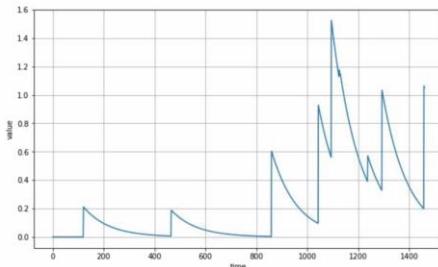
Pola seasonality juga biasa disebut dengan pola musiman, yang merupakan suatu pola dimana bentuk polanya berulang pada interval yang dapat diprediksi[11]. Berikut merupakan contoh grafik dari pola seasonality.



Gambar 1.3. Contoh Grafik Pola Seasonality

c. Autocorrelations

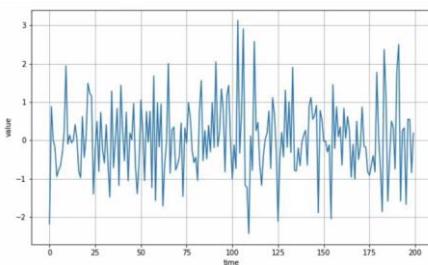
Autocorrelations merupakan pola dimana tidak ada trend dan seasonality[11]. Grafiknya muncul dalam rentang waktu yang acak. Berikut merupakan contoh grafik autocorrelations.



Gambar 1.4. Contoh Grafik Pola Autocorrelations

d. Noise

Noise merupakan pola kombinasi antara pola trend dan seasonality. Namun, pola ini memiliki bentuk yang tidak dapat diprediksi karena nilainya terkesan sangat acak[11]. Berikut merupakan contoh dari pola noise.



Gambar 1.5. Contoh Grafik Pola Noise

1.3.3. Contoh Time Series Forecasting

Permasalahan di dunia ini memang tidak ada habisnya, termasuk juga masalah yang berhubungan dengan time series. Namun, dengan adanya time series forecasting ini bisa membantu mengatasi masalah yang ada di dunia ini. Berikut merupakan beberapa contoh penerapan time series forecasting di berbagai bidang[12].

1. Peramalan hasil panen jagung dalam ton berdasarkan wilayah negara bagian setiap tahunnya.
2. Prediksi EEG dalam hitungan detik untuk menunjukkan apakah pasien mengalami kejang atau tidak.
3. Peramalan harga penutupan saham setiap hari.
4. Peramalan angka kelahiran di semua rumah sakit di suatu kota setiap tahun.
5. Peramalan penjualan produk dalam unit yang terjual setiap hari untuk sebuah toko.
6. Peramalan jumlah penumpang yang menggunakan kereta api setiap hari.
7. Peramalan pengangguran di suatu wilayah setiap kuartal tahun.
8. Peramalan harga rata-rata bensin di suatu kota setiap hari.

BAB 2

PERSIAPAN PRAKTEK

2.1 Alat dan Bahan

Untuk bisa membuat suatu model time series forecasting, tentunya kita membutuhkan suatu tools atau alat dalam membuatnya. Pada buku ini, penulis akan menggunakan windows sebagai sistem operasinya. Berikut merupakan detail alat dan bahan yang kita gunakan dalam pembuatan model time series forecasting pada buku ini.

Persyaratan utama yaitu harus memiliki laptop/komputer dengan dengan spesifikasi :

- Minimal RAM 2 GB
- Sistem operasi Chrome OS, Linux, Windows 7/8/10, MAC OS X. Namun pada buku ini penulis menggunakan Windows 10.
- Prosesor 32-bit atau 64-bit
- Terdapat Google Chrome atau browser sejenis
- Dapat terhubung ke Internet. Karena kita akan menggunakan Google Colab untuk praktek kali ini.

Kurang lebih persyaratan di atas harus dipenuhi agar pembaca bisa mengikuti tutorial pembuatan model Time Series Forecasting pada buku ini. Selanjutnya, penulis akan menjelaskan mengenai cara penggunaan tools yang digunakan dalam praktek kali ini.

2.2 Petunjuk Penggunaan Google Colab

Google Colab merupakan layanan free cloud service dari google dimana Google Colab ini berbentuk coding environment bahasa pemrograman Python dengan format notebook (mirip dengan Jupyter

notebook). Jadi dengan menggunakan Google Colab kita bisa membuat program dengan bahasa python dengan mudah tanpa menginstall IDE yang berat. Cukup dengan internet kita bisa mengakses Google Colab.

Google Colab ini menyediakan tiga jenis prosesor yaitu Central Processing Unit (CPU), Graphic Processing Unit (GPU), dan Tensor Processing Unit (TPU). Saat ini Google Colab banyak digunakan untuk pengembangan deep learning. Baiklah kita sudah tau mengenai apa itu Google Colab, selanjutnya penulis akan menunjukkan bagaimana cara penggunaan Google Colab.

1. Pertama, untuk bisa mengakses Google Colab. Kita harus memiliki koneksi internet dan browser. Disini penulis memakai Google Chrome.
2. Buka Google Chrome dan menuju ke link <https://colab.research.google.com/> atau bisa langsung ketik Google Colab di kolom pencarian Google.



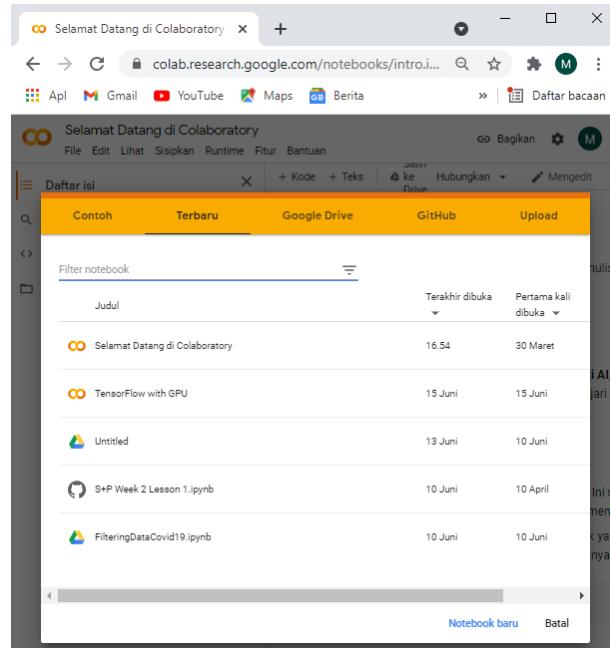
Gambar 2.1. Tampilan Search Link Google Colab

atau



Gambar 2.2. Tampilan Pencarian Google Colab

3. Setelah itu akan muncul tampilan seperti gambar 2.3.. Dimana memuat mengenai daftar notebook kita.



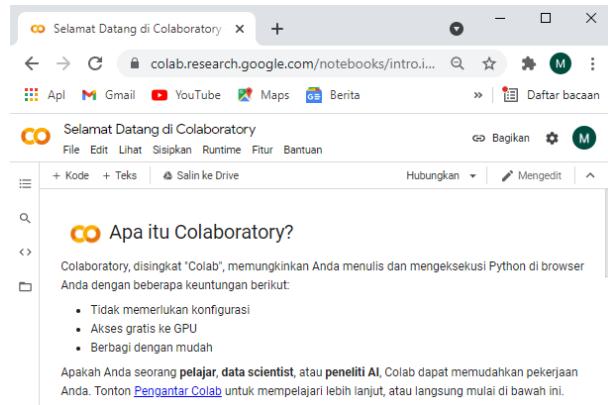
The screenshot shows a web browser window for Google Colaboratory. The title bar says "Selamat Datang di Colaboratory". The main content area is titled "Daftar Isi" and lists several notebooks:

Judul	Terakhir dibuka	Pertama kali dibuka
Selamat Datang di Colaboratory	16.54	30 Maret
TensorFlow with GPU	15 Juni	15 Juni
Untitled	13 Juni	10 Juni
S+P Week 2 Lesson 1.ipynb	10 Juni	10 April
FilteringDataCovid19.ipynb	10 Juni	10 Juni

At the bottom right of the list, there are buttons for "Notebook baru" and "Batal".

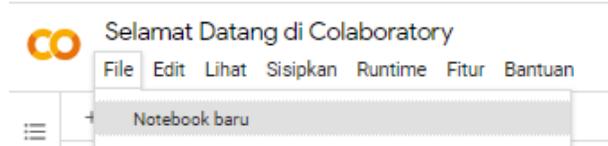
Gambar 2.3. Tampilan Daftar Notebook

4. Jika kita pilih batal pada tampilan gambar 2.3, maka kita akan mengarah ke halaman utama dari Google Colab.



Gambar 2.4. Tampilan Beranda Google Colab

5. Untuk membuat notebook baru, maka pilih tab file. Lalu pilih “Notebook baru”.



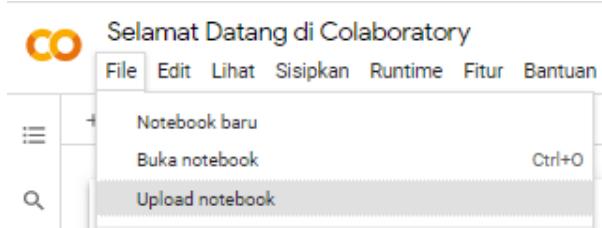
Gambar 2.5. Tampilan Pilihan Notebook Baru

6. Untuk membuka notebook yang telah dibuat, maka pilih tab file. Pilih “Buka notebook” atau kita bisa menekan Ctrl + O. Setelah itu pilih notebook yang ingin dibuka.



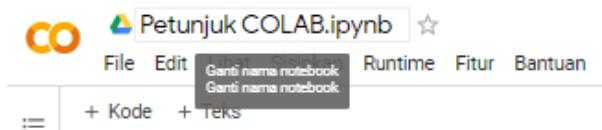
Gambar 2.6. Tampilan Pilihan Buka Notebook

7. Jika ingin mengupload notebook maka pilih File. Lalu klik Upload notebook. Kemudian pilih file yang ingin di upload.



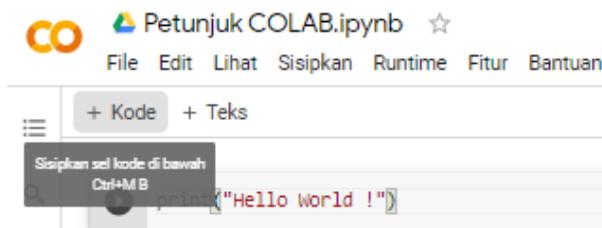
Gambar 2.7. Tampilan Pilihan Upload Notebook

8. Setelah membuat notebook baru. Kita bisa memberi nama notebook kita dengan bebas di bagian kiri atas sebelah logo Google Colab. Disini Penulis memberi nama Petunjuk COLAB.



Gambar 2.8. Tampilan Memberi Nama Notebook

9. Untuk menambahkan kode Python. Kita bisa klik "+Kode". Kemudian tulis kode di kotak yang sudah disediakan.



Gambar 2.9. Tampilan Menambahkan Kode

10. Untuk menjalankan kode yang sudah ditulis. Klik ikon bulat yang ada di samping kotak kode.

A screenshot of the Google Colab interface. On the left, there are icons for file, edit, and cell type. A code cell is selected, containing the Python code `print("Hello World!")`. To the right of the cell, a tooltip box appears with the text "Jalankan sel (Ctrl+Enter)" and "sel belum dijalankan di sesi ini".

Gambar 2.10. Tampilan Menjalankan Kode

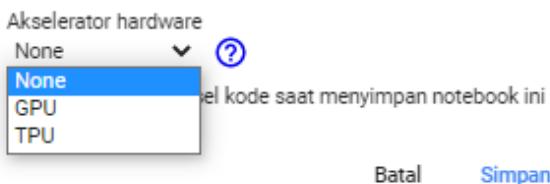
11. Jika ingin menambahkan teks, maka pilih “+ Teks”. Lalu tulis teks yang ingin ditulis.

A screenshot of the Google Colab interface. A text cell is selected, indicated by a toolbar above it with options like bold, italic, and code. The text area contains the placeholder text "Membuat Model Time Series".

Gambar 2.11. Tampilan Menambahkan Teks

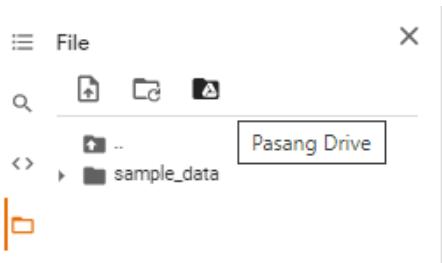
12. Google Colab memiliki 3 jenis prosesor. Jika ingin mengatur jenis prosesor, pilih tab Edit. Lalu pilih “Setelan notebook”. Kemudian pilih jenis prosesor yang akan digunakan.

Setelan notebook



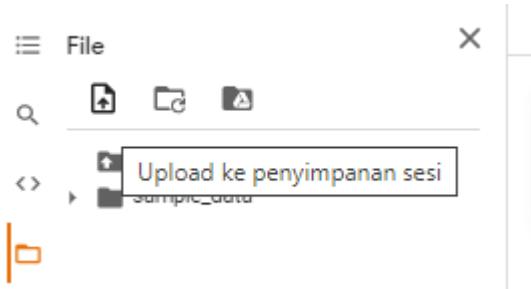
Gambar 2.12. Tampilan Mengatur Prosesor

13. Untuk menghubungkan ke google drive. Kita dapat memilih ikon file di samping kiri bawah. Lalu kita klik ikon yang terdapat ikon google drive.



Gambar 2.13. Tampilan Menghubungkan G-Drive

14. Untuk mengupload file ke google colab, maka pilih ikon folder di sebelah kiri bawah. Lalu pilih ikon dengan gambar file panah ke atas.



Gambar 2.14. Tampilan Mengupload File

15. Jika ingin menyimpan notebook, maka pilih tab File. Dan pilih Simpan, atau bisa juga cukup tekan Ctrl + S.



Gambar 2.15. Tampilan Menyimpan Notebook

2.3 Instalasi Library

Dalam pembuatan model time series forecasting, tentunya kita harus menginstall beberapa Library. Library merupakan gabungan dari sekumpulan package dan modul dengan fungsionalitas yang sama dengan tujuan untuk memudahkan kalian dalam membuat suatu aplikasi, tanpa harus menulis banyak kode.

Untuk menginstal suatu library pada Google Colab kita hanya perlu menggunakan perintah berikut :

```
!pip install nama_library
```

Berikut merupakan library yang akan kita gunakan kali ini.

a. Tensorflow



Gambar 2.16. Tampilan Logo Tensorflow

TensorFlow adalah library perangkat lunak open-source dan gratis untuk machine learning. Library ini dapat digunakan untuk berbagai keperluan tetapi memiliki fokus khusus pada pelatihan dan inferensi jaringan saraf dalam (Deep neural networks)[13].

Berikut merupakan perintah yang digunakan untuk menginstall tensorflow pada Google Colab :

!pip install tensorflow

Untuk mengecek versi dari tensorflow, kita dapat menggunakan perintah seperti berikut :

import tensorflow as tf

tf.__version__

Lebih jelasnya ini adalah screenshoot dari hasil google colab :

```
[1] !pip install tensorflow
```

```
[2] import tensorflow as tf
    tf.__version__
```

```
'2.5.0'
```

Gambar 2.17. Tampilan Kode Tensorflow

b. NumPy



Gambar 2.18. Tampilan Logo NumPy

NumPy adalah library untuk Bahasa pemrograman Python, NumPy berfungsi untuk memudahkan operasi komputasi tipe data numerik. NumPy menyimpan data dalam bentuk array multidimensional yang bentuknya dapat berupa 1 dimensi, 2 dimensi, atau lebih[14].

Untuk menginstall NumPy pada Google Colab, kita bisa menggunakan perintah seperti berikut :

!pip install numpy

Jika ingin melihat versi NumPy yang digunakan, kita bisa menggunakan perintah seperti berikut :

**import numpy as np
np.__version__**

Lebih jelasnya, berikut merupakan screenshoot kodenya pada Google Colab :

```
[1] !pip install numpy
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (1.19.5)

[2] import numpy as np
np.__version__
'1.19.5'
```

Gambar 2.19. Tampilan Kode NumPy

c. Matplotlib



Gambar 2.20. Tampilan Logo Matplotlib

Matplotlib adalah library untuk membuat visualisasi statis, animasi, dan interaktif dengan Python. Matplotlib dapat menghasilkan plot dengan kualitas tinggi dalam berbagai format dan dapat digunakan di banyak platform[15].

Untuk menginstall library matplotlib, kita bisa gunakan perintah seperti berikut :

!pip install matplotlib

Jika ingin memeriksa versi matplotlib yang digunakan, kita bisa menggunakan perintah seperti berikut :

**import matplotlib as plt
plt.__version__**

Untuk lebih jelasnya, bisa melihat hasil screenshoot di bawah ini :

```
[3] !pip install matplotlib
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (3.2.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib)
Requirement already satisfied: pytz!=2019.3,>=2019.2 in /usr/local/lib/python3.7/dist-packages (from matplotlib)
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.7/dist-packages (from matplotlib)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from cycler>=0.10)
[3] >
import matplotlib as plt
plt.__version__
'3.2.2'
```

Gambar 2.21. Tampilan Kode Matplotlib

BAB 3

TIME SERIES FORECASTING DENGAN NEURAL NETWORK

3.1 Menyiapkan Feature dan Label

Agar dapat melakukan prediksi, model Machine learning membutuhkan data. Di dalam data terdapat yang namanya feature dan label. Setelah ini kita akan membahasnya.

3.1.1. Pengertian Feature dan Label

Feature merupakan data yang dipilih untuk melatih (training) atau sering juga disebut sebagai x variabel. Sedangkan label adalah data yang diprediksi oleh model atau sering disebut sebagai y variabel[16]. Berikut merupakan contoh gambaran feature dan label dari dataset Iris.

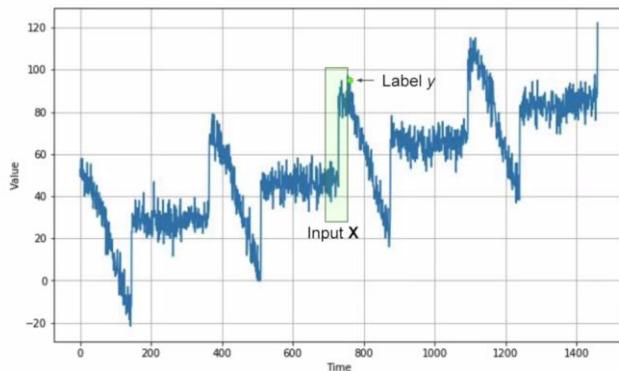
	Feature				Label	
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

Gambar 3.1. Ilustrasi Feature dan Label

Saat akan menyelesaikan masalah machine learning, pada umumnya langkah awal yang dilakukan yaitu kita harus membagi data kita menjadi feature dan label. Dalam kasus time series forecasting, feature adalah sejumlah nilai dalam series, dengan label sebagai nilai lanjutannya. Bisa dibilang bahwa

jumlah nilai yang akan digunakan itu sebagai feature, lalu kita mengambil windowed data dan melakukan training pada model machine learning untuk memprediksi nilai berikutnya atau memprediksi labelnya. Berikut merupakan gambaran mengenai feature dan label pada data time series.

Machine Learning on Time Windows



Gambar 3.2. Contoh Feature dan Label

Contoh sederhananya jika kita mengambil data time series, katakanlah 30 hari setiap kalinya. Kita akan menggunakan 30 nilai tersebut sebagai features dan nilai berikutnya adalah labelnya. Kemudian seiring berjalannya waktu, kita akan melatih (men-training) jaringan saraf untuk mencocokkan 30 feature dengan satu label.

3.1.2. Membagi Feature dan Label

Untuk dapat membagi data menjadi feature dan label, pertama kita perlu membuat datanya. Langsung saja setelah ini

akan dijelaskan mengenai cara membuat datasetnya hingga membagi feature dan label.

- **Mengimpor Library dan Memeriksa Versi Tensorflow**

Pertama kita perlu untuk mengimpor library yang akan kita gunakan dan kita perlu untuk memeriksa versi tensorflow yang digunakan. Berikut merupakan kodennya :

```
#Mengimpor library tensorflow
import tensorflow as tf

#Mengimpor library NumPy
import numpy as np

#Mengimpor library matplotlib
import matplotlib.pyplot as plt
```

Gambar 3.3. Kode Mengimpor Library

```
#Memeriksa versi tensorflow
print(tf.__version__)

2.5.0
```

Gambar 3.4. Kode Memeriksa Versi Tensorflow

Berdasarkan hasil gambar 3.4., dapat kita lihat bahwa kita menggunakan tensorflow versi 2.5.0. Selanjutnya kita akan mulai membuat dataset sederhana.

- **Membuat dataset sederhana dengan tf.data.Dataset**

Pada contoh kali ini kita akan membuat dataset sederhana yang berisi 10 nilai. Berikut merupakan kode yang akan kita buat.

```
#Membuat dataset dengan nilai sebanyak 10
dataset = tf.data.Dataset.range(10)
for nilai in dataset:
    print(nilai.numpy())

0
1
2
3
4
5
6
7
8
9
```

Gambar 3.5. Kode Membuat Dataset Sederhana

Kode pada gambar 3.5. menggunakan `tf.data.Dataset` yang berfungsi untuk membuat dataset. Kemudian juga menggunakan `numpy`. Ketika kita run kode pada gambar 3.5, maka akan muncul serangkaian data dari 0 hingga 9. Selanjutnya mari kita buat data ini menjadi lebih menarik dengan menggunakan `dataset.window`.

- **Menambahkan `dataset.window` pada kode `dataset` sebelumnya**

Kali ini kita menambahkan `dataset.window` untuk memperluas kumpulan datanya menggunakan windowing. Berikut kode yang akan kita buat beserta outputnya :

```
#Menambahkan dataset.window untuk membuat windowed dataset
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1)
for window_dataset in dataset:
    for nilai in window_dataset:
        print(nilai.numpy(), end=" ")
    print()

0 1 2 3 4
1 2 3 4 5
2 3 4 5 6
3 4 5 6 7
4 5 6 7 8
5 6 7 8 9
6 7 8 9
7 8 9
8 9
9
```

Gambar 3.6. Kode dan Hasil Menambahkan dataset.window

Pada kode di gambar 3.6, kita menambahkan dataset.window(5, shift=1) yang berarti kita mengatur ukuran jendelanya menjadi 5, dengan pergeseran 1. Ketika kita run kodennya, maka yang muncul akan 01234 pada baris pertama, lalu baris kedua angkanya bergeser 1, jadi 12345 sampai seterusnya. Tahap selanjutnya kita akan mengedit sedikit jendelanya, sehingga ukuran datanya menjadi teratur.

- **Menambahkan drop_remainder**

Kali ini kita akan menambahkan drop_remainder dan mengaturnya menjadi True. Hal ini bertujuan agar datanya menjadi lebih teratur. Sehingga kodennya akan menjadi seperti ini :

```

dataset = tf.data.Dataset.range(10)
#Menambahkan drop remainder
dataset = dataset.window(5, shift=1,
                        drop_remainder=True)
for window_dataset in dataset:
    for nilai in window_dataset:
        print(nilai.numpy(), end=" ")
    print()

```

```

0 1 2 3 4
1 2 3 4 5
2 3 4 5 6
3 4 5 6 7
4 5 6 7 8
5 6 7 8 9

```

Gambar 3.7. Kode Menambahkan drop_remainder

Berdasarkan hasil output di gambar 3.7, dapat kita lihat bahwa bentuk datanya menjadi lebih teratur. Pada baris pertama hasilnya angka 01234 dan berakhir dengan nilai 56789. Langkah selanjutnya kita mengubah nilainya ke dalam daftar numpy.

- **Memasukkan nilai ke daftar numpy**

Agar datasetnya bisa digunakan untuk proses machine learning, maka kita perlu untuk memasukkan datanya ke dalam daftar numpy dengan menggunakan dataset.flat_map. Sehingga kodennya berubah menjadi seperti berikut :

```

dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1,
                        drop_remainder=True)
dataset = dataset.flat_map(lambda window: window.batch(5))
for nilai in dataset:
    print(nilai.numpy())

```

```

[0 1 2 3 4]
[1 2 3 4 5]
[2 3 4 5 6]
[3 4 5 6 7]
[4 5 6 7 8]
[5 6 7 8 9]

```

Gambar 3.8. Kode Memasukkan Nilai ke NumPy

Data kita telah menjadi bentuk array. Maka datasetnya siap digunakan dalam proses machine learning dan langkah selanjutnya adalah membagi datanya menjadi feature dan label.

- **Membagi data menjadi feature dan label**

Untuk membagi datanya menjadi feature dan label, maka kita menggunakan pemetaan dataset.map . Berikut kodenya :

```
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1, drop_remainder=True)
dataset = dataset.flat_map(lambda window: window.batch(5))

#Membagi data dengan pemetaan
dataset = dataset.map(lambda window:
                      (window[:-1], window[-1:]))
for x,y in dataset:
    print(x.numpy(), y.numpy())

[0 1 2 3] [4]
[1 2 3 4] [5]
[2 3 4 5] [6]
[3 4 5 6] [7]
[4 5 6 7] [8]
[5 6 7 8] [9]
```

Gambar 3.9. Kode Membagi Data

Setelah ini kita akan menambahkan metode shuffle agar datanya bisa acak, sehingga bagus untuk pelatihan machine learning.

- **Menambahkan metode shuffle**

Langkah terakhir agar datanya bagus untuk pelatihan machine learning yaitu kita perlu mengacak nilainya. Disini kita menggunakan metode shuffle. Sehingga kodenya menjadi seperti berikut :

```
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1, drop_remainder=True)
dataset = dataset.flat_map(lambda window: window.batch(5))
dataset = dataset.map(lambda window:
                      (window[:-1], window[-1:]))
#Mengacak Data
dataset = dataset.shuffle(buffer_size=10)
for x,y in dataset:
    print(x.numpy(), y.numpy())

[2 3 4 5] [6]
[5 6 7 8] [9]
[3 4 5 6] [7]
[1 2 3 4] [5]
[0 1 2 3] [4]
[4 5 6 7] [8]
```

Gambar 3.10. Kode Mengacak Data

Baiklah, dengan begini kita telah berhasil untuk membagi data menjadi feature dan label. Pada sub bab selanjutnya kita akan menyiapkan windowed dataset.

3.2 Menyiapkan Windowed Dataset

Dalam membuat model time series machine learning, tentunya kita harus menyiapkan dataset. Dalam time series, kita biasa menyebutnya dengan windowed dataset. Windowed dataset adalah dataset dari time series yang sudah di atur ukurannya dan konsisten. Pada sub bab sebelumnya kita sudah membuat windowed dataset sederhana dengan rentang nilai sampai 10.

Kali ini kita akan membuat windowed dataset yang sesungguhnya yang akan kita gunakan dalam deep learning Neural Network. Berikut merupakan kode yang harus kita buat untuk membuat windowed datasetnya :

```
import tensorflow as tf

def windowed_dataset(series, window_size, batch_size, shuffle_buffer):
    dataset = tf.data.Dataset.from_tensor_slices(series)
    dataset = dataset.window(window_size + 1, shift=1, drop_remainder=True)
    dataset = dataset.flat_map(lambda window: window.batch(window_size + 1))
    dataset = dataset.shuffle(shuffle_buffer)
        .map(lambda window: (window[:-1], window[-1]))
    dataset = dataset.batch(batch_size).prefetch(1)
    return dataset
```

Gambar 3.11. Kode Membuat Windowed Dataset

Penjelasan kode :

- Pertama kita harus mengimpor library tensorflow
- Kemudian kita membuat sebuah fungsi untuk datasetnya, dan kita beri nama windowed_dataset
- tf.data.Dataset kita gunakan untuk membuat datasetnya, kemudian kita membuat datanya menjadi series dengan metode from_tensor_slices.
- Lalu kita menggunakan metode dataset.window untuk membuat bentuk data menjadi teratur, dan kita membuat semua ukurannya sama dengan drop_remainder=True.
- Kemudian kita meratakan datanya dengan dataset.flat_map agar datanya lebih mudah digunakan.
- Setelah meratakan data, maka kita akan menggunakan data.shuffle untuk mengacak datanya. Selain itu, kita menggunakan buffer_shuffle untuk mempercepat prosesnya.
- Dataset yang sudah diacak kemudian dikelompokkan ke dalam ukuran batch yang dipilih dengan metode dataset.batch
- Yang terakhir kita me return dataset, agar prosesnya dapat dilaksanakan. Maka windowed dataset kita sudah siap digunakan.

3.3 Percobaan dengan Single Layer Neural Network

Sebelumnya kita sudah membuat window dataset. Sekarang kita bisa memulai untuk melatih datanya dengan menggunakan neural network. Dalam percobaan kali ini metode yang digunakan adalah regresi linear. Baiklah langsung saja kita mulai percobaannya.

- **Mengimpor Library dan Memeriksa Versi Tensorflow**

Langkah pertama yang perlu kita lakukan adalah mengimpor library, disini kita membutuhkan library tensorflow, numpy dan matplotlib. Tak lupa kita juga memeriksa versi tensorflow yang digunakan. Berikut merupakan kode yang perlu kita tulis :

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
print(tf.__version__)
```

2.5.0

Gambar 3.12. Kode Impor Library dan Periksa Versi Tensorflow

Sesuai hasil di gambar 3.12, bisa diketahui penulis menggunakan tensorflow versi 2.5.0.

- **Membuat Fungsi Data Series**

Setelah mengimpor library, langkah selanjutnya yaitu kita membuat fungsi pembantu yang menyiapkan data series serta membuat data series itu sendiri. Berikut merupakan kodennya :

```

#Fungsi untuk membuat visualisasi grafik time series
def plot_series(time, series, format="-", start=0, end=None):
    plt.plot(time[start:end], series[start:end], format)
    plt.xlabel("Waktu") #Sumbu x
    plt.ylabel("Nilai") #Sumbu y
    plt.grid(True) #menampilkan grid

#Fungsi untuk membuat pola trend
#Nilainya boleh diubah diatur sesuai keinginan
def trend(time, slope=0):
    return slope * time

#Fungsi untuk membuat pola seasonal
#Nilainya boleh diubah diatur sesuai keinginan
def seasonal_pattern(season_time):
    return np.where(season_time < 0.4,
                   np.cos(season_time * 2 * np.pi),
                   1 / np.exp(3 * season_time))

#Fungsi untuk membuat pola seasonality
def seasonality(time, period, amplitude=1, phase=0):
    season_time = ((time + phase) % period) / period
    return amplitude * seasonal_pattern(season_time)

#Fungsi untuk membuat pola noise
def noise(time, noise_level=1, seed=None):
    rnd = np.random.RandomState(seed)
    return rnd.randn(len(time)) * noise_level

#Mengatur nilai time (waktu)
time = np.arange(4 * 365 + 1, dtype="float32")
baseline = 10 #Mengatur nilai baseline
series = trend(time, 0.1) #Mengatur seriesnya
baseline = 10
amplitude = 40 #Mengatur amplitude
slope = 0.05 #Mengatur slope
noise_level = 5 #Mengatur level noise

#Membuat Seriesnya
series = baseline + trend(time, slope) + seasonality(time,
                                                       period=365,
                                                       amplitude=amplitude)
#Diperbarui dengan menambahkan noise
series += noise(time, noise_level, seed=42)

```

Gambar 3.13. Kode Membuat Bentuk Data Series

- **Membagi Dataset**

Sebelum melakukan training, kita harus membagi dataset dulu menjadi training dan validation set. Berikut merupakan kode yang digunakan :

```
split_time = 1000 #Membagi time menjadi 1000
time_train = time[:split_time] #data time untuk training
x_train = series[:split_time] #data series untuk training
time_valid = time[split_time:] #data time untuk validasi
x_valid = series[split_time:] #data series untuk validasi
```

Gambar 3.14. Kode Membagi Dataset

Setelah membagi dataset, langkah selanjutnya kita bisa mengatur konstanta untuk digunakan sebagai ukuran window, batch, dan shuffle_buffer. Berikut kode yang ditambahkan :

```
window_size = 20 #mengatur ukuran window
batch_size = 32 #mengatur ukuran batch
shuffle_buffer_size = 1000 #mengatur ukuran buffer acak
```

Gambar 3.15. Kode Mengatur Window, Batch, Shuffle Buffer

Sesuai gambar 3.15, window sizenya di atur menjadi 20, batchnya 32, dan shuffle buffernya menjadi 1000.

- **Menambahkan Fungsi Windowed Dataset**

Selanjutnya kita menambahkan dataset yang telah kita buat sebelumnya. Berikut kodennya :

```
def windowed_dataset(series, window_size, batch_size, shuffle_buffer):
    dataset = tf.data.Dataset.from_tensor_slices(series)
    dataset = dataset.window(window_size + 1, shift=1,
                            drop_remainder=True)
    dataset = dataset.flat_map(lambda window:
                                window.batch(window_size + 1))
    dataset = dataset.shuffle(shuffle_buffer).map(lambda window:
                                                   (window[:-1], window[-1]))
    dataset = dataset.batch(batch_size).prefetch(1)
    return dataset
```

Gambar 3.16. Kode Membuat Windowed Dataset

- **Membuat Model**

Kita telah berhasil menambahkan datasetnya, langkah selanjutnya kita akan membuat model machine learningnya. Pertama kita akan memanggil fungsi datasetnya dan di sesuaikan seriesnya dengan ukuran window, batch, dan shuffle buffernya. Kemudian kita membuat single layer neural network dengan variabel layer, dan mendefinisikan model kita dengan sequential. Berikut merupakan kodennya :

```
#Memanggil Fungsi Windowed Dataset
dataset = windowed_dataset(x_train, window_size,
                           batch_size,
                           shuffle_buffer_size)
print(dataset)

#Membuat Dense Layer
layer = tf.keras.layers.Dense(1, input_shape=[window_size])

#Membuat single layer neural network
model = tf.keras.models.Sequential([layer])

#Membuat simpulan model
model.summary()
```

Gambar 3.17. Kode Membuat Model Neural Network

Hasil Summary Model :

```
<PrefetchDataset shapes: ((None, None), (None,)), types: (tf.float32,
Model: "sequential_1"

Layer (type)          Output Shape         Param #
=====
dense_1 (Dense)      (None, 1)            21
=====
Total params: 21
Trainable params: 21
Non-trainable params: 0
```

Gambar 3.18. Hasil Model Summary

Setelah membuat model, selanjutnya kita akan mengcompile dan mem fit kan modelnya, disini kita menggunakan mean square error sebagai loss function. Berikut kodennya :

```

#Mengcompile model
model.compile(loss="mse",
              optimizer=tf.keras.optimizers.SGD(learning_rate=1e-6,
                                                momentum=0.9))
#Mentraining Model
model.fit(dataset,epochs=100,verbose=0)

#Menampilkan Hasil Prediksi
print("Layer weights {}".format(layer.get_weights()))

```

Gambar 3.19. Kode Compile dan Training Model

- **Hasil Prediksi**

Setelah kita menjalankan kode seperti gambar 3.19, maka akan menghasilkan hasil prediksi dari model yang telah dibuat.

Berikut merupakan hasil prediksinya :

```

Layer weights [array([[-0.03994585],
[-0.02073532],
[ 0.04256446],
[ 0.01644825],
[ 0.0195014 ],
[ 0.00051015],
[-0.04407081],
[ 0.02765848],
[ 0.00514269],
[ 0.03083453],
[-0.03844512],
[-0.00114161],
[-0.02529022],
[ 0.05769587],
[-0.00324716],
[ 0.06649581],
[ 0.00756234],
[ 0.1824707 ],
[ 0.28489405],
[ 0.43905395]], dtype=float32), array([0.02710244], dtype=float32)]

```

Gambar 3.20. Hasil Prediksi

Gambar 3.20 merupakan nilai bobot layer yang memberikan koefisien untuk nilai x dan regresi linear, serta nilai bias untuk regresinya. Selanjutnya kita bisa mengetahui hasil forecastingnya dengan kode berikut :

```

#Membuat Prediksi
prediksi = []

for time in range(len(series) - window_size):
    prediksi.append(model.predict(series[time:time + window_size]
                                [np.newaxis]))

prediksi = prediksi[split_time-window_size:]
Hasil = np.array(prediksi)[:, 0, 0]

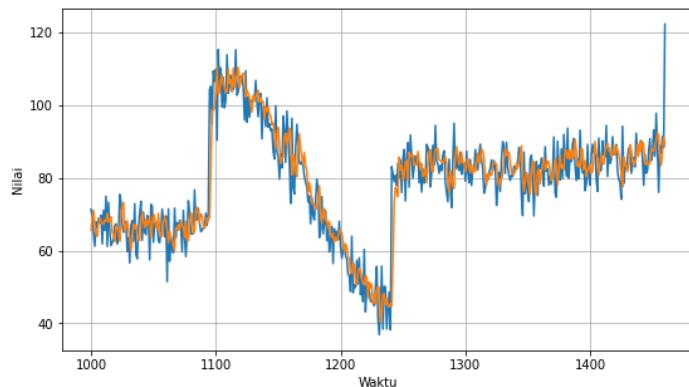
#Mengatur ukuran grafik
plt.figure(figsize=(9, 5))

plot_series(time_valid, x_valid) #Visualisasi data aktual
plot_series(time_valid, Hasil) #Visualisasi prediksi

```

Gambar 3.21. Kode Visualisasi Prediksi

Hasil Output :



Gambar 3.22. Hasil Visualisasi Prediksi

Gambar 3.22 merupakan hasil dari prediksi time series forecasting yang divisualisasikan dengan matplotlib. Grafik yang berwarna biru merupakan nilai yang sesungguhnya. Sedangkan grafik yang berwarna oranye merupakan nilai prediksinya. Melihat hasil pada gambar 3.22, bisa dikategorikan hasil model kita cukup bagus.

- **Mengukur Hasil Mean Absolute Error**

Langkah terakhir kita dapat mengukur hasil Mean absolute error dari model yang telah dibuat. Mean Absolute Error (MAE) adalah rata rata kesalahan mutlak selama periode tertentu tanpa memperhatikan apakah hasil peramalan lebih besar atau lebih kecil dibanding kenyataannya, dengan kata lain MAE adalah rata-rata dari nilai absolut simpangan[17]. Berikut rumus dari MAE :

$$\text{MAE} = \frac{\sum_{i=1}^n |y_i - x_i|}{n}$$

MAE = rata-rata kesalahan mutlak

y_i = prediksi

x_i = nilai benar

n = jumlah total poin data

Gambar 3.23. Rumus MAE

Baiklah kita sebagai langkah terakhir, kita akan mengukur nilai MAE pada model yang telah dibuat dengan kode berikut :

```
tf.keras.metrics.mean_absolute_error(x_valid, Hasil).numpy()
```

5.10186

Gambar 3.24. Kode dan Hasil Output MAE

Berdasarkan hasil pada gambar 3.24, hasil MAE dari model yang telah dibuat yaitu sebesar 5.1816177 . Nilai ini bisa saja berubah ubah di setiap notebook yang dijalankan. Karena kita menggunakan data acak. Namun, hasilnya biasanya tidak berbeda jauh.

- **Kode Keseluruhan**

Kita telah mempelajari setiap langkah-langkah dalam membuat model time series forecasting dengan neural network. Berikut ini merupakan kode keseluruhan gabungan dari kode pada setiap langkah di atas. Kalian bisa mencocokkan kode yang telah kalian praktekkan, silahkan periksa sudah benar apa belum.

Menyediakan Feature dan Label

Mengimpor library dan Memeriksa versi tensorflow

```
[ ] #Mengimpor library tensorflow
      import tensorflow as tf

#Mengimpor library NumPy
      import numpy as np

#Mengimpor library matplotlib
      import matplotlib.pyplot as plt

#Memeriksa versi tensorflow
      print(tf.__version__)
```

2.5.0

Membuat dataset sederhana dengan tf.data.Dataset

```
[ ] #Membuat dataset dengan nilai sebanyak 10
      dataset = tf.data.Dataset.range(10)
      for nilai in dataset:
          print(nilai.numpy())
```

0
1
2
3
4
5
6
7
8
9

Menambahkan dataset.window pada dataset sebelumnya

```
[ ] #Menambahkan dataset.window untuk membuat windowed dataset
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1)
for window_dataset in dataset:
    for nilai in window_dataset:
        print(nilai.numpy(), end=" ")
    print()

0 1 2 3 4
1 2 3 4 5
2 3 4 5 6
3 4 5 6 7
4 5 6 7 8
5 6 7 8 9
6 7 8 9
7 8 9
8 9
9
```

Menambahkan drop_remainder

```
[ ] dataset = tf.data.Dataset.range(10)
#Menambahkan drop remainder
dataset = dataset.window(5, shift=1,
                       drop_remainder=True)
for window_dataset in dataset:
    for nilai in window_dataset:
        print(nilai.numpy(), end=" ")
    print()

0 1 2 3 4
1 2 3 4 5
2 3 4 5 6
3 4 5 6 7
4 5 6 7 8
5 6 7 8 9
```

Memasukkan nilai ke daftar numpy

```
[ ] dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1,
                        drop_remainder=True)
dataset = dataset.flat_map(lambda window: window.batch(5))
for nilai in dataset:
    print(nilai.numpy())

[0 1 2 3 4]
[1 2 3 4 5]
[2 3 4 5 6]
[3 4 5 6 7]
[4 5 6 7 8]
[5 6 7 8 9]
```

Membagi data menjadi feature dan label

```
[ ] dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1, drop_remainder=True)
dataset = dataset.flat_map(lambda window: window.batch(5))

#Membagi data dengan pemetaan
dataset = dataset.map(lambda window:
                      (window[:-1], window[-1:]))
for x,y in dataset:
    print(x.numpy(), y.numpy())

[0 1 2 3] [4]
[1 2 3 4] [5]
[2 3 4 5] [6]
[3 4 5 6] [7]
[4 5 6 7] [8]
[5 6 7 8] [9]
```

Menambahkan metode shuffle

```
[ ] dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1, drop_remainder=True)
dataset = dataset.flat_map(lambda window: window.batch(5))
dataset = dataset.map(lambda window:
                      (window[:-1], window[-1:]))
#Mengacak Data
dataset = dataset.shuffle(buffer_size=10)
for x,y in dataset:
    print(x.numpy(), y.numpy())

[2 3 4 5] [6]
[5 6 7 8] [9]
[3 4 5 6] [7]
[1 2 3 4] [5]
[0 1 2 3] [4]
[4 5 6 7] [8]
```

Menyiapkan Windowed Dataset

```
[ ] import tensorflow as tf

def windowed_dataset(series, window_size, batch_size, shuffle_buffer):
    dataset = tf.data.Dataset.from_tensor_slices(series)
    dataset = dataset.window(window_size + 1, shift=1, drop_remainder=True)
    dataset = dataset.flat_map(lambda window: window.batch(window_size + 1))
    dataset = dataset.shuffle(shuffle_buffer
                             ).map(lambda window: (window[:-1], window[-1]))
    dataset = dataset.batch(batch_size).prefetch(1)
    return dataset
```

Time Series Forecasting dengan Single Layer Neural Network

Mengimpor library dan Memeriksa versi tensorflow

```
[ ] import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
print(tf.__version__)

2.5.0
```

Membuat fungsi data series

```
[ ] #Fungsi untuk membuat visualisasi grafik time series
def plot_series(time, series, format="--", start=0, end=None):
    plt.plot(time[start:end], series[start:end], format)
    plt.xlabel("Waktu") #Sumbu x
    plt.ylabel("Nilai") #Sumbu y
    plt.grid(True) #menampilkan grid

#Fungsi untuk membuat pola trend
#Nilainya boleh diubah diatur sesuai keinginan
def trend(time, slope=0):
    return slope * time

#Fungsi untuk membuat pola seasonal
#Nilainya boleh diubah diatur sesuai keinginan
def seasonal_pattern(season_time):
    return np.where(season_time < 0.4,
                   np.cos(season_time * 2 * np.pi),
                   1 / np.exp(3 * season_time))

[ ] #Fungsi untuk membuat pola seasonality
def seasonality(time, period, amplitude=1, phase=0):
    season_time = ((time + phase) % period) / period
    return amplitude * seasonal_pattern(season_time)

#Fungsi untuk membuat pola noise
def noise(time, noise_level=1, seed=None):
    rnd = np.random.RandomState(seed)
    return rnd.randn(len(time)) * noise_level

#Mengatur nilai time (waktu)
time = np.arange(4 * 365 + 1, dtype="float32")
baseline = 10 #Mengatur nilai baseline
series = trend(time, 0.1) #Mengatur seriesnya
baseline = 10
amplitude = 40 #Mengatur amplitude
slope = 0.05 #Mengatur slope
noise_level = 5 #Mengatur level noise
```

```

#Membuat Seriesnya
series = baseline + trend(time, slope) + seasonality(time,
                                                       period=365,
                                                       amplitude=amplitude)
#Diperbarui dengan menambahkan noise
series += noise(time, noise_level, seed=42)

```

Membagi Dataset

```

[ ] split_time = 1000 #Membagi time menjadi 1000
time_train = time[:split_time] #data time untuk training
x_train = series[:split_time] #data series untuk training
time_valid = time[split_time:] #data time untuk validasi
x_valid = series[split_time:] #data series untuk validasi

window_size = 20 #mengatur ukuran window
batch_size = 32 #mengatur ukuran batch
shuffle_buffer_size = 1000 #mengatur ukuran buffer acak

```

Menambahkan Fungsi Windowed Dataset

```

[ ] def windowed_dataset(series, window_size, batch_size, shuffle_buffer):
    dataset = tf.data.Dataset.from_tensor_slices(series)
    dataset = dataset.window(window_size + 1, shift=1,
                            drop_remainder=True)
    dataset = dataset.flat_map(lambda window:
                               window.batch(window_size + 1))
    dataset = dataset.shuffle(shuffle_buffer).map(lambda window:
                                                   (window[:-1], window[-1]))
    dataset = dataset.batch(batch_size).prefetch(1)
    return dataset

```

Membuat Model

```

[ ] #Memanggil Fungsi Windowed Dataset
dataset = windowed_dataset(x_train, window_size,
                           batch_size,
                           shuffle_buffer_size)
print(dataset)

#Membuat Dense Layer
layer = tf.keras.layers.Dense(1, input_shape=[window_size])

#Membuat single layer neural network
model = tf.keras.models.Sequential([layer])

#Membuat simpulan model
model.summary()

```

```

<PrefetchDataset shapes: ((None, None), (None,)), types: (tf.float32, tf.float
Model: "sequential"

Layer (type)          Output Shape         Param #
=====
dense (Dense)         (None, 1)            21
=====
Total params: 21
Trainable params: 21
Non-trainable params: 0

```

Hasil Prediksi

```

[ ] #Mengcompile model
model.compile(loss="mse",
              optimizer=tf.keras.optimizers.SGD(learning_rate=1e-6,
                                                momentum=0.9))
#Menraining Model
model.fit(dataset,epochs=100,verbose=0)

#Menampilkan Hasil Prediksi
print("Layer weights {}".format(layer.get_weights()))

```

```

Layer weights [array([[ 0.00856906,
[ 0.01529512],
[-0.01067554],
[-0.02653188],
[-0.00493264],
[-0.03104859],
[ 0.09642483],
[-0.09486473],
[ 0.06517576],
[ 0.01495427],
[ 0.01387533],
[-0.05110552],
[-0.02742474],
[-0.01197842],
[ 0.12855123],
[ 0.03456542],
[-0.018751 ],
[ 0.13765533],
[ 0.3801089 ],
[ 0.389866 ]], dtype=float32), array([0.01546765], dtype=float32)]
```

```

[ ] #Membuat Prediksi
prediksi = []

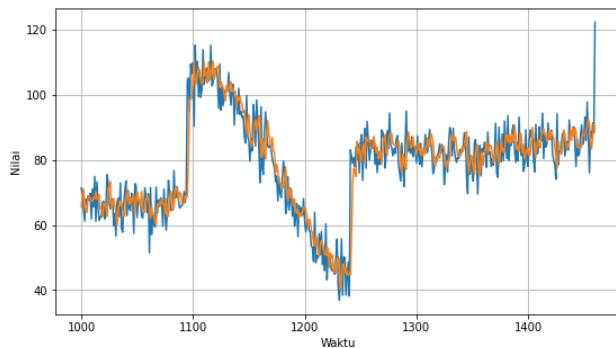
for time in range(len(series) - window_size):
    prediksi.append(model.predict(series[time:time + window_size]
                                  [np.newaxis]))

prediksi = prediksi[split_time-window_size:]
Hasil = np.array(prediksi)[:, 0, 0]

#Mengatur ukuran grafik
plt.figure(figsize=(9, 5))

plot_series(time_valid, x_valid) #Visualisasi data aktual
plot_series(time_valid, Hasil) #Visualisasi prediksi

```



Mengukur Hasil Mean Absolute Error

```
[ ] tf.keras.metrics.mean_absolute_error(x_valid, Hasil).numpy()  
5.10186
```

Gambar 3.25. Kode Keseluruhan Bab 3

BAB 4

TIME SERIES FORECASTING DENGAN DEEP NEURAL NETWORK (DNN)

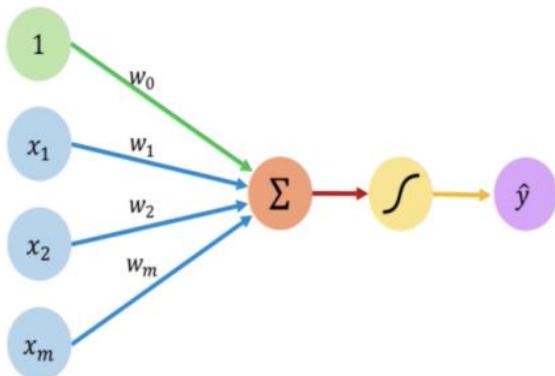
4.1 Konsep Deep Neural Network

Pada bab 1, sudah dijelaskan secara singkat mengenai apa itu Deep Neural Network. Namun, disini akan dibahas lebih mendalam mengenai bagaimana konsep dari Deep Neural Network. Seperti yang sudah diketahui, Deep Neural Network merupakan salah satu algoritma berbasis jaringan saraf yang dapat digunakan untuk pengambilan keputusan.

Konsep DNN terinspirasi dari cara kerja saraf otak manusia. Dimana saraf manusia saling terhubung dan berfungsi sebagai pembawa pesan atau informasi. Cara kerja dari neuron (saraf) sangat sederhana. Setiap neuron terhubung dengan beberapa ribu neuron lainnya.

Perceptron merupakan komponen dasar pembangun saraf tiruan. Perceptron pada jaringan saraf tiruan terinspirasi dari neuron pada jaringan saraf di otak manusia[18]. Pada jaringan saraf tiruan, perceptron dan neuron merujuk pada hal yang sama.

Pada Deep Neural Network, sebuah perceptron menerima masukan berupa bilangan numerik. Kemudian inputan tersebut diproses untuk menghasilkan keluaran (output). Gambaran prosesnya seperti berikut :



Gambar 4.1. Ilustrasi Proses DNN

Sebuah perceptron tersusun atas 5 komponen, diantaranya :

1. Input (x_i)
2. Bobot atau weights (W_i) dan bias (W_0)
3. Penjumlahan atau sum (Σ)
4. Fungsi aktivasi atau non linearity function (f)
5. Output (y)

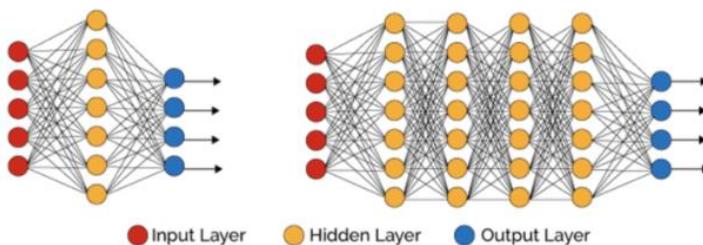
Lalu bagaimakah cara kerja perceptron, berikut merupakan penjelasan mengenai cara kerjanya :

- Pertama memasukkan input yang menerima masukan berupa angka-angka. Setiap input memiliki bobot masing-masing.
- Selanjutnya tahap penjumlahan input. Pada tahap ini setiap input akan dikalikan dengan bobotnya masing-masing, lalu hasilnya akan ditambahkan dengan bias yang merupakan konstanta atau angka. Nilai bias berguna untuk mengubah

kurva fungsi aktivasi ke atas atau ke bawah sehingga bisa lebih fleksibel dalam meminimalisasi error. Hasil penjumlahan pada tahap ini disebut weighted sum.

- Setelah penjumlahan input, hasil weighted sum diaplikasikan pada fungsi aktivasi atau disebut dengan Non-Linearity Function. Fungsi ini memungkinkan perceptron dapat menyesuaikan pola untuk data yang non linier.
- Setelah proses di atas sudah dilewati, tahap terakhir yaitu hasil output, yaitu hasil perhitungan sebuah percepton.

Deep Neural Network memiliki banyak hidden layer. Hidden layer merupakan dense layer yang berada di antara input layer dan output layer. Berikut merupakan ilustrasi dari jaringan saraf yang memiliki 1 hidden layer (kiri) dan 4 hidden layer (kanan) :



Gambar 4.2. Ilustrasi Layer DNN

Alasan mengapa hidden layer dinamai dengan hidden karena sifatnya yang tersembunyi. Pada sebuah sistem jaringan saraf, input dan output layer merupakan lapisan yang dapat diamati, namun hidden layer tidak bisa. Semakin banyak hidden layer, maka jaringan saraf tersebut akan bisa menyelesaikan masalah yang kompleks.

4.2 Percobaan dengan Deep Neural Network

Pada bab sebelumnya kita telah belajar membuat model time series forecasting dengan neural network. Pada bab ini, kita akan menggunakan algoritma yang berbeda, yaitu deep neural network. Sebenarnya ada sedikit persamaan antara neural network dan deep neural network, yang membedakan yaitu jumlah layer yang digunakan pada modelnya. Selanjutnya langsung saja mari kita pelajari cara membuat model time series forecasting dengan deep neural network.

- **Mengimpor library dan Memeriksa versi Tensorflow**

Seperti pada percobaan sebelumnya, pertama kita perlu mengimpor library yang diperlukan dengan kode berikut :

```
import tensorflow as tf  
import numpy as np  
import matplotlib.pyplot as plt  
print(tf.__version__)
```

2.5.0

Gambar 4.3. Kode Impor Library dan Cek Versi Tensorflow

Dalam membuat model kali ini, kita memerlukan library tensorflow, numpy dan matplotlib. Dan tensorflow yang kita gunakan adalah tensorflow versi 2.5.0.

- **Membuat Fungsi Data Series**

Sama seperti sebelumnya, fungsi series yang akan kita buat juga sama. Berikut adalah kodenya :

```

def plot_series(time, series, format="-", start=0, end=None):
    plt.plot(time[start:end], series[start:end], format)
    plt.xlabel("Waktu")
    plt.ylabel("Nilai")
    plt.grid(True)

def trend(time, slope=0):
    return slope * time

def seasonal_pattern(season_time):
    return np.where(season_time < 0.4,
                   np.cos(season_time * 2 * np.pi),
                   1 / np.exp(3 * season_time))

def seasonality(time, period, amplitude=1, phase=0):
    season_time = ((time + phase) % period) / period
    return amplitude * seasonal_pattern(season_time)

def noise(time, noise_level=1, seed=None):
    rnd = np.random.RandomState(seed)
    return rnd.randn(len(time)) * noise_level

time = np.arange(4 * 365 + 1, dtype="float32")
baseline = 10
series = trend(time, 0.1)
baseline = 10
amplitude = 20
slope = 0.09
noise_level = 5

# Membuat Seriesnya
series = baseline + trend(time, slope) + seasonality(time,
                                                       period=365,
                                                       amplitude=amplitude)
# Menambahkan noise
series += noise(time, noise_level, seed=42)

```

Gambar 4.4. Kode Membuat Fungsi Data Series

Pada fungsi series kali ini penulis mengubah amplitudennya menjadi 20 dan nilai slopennya menjadi 0.09. Sebenarnya untuk fungsi series ini kalian bisa mengatur nilainya sesuai dengan kebutuhan anda.

- **Membagi Dataset**

Setelah membuat dan memberi nilai fungsi data series, selanjutnya Membagi datasetnya menjadi train dan validation. Tak

lupa kita juga mengatur nilai konstan untuk ukuran window dan batchnya. Berikut kodenya :

```
split_time = 1000
time_train = time[:split_time]
x_train = series[:split_time]
time_valid = time[split_time:]
x_valid = series[split_time:]

window_size = 20
batch_size = 32
shuffle_buffer_size = 1000
```

Gambar 4.5. Kode Membagi Dataset

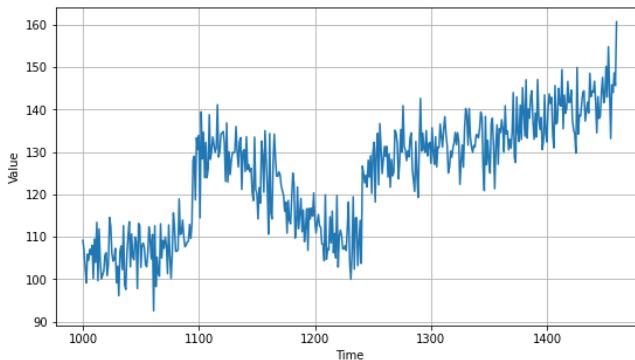
- **Membuat visualisasi dari time_valid dan x_valid**

Setelah membagi dataset, kita coba untuk membuat visualisasi series dari time_valid dan x_valid. Berikut kodenya :

```
plt.figure(figsize=(9, 5))
plot_series(time_valid, x_valid)
```

Gambar 4.6. Kode Visualisasi Data Series

Hasil Output :



Gambar 4.7. Hasil Visualisasi Data Series

- **Membuat Fungsi Windowed Dataset**

Dataset yang gunakan sama seperti windowed dataset sebelumnya. Berikut merupakan kodenya :

```

def windowed_dataset(series, window_size, batch_size, shuffle_buffer):
    dataset = tf.data.Dataset.from_tensor_slices(series)
    dataset = dataset.window(window_size + 1, shift=1, drop_remainder=True)
    dataset = dataset.flat_map(lambda window: window.batch(window_size + 1))
    dataset = dataset.shuffle(shuffle_buffer).map(lambda window:
                                                    (window[:-1], window[-1]))
    dataset = dataset.batch(batch_size).prefetch(1)
    return dataset

```

Gambar 4.8. Kode Membuat Windowed Dataset

- **Membuat Model dengan DNN**

Ini merupakan inti dari percobaan kita. Disini penulis membuat model DNN dengan 3 layer. Berikut kodennya :

```

#Memanggil Fungsi Windowed Dataset
dataset = windowed_dataset(x_train, window_size,
                            batch_size, shuffle_buffer_size)

#Membuat DNN Model
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(10, input_shape=[window_size], activation="relu"),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(1)
])

#Menampilkan simpulan model
model.summary()

```

Gambar 4.9. Kode Membuat Model DNN

Hasil :

```

Model: "sequential"
-----
Layer (type)          Output Shape       Param #
dense (Dense)         (None, 10)        210
dense_1 (Dense)        (None, 10)        110
dense_2 (Dense)        (None, 1)         11
-----
Total params: 331
Trainable params: 331
Non-trainable params: 0

```

Gambar 4.10. Hasil Summary Model DNN

Disini tf.keras.models.Sequential digunakan untuk membuat modelnya dan tf.keras.layers.Dense digunakan untuk membuat

layernya atau bisa disebut dengan hidden layer. Sedangkan untuk fungsi aktivasinya kita menggunakan relu.

- **Mentraining Model**

Setelah membuat model, langkah selanjutnya yaitu melakukan training pada model. Berikut kodenya :

```
model.compile(loss="mse",
              optimizer=tf.keras.optimizers.SGD(learning_rate=1e-6,
                                                momentum=0.9))
model.fit(dataset, epochs=100, verbose=0)
```

Gambar 4.11. Kode Mentraining Model

Pada compile model di gambar 4.11, kita menggunakan fungsi loss mean squared error, lalu kita tambahkan optimizer dan learning rate yang digunakan adalah 1e-6. Sedangkan kita mentraining modelnya dengan 100 epochs.

- **Membuat Visualisasi Hasil Prediksi**

Untuk membuat prediksi kita menggunakan model.predict, selanjutnya hasil prediksinya akan kita visualisasikan. Berikut kodenya :

```
prediksi = []
for time in range(len(series) - window_size):
    prediksi.append(model.predict(series[time:time + window_size]
                                  [np.newaxis]))

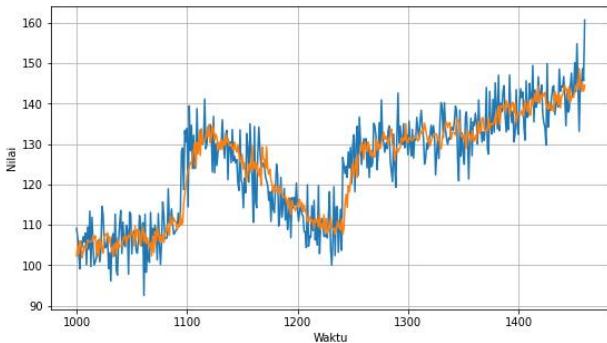
prediksi = prediksi[split_time-window_size:]
Hasil = np.array(prediksi)[:, 0, 0]

plt.figure(figsize=(9, 5))

plot_series(time_valid, x_valid)
plot_series(time_valid, Hasil)
```

Gambar 4.12. Kode Visualisasi Hasil Prediksi

Hasil plot :



Gambar 4.13. Visualisasi Hasil Prediksi

Ingat yang berwarna biru merupakan data aktualnya, sedangkan yang oranye merupakan hasil prediksinya.

- **Mengukur MAE**

Yang terakhir, kita mengukur metrik mean absolute errornya menggunakan `tf.keras.metrics.mean_absolute_error`. Berikut kodenya beserta hasil yang didapat :

```
tf.keras.metrics.mean_absolute_error(x_valid, Hasil).numpy()
```

```
4.9936485
```

Gambar 4.14. Kode dan Hasil MAE

4.3 Memperbaiki Model DNN

Sebelumnya kita sudah berhasil membuat model time series forecasting dengan Deep Neural Network. Namun, hasil metrik yang kita dapatkan masih berkisar di angka 4,99 . Pada materi kali ini, kita akan memperbaiki modelnya sehingga hasil metriknya MAEnya lebih rendah, dan prediksinya menjadi lebih akurat. Baik langsung saja kita bahas bagaimana cara memperbaiki modelnya.

- **Model Sebelumnya**

Dalam memperbaiki model kali ini, kita masih sama menggunakan 3 layer dan aktivasinya menggunakan “relu”. Sehingga kodennya tetap seperti berikut :

```
dataset = windowed_dataset(x_train, window_size,
                           batch_size, shuffle_buffer_size)

model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(10, input_shape=[window_size],
                          activation="relu"),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(1)
])

model.summary()
```

Gambar 4.15. Kode Model Sebelumnya

- **Menambahkan Callback dan Training Model**

Untuk membuat modelnya jadi lebih efisien, disini perlu ditambahkan callback pada modelnya. `tf.keras.callbacks.LearningRateScheduler` ini berfungsi untuk mempercepat learning ratenya. Berikut merupakan kodennya :

```
#Membuat callback learning rate
lr_schedule = tf.keras.callbacks.LearningRateScheduler(
    lambda epoch: 1e-8 * 10**(epoch / 20))
#Membuat optimizer
optimizer = tf.keras.optimizers.SGD(learning_rate=1e-8, momentum=0.9)

#Mengcompile Model
model.compile(loss="mse", optimizer=optimizer)

#Menraining Model
history = model.fit(dataset, epochs=100, callbacks=[lr_schedule], verbose=0)
```

Gambar 4.16. Kode Menambahkan Callback dan Training Model

Callback ini nantinya akan dipanggil di akhir setiap epoch. Disitu kita akan mengganti nilai learning ratenya berdasarkan nomor epochnya. Jadi epoch 1, $1e-8 \cdot 10^{10}$ kali 10^{10} pangkat 1 di atas 20. Disaat kita mencapai 100 epoch, itu akan menjadi $1e-8 \cdot 10^{10}$

pangkat 5, dan itu 100 di atas 20. Ini akan terjadi pada setiap callback, karena kita mengurnya di parameter modelnya.

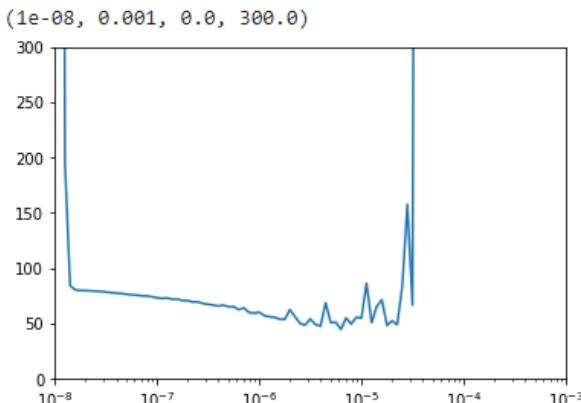
- **Memvisualisasikan Kecepatan Epoch**

Setelah mentraining modelnya, kita dapat membuat plot atau visualisasi kecepatan learning rate per epoch dengan kode berikut :

```
lrs = 1e-8 * (10 ** (np.arange(100) / 20))
plt.semilogx(lrs, history.history["loss"])
plt.axis([1e-8, 1e-3, 0, 300])
```

Gambar 4.17. Kode Visualiasi Kecepatan Epoch

Hasilnya :



Gambar 4.18. Hasil Visualiasi Kecepatan Epoch

Grafik pada gambar 4.18, menunjukkan bahwa terdapat sumbu y dan sumbu x. Sumbu y menunjukkan loss dari setiap epoch, sedangkan sumbu x menunjukkan learning ratenya. Kemudian dari situ kita pilih titik terendah dari kurvanya.

- **Membuat Model Baru**

Selanjutnya kita buat model baru lagi. Modelnya tetap sama memakai 3 layer, namun disini kita ubah nilai learning ratenya.

Pakai nilai learning rate titik kurva paling bawah sesuai dengan grafik sebelumnya. Disini penulis mengubah nilai learning ratenya menjadi 7e-6 dan kita akan mentrainingnya lebih lama, kita akan mentrainingnya menjadi 500 epoch. Jadi, berikut kodenya :

```
window_size = 30
dataset = windowed_dataset(x_train, window_size,
                           batch_size, shuffle_buffer_size)

model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(10, activation="relu", input_shape=[window_size]),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(1)
])

optimizer = tf.keras.optimizers.SGD(learning_rate=8e-6,
                                    momentum=0.9)
model.compile(loss="mse", optimizer=optimizer)
history = model.fit(dataset, epochs=500, verbose=0)
```

Gambar 4.19. Kode Membuat Model Baru

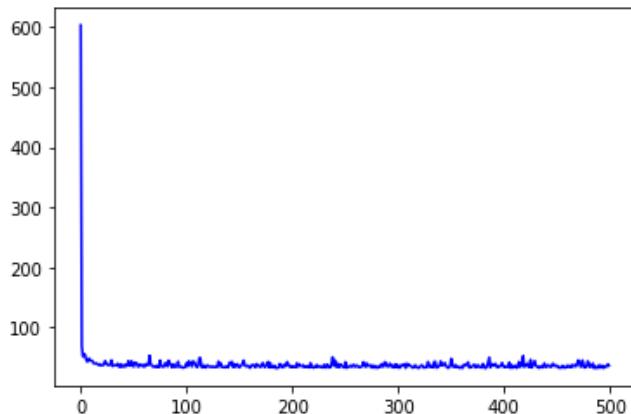
- **Memvisualisasikan Hasil Loss**

Setelah mentraining modelnya, mari kita lihat hasil visualisasi lossnya dengan kode berikut :

```
loss = history.history['loss']
epochs = range(len(loss))
plt.plot(epochs, loss, 'b', label='Training Loss')
plt.show()
```

Gambar 4.20. Kode Visualisasi Hasil Loss

Hasilnya :



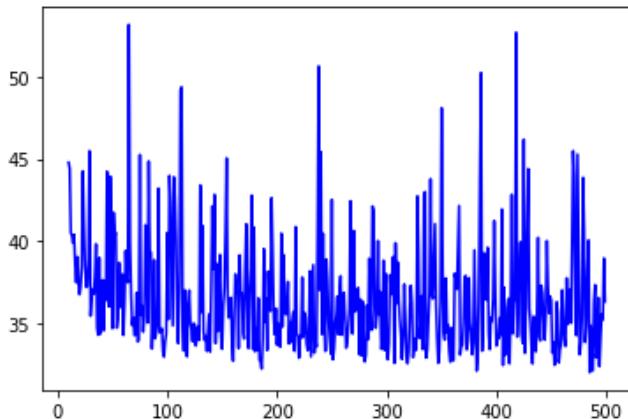
Gambar 4.21. Visualisasi Hasil Loss

Berdasarkan grafik pada gambar 4.21, bisa kita lihat bahwa nilai lossnya saat di awal cukup tinggi. Maka mari kita lihat lebih detail hasil loss 10 epoch pertama dengan kode berikut :

```
loss = history.history['loss']
epochs = range(10, len(loss))
plot_loss = loss[10:]
print(plot_loss)
plt.plot(epochs, plot_loss, 'b', label='Training Loss')
plt.show()
```

Gambar 4.22. Kode Visualisasi Hasil Loss Detail

Hasilnya :



Gambar 4.23. Visualisasi Hasil Loss Detail

Bisa kita lihat lebih detail bahwa hasil loss cenderung lebih menurun bahkan setelah 500 epoch. Hal itu menunjukkan bahwa model DNN kita bekerja dengan cukup baik.

- **Memvisualisasikan Hasil Prediksi**

Selanjutnya mari kita lihat visualisasi hasil prediksinya dengan kode berikut :

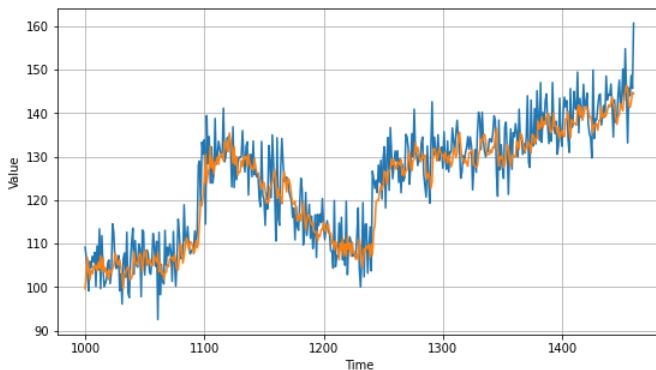
```
prediksi = []
for time in range(len(series) - window_size):
    prediksi.append(model.predict(series[time:time + window_size]
                                  [np.newaxis]))

prediksi = prediksi[split_time-window_size:]
Hasil = np.array(prediksi)[:, 0, 0]

plt.figure(figsize=(9, 5))
plot_series(time_valid, x_valid)
plot_series(time_valid, Hasil)
```

Gambar 4.24. Kode Visualisasi Hasil Prediksi

Hasilnya :



Gambar 4.25. Visualisasi Hasil Prediksi

Bisa dilihat bahwa hasil prediksi kita jauh lebih baik dari hasil sebelumnya. Disini, garis oranye terlihat lebih pas dengan garis birunya. Artinya hasil prediksinya hampir sama dengan data aktualnya.

- **Mengukur Hasil MAE**

Seperti biasa, langkah akhir yaitu mengukur hasil metrik MAE. Berikut kode beserta hasilnya :

```
tf.keras.metrics.mean_absolute_error(x_valid, Hasil).numpy()
```

```
4.627793
```

Gambar 4.26. Kode dan Hasil MAE

Dari hasil pada gambar 4.26, terbukti bahwa model kita bekerja dengan lebih baik. Angka MAEnya lebih rendah dari sebelumnya. Dimana sebelumnya kita memperoleh nilai 4.9936485 sekarang turun menjadi 4.627793. Namun, harus kita ingat sebelumnya bahwa nilai MAE ini berbeda-beda saat kita

mentraining modelnya, karena data yang kita gunakan acak. Tapi biasanya hasilnya tidak akan jauh berbeda.

4.4 Kode Keseluruhan DNN

Kita telah mempelajari langkah-langkah untuk membuat model time series forecasting dengan Deep Neural Network (DNN). Disini penulis akan mencantumkan kode keseluruhan dari notebooknya, agar kalian bisa mencocokkan apakah kodennya sudah sesuai apa belum. Berikut merupakan kode keseluruhan pada bab ini.

Percobaan Dengan Deep Neural Network

Mengimpor Library dan Memeriksa Versi Tensorflow

```
[ ] import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
print(tf.__version__)

2.5.0
```

Membuat Fungsi Data Series

```
[ ] def plot_series(time, series, format="-", start=0, end=None):
    plt.plot(time[start:end], series[start:end], format)
    plt.xlabel("Waktu")
    plt.ylabel("Nilai")
    plt.grid(True)

def trend(time, slope=0):
    return slope * time

def seasonal_pattern(season_time):
    return np.where(season_time < 0.4,
                   np.cos(season_time * 2 * np.pi),
                   1 / np.exp(3 * season_time))

def seasonality(time, period, amplitude=1, phase=0):
    season_time = ((time + phase) % period) / period
    return amplitude * seasonal_pattern(season_time)

def noise(time, noise_level=1, seed=None):
    rnd = np.random.RandomState(seed)
    return rnd.randn(len(time)) * noise_level
```

```

time = np.arange(4 * 365 + 1, dtype="float32")
baseline = 10
series = trend(time, 0.1)
baseline = 10
amplitude = 20
slope = 0.09
noise_level = 5

# Membuat Seriesnya
series = baseline + trend(time, slope) + seasonality(time,
                                                       period=365,
                                                       amplitude=amplitude)
# Menambahkan noise
series += noise(time, noise_level, seed=42)

```

Membagi Dataset

```

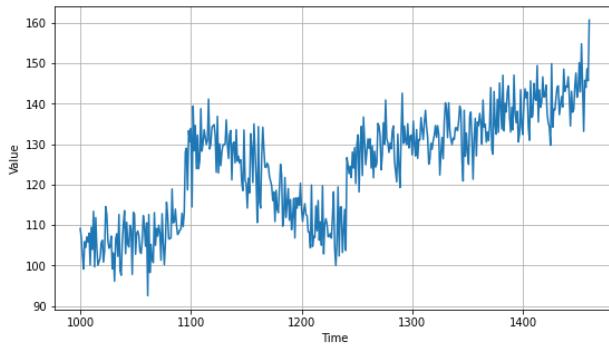
[ ] split_time = 1000
time_train = time[:split_time]
x_train = series[:split_time]
time_valid = time[split_time:]
x_valid = series[split_time:]

window_size = 20
batch_size = 32
shuffle_buffer_size = 1000

```

Membuat Visualisasi

```
[ ] plt.figure(figsize=(9, 5))
plot_series(time_valid, x_valid)
```



Membuat Fungsi Windowed Dataset

```
[ ] def windowed_dataset(series, window_size, batch_size, shuffle_buffer):
    dataset = tf.data.Dataset.from_tensor_slices(series)
    dataset = dataset.window(window_size + 1, shift=1, drop_remainder=True)
    dataset = dataset.flat_map(lambda window: window.batch(window_size + 1))
    dataset = dataset.shuffle(shuffle_buffer).map(lambda window:
                                                (window[:-1], window[-1]))
    dataset = dataset.batch(batch_size).prefetch(1)
    return dataset
```

Membuat Model dengan DNN

```
[ ] #Memanggil Fungsi Windowed Dataset
dataset = windowed_dataset(x_train, window_size,
                           batch_size, shuffle_buffer_size)

#Membuat DNN Model
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(10, input_shape=[window_size], activation="relu"),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(1)
])

#Menampilkan simpulan model
model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
<hr/>		
dense_3 (Dense)	(None, 10)	210
dense_4 (Dense)	(None, 10)	110
dense_5 (Dense)	(None, 1)	11
<hr/>		
Total params:	331	
Trainable params:	331	
Non-trainable params:	0	

Mentraining Model

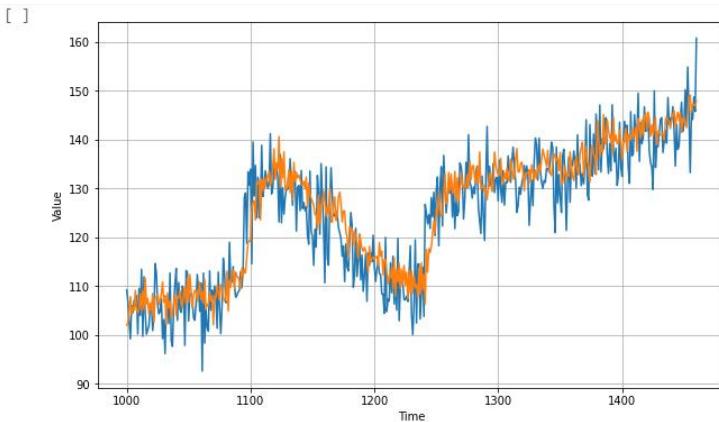
```
[ ] model.compile(loss="mse",
                  optimizer=tf.keras.optimizers.SGD(learning_rate=1e-6,
                                                   momentum=0.9))
model.fit(dataset, epochs=100, verbose=0)

<tensorflow.python.keras.callbacks.History at 0x7f880df86c10>
```

Membuat Visualisasi hasil prediksi

```
[ ] prediksi = []
for time in range(len(series) - window_size):
    prediksi.append(model.predict(series[time:time + window_size]
                                  [np.newaxis]))
prediksi = prediksi[split_time-window_size:]
Hasil = np.array(prediksi)[:, 0, 0]

plt.figure(figsize=(9, 5))
plot_series(time_valid, x_valid)
plot_series(time_valid, Hasil)
```



Mengukur hasil MAE

```
[ ] tf.keras.metrics.mean_absolute_error(x_valid, Hasil).numpy()
4.9936485
```

Memperbaiki Model dengan Deep Neural Network

Model Sebelumnya

```
[ ] dataset = windowed_dataset(x_train, window_size,
                                batch_size, shuffle_buffer_size)

model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(10, input_shape=[window_size],
                          activation="relu"),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(1)
])

model.summary()
Model: "sequential_2"

Layer (type)                 Output Shape              Param #
=====
dense_6 (Dense)             (None, 10)               210
dense_7 (Dense)             (None, 10)               110
dense_8 (Dense)             (None, 1)                11
=====
Total params: 331
Trainable params: 331
Non-trainable params: 0
```

Menambahkan Callback dan Training Model

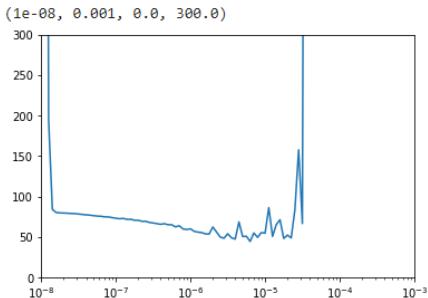
```
[ ] #Membuat callback learning rate
lr_schedule = tf.keras.callbacks.LearningRateScheduler(
    lambda epoch: 1e-8 * 10**(epoch / 20))
#Membuat optimizer
optimizer = tf.keras.optimizers.SGD(learning_rate=1e-8, momentum=0.9)

#Mengcompile Model
model.compile(loss="mse", optimizer=optimizer)

#Mentraining Model
history = model.fit(dataset, epochs=100, callbacks=[lr_schedule], verbose=0)
```

Memvisualisasikan Kecepatan Epoch

```
[ ] lrs = 1e-8 * (10 ** (np.arange(100) / 20))
plt.semilogx(lrs, history.history["loss"])
plt.axis([1e-8, 1e-3, 0, 300])
```



Membuat Model Baru

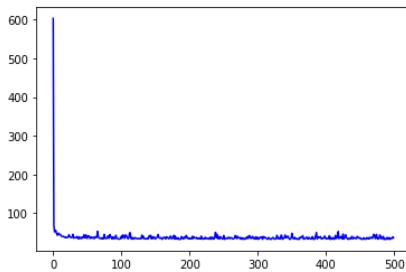
```
[ ] window_size = 30
dataset = windowed_dataset(x_train, window_size,
                           batch_size, shuffle_buffer_size)

model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(10, activation="relu", input_shape=[window_size]),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(1)
])

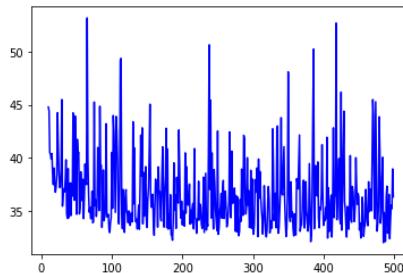
optimizer = tf.keras.optimizers.SGD(learning_rate=8e-6,
                                    momentum=0.9)
model.compile(loss="mse", optimizer=optimizer)
history = model.fit(dataset, epochs=500, verbose=0)
```

Memvisualisasikan Hasil Loss

```
[ ] loss = history.history['loss']
epochs = range(len(loss))
plt.plot(epochs, loss, 'b', label='Training Loss')
plt.show()
```



```
[ ] loss = history.history['loss']
epochs = range(10, len(loss))
plot_loss = loss[10:]
print(plot_loss)
plt.plot(epochs, plot_loss, 'b', label='Training Loss')
plt.show()
```

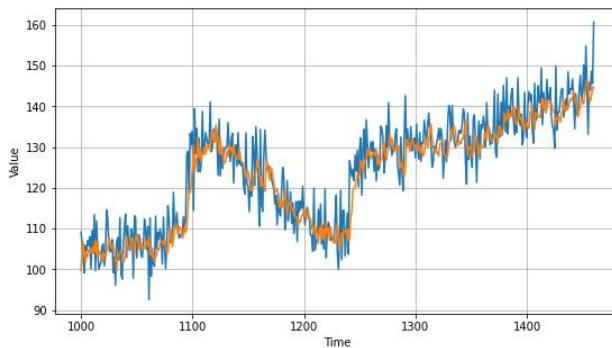


Memvisualisasikan Hasil Prediksi

```
[ ] prediksi = []
for time in range(len(series) - window_size):
    prediksi.append(model.predict(series[time:time + window_size]
                                  [np.newaxis]))
prediksi = prediksi[split_time-window_size:]
Hasil = np.array(prediksi)[:, 0, 0]

plt.figure(figsize=(9, 5))

plot_series(time_valid, x_valid)
plot_series(time_valid, Hasil)
```



Mengukur Hasil MAE

```
[ ] tf.keras.metrics.mean_absolute_error(x_valid, Hasil).numpy()  
4.627793
```

Gambar 4.27. Kode Keseluruhan Bab 4

BAB 5

TIME SERIES FORECASTING DENGAN RECURRENT NEURAL NETWORK (RNN)

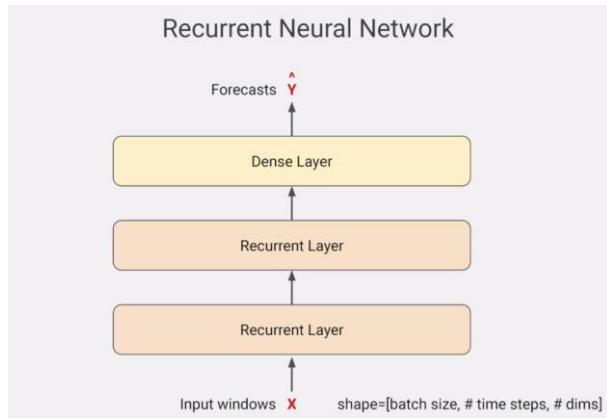
5.1. Konsep RNN

Pada bab sebelumnya kita sudah mempelajari cara membuat model time series forecasting dengan Deep Neural Network. Namun, time series merupakan data temporal, yang berarti kita harus menerapkan sequence model seperti RNN agar hasil prediksi kita menjadi lebih akurat. Oleh karena itu, pada bab ini kita akan mempelajari cara membuat model time series forecasting dengan Recurrent Neural Network (RNN).

5.1.1. Pengertian RNN

Seperti kita tahu bahwa RNN ini merupakan salah satu algoritma dari Deep Learning. Recurrent Neural Network (RNN) merupakan neural network atau jaringan saraf tiruan dimana layernya diproses secara berulang-ulang, oleh karena itu algoritma ini dinamai recurrent[6].

RNN dirancang untuk memproses urutan input secara berurutan. RNN ini cukup fleksibel, mampu memproses semua jenis urutan (sequential). Berikut merupakan ilustrasi konsep dari RNN :



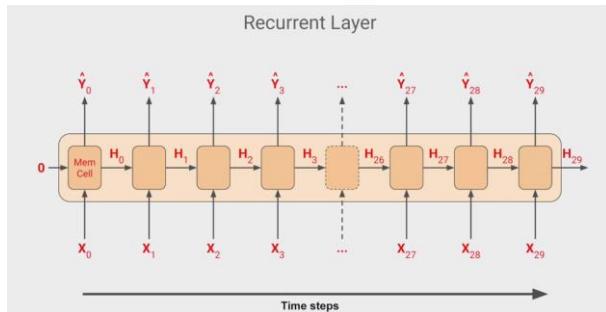
Gambar 5.1. Ilustrasi Konsep RNN

Konsep RNN pada gambar 5.1 terdiri dari 2 recurrent layer dan sebuah dense layer, yang dimana nantinya akan berfungsi sebagai output. Dengan RNN, kita bisa memasukkan batch ke dalam sequence, dan nantinya akan menampilkan hasil forecast atau prediksi.

Hal yang membedakan jika menggunakan RNN adalah bentuknya akan menjadi dimensional tiga. Dimensi pertama adalah batch size, dimensi yang kedua adalah time steps, dan dimensi ketiga adalah dimensi input pada setiap time step. Contohnya, pada univariate time series, nilainya akan menjadi satu, sedangkan multivariate nilainya akan lebih dari satu.

5.1.2. Cara Kerja RNN

Selanjutnya mari kita cari tau cara kerja dari RNN. Berikut merupakan ilustrasi dari cara kerja RNN :



Gambar 5.2. Cara Kerja RNN

Pada gambar 5.2, terlihat bahwa ada banyak cell (kotak). Namun, sebenarnya hanya ada satu cell dan itu digunakan berulang kali untuk menghitung output. Pada setiap time step, memori cell mengambil nilai input untuk langkah tersebut.

Jadi misalnya, itu nilainya 0 pada waktu ke 0, dan input dalam keadaan 0. Kemudian menghitung output untuk langkah itu, dalam hal ini Y_0 dan vektor H_0 yang akan dimasukkan ke langkah berikutnya. H_0 diumpulkan ke dalam cell dengan X_1 untuk menghasilkan Y_1 dan H_1 , yang kemudian diumpulkan ke dalam cell pada langkah berikutnya dengan X_2 untuk meghasilkan Y_2 dan H_2 .

Langkah ini berlanjut terus sampai kita mencapai akhir dimensi input, dimana dalam gambar 5.2, memiliki 30 nilai. Ini lah sebab mengapa algoritma ini dinamai dengan recurrent, karena proses dilakukan secara berulang.

5.2. Percobaan dengan RNN

Sebelumnya kita telah mempelajari cara membuat model time series forecasting dengan Neural Network dan Deep Neural Network. Kali ini kita akan menggunakan metode algoritma baru, yaitu

Recurrent Neural Network. Langkah-langkah yang digunakan hampir sama dengan metode deep learning lainnya, namun yang membedakan hanya kode di bagian modelnya. Baik langsung saja kita pelajari caranya.

- **Mengimpor Library dan Memeriksa Versi Tensorflow**

Seperti biasa, pertama kita perlu mengimpor library yang diperlukan, yaitu library tensorflow, numpy dan matplotlib. Setelah itu kita memeriksa versi dari tensorflow. Berikut merupakan kodennya :

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
print(tf.__version__)
```

Gambar 5.3. Kode Impor Library dan Cek Versi Tensorflow

- **Membuat Bentuk Fungsi Data Series**

Selanjutnya kita perlu membuat fungsi untuk pola data seriesnya. Disini ada 4 bentuk pola, yaitu trend, seasonal, seasonality dan noise. Kita juga mengatur nilai dari time, baseline, series, amplitude, slope dan level dari noisenya. Berikut merupakan kodennya :

```
def plot_series(time, series, format="-", start=0, end=None):
    plt.plot(time[start:end], series[start:end], format)
    plt.xlabel("Waktu")
    plt.ylabel("Nilai")
    plt.grid(True)
```

```

def trend(time, slope=0):
    return slope * time

def seasonal_pattern(season_time):
    return np.where(season_time < 0.4,
                   np.cos(season_time * 2 * np.pi),
                   1 / np.exp(3 * season_time))

def seasonality(time, period, amplitude=1, phase=0):
    season_time = ((time + phase) % period) / period
    return amplitude * seasonal_pattern(season_time)

def noise(time, noise_level=1, seed=None):
    rnd = np.random.RandomState(seed)
    return rnd.randn(len(time)) * noise_level

time = np.arange(4 * 365 + 1, dtype="float32")
baseline = 10
series = trend(time, 0.1)
baseline = 10
amplitude = 40
slope = 0.05
noise_level = 5

#Membuat Series
series = baseline + trend(time, slope) + seasonality(time,
                                                       period=365,
                                                       amplitude=amplitude)
#Menambahkan Noise
series += noise(time, noise_level, seed=42)

```

Gambar 5.4. Kode Membuat Fungsi Data Series

- **Membagi Dataset**

Disini kita membagi datanya menjadi data train dan validasi. Pertama kita split menjadi 1000, lalu buat variabel time_train, x_train, time_valid, dan x_valid. Setelah itu, kita mengatur nilai window size, batch size, dan shuffle buffer. Berikut merupakan kodennya :

```
split_time = 1000
time_train = time[:split_time]
x_train = series[:split_time]
time_valid = time[split_time:]
x_valid = series[split_time:]

window_size = 20
batch_size = 32
shuffle_buffer_size = 1000
```

Gambar 5.5. Kode Membagi Dataset

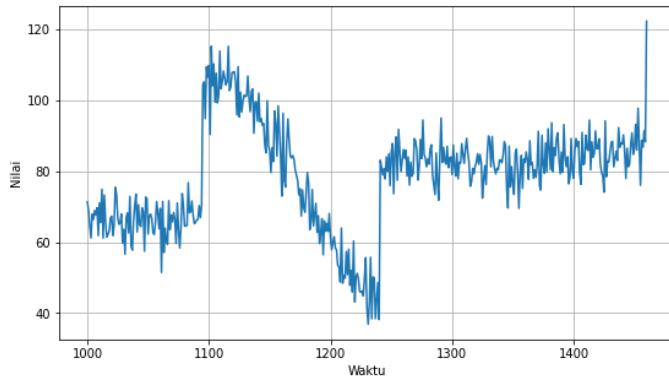
- **Memvisualisasikan Data Series**

Selanjutnya kita memvisualisasikan time valid (time) dan x valid (series) ke dalam bentuk grafik. Disini kita menggunakan matplotlib. Berikut merupakan kodenya :

```
plt.figure(figsize=(9, 5))
plot_series(time_valid, x_valid)
```

Gambar 5.6. Kode Memvisualisasikan Data Series

Hasil :



Gambar 5.7. Visualisasi Data Series

- **Membuat Windowed Dataset**

Cara membuat windowed dataset ini sudah dijelaskan di bab sebelumnya. Disini kita membuat datasetnya dengan menggunakan tf.data.Dataset. Berikut merupakan kodenya :

```
def windowed_dataset(series, window_size, batch_size, shuffle_buffer):
    dataset = tf.data.Dataset.from_tensor_slices(series)
    dataset = dataset.window(window_size + 1, shift=1, drop_remainder=True)
    dataset = dataset.flat_map(lambda window: window.batch(window_size + 1))
    dataset = dataset.shuffle(shuffle_buffer).map(lambda window: (window[:-1], window[-1]))
    dataset = dataset.batch(batch_size).prefetch(1)
    return dataset
```

Gambar 5.8. Kode Membuat Windowed Dataset

- **Membuat Model dengan RNN**

Langkah pembuatan time series yang berbeda terletak pada kode modelnya. Kali ini model yang kita gunakan menggunakan RNN. Berikut merupakan kodenya :

```
tf.keras.backend.clear_session()
tf.random.set_seed(51)
np.random.seed(51)

#Memanggil Fungsi Windowed Dataset
train_set = windowed_dataset(x_train, window_size, batch_size=128,
                             shuffle_buffer=shuffle_buffer_size)
#Membuat Sequential Model
model = tf.keras.models.Sequential([
    tf.keras.layers.Lambda(lambda x: tf.expand_dims(x, axis=-1),
                          input_shape=[None]), #Membuat Lambda Layer
    #Membuat RNN Layer
    tf.keras.layers.SimpleRNN(40, return_sequences=True),
    tf.keras.layers.SimpleRNN(40),
    tf.keras.layers.Dense(1), #Membuat Dense Layer
    tf.keras.layers.Lambda(lambda x: x * 100.0) #Membuat Lambda Layer
])
#Membuat Callback
lr_schedule = tf.keras.callbacks.LearningRateScheduler(
    lambda epoch: 1e-8 * 10**(epoch / 20))
optimizer = tf.keras.optimizers.SGD(learning_rate=1e-8,
                                    momentum=0.9)
#Mengcompile Model
model.compile(loss=tf.keras.losses.Huber(),
              optimizer=optimizer,
              metrics=['mae'])
```

Gambar 5.9. Kode Membuat Model RNN

Pada pembuatan model kali ini, pertama kita menggunakan `tf.keras.backend.clear_session()` untuk menghemat ukuran memori. Lalu `tf.random.set_seed` dan `np.random.seed()` berguna untuk mengatur global random seednya. Setelah itu kita membuat variabel training set yang berisi pemanggilan dataset yang akan ditraining.

Lalu kita membuat modelnya dengan `tf.keras.models.Sequential`. Disini yang membedakan yaitu `tf.keras.layers.SimpleRNN` yang berfungsi untuk memproses layernya dengan metode RNN, `return_sequences` di set True agar inputnya diproses beberapa kali sampai selesai. Kemudian tak lupa kita gunakan learning rate dan optimizer untuk mempercepat proses kinerja model. Lalu yang terakhir kita compile modelnya dengan `model.compile()`.

- **Mentraining Model**

Setelah selesai membuat model, langkah selanjutnya yaitu mentraining model. Kali ini penulis mengatur epochnya 100. Berikut merupakan kodennya :

```
history = model.fit(train_set, epochs=100, callbacks=[lr_schedule])
```

Gambar 5.10. Kode Mentraining Model RNN

- **Memvisualisasikan Hasil Prediksi**

Kode yang digunakan untuk memvisualisasikan hasil prediksi sama dengan sebelumnya. Jadi menggunakan matplotlib. Berikut merupakan kodennya :

```

prediksi = []
for time in range(len(series) - window_size):
    prediksi.append(model.predict(series[time:time + window_size]
                                [np.newaxis]))

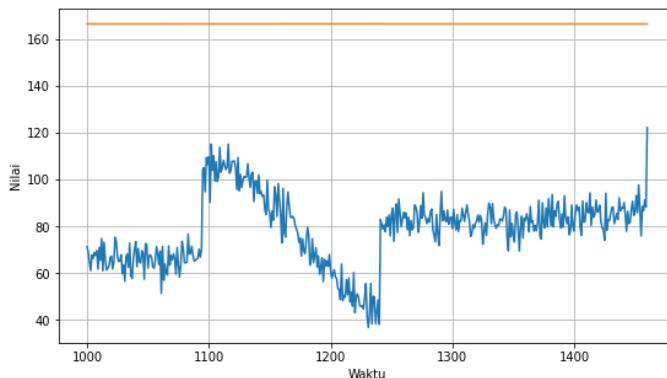
prediksi = prediksi[split_time-window_size:]
hasil = np.array(prediksi)[:, 0, 0]

plt.figure(figsize=(9, 5))
plot_series(time_valid, x_valid)
plot_series(time_valid, hasil)

```

Gambar 5.11. Kode Visualsasi Prediksi Model RNN

Hasil :



Gambar 5.12. Visualsasi Prediksi Model RNN

Berdasarkan hasil pada gambar 5.12, dapat kita ketahui bahwa prediksi kita sangat jauh dari data aktual. Hal ini menunjukkan bahwa model kita belum bekerja dengan baik. Selanjutnya mari kita periksa berapa hasil metrik MAE nya.

- **Mengukur Hasil MAE**

Dari gambar hasil prediksi sebelumnya, kita ketahui bahwa prediksinya sangat jauh dari data aktual. Namun berapakah sebenarnya nilai lossnya. Mari kita ukur dengan kode berikut :

```
tf.keras.metrics.mean_absolute_error(x_valid, hasil).numpy()  
88.18091
```

Gambar 5.13. Kode dan Hasil MAE

Ternyata nilai metrik MAEnya sangat tinggi, mencapai angka 88. Namun, kita masih bisa memperbaiki modelnya. Pada sub bab selanjutnya akan kita bahas tentang memperbaiki modelnya.

5.3. Memperbaiki Model RNN

Pada sub bab sebelumnya kita telah berhasil membuat model time series forecasting dengan RNN. Namun, model yang kita buat ternyata masih kurang bagus. Hasil metrik MAEnya terlalu tinggi. Oleh karena itu disini kita akan mencoba untuk memperbaiki modelnya.

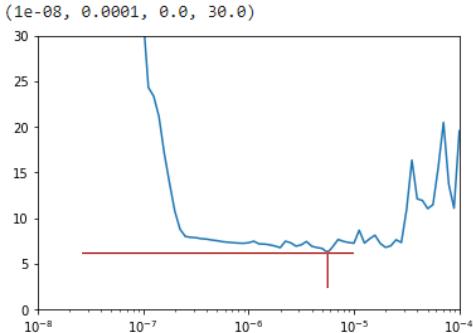
- **Mencari Nilai Optimum Learning Rate**

Salah satu mengapa model kita kurang bagus bisa jadi karena nilai learning rate yang digunakan tidak sesuai, sehingga prediksi modelnya kurang bagus. Selanjutnya kita akan mencari tahu nilai optimum learning ratenya dengan kode berikut :

```
plt.semilogx(history.history["lr"], history.history["loss"])  
plt.axis([1e-8, 1e-4, 0, 30])
```

Gambar 5.14. Kode Mencari Nilai Optimum

Hasil :



Gambar 5.15. Hasil Nilai Optimum

Cara mencari nilai optimum learningnya yaitu dengan melihat titik terbawah dari kurvanya. Sesuai garis oranye, bisa kita ketahui bahwa nilai optimumnya berada di kisaran angka 6e-5. Setelah ini langsung saja kita buat model baru.

- **Membuat Model Baru**

Model yang akan kita buat, kodennya sama dengan kode membuat model RNN sebelumnya. Namun, yang berbeda yaitu disini kita mengganti nilai learning ratenya. Sebelumnya bernilai 1e-8 diganti dengan 6e-5 sesuai dengan nilai optimum. Baik berikut merupakan kodennya :

```

tf.keras.backend.clear_session()
tf.random.set_seed(51)
np.random.seed(51)

dataset = windowed_dataset(x_train, window_size, batch_size=128,
                           shuffle_buffer=shuffle_buffer_size)

model = tf.keras.models.Sequential([
    tf.keras.layers.Lambda(lambda x: tf.expand_dims(x, axis=-1),
                           input_shape=[None]),
    tf.keras.layers.SimpleRNN(40, return_sequences=True),
    tf.keras.layers.SimpleRNN(40),
    tf.keras.layers.Dense(1),
    tf.keras.layers.Lambda(lambda x: x * 100.0)
])

optimizer = tf.keras.optimizers.SGD(learning_rate=6e-5, momentum=0.9)
model.compile(loss=tf.keras.losses.Huber(),
              optimizer=optimizer,
              metrics=[ "mae"])

```

Gambar 5.16. Kode Membuat Perbaikan Model RNN

- **Mentraining Model**

Setelah membuat model terbaru, langkah selanjutnya yaitu mentraining modelnya. Kali ini epochnya diganti menjadi 400 kali. Berikut kodenya :

```
history = model.fit(dataset, epochs=400)
```

Gambar 5.17. Kode Mentraining Model RNN

- **Memvisualisasikan Hasil Prediksi**

Selanjutnya kita buat visualisasi hasil prediksinya untuk mengetahui apakah model kita sudah bekerja dengan baik atau belum. Seperti biasa, berikut kodenya :

```

prediksi=[]
for time in range(len(series) - window_size):
    prediksi.append(model.predict(series[time:time + window_size]
                                  [np.newaxis]))

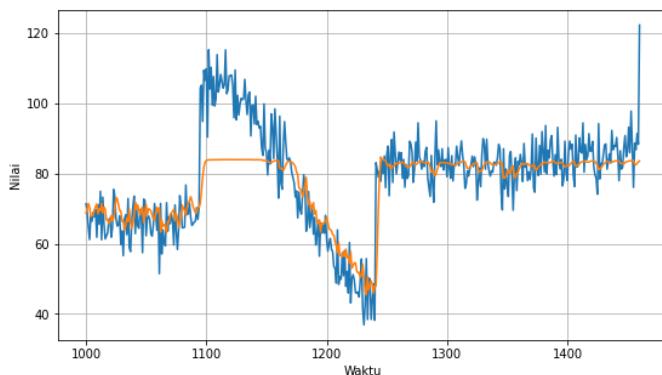
prediksi = prediksi[split_time-window_size:]
hasil = np.array(prediksi)[:, 0, 0]

plt.figure(figsize=(9, 5))
plot_series(time_valid, x_valid)
plot_series(time_valid, hasil)

```

Gambar 5.18. Kode Visualisasi Hasil Prediksi

Hasil :



Gambar 5.19. Visualisasi Hasil Prediksi

Berdasarkan hasil pada gambar 5.19, bisa dikatakan bahwa prediksinya cukup bagus. Walaupun di nilai waktu 1100 dan 1200 hasil kurva tren nya kurang pas. Untuk mengetahui nilainya, mari kita lihat hasil metrik MAE nya.

- **Menghitung Hasil MAE**

Berdasarkan hasil pada gambar 5.19, terlihat bahwa prediksinya cukup bagus. Selanjutnya, mari kita cek nilai metrik MAE nya untuk mengetahui tingkat errornya. Berikut kodennya :

```
tf.keras.metrics.mean_absolute_error(x_valid, hasil).numpy()
```

```
6.4636235
```

Gambar 5.20. Kode dan Hasil MAE

Ternyata hasil MAEnya cukup bagus. Nilai 6 bisa dikatakan tidak terlalu tinggi, dibanding sebelumnya yaitu hasil MAEnya mencapai 88. Oleh karena itu disini bisa disimpulkan bahwa penting bagi kita untuk menentukan nilai dari learning ratenya.

5.4. Kode Keseluruhan RNN

Disini penulis akan menampilkan kode keseluruhan cara membuat model time series forecasting dengan Recurrent Neural Network (RNN). Buat kalian yang sudah mencoba praktek, boleh mencocoknya hasilnya dengan kode keseluruhan berikut.

Percobaan Time Series Forecasting dengan RNN

Mengimpor library dan Memeriksa Versi Tensorflow

```
[ ] import tensorflow as tf  
import numpy as np  
import matplotlib.pyplot as plt  
print(tf.__version__)
```

2.5.0

Membuat Bentuk Fungsi Data Series

```
[ ] def plot_series(time, series, format="-", start=0, end=None):  
    plt.plot(time[start:end], series[start:end], format)  
    plt.xlabel("Waktu")  
    plt.ylabel("Nilai")  
    plt.grid(True)  
  
def trend(time, slope=0):  
    return slope * time  
  
def seasonal_pattern(season_time):  
    return np.where(season_time < 0.4,  
                   np.cos(season_time * 2 * np.pi),  
                   1 / np.exp(3 * season_time))  
  
def seasonality(time, period, amplitude=1, phase=0):  
    season_time = ((time + phase) % period) / period  
    return amplitude * seasonal_pattern(season_time)
```

```

def noise(time, noise_level=1, seed=None):
    rnd = np.random.RandomState(seed)
    return rnd.randn(len(time)) * noise_level

time = np.arange(4 * 365 + 1, dtype="float32")
baseline = 10
series = trend(time, 0.1)
baseline = 10
amplitude = 40
slope = 0.05
noise_level = 5

#Membuat Series
series = baseline + trend(time, slope) + seasonality(time,
                                                       period=365,
                                                       amplitude=amplitude)
#Menambahkan Noise
series += noise(time, noise_level, seed=42)

```

Membagi Dataset

```

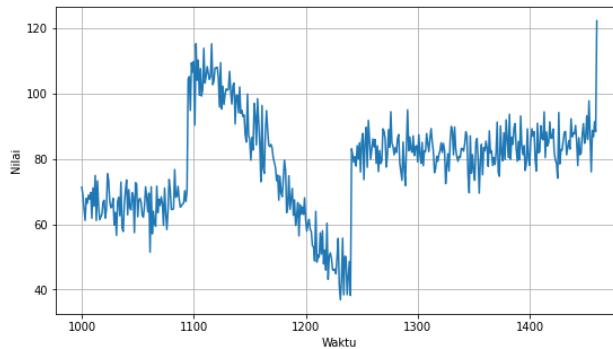
[ ] split_time = 1000
time_train = time[:split_time]
x_train = series[:split_time]
time_valid = time[split_time:]
x_valid = series[split_time:]

window_size = 20
batch_size = 32
shuffle_buffer_size = 1000

```

Memvisualisasikan Data

```
[ ] plt.figure(figsize=(9, 5))
plot_series(time_valid, x_valid)
```



Membuat Windowed Dataset

```
[ ] def windowed_dataset(series, window_size, batch_size, shuffle_buffer):
    dataset = tf.data.Dataset.from_tensor_slices(series)
    dataset = dataset.window(window_size + 1, shift=1, drop_remainder=True)
    dataset = dataset.flat_map(lambda window: window.batch(window_size + 1))
    dataset = dataset.shuffle(shuffle_buffer).map(lambda window: (window[:-1],
                                                               window[-1]))
    dataset = dataset.batch(batch_size).prefetch(1)
    return dataset
```

Membuat Model dengan RNN

```
[ ] tf.keras.backend.clear_session()
tf.random.set_seed(51)
np.random.seed(51)

#Memanggil Fungsi Windowed Dataset
train_set = windowed_dataset(x_train, window_size, batch_size=128,
                             shuffle_buffer=shuffle_buffer_size)

#Membuat Sequential Model
model = tf.keras.models.Sequential([
    tf.keras.layers.Lambda(lambda x: tf.expand_dims(x, axis=-1),
                          input_shape=[None]), #Membuat Lambda Layer
    #Membuat RNN Layer
    tf.keras.layers.SimpleRNN(40, return_sequences=True),
    tf.keras.layers.SimpleRNN(40),
    tf.keras.layers.Dense(1), #Membuat Dense Layer
    tf.keras.layers.Lambda(lambda x: x * 100.0) #Membuat Lambda Layer
])

#Membuat Callback
lr_schedule = tf.keras.callbacks.LearningRateScheduler(
    lambda epoch: 1e-8 * 10**(epoch / 20))
optimizer = tf.keras.optimizers.SGD(learning_rate=1e-8,
                                    momentum=0.9)

#Mengcompile Model
model.compile(loss=tf.keras.losses.Huber(),
              optimizer=optimizer,
              metrics=["mae"])
```

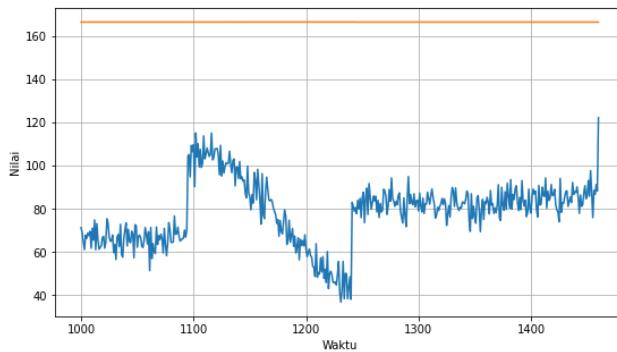
Mentraining Model

```
[ ] history = model.fit(train_set, epochs=100, callbacks=[lr_schedule])
```

Membuat Visualisasi Prediksi

```
[ ] prediksi = []
for time in range(len(series) - window_size):
    prediksi.append(model.predict(series[time:time + window_size]
                                  [np.newaxis]))
prediksi = prediksi[split_time-window_size:]
hasil = np.array(prediksi)[:, 0, 0]

plt.figure(figsize=(9, 5))
plot_series(time_valid, x_valid)
plot_series(time_valid, hasil)
```



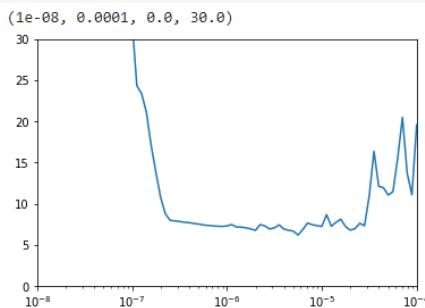
Mengukur Hasil MAE

```
[ ] tf.keras.metrics.mean_absolute_error(x_valid, hasil).numpy()
88.18091
```

Memperbaiki Model

Mencari Nilai Optimum Learning Rate

```
[ ] plt.semilogx(history.history["lr"], history.history["loss"])
plt.axis([1e-8, 1e-4, 0, 30])
```



Membuat Model Baru

```
[ ] tf.keras.backend.clear_session()
tf.random.set_seed(51)
np.random.seed(51)

dataset = windowed_dataset(x_train, window_size, batch_size=128,
                           shuffle_buffer=shuffle_buffer_size)

model = tf.keras.models.Sequential([
    tf.keras.layers.Lambda(lambda x: tf.expand_dims(x, axis=-1),
                           input_shape=[None]),
    tf.keras.layers.SimpleRNN(40, return_sequences=True),
    tf.keras.layers.SimpleRNN(40),
    tf.keras.layers.Dense(1),
    tf.keras.layers.Lambda(lambda x: x * 100.0)
])

optimizer = tf.keras.optimizers.SGD(learning_rate=6e-5, momentum=0.9)
model.compile(loss=tf.keras.losses.Huber(),
              optimizer=optimizer,
              metrics=["mae"])
```

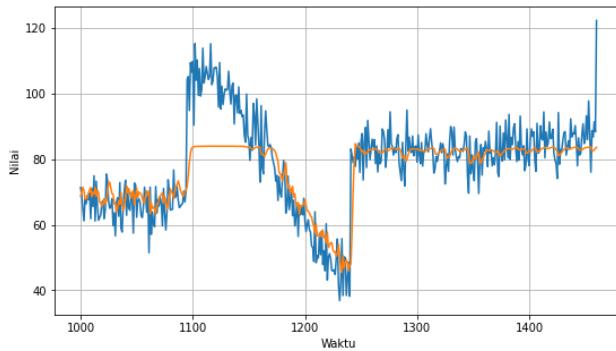
Mentraining Model

```
[ ] history = model.fit(dataset, epochs=400)
```

Memvisualisasikan Hasil Prediksi

```
[ ] prediksi=[]
    for time in range(len(series) - window_size):
        prediksi.append(model.predict(series[time:time + window_size]
                                      [np.newaxis]))
prediksi = prediksi[split_time-window_size:]
hasil = np.array(prediksi)[:, 0, 0]

plt.figure(figsize=(9, 5))
plot_series(time_valid, x_valid)
plot_series(time_valid, hasil)
```



Menghitung Hasil MAE

```
[ ] tf.keras.metrics.mean_absolute_error(x_valid, hasil).numpy()
6.4636235
```

Gambar 5.21. Kode Keseluruhan Bab 5

BAB 6

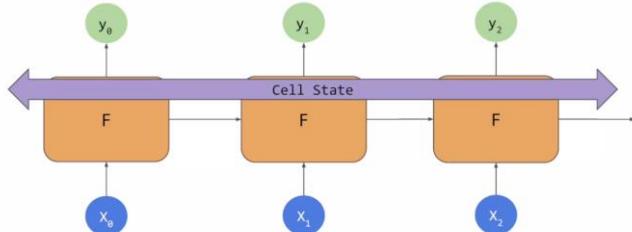
TIME SERIES FORECASTING DENGAN LSTM

6.1. Konsep LSTM

Pada bab sebelumnya, kita telah mempelajari bagaimana cara membuat model time series forecasting dengan RNN. Model yang kita dapatkan dengan RNN sudah cukup bagus. Namun, ada prediksi di bagian kurva yang belum maksimal. Oleh karena itu, kali ini kita akan mengembangkan hasil prediksinya dengan menggunakan LSTM.

LSTM merupakan kepanjangan dari Long Short Term Memory. LSTM termasuk dalam jenis RNN. LSTM merupakan salah satu pengembangan neural network yang digunakan untuk pemodelan time series. LSTM mampu mengatasi ketergantungan jangka panjang pada inputannya. Sebuah cell dalam LSTM menyimpan sebuah nilai atau keadaan (cell state)[19].

LSTM merupakan cell state yang mempertahankan nilainya sepanjang masa pelatihan sehingga nilainya diteruskan dari cell ke cell, timestamp ke timestamp. Ini berarti data dari window sebelumnya dapat memiliki dampak yang lebih baik dalam proyeksi kasus RNN. State nya juga dapat berupa bidirectional, jadi nilai atau statenya bergerak maju dan mundur. Berikut merupakan gambaran ilustrasi dari LSTM.



Gambar 6.1. Ilustrasi LSTM

6.2. Percobaan dengan LSTM

Kali ini kita akan mencoba untuk membuat model time series forecasting dengan LSTM. Seperti yang sudah dijelaskan sebelumnya, langkah-langkah untuk membuat LSTM hampir sama dengan algoritma deep learning lainnya. Namun, yang berbeda hanyalah terletak di bagian modelnya. Baik setelah ini langsung kita pelajari cara membuat model time series forecasting dengan LSTM.

- **Mengimpor Library dan Memeriksa Versi Tensorflow**

Sebelum membuat model, pertama kita perlu untuk mengimpor library yang diperlukan. Sama seperti pembuatan model sebelumnya, library yang dibutuhkan yaitu tensorflow, numpy dan matplotlib. Berikut merupakan kodennya :

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
print(tf.__version__)
```

Gambar 6.2. Kode Impor Library dan Cek Versi Tensorflow

- **Membuat Bentuk Fungsi Data Series**

Kode langkah ini juga sama seperti sebelumnya, kita membentuk fungsi data series dan nilainya bisa di atur sesuai dengan kita. Berikut merupakan kodonya :

```
def plot_series(time, series, format="-", start=0, end=None):
    plt.plot(time[start:end], series[start:end], format)
    plt.xlabel("Waktu")
    plt.ylabel("Nilai")
    plt.grid(True)

def trend(time, slope=0):
    return slope * time

def seasonal_pattern(season_time):
    return np.where(season_time < 0.4,
                   np.cos(season_time * 2 * np.pi),
                   1 / np.exp(3 * season_time))

def seasonality(time, period, amplitude=1, phase=0):
    season_time = ((time + phase) % period) / period
    return amplitude * seasonal_pattern(season_time)

def noise(time, noise_level=1, seed=None):
    rnd = np.random.RandomState(seed)
    return rnd.randn(len(time)) * noise_level

time = np.arange(4 * 365 + 1, dtype="float32")
baseline = 10
series = trend(time, 0.1)
baseline = 10
amplitude = 40
slope = 0.05
noise_level = 5

# Membuat Series
series = baseline + trend(time, slope) + seasonality(time,
                                                       period=365,
                                                       amplitude=amplitude)
# Menambahkan Noise
series += noise(time, noise_level, seed=42)
```

Gambar 6.3. Kode Membuat Fungsi Data Series

- **Membagi Dataset**

Sebelum bisa menggunakan datanya, kita perlu membagi datasetnya menjadi data train dan validation. Disini kode yang

digunakan sama seperti kode pembagian dataset sebelumnya.

Berikut merupakan kodennya :

```
split_time = 1000
time_train = time[:split_time]
x_train = series[:split_time]
time_valid = time[split_time:]
x_valid = series[split_time:]

window_size = 20
batch_size = 32
shuffle_buffer_size = 1000
```

Gambar 6.4. Kode Membagi Dataset

- **Membuat Windowed Dataset**

Dalam data series, dataset yang digunakan adalah windowed dataset. Setelah membuat pembagian dataset, maka selanjutnya membuat fungsi windowed datasetnya dengan tf.data.Dataset. Langkahnya sama seperti di bab sebelumnya. Berikut merupakan kodennya :

```
def windowed_dataset(series, window_size, batch_size, shuffle_buffer):
    dataset = tf.data.Dataset.from_tensor_slices(series)
    dataset = dataset.window(window_size + 1, shift=1, drop_remainder=True)
    dataset = dataset.flat_map(lambda window: window.batch(window_size + 1))
    dataset = dataset.shuffle(shuffle_buffer).map(lambda window:
                                                (window[:-1], window[-1]))
    dataset = dataset.batch(batch_size).prefetch(1)
    return dataset
```

Gambar 6.5. Kode Membuat Windowed Dataset

- **Membuat Model**

Ini merupakan hal yang membedakan dengan kode sebelumnya. Disini kita akan membuat model dengan bidirectional lstm. Berikut merupakan kodennya :

```

#Membersihkan variabel internal
tf.keras.backend.clear_session()
tf.random.set_seed(51)
np.random.seed(51)

tf.keras.backend.clear_session()
#Memanggil Fungsi Windowed Dataset
dataset = windowed_dataset(x_train, window_size, batch_size,
                           shuffle_buffer_size)

#Membuat Sequential Model
model = tf.keras.models.Sequential([
    #Membuat Lambda layer
    tf.keras.layers.Lambda(lambda x: tf.expand_dims(x, axis=-1),
                           input_shape=[None]),
    #Membuat LSTM Bidirectional Layer
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(
        32, return_sequences=True)),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32)),
    tf.keras.layers.Dense(1), #Membuat Dense layer
    tf.keras.layers.Lambda(lambda x: x * 100.0) #Membuat lambda layer
])
#Membuat Callbacks
lr_schedule = tf.keras.callbacks.LearningRateScheduler(
    lambda epoch: 1e-8 * 10***(epoch / 20))
optimizer = tf.keras.optimizers.SGD(learning_rate=1e-6, momentum=0.9)
#Mengcompile model
model.compile(loss=tf.keras.losses.Huber(),
              optimizer=optimizer,
              metrics=["mae"])

```

Gambar 6.6. Kode Membuat Model LSTM

Pertama, `tf.keras.backend.clear_session()` berfungsi untuk menghapus semua variabel internal atau untuk menghemat memori. Hal ini memudahkan untuk melakukan percobaan tanpa mempengaruhi model dengan versi terbaru.

Variabel dataset berguna untuk memanggil fungsi windowed dataset dan didalamnya terdapat parameter `x_train, window_size, batch_size, shuffle_buffer_size`.

Lalu membuat modelnya dengan `tf.keras.models.sequential()`. Didalam modelnya terdapat lambda layer yang memperluas dimensinya. Kemudian, menambahkan layer dengan bidirectional lstm, untuk membuat dampak pada hasil prediksi. Disini penulis

membuat lapisan lstm dengan 32 sel. Setelah itu ditambah dengan dense layer dan lambda layer lagi.

Kemudian menambahkan callback untuk learning ratenya. Menambahkan optimizer dan disini penulis mengatur learningnya bernilai 1e-6. Metrik yang digunakan pada model compile yaitu mae, dan huber loss.

- **Mentraining Model**

Berikut merupakan kode untuk mentraining modelnya. disini penulis mengaturnya menjadi 100 epoch.

```
history = model.fit(dataset, epochs=100, callbacks=[lrc_schedule])
```

Gambar 6.7. Kode Mentraining Model

- **Memvisualisasikan Hasil Prediksi**

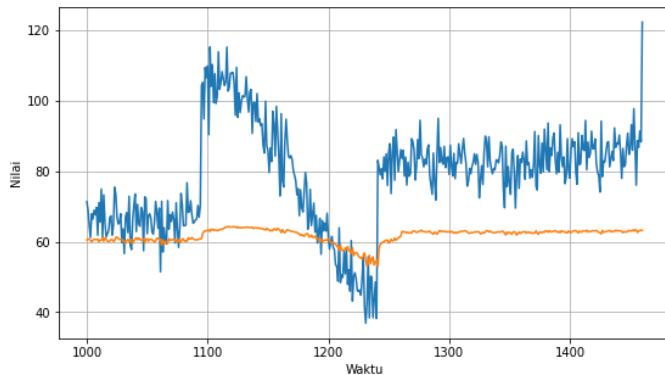
Setelah mentraining model, kita perlu untuk mengetahui hasil prediksi. Pertama membuat variabel forecast dan result untuk menampung hasil dari prediksi dan hasil, forecast dan result bersifat array. Selanjutnya, membuat perulangan dimana berisi prediksi model. Lalu kemudian membuat visualisasi series time valid, x valid dan time valid, result dengan matplotlib Berikut merupakan kodennya :

```
prediksi = []
hasil = []
for time in range(len(series) - window_size):
    prediksi.append(model.predict(series[time:time + window_size]
                                [np.newaxis]))
prediksi = prediksi[split_time-window_size:]
hasil = np.array(prediksi)[:, 0, 0]

plt.figure(figsize=(9, 5))
plot_series(time_valid, x_valid)
plot_series(time_valid, hasil)
```

Gambar 6.8. Kode Visualisasi Hasil Prediksi

Hasil :



Gambar 6.9. Visualisasi Hasil Prediksi

Berdasarkan grafik pada gambar 6.9, bisa kita lihat ternyata hasil prediksinya sangat jauh dari hasil aktual. Selanjutnya mari kita lihat berapa nilai MAE nya.

- **Mengukur Metrik MAE**

Seperti sebelumnya berikut merupakan kode untuk melihat hasil metrik MAE :

```
tf.keras.metrics.mean_absolute_error(x_valid, hasil).numpy()
```

```
18.120106
```

Gambar 6.10. Kode dan Hasil MAE

Ternyata hasil MAE nya sungguh tinggi, errornya mencapai 18,1. Hal ini menandakan model kita belum bekerja dengan maksimal. Oleh karena itu selanjutnya mari memperbaiki modelnya.

6.3. Memperbaiki Model LSTM

Karena hasil prediksi model sebelumnya belum bagus, maka sekarang kita akan memperbaiki modelnya. langkah-langkahnya hampir sama dengan bab sebelumnya.

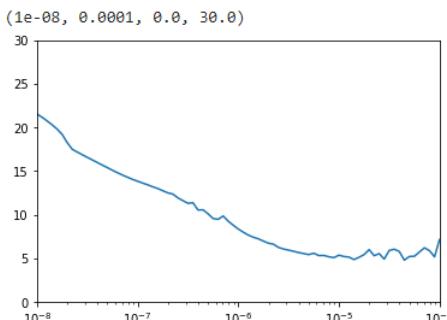
- **Mencari Nilai Optimum Learning Rate**

Agar bisa mendapatkan hasil prediksi yang baik, kita perlu untuk mengetahui nilai learning rate yang sesuai. Berikut merupakan kodennya :

```
plt.semilogx(history.history["lr"], history.history["loss"])
plt.axis([1e-8, 1e-4, 0, 30])
```

Gambar 6.11. Kode Mencari Nilai Optimum

Hasil :



Gambar 6.12. Hasil Nilai Optimum

Dari grafik pada gambar 6.12, titik terbawah kurva merupakan nilai optimum dari learning ratenya. Dan jika dilihat, nilai terendah kurvanya berada di nilai 10^{-5} . Oleh karena itu, pada model selanjutnya kita akan menggunakan nilai $1e-5$ sebagai nilai learning ratenya.

- **Membuat Model Baru**

Kode untuk modelnya sama dengan kode pembuatan model lstm sebelumnya. Namun, disini penulis mengubah nilai learning ratenya yang awalnya 1e-6 menjadi 1e-5. Berikut merupakan kodenya :

```
tf.keras.backend.clear_session()
tf.random.set_seed(51)
np.random.seed(51)

tf.keras.backend.clear_session()
dataset = windowed_dataset(x_train, window_size, batch_size,
                           shuffle_buffer_size)

model = tf.keras.models.Sequential([
    tf.keras.layers.Lambda(lambda x: tf.expand_dims(x, axis=-1),
                           input_shape=[None]),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM
                                 (32, return_sequences=True)),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32)),
    tf.keras.layers.Dense(1),
    tf.keras.layers.Lambda(lambda x: x * 100.0)
])
model.compile(loss="mse",
              optimizer=tf.keras.optimizers.SGD(learning_rate=1e-5,
                                                momentum=0.9),
              metrics=["mae"])
```

Gambar 6.13. Kode Membuat Model LSTM Baru

- **Mentraining Model**

Proses training model sama dengan sebelumnya. Disini kita akan menambahkan jumlah epoch sehingga menjadi 500. Berikut kodenya :

```
history = model.fit(dataset, epochs=500, verbose=0)
```

Gambar 6.14. Kode Mentraining Model LSTM Baru

- **Memvisualisasikan Hasil Prediksi**

Kode untuk ini sama dengan sebelumnya. Berikut merupakan kodenya :

```

prediksi = []
hasil = []
for time in range(len(series) - window_size):
    prediksi.append(model.predict(series[time:time + window_size]
                                  [np.newaxis]))

prediksi = prediksi[split_time-window_size:]
hasil = np.array(prediksi)[:, 0, 0]

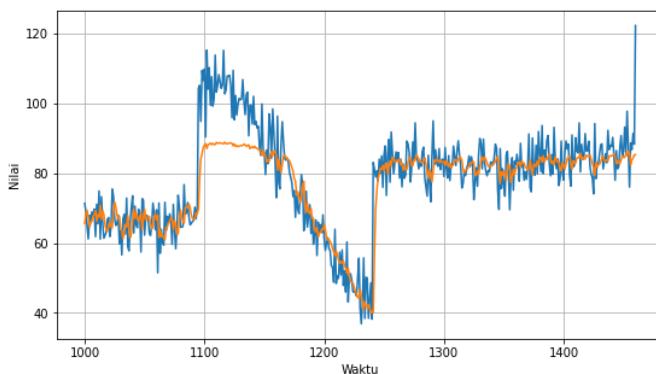
plt.figure(figsize=(9, 5))

plot_series(time_valid, x_valid)
plot_series(time_valid, hasil)

```

Gambar 6.15. Kode Visualisasi Hasil Prediksi

Hasilnya :



Gambar 6.16. Visualisasi Hasil Prediksi

Setelah memperbaiki modelnya, tampak bahwa hasil prediksinya lebih bagus. Hampir sama dengan data aktualnya. Berikutnya mari kita periksa berapa nilai MAEnya.

- **Mengukur Hasil MAE**

Kode untuk mengukur hasil mae sama seperti sebelumnya.

Berikut merupakan kodennya :

```
tf.keras.metrics.mean_absolute_error(x_valid, hasil).numpy()
```

```
5.782226
```

Gambar 6.17. Kode dan Hasil MAE

Hasil yang kita dapat ternyata bagus, nilai errornya hanya berkisar 5,7. Baiklah kita telah berhasil membuat model time series forecasting dengan lstm.

6.4. Kode Keseluruhan LSTM

Pada sub bab terakhir ini berisi kode keseluruhan. Kalian dapat mencocokkan dengan hasil percobaan kalian, apakah kodennya sudah sama atau belum.

Percobaan Time Series Forecasting dengan LSTM

Mengimpor Library dan Memeriksa Versi Tensorflow

```
[ ] import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
print(tf.__version__)

2.5.0
```

Membuat Bentuk Fungsi Data Series

```
[ ] def plot_series(time, series, format="-", start=0, end=None):
    plt.plot(time[start:end], series[start:end], format)
    plt.xlabel("Waktu")
    plt.ylabel("Nilai")
    plt.grid(True)
```

```

def trend(time, slope=0):
    return slope * time

def seasonal_pattern(season_time):
    return np.where(season_time < 0.4,
                   np.cos(season_time * 2 * np.pi),
                   1 / np.exp(3 * season_time))

def seasonality(time, period, amplitude=1, phase=0):
    season_time = ((time + phase) % period) / period
    return amplitude * seasonal_pattern(season_time)

def noise(time, noise_level=1, seed=None):
    rnd = np.random.RandomState(seed)
    return rnd.randn(len(time)) * noise_level

time = np.arange(4 * 365 + 1, dtype="float32")
baseline = 10
series = trend(time, 0.1)
baseline = 10
amplitude = 40
slope = 0.05
noise_level = 5

# Membuat Series
series = baseline + trend(time, slope) + seasonality(time,
                                                       period=365,
                                                       amplitude=amplitude)
# Menambahkan Noise
series += noise(time, noise_level, seed=42)

```

Membagi Dataset

```

[ ] split_time = 1000
time_train = time[:split_time]
x_train = series[:split_time]
time_valid = time[split_time:]
x_valid = series[split_time:]

window_size = 20
batch_size = 32
shuffle_buffer_size = 1000

```

Membuat Windowed Dataset

```
[ ] def windowed_dataset(series, window_size, batch_size, shuffle_buffer):
    dataset = tf.data.Dataset.from_tensor_slices(series)
    dataset = dataset.window(window_size + 1, shift=1, drop_remainder=True)
    dataset = dataset.flat_map(lambda window: window.batch(window_size + 1))
    dataset = dataset.shuffle(shuffle_buffer).map(lambda window:
                                                    (window[:-1], window[-1]))
    dataset = dataset.batch(batch_size).prefetch(1)
    return dataset
```

Membuat Model

```
[ ] #Membersihkan variabel internal
tf.keras.backend.clear_session()
tf.random.set_seed(51)
np.random.seed(51)

tf.keras.backend.clear_session()
#Memanggil Fungsi Windowed Dataset
dataset = windowed_dataset(x_train, window_size, batch_size,
                           shuffle_buffer_size)

#Membuat Sequential Model
model = tf.keras.models.Sequential([
    #Membuat Lambda layer
    tf.keras.layers.Lambda(lambda x: tf.expand_dims(x, axis=-1),
                           input_shape=[None]),
    #Membuat LSTM Bidirectional Layer
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM
                                  (32, return_sequences=True)),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32)),
    tf.keras.layers.Dense(1), #Membuat Dense layer
    tf.keras.layers.Lambda(lambda x: x * 100.0) #Membuat lambda layer
])
#Membuat Callbacks
lr_schedule = tf.keras.callbacks.LearningRateScheduler(
    lambda epoch: 1e-8 * 10**(epoch / 20))
optimizer = tf.keras.optimizers.SGD(learning_rate=1e-6, momentum=0.9)
#Mengcompile model
model.compile(loss=tf.keras.losses.Huber(),
               optimizer=optimizer,
               metrics=["mae"])
```

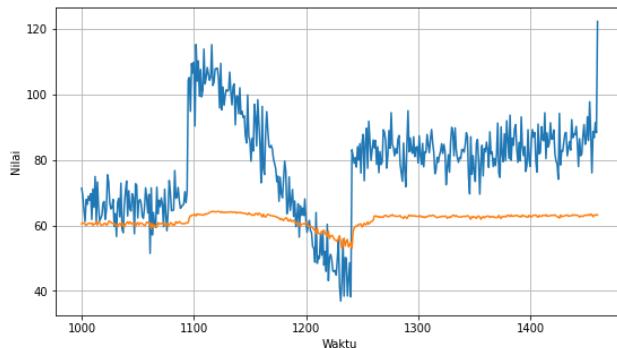
Mentraining Model

```
[ ] history = model.fit(dataset, epochs=100, callbacks=[lr_schedule])
```

Memvisualisasikan Hasil Prediksi

```
[ ] prediksi = []
hasil = []
for time in range(len(series) - window_size):
    prediksi.append(model.predict(series[time:time + window_size]
                                [np.newaxis]))
prediksi = prediksi[split_time-window_size:]
hasil = np.array(prediksi)[:, 0, 0]

plt.figure(figsize=(9, 5))
plot_series(time_valid, x_valid)
plot_series(time_valid, hasil)
```



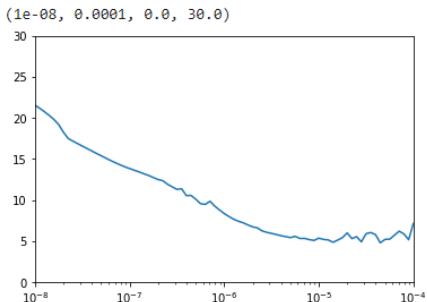
Mengukur Hasil MAE

```
[ ] tf.keras.metrics.mean_absolute_error(x_valid, hasil).numpy()
18.120106
```

Memperbaiki Model

Mencari Nilai Optimum Learning Rate

```
[ ] plt.semilogx(history.history["lr"], history.history["loss"])
plt.axis([1e-8, 1e-4, 0, 30])
```



Membuat Model Baru

```
[ ] tf.keras.backend.clear_session()
tf.random.set_seed(51)
np.random.seed(51)

tf.keras.backend.clear_session()
dataset = windowed_dataset(x_train, window_size, batch_size,
                           shuffle_buffer_size)

model = tf.keras.models.Sequential([
    tf.keras.layers.Lambda(lambda x: tf.expand_dims(x, axis=-1),
                           input_shape=[None]),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM
                                 (32, return_sequences=True)),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32)),
    tf.keras.layers.Dense(1),
    tf.keras.layers.Lambda(lambda x: x * 100.0)
])

model.compile(loss="mse",
              optimizer=tf.keras.optimizers.SGD(learning_rate=1e-5,
                                                momentum=0.9),
              metrics=["mae"])
```

Mentraining Model

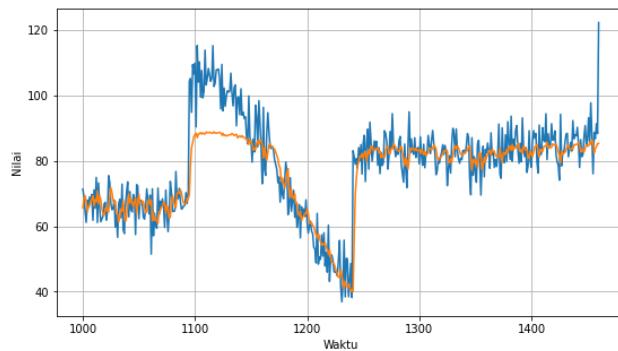
```
[ ] history = model.fit(dataset,epochs=500,verbose=0)
```

Memvisualisasikan Hasil Prediksi

```
[ ] prediksi = []
hasil = []
for time in range(len(series) - window_size):
    prediksi.append(model.predict(series[time:time + window_size]
                                    [np.newaxis]))
prediksi = prediksi[split_time-window_size:]
hasil = np.array(prediksi)[:, 0, 0]

plt.figure(figsize=(9, 5))

plot_series(time_valid, x_valid)
plot_series(time_valid, hasil)
```



Mengukur Hasil MAE

```
[ ] tf.keras.metrics.mean_absolute_error(x_valid, hasil).numpy()
```

```
5.782226
```

Gambar 6.18. Kode Keseluruhan Bab 6

BAB 7

TIME SERIES FORECASTING DENGAN REAL DATASET

7.1. Dataset

Akhirnya kita sudah mencapai bab terakhir dari buku ini. Sebelumnya kita telah memperlajari cara membuat model time series forecasting dengan beberapa algoritma deep learning, diantaranya yaitu Neural Network, Deep Neural Network, Recurrent Neural Network dan LSTM. Namun, pada percobaan sebelumnya kita masih menggunakan data biasa yang kita buat dengan rumus. Kali ini, kita akan mencoba membuat model time series forecasting dengan menggunakan Dataset sebenarnya. Jadi, datanya merupakan hasil nyata.

7.1.1. Pengertian Dataset

Dataset merupakan hal yang paling penting dalam pembuatan machine learning, karena mesin membutuhkan data untuk belajar. Dataset adalah kumpulan dari koleksi data. Dataset memuat nilai dari setiap variabel. Dataset juga bisa berisi dari kumpulan dokumen atau file. Selain itu, dataset juga bisa kita sebut sebagai himpunan data. Dataset juga bisa mengacu pada isi sebuah tabel database atau matriks data statistik[20].

7.1.2. Sumber Dataset

Saat ini, terdapat banyak sekali website yang sudah menyediakan dataset untuk bisa digunakan sebagai bahan penelitian. Jadi, dengan adanya ini akan memberikan kemudahan bagi kita. Kita sudah tidak perlu untuk membuat datasetnya sendiri. Berikut merupakan website yang bisa dijadikan untuk mencari dataset :

1. Kaggle



Gambar 7.1. Logo Kaggle

Jika anda seorang yang menyukai data science atau machine learning, tentu anda tidak akan asing dengan website satu ini. Kaggle merupakan website yang berisi komunitas online yang dibentuk pada tahun 2010 oleh Anthony Goldbloom dan Ben Hamner. Di kaggle terdapat banyak sekali dataset. Tak hanya dataset, dengan kaggle kalian juga bisa menulis dan berbagi kode, bahkan mengikuti kompetisi mengenai pengelolaan data. Berikut merupakan link untuk membuka kaggle : <https://www.kaggle.com/>.

2. UCI Machine Learning



Gambar 7.2. Logo UCI Machine Learning

UCI Machine Learning merupakan machine learning repository yang dibuat oleh David Aha sejak tahun 1987. UCI Machine learning repository ini berisi kumpulan dataset, database, teori domain dan generator data yang digunakan untuk komunitas pembelajar machine learning. Berikut merupakan link menuju UCI Machine Learning Repository : <https://archive.ics.uci.edu/ml/index.php> .

3. Google Dataset Search



Gambar 7.3. Logo Google Dataset Search

Google Dataset Search merupakan layanan dari Google untuk mencari dataset yang bisa digunakan oleh peneliti untuk mendapatkan data. Google dataset ini di launch pada 5 September 2018. Berikut merupakan link menuju Google Dataset Search : <https://datasetsearch.research.google.com/> .

4. Portal Satu Data Indonesia



Gambar 7.4. Logo Portal Satu Data Indonesia

Portal Satu Data Indonesia merupakan portal resmi data terbuka Indonesia yang dikelola oleh Sekretariat Satu Data Indonesia tingkat Pusat, Kementerian Perencanaan Pembangunan Nasional / Bappenas. Jadi, disini berisi data-data yang didapat oleh pemerintah. Berikut merupakan link menuju Portal Satu Data Indonesia : <https://data.go.id/> .

7.1.3. Sunspots Dataset



Gambar 7.5. Tampilan Dataset Sunspots Kaggle

Pada bab kali ini, kita akan belajar membuat model time series forecasting dengan Sunspots Dataset. Dataset ini kerap kali digunakan untuk pemula yang baru mempelajari machine learning.

Sunspots merupakan fenomena sementara pada fotosfer matahari yang tampak seperti bintik-bintik gelap di daerah sekitarnya. Sunspot terjadi akibat penurunan suhu yang disebabkan oleh konsentrasi fluks medan magnet yang menghambat konveksi. Sunspots biasanya muncul berpasangan dengan polaritas magnet yang berlawanan. Jumlahnya bervariasi sesuai dengan siklus matahari sekitar 11 tahun.

Dataset Sunspots ini berisi nilai rata-rata bulanan sunspot dari 1 Januari 1979 sampai 31 Agustus 2017. Dataset ini terdiri dari 3235 data yang berisi data tanggal bulan dan nilai rata-rata bulanan sunspot. Dataset ini di dapat dari database SIDC (Solar Influence Data Analysis Center) Departemen penelitian fisika matahari dari royal observatory Belgia[21].

Berikut merupakan link untuk mengakses datasetnya :
<https://www.kaggle.com/robervalt/sunspots> .

7.2. Membuat Model

Sebelumnya, kita telah belajar membuat model dengan deep learning. Prosesnya hampir sama dengan sebelumnya. Jadi, disini kita akan mencoba untuk mengimplementasikan hasil percobaan kita dengan data sebenarnya. Dalam pembuatan model time series ini, penulis menggunakan model RNN LSTM. Langsung saja kita lihat langkah-langkahnya.

- **Mengimpor Library dan Memeriksa Versi Tensorflow**

Seperti biasa langkah pertama yaitu mengimpor library yang kita butuhkan. Tak lupa cek juga versi tensorflownya. Berikut merupakan kodenya :

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
print(tf.__version__)
```

Gambar 7.6. Kode Impor Library dan Cek Versi Tensorflow

Pada awal ini, kita perlu mengimpor library tensorflow, numpy dan matplotlib. Pastikan juga versi tensorflownya sudah menggunakan versi 2 ke atas.

- **Membuat Fungsi Grafik Data Series**

Selanjutnya kita membuat fungsi plot series untuk membuat visualisasi dari datasetnya. Berikut merupakan kodenya :

```
def plot_series(time, series, format="-", start=0, end=None):
    plt.plot(time[start:end], series[start:end], format)
    plt.xlabel("Waktu")
    plt.ylabel("Nilai")
    plt.grid(True)
```

Gambar 7.7. Kode Fungsi Grafik Data Series

Penjelasan kodenya, def digunakan untuk membuat fungsi. plt.plot digunakan untuk membuat format grafiknya. Lalu kita beri keterangan sumbu x dengan plt.xlabel, dan sumbu y dengan plt.ylabel. lalu plt.grid kita set True agar digrafiknya muncul grid.

- **Menghubungkan dengan Google Drive**

Karena penyimpanan Google Colab bersifat sementara, oleh karena itu penulis menyimpan datasetnya di Google Drive. Selanjutnya yang perlu dilakukan yaitu menghubungkan notebook

Google Colab kita dengan Goole Drive kita. Berikut merupakan kodenya :

```
from google.colab import drive #Mengimpor G-Drive  
drive.mount('/content/drive') #Mengatur lokasi Drive
```

Gambar 7.8. Kode Menghubungkan GDrive

Setelah menjalankan kode seperti gambar 7.8, Anda akan disuruh untuk memasukkan kode autorisasi. Untuk mendapatkan kodenya, maka kita perlu mengklik link yang sudah disediakan.

Menghubungkan dengan Google Drive

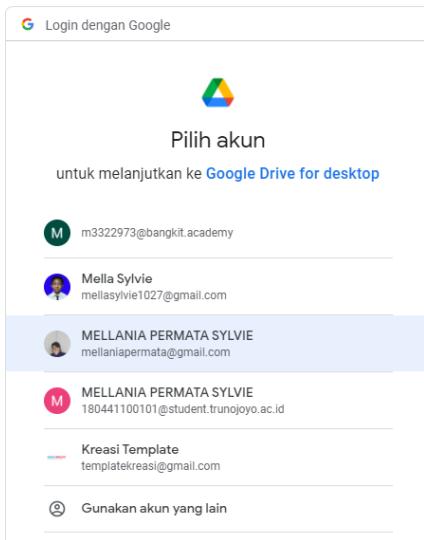
The screenshot shows a Google Colab notebook cell. The code is:

```
from google.colab import drive #Mengimpor G-Drive  
drive.mount('/content/drive') #Mengatur lokasi Drive
```

Below the code, there is a message: "Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=...". A text input field is present with the placeholder "Enter your authorization code:" followed by a redacted box. The Colab toolbar is visible at the top.

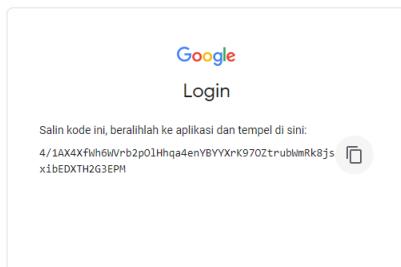
Gambar 7.9. Tampilan Kode Autorisasi

Setelah mengklik linknya, maka kita akan disuruh untuk memilih akun mana yang akan kita akses Google Drivenya. Berikut contohnya :



Gambar 7.10. Tampilan Memilih Akun

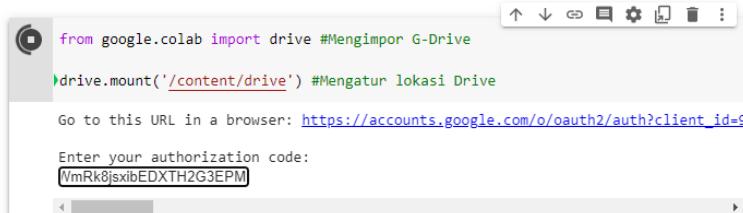
Setelah memilih akunnya, tekan tombol login dan kita akan mendapatkan kodenya. Copy kodenya.



Gambar 7.11. Tampilan Link Kode Autorisasi

Selanjutnya paste kodenya di kotak kode autorisasi pada Google Colab. Maka kita sudah berhasil menghubungkan Notebook kita dengan Google Drive kita.

Menghubungkan dengan Google Drive



```
from google.colab import drive #Mengimpor G-Drive
drive.mount('/content/drive') #Mengatur lokasi Drive
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=...
Enter your authorization code:
VmRk8jsxibEDXTH2G3EPM
```

Gambar 7.12. Tampilan Mengisi Kode Autorisasi

- **Mengakses Dataset**

Selanjutnya kita akan mengakses datasetnya. Disini untuk membaca file csv nya kita memerlukan library pandas. Maka disini kita mengimpor library baru. Berikut merupakan kodenya :

```
import pandas as pd #Mengimpor Library Pandas
#Membaca File Dataset CSV
Dataset = pd.read_csv('/content/drive/MyDrive/Dataset/Sunspots.csv')
#Melihat Info Dataset
Dataset.info()
```

Gambar 7.13. Kode Mengakses Dataset

Dalam kode pada gambar 7.13, pd.read_csv digunakan untuk membaca file dataset csv. Lalu bisa melihat info mengenai datasetnya dengan dataset.info(). Berikut hasil informasi dataset sunspots nya :

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3235 entries, 0 to 3234
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype  
 ---  --  
 0   Unnamed: 0        3235 non-null   int64  
 1   Date             3235 non-null   object  
 2   Monthly Mean Total Sunspot Number 3235 non-null   float64 
dtypes: float64(1), int64(1), object(1)
memory usage: 75.9+ KB
```

Gambar 7.14. Tampilan Info Dataset

- **Melihat Isi Dataset**

Kita juga bisa melihat data dari datasetnya dengan dataset.head(), head ini akan menampilkan 5 data teratas. Berikut merupakan kode dan hasilnya :

```
#Melihat Isi Dataset
Dataset.head()
```

	Unnamed: 0	Date	Monthly Mean Total Sunspot Number
0	0	1749-01-31	96.7
1	1	1749-02-28	104.3
2	2	1749-03-31	116.7
3	3	1749-04-30	92.8
4	4	1749-05-31	141.7

Gambar 7.15. Kode dan Isi Dataset

- **Membuka Dataset**

Untuk bisa menggunakan datasetnya, maka kita perlu untuk membuka datasetnya dan memvisualisasikan dalam bentuk grafik. Berikut merupakan kodennya :

```
import csv #Mengimpor CSV
time_step = []
sunspot = []

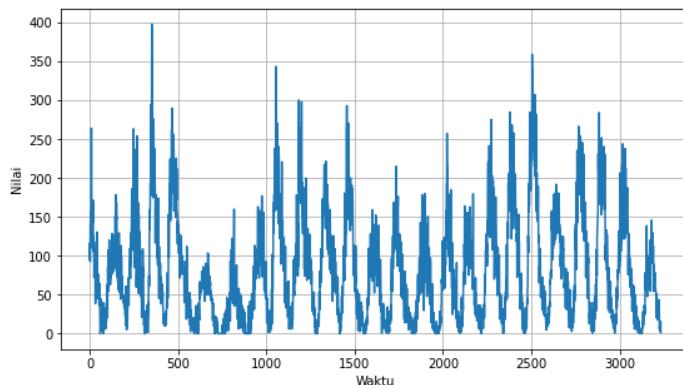
#Membuka File CSV
with open('/content/drive/MyDrive/Dataset/Sunspots.csv') as csvfile:
    reader = csv.reader(csvfile, delimiter=',')
    next(reader)
    for row in reader:
        sunspot.append(float(row[2]))
        time_step.append(int(row[0]))

#Membuat Visualisasi Dataset
series = np.array(sunspot)
time = np.array(time_step)
plt.figure(figsize=(9, 5))
plot_series(time, series)
```

Gambar 7.16. Kode Membuka Dataset

Pertama, kita perlu mengimpor csv. Kemudian membuat variabel time_step dan sunspot yang bernilai array. Selanjutnya membuka file csvnya dengan with open, dan membaca file csvnya dengan csv.reader. lalu kita menggunakan perulangan agar dapat mengakses data dari kolom sunspot(nilai rata-rata) dan time_stepnya (jumlah data).

Selanjutnya, memvisualisasikan datasetnya ke dalam bentuk grafik dengan menggunakan matplotlib. Berikut merupakan hasil grafiknya :



Gambar 7.17. Visualisasi Dataset Sunspots

Dapat kita ketahui dari gambar 7.17, bahwa dataset sunspots ini berbentuk seasonal.

- **Membagi Dataset**

Untuk bisa mentraining datasetnya, kita juga perlu untuk membagi datasetnya ke dalam data training dan validasi. Kodenya sama seperti sebelumnya, berikut kodenya :

```

split_time = 3000
time_train = time[:split_time]
x_train = series[:split_time]
time_valid = time[split_time:]
x_valid = series[split_time:]

window_size = 30
batch_size = 32
shuffle_buffer_size = 1000

```

Gambar 7.18. Kode Membagi Dataset

Kali ini penulis membagi datasetnya menjadi 3000, window_size nya di atur menjadi 30, batch_sizenya 32, dan shuffle_buffernya 1000.

- **Membuat Windowed Dataset**

Karena dalam time series data yang digunakan harus window. Maka selanjutnya kita membuat windowed datasetnya. Berikut merupakan kodennya :

```

def windowed_dataset(series, window_size, batch_size, shuffle_buffer):
    series = tf.expand_dims(series, axis=-1)
    dataset = tf.data.Dataset.from_tensor_slices(series)
    dataset = dataset.window(window_size + 1, shift=1, drop_remainder=True)
    dataset = dataset.flat_map(lambda w: w.batch(window_size + 1))
    dataset = dataset.shuffle(shuffle_buffer)
    dataset = dataset.map(lambda w: (w[:-1], w[1:]))
    return dataset.batch(batch_size).prefetch(1)

```

Gambar 7.19. Kode Membuat Windowed Dataset

- **Membuat Fungsi Untuk Memprediksi Model**

Fungsi ini digunakan agar kita bisa menggunakan prediksi ini ke dalam visualisasi hasilnya. Berikut merupakan kodennya :

```

def prediksi_model(model, series, window_size):
    dataset = tf.data.Dataset.from_tensor_slices(series)
    dataset = dataset.window(window_size, shift=1, drop_remainder=True)
    dataset = dataset.flat_map(lambda w: w.batch(window_size))
    dataset = dataset.batch(32).prefetch(1)
    prediksi = model.predict(dataset)
    return prediksi

```

Gambar 7.20. Kode Fungsi Prediksi Model

- **Membuat Model**

Selanjutnya, ini merupakan inti dari pembelajaran kita. Disini kita menggunakan RNN Lstm model sebagai modelnya. Berikut merupakan kodennya :

```
#Membersihkan Variabel luar
tf.keras.backend.clear_session()
tf.random.set_seed(51)
np.random.seed(51)

window_size = 64 #Mengatur Ukuran Window
batch_size = 256 #Mengatur Ukuran Batch

#Memanggil Windowed Dataset
train_set = windowed_dataset(x_train, window_size, batch_size,
                             shuffle_buffer_size)

print(train_set) #Melihat shape train_set
print(x_train.shape) #Melihat shape x_train
#Membuat Model
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv1D(filters=32, kernel_size=5,
                          strides=1, padding="causal",
                          activation="relu",
                          input_shape=[None, 1]), #Membuat Layer CNN
    tf.keras.layers.LSTM(64, return_sequences=True), #Membuat Layer lstm
    tf.keras.layers.LSTM(64, return_sequences=True),
    tf.keras.layers.Dense(30, activation="relu"), #Membuat Dense Layer
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(1),
    tf.keras.layers.Lambda(lambda x: x * 400) #Membuat Lambda Layer
])

#Membuat Callbacks
lr_schedule = tf.keras.callbacks.LearningRateScheduler(
    lambda epoch: 1e-8 * 10**((epoch / 20)))
optimizer = tf.keras.optimizers.SGD(learning_rate=1e-8, momentum=0.9)

#Mengcompile Model
model.compile(loss=tf.keras.losses.Huber(),
              optimizer=optimizer,
              metrics=["mae"])
```

Gambar 7.21. Kode Membuat Model

Seperti biasa, kita perlu membersihkan variabel luar untuk menghemat memory dengan tf.keras.backend.clear_session(). Selanjutnya kita mengacak datanya, dan mengatur ukuran window

dan batch. Kemudian, kita panggil fungsi windowed dataset yang sudah dibuat sebelumnya.

Lalu kita membuat modelnya dengan tf.keras.model disini layer kita menggunakan convolutional, lstm dan lambda layer. Aktivasi yang digunakan adalah relu. Setelah membuat model, maka kita buat callbacknya untuk mempercepat dan memperbagus hasil prediksi. Dan yang terakhir adalah mengcompile modelnya dengan kode pada gambar 7.21. Kita juga bisa melihat summary bentuk modelnya dengan model.summary .

```
model.summary() #Melihat Summary Model
```

Model: "sequential"

Layer (type)	Output Shape	Param #
<hr/>		
conv1d (Conv1D)	(None, None, 32)	192
lstm (LSTM)	(None, None, 64)	24832
lstm_1 (LSTM)	(None, None, 64)	33024
dense (Dense)	(None, None, 30)	1950
dense_1 (Dense)	(None, None, 10)	310
dense_2 (Dense)	(None, None, 1)	11
lambda (Lambda)	(None, None, 1)	0
<hr/>		
Total params:	60,319	
Trainable params:	60,319	
Non-trainable params:	0	

Gambar 7.22. Kode dan Summary Model

- **Mentraining Model**

Setelah membuat model, tentunya kita latih modelnya. disini penulis mentrainingnya dengan 100 epochs. Berikut kodennya :

```
history = model.fit(train_set, epochs=100, callbacks=[lr_schedule])
```

Gambar 7.23. Kode Mentraining Model

- **Visualisasi Hasil Prediksi**

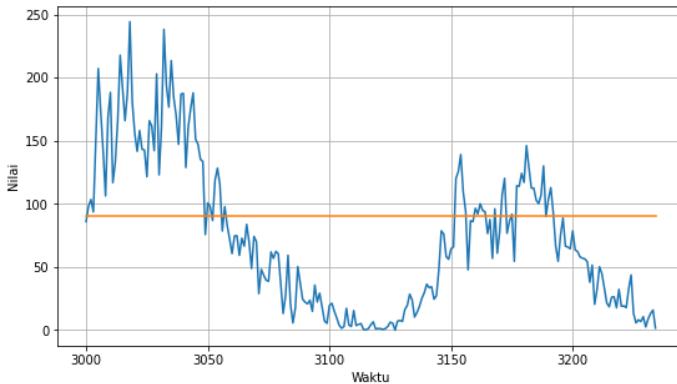
Setelah mentraining modelnya, maka kita lihat apakah hasilnya sudah bagus atau belum. Oleh karena itu kita membuat visualisasi hasil prediksi dengan kode berikut :

```
prediksi_rnn = prediksi_model(model, series[...], np.newaxis), window_size)
prediksi_rnn = prediksi_rnn[split_time - window_size:-1, -1, 0]

plt.figure(figsize=(9, 5))
plot_series(time_valid, x_valid)
plot_series(time_valid, prediksi_rnn)
```

Gambar 7.24. Kode Visualisasi Hasil Prediksi

Hasilnya :



Gambar 7.25. Visualisasi Hasil Prediksi

Berdasarkan hasil pada gambar 7.25, ternyata prediksinya sangat jauh dari data aktual. Ini menandakan model kita kurang baik, oke mari kita cek berapa nilai MAE nya.

- **Mengukur Hasil MAE**

Selanjutnya kita mengukur tingkat errornya dengan metrik MAE. Berikut merupakan kode dan hasilnya :

```
tf.keras.metrics.mean_absolute_error(x_valid, prediksi_rnn).numpy()  
52.4056
```

Gambar 7.26. Kode dan Hasil MAE

Ternyata hasil yang didapat sangat tinggi errornya. Oleh karena itu prediksinya masih jauh data aktual. Baik di sub bab selanjutnya kita akan memperbaiki modelnya agar nilai prediksinya lebih bagus.

7.3. Perbaikan Model

Model kita sebelumnya ternyata kurang bekerja dengan baik, sehingga nilai prediksinya masih buruk. Oleh karena itu, sekarang kita perbaiki modelnya.

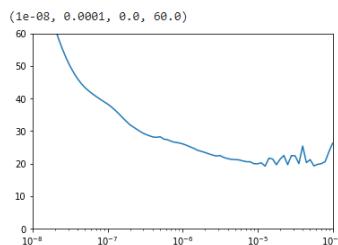
- **Mencari Nilai Optimum**

Nilai dari learning rate juga mempengaruhi hasil. Oleh karena itu, untuk membuat model yang lebih baik kita perlu mencari tau berapa nilai optimum untuk learning ratenya. Berikut merupakan kodennya :

```
plt.semilogx(history.history["lr"], history.history["loss"])  
plt.axis([1e-8, 1e-4, 0, 60])
```

Gambar 7.27. Kode Mencari Nilai Optimum

Hasil :



Gambar 7.28. Hasil Grafik Nilai Optimum

Untuk mengetahui nilai optimumnya, maka kita lihat titik terendah dari kurvanya. Dari visualisasi loss training pada gambar 7.28, ternyata titik terendahnya terletak pada rate 10^{-5} . Oleh karena itu, learning ratenya kita menggunakan $1e-5$.

- **Membuat Model Baru**

Selanjutnya mari buat model baru. Berikut merupakan kodennya :

```
tf.keras.backend.clear_session()
tf.random.set_seed(51)
np.random.seed(51)
train_set = windowed_dataset(x_train, window_size=60, batch_size=100,
                             shuffle_buffer=shuffle_buffer_size)
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv1D(filters=60, kernel_size=5,
                          strides=1, padding="causal",
                          activation="relu",
                          input_shape=[None, 1]),
    tf.keras.layers.LSTM(60, return_sequences=True),
    tf.keras.layers.LSTM(60, return_sequences=True),
    tf.keras.layers.Dense(30, activation="relu"),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(1),
    tf.keras.layers.Lambda(lambda x: x * 400)
])

optimizer = tf.keras.optimizers.SGD(learning_rate=1e-5, momentum=0.9)
model.compile(loss=tf.keras.losses.Huber(),
              optimizer=optimizer,
              metrics=["mae"])
```

Gambar 7.29. Kode Membuat Model Baru

Kode untuk perbaikan modelnya hampir sama dengan kode model sebelumnya. Namun, disini penulis mengubah ubah nilainya, sampai mendapatkan hasil yang bagus. Disini penulis mengatur window sizenya menjadi 60, batch sizenya menjadi 100. Lalu filtersnya menjadi 60, dan kernelnya 5. Untuk lstmnya menggunakan 60 kali, sebelumnya 64. Lalu disini optimizernya learningnya sebelumnya $1e-8$ diubah menjadi $1e-5$ sesuai hasil nilai optimum learning rate. Selanjutnya mengcompile model.

- **Mentraining Model**

Setelah membuat perbaikan model, kita perlu melatih modelnya apakah sudah bekerja dengan baik atau belum. Disini penulis mentraining modelnya dengan 500 epoch. Berikut merupakan kodenya :

```
history = model.fit(train_set, epochs=500)
```

Gambar 7.30. Kode Mentraining Model

- **Visualisasi Hasil Prediksi**

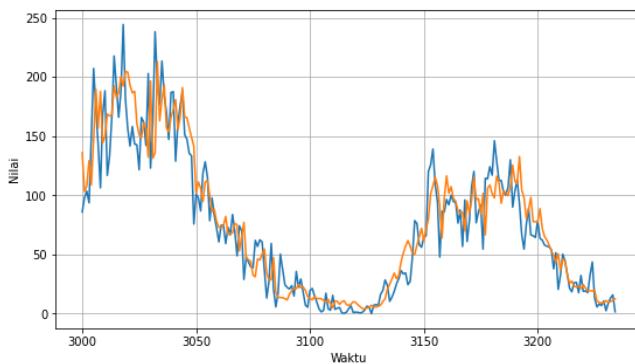
Setelah mendapatkan hasil training, maka kita perlu menvisualisasikan hasil prediksinya apakah sudah baik apa belum. Berikut merupakan kodenya :

```
rnn_baru = prediksi_model(model, series[..., np.newaxis], window_size)
rnn_baru = rnn_baru[split_time - window_size:-1, -1, 0]

plt.figure(figsize=(9, 5))
plot_series(time_valid, x_valid)
plot_series(time_valid, rnn_baru)
```

Gambar 7.31. Kode Visualisasi Hasil Prediksi

Hasil :



Gambar 7.32. Visualisasi Hasil Prediksi

Berdasarkan hasil prediksi pada gambar 7.32, ternyata hasil prediksinya cukup bagus, polanya hampir mendekati data aktualnya.

- **Mengukur MAE**

Untuk mengetahui tingkat errornya, maka seperti biasa kita mengukur hasil metrik MAEnya dengan kode berikut :

```
tf.keras.metrics.mean_absolute_error(x_valid, rnn_baru).numpy()  
15.620486
```

Gambar 7.33. Kode dan Hasil MAE

Ternyata hasil MAE nya adalah 15.6, bisa dikatakan cukup bagus tidak terlalu buruk. Ini menandakan bahwa model kita bekerja dengan lebih baik.

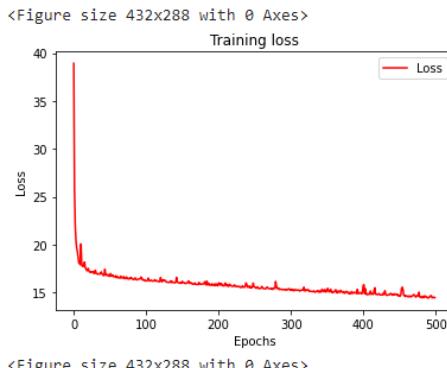
- **Visualisasi Grafik Training Loss**

Selanjutnya kita bisa melihat grafik training lossnya dengan kode berikut. Hal ini dilakukan untuk mengetahui bagaimana hasil training lossnya selama pelatihan.

```
#impor library matplotlib  
import matplotlib.image as mpimg  
import matplotlib.pyplot as plt  
  
loss=history.history['loss'] #mengambil data loss  
epochs=range(len(loss)) #Menghitung jumlah epoch  
  
#Membuat visualisasi grafik loss  
plt.plot(epochs, loss, 'r')  
plt.title('Training loss')  
plt.xlabel("Epochs")  
plt.ylabel("Loss")  
plt.legend(["Loss"])  
  
plt.figure()
```

Gambar 7.34. Kode Visualisasi Training Loss

Hasil :



Gambar 7.35. Hasil Visualisasi Training Loss

Dari hasil pada gambar 7.35, bisa dilihat bahwa training lossnya semakin lama semakin turun. Hal ini menandakan bagus. Lalu mari kita zoom, training lossnya dari epoch 200 sampai 500. Berikut kodenya :

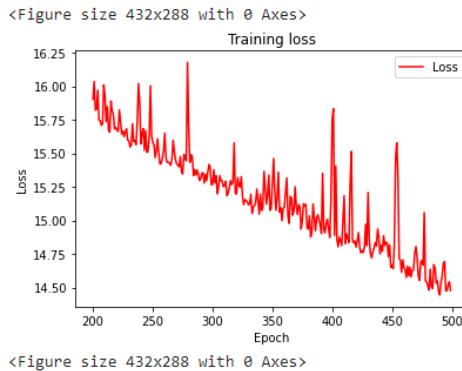
```
zoom_loss = loss[200:] #Mengambil data dari 200
zoom_epoch = range(200,500) #Range data dari 200 sampai 500

#Memvisualisasikan grafik
plt.plot(zoom_epoch, zoom_loss, 'r')
plt.title('Training loss')
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend(["Loss"])

plt.figure()
```

Gambar 7.36. Kode Zoom Visualisasi Training Loss

Hasil :



Gambar 7.37. Hasil Zoom Visualisasi Training Loss

Setelah dilihat lebih detail. Ya memang training lossnya cenderung semakin turun, ini merupakan pertanda bagus.

- **Melihat Hasil Prediksi**

Baik yang terakhir kita bisa melihat nilai dari prediksinya. Baik berikut merupakan kodennya :

```
print(rnn_baru)
```

Gambar 7.38. Kode Melihat Hasil Prediksi

Hasil :

[135.93948	103.12235	107.854065	129.22385	108.84819	147.39487
189.78087	156.79062	187.81398	144.31122	150.6124	168.67754
166.9058	167.96768	192.48328	182.7197	186.66794	200.11948
192.28748	284.5711	203.98186	192.89357	186.52187	187.82918
160.57487	151.08842	148.91212	161.18733	156.05412	132.18439
196.56184	131.38285	136.06392	212.91818	162.95776	181.05084
193.0767	157.0992	153.59818	165.92699	172.63889	180.77162
155.54376	166.10939	191.04233	166.12889	165.59535	156.55511
148.85011	141.43314	102.08545	111.27963	106.03841	94.656075
111.22477	112.19361	99.72339	86.33762	87.99869	77.85541
72.827484	72.42288	74.6223	82.11267	66.877884	67.77999
68.20702	76.20557	75.08509	52.962357	64.73747	77.068375
46.685688	45.63853	44.194557	32.430347	31.180859	46.316635
45.280518	50.386272	54.557896	34.109955	28.836811	28.727839
47.2776	17.91089	14.0321865	13.622933	13.497494	12.69638
11.581974	16.43831	19.259253	21.55281	21.041985	24.788696
21.761194	22.558731	19.731035	14.582336	12.897987	11.788781
14.488938	12.563949	12.8702345	12.583889	10.987878	9.625046
12.106937	8.611035	5.0987373	10.786907	10.411265	7.8480825
10.081189	18.927359	7.637983	6.963053	7.954654	10.097498
9.621684	7.6802	6.0288986	4.386717	4.108143	4.6765475
5.578646	7.119054	5.594024	5.6613326	6.3322678	7.164158
10.290985	12.593584	20.88556	24.778852	29.313897	34.24381
29.524559	32.3227	43.777554	51.188503	57.03432	61.741848
57.025562	51.106476	50.24871	58.000607	64.55911	72.112335
61.50855	71.31736	79.54353	101.18656	107.90452	115.040794
108.574	98.5965	64.021126	98.0149	116.394714	101.63542
187.30144	97.21209	95.027245	87.01589	84.5397	85.66534
69.92878	95.28463	84.83131	96.3338	115.1716	96.16431
97.32599	84.18473	101.58494	66.59196	105.92151	108.65871
101.373314	97.82232	116.24684	110.46861	93.28166	103.72418
182.841855	99.70088	107.042816	125.60128	111.761765	109.58706
132.7364	104.63147	97.961006	88.62893	88.136314	98.253265
77.87444	77.54417	77.323425	88.7316	72.1582	65.22426
62.3559	58.03457	52.761154	49.708244	43.34345	50.544857
36.742672	46.267952	44.044575	33.271946	25.456364	27.537779
23.141676	22.526178	23.30675	21.257904	24.558527	21.752968
19.125467	19.41524	19.124939	19.812616	10.813507	9.524088
8.9117985	9.221161	11.106175	9.440088	10.182383	11.372046
12.518983]					

Gambar 7.39. Hasil Prediksi

Baiklah, dengan ini kita telah berhasil membuat model time series forecasting dengan real dataset. Selamat buat kalian, tetap semangat untuk terus belajar dan eksplorasi.

7.4. Kode Keseluruhan

Untuk mengetahui apakah kalian sudah mengikuti dengan baik atau belum. Berikut merupakan kode keseluruhan notebook di bab ini, kalian boleh memeriksa dan mencocokknya dengan hasil yang sudah kalian kerjakan.

TIME SERIES FORECASTING DENGAN REAL DATASET

Mengimpor Library dan Memeriksa Versi Tensorflow

```
[ ] import tensorflow as tf  
import numpy as np  
import matplotlib.pyplot as plt  
print(tf.__version__)
```

2.5.0

Membuat Fungsi Grafik Data Series

```
[ ] def plot_series(time, series, format="-", start=0, end=None):  
    plt.plot(time[start:end], series[start:end], format)  
    plt.xlabel("Waktu")  
    plt.ylabel("Nilai")  
    plt.grid(True)
```

Menghubungkan dengan Google Drive

```
[ ] from google.colab import drive #Mengimpor G-Drive  
  
drive.mount('/content/drive') #Mengatur lokasi Drive
```

Mounted at /content/drive

Mengakses Dataset

```
[ ] import pandas as pd #Mengimpor Library Pandas

#Membaca File Dataset CSV
Dataset = pd.read_csv('/content/drive/MyDrive/Dataset/Sunspots.csv')
#Melihat Info Dataset
Dataset.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3235 entries, 0 to 3234
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Unnamed: 0        3235 non-null   int64  
 1   Date              3235 non-null   object  
 2   Monthly Mean Total Sunspot Number 3235 non-null   float64 
dtypes: float64(1), int64(1), object(1)
memory usage: 75.9+ KB
```

```
[ ] #Melihat Isi Dataset
Dataset.head()
```

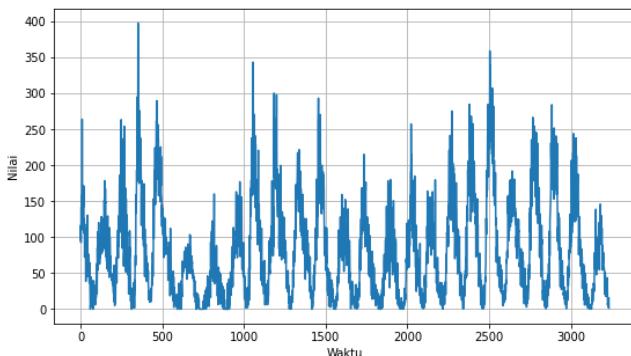
	Unnamed: 0	Date	Monthly Mean Total Sunspot Number
0	0	1749-01-31	96.7
1	1	1749-02-28	104.3
2	2	1749-03-31	116.7
3	3	1749-04-30	92.8
4	4	1749-05-31	141.7

Membuka Dataset

```
[ ] import csv #Mengimpor CSV
time_step = []
sunspot = []

#Membuka File CSV
with open('/content/drive/MyDrive/Dataset/Sunspots.csv') as csvfile:
    reader = csv.reader(csvfile, delimiter=',')
    next(reader)
    for row in reader:
        sunspot.append(float(row[2]))
        time_step.append(int(row[0]))

#Membuat Visualisasi Dataset
series = np.array(sunspot)
time = np.array(time_step)
plt.figure(figsize=(9, 5))
plot_series(time, series)
```



Membagi Dataset

```
[ ] split_time = 3000
time_train = time[:split_time]
x_train = series[:split_time]
time_valid = time[split_time:]
x_valid = series[split_time:]

window_size = 30
batch_size = 32
shuffle_buffer_size = 1000
```

Membuat Windowed Dataset

```
[ ] def windowed_dataset(series, window_size, batch_size, shuffle_buffer):
    series = tf.expand_dims(series, axis=-1)
    dataset = tf.data.Dataset.from_tensor_slices(series)
    dataset = dataset.window(window_size + 1, shift=1, drop_remainder=True)
    dataset = dataset.flat_map(lambda w: w.batch(window_size + 1))
    dataset = dataset.shuffle(shuffle_buffer)
    dataset = dataset.map(lambda w: (w[:-1], w[1:]))
    return dataset.batch(batch_size).prefetch(1)
```

Membuat Fungsi Untuk Memprediksi Model

```
[ ] def prediksi_model(model, series, window_size):
    dataset = tf.data.Dataset.from_tensor_slices(series)
    dataset = dataset.window(window_size, shift=1, drop_remainder=True)
    dataset = dataset.flat_map(lambda w: w.batch(window_size))
    dataset = dataset.batch(32).prefetch(1)
    prediksi = model.predict(dataset)
    return prediksi
```

Membuat Model

```
[ ] #Membersihkan Variabel luar
tf.keras.backend.clear_session()
tf.random.set_seed(51)
np.random.seed(51)

window_size = 64 #Mengatur Ukuran Window
batch_size = 256 #Mengatur Ukuran Batch

#Memanggil Windowed Dataset
train_set = windowed_dataset(x_train, window_size, batch_size,
                             shuffle_buffer_size)

print(train_set) #Melihat shape train_set
print(x_train.shape) #Melihat shape x_train

[ ] #Membuat Model
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv1D(filters=32, kernel_size=5,
                          strides=1, padding="causal",
                          activation="relu",
                          input_shape=[None, 1]), #Membuat Layer CNN
    tf.keras.layers.LSTM(64, return_sequences=True), #Membuat Layer lstm
    tf.keras.layers.LSTM(64, return_sequences=True),
    tf.keras.layers.Dense(30, activation="relu"), #Membuat Dense Layer
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(1),
    tf.keras.layers.Lambda(lambda x: x * 400) #Membuat Lambda Layer
])

#Membuat Callbacks
lr_schedule = tf.keras.callbacks.LearningRateScheduler(
    lambda epoch: 1e-8 * 10**((epoch / 20)))
optimizer = tf.keras.optimizers.SGD(learning_rate=1e-8, momentum=0.9)

#Mengcompile Model
model.compile(loss=tf.keras.losses.Huber(),
              optimizer=optimizer,
              metrics=["mae"])
```

```
[ ] model.summary() #Melihat Summary Model
Model: "sequential"
-----
```

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, None, 32)	192
lstm (LSTM)	(None, None, 64)	24832
lstm_1 (LSTM)	(None, None, 64)	33024
dense (Dense)	(None, None, 30)	1950
dense_1 (Dense)	(None, None, 10)	310
dense_2 (Dense)	(None, None, 1)	11
lambda (Lambda)	(None, None, 1)	0

```
-----  
Total params: 60,319  
Trainable params: 60,319  
Non-trainable params: 0
```

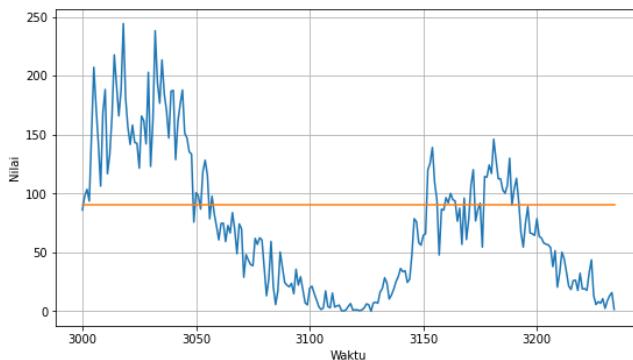
Mentraining Model

```
[ ] history = model.fit(train_set, epochs=100, callbacks=[lr_schedule])
```

Visualisasi Hasil Prediksi

```
[ ] prediksi_rnn = prediksi_model(model, series[...], np.newaxis, window_size)
prediksi_rnn = prediksi_rnn[split_time - window_size:-1, -1, 0]

plt.figure(figsize=(9, 5))
plot_series(time_valid, x_valid)
plot_series(time_valid, prediksi_rnn)
```



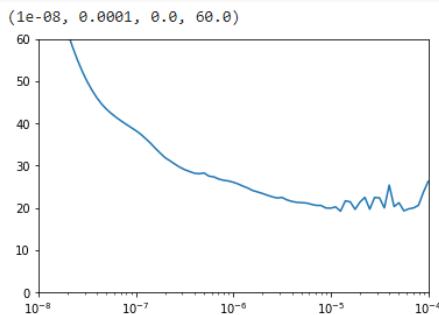
Mengukur MAE

```
[ ] tf.keras.metrics.mean_absolute_error(x_valid, prediksi_rnn).numpy()  
52.40565
```

Memperbaiki Model

Mencari Nilai Optimum

```
[ ] plt.semilogx(history.history["lr"], history.history["loss"])  
plt.axis([1e-8, 1e-4, 0, 60])
```



Membuat Model Baru

```
[ ] tf.keras.backend.clear_session()  
tf.random.set_seed(51)  
np.random.seed(51)  
train_set = windowed_dataset(x_train, window_size=60, batch_size=100,  
                             shuffle_buffer=shuffle_buffer_size)  
model = tf.keras.models.Sequential([  
    tf.keras.layers.Conv1D(filters=60, kernel_size=5,  
                          strides=1, padding="causal",  
                          activation="relu",  
                          input_shape=[None, 1]),  
    tf.keras.layers.LSTM(60, return_sequences=True),  
    tf.keras.layers.LSTM(60, return_sequences=True),  
    tf.keras.layers.Dense(30, activation="relu"),  
    tf.keras.layers.Dense(10, activation="relu"),  
    tf.keras.layers.Dense(1),  
    tf.keras.layers.Lambda(lambda x: x * 400)  
])
```

```
optimizer = tf.keras.optimizers.SGD(learning_rate=1e-5, momentum=0.9)
model.compile(loss=tf.keras.losses.Huber(),
              optimizer=optimizer,
              metrics=["mae"])
```

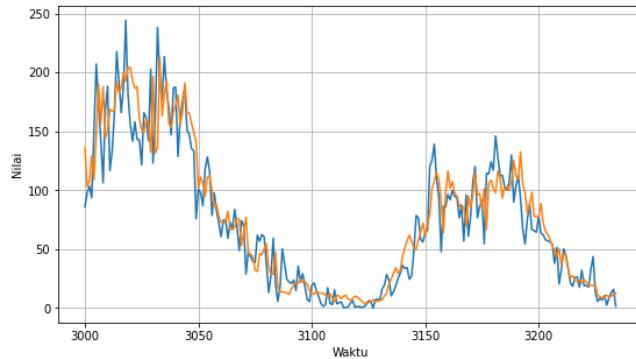
Mentraining Model

```
[ ] history = model.fit(train_set, epochs=500)
```

Visualisasi Hasil Prediksi

```
[ ] rnn_baru = prediksi_model(model, series[..., np.newaxis], window_size)
rnn_baru = rnn_baru[split_time - window_size:-1, -1, 0]
```

```
[ ] plt.figure(figsize=(9, 5))
plot_series(time_valid, x_valid)
plot_series(time_valid, rnn_baru)
```



Mengukur MAE

```
[ ] tf.keras.metrics.mean_absolute_error(x_valid, rnn_baru).numpy()
```

```
15.620486
```

Visualisasi Grafik Training Loss

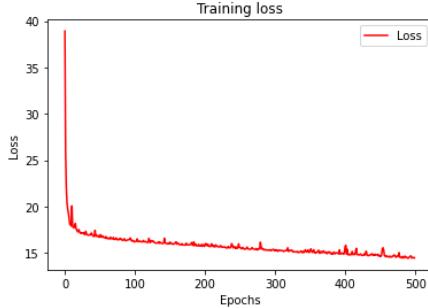
```
[ ] #import library matplotlib
import matplotlib.image as mpimg
import matplotlib.pyplot as plt

loss=history.history['loss'] #mengambil data loss
epochs=range(len(loss)) #Menghitung jumlah epoch

#Membuat visualisasi grafik loss
plt.plot(epochs, loss, 'r')
plt.title('Training loss')
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend(["Loss"])

plt.figure()
```

<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>

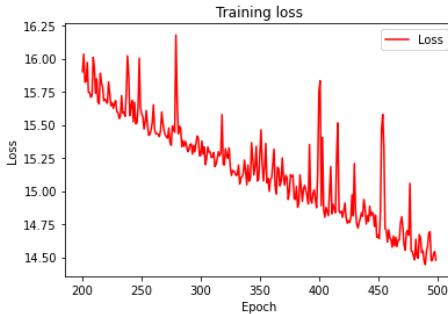
Visualisasi Grafik Training Loss lebih Detail

```
[ ] zoom_loss = loss[200:] #Mengambil data dari 200
zoom_epoch = range(200,500) #Range data dari 200 sampai 500

#Memvisualisasikan grafik
plt.plot(zoom_epoch, zoom_loss, 'r')
plt.title('Training loss')
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend(["Loss"])

plt.figure()
```

<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>

Hasil Prediksi RNN

```
[ ] print(rnn_baru)

[135.93948 103.12235 107.854065 129.22385 108.84819 147.39487
189.78087 156.79062 187.81398 144.31122 150.0124 168.67754
166.9058 167.96768 192.48328 182.7197 186.66794 200.11948
192.28748 204.5711 203.98186 192.89357 186.52187 187.82918
160.57487 151.08842 148.91212 161.18733 156.05412 132.18439
196.56184 131.38205 136.06392 212.91818 162.95776 181.05084
193.0767 157.0092 153.59818 165.92699 172.63889 180.77162
155.54376 166.10939 191.04233 166.12889 165.59535 156.55511
148.85811 141.43314 182.08545 111.27963 106.03841 94.656075
111.22477 112.19361 99.72339 86.33762 87.99869 77.85541
72.827484 72.42288 74.62233 82.11267 66.877884 67.77999
68.20702 76.20557 75.08509 52.962357 64.73747 77.068375
46.685688 45.63053 44.194557 32.430347 31.180859 46.316635
45.288518 50.386272 54.557896 34.189955 28.836811 28.727839
47.2776 17.91009 14.0321865 13.622933 13.497494 12.69638
11.581974 16.43831 19.259253 21.55281 21.041985 24.788696
21.761194 22.558731 19.731035 14.582336 12.879787 11.788781
14.488938 12.563949 12.8702345 12.583889 10.987878 9.625046
12.106937 8.611035 5.0097373 10.786907 10.411265 7.8488025
10.0881109 10.927359 7.637903 6.963053 7.954654 10.097498
9.621684 7.60202 6.0288906 4.388717 4.108143 4.6765475
5.578646 7.119854 5.594924 5.6613236 6.3322678 7.164158
10.290985 12.593584 20.88556 24.778852 29.313097 34.24381
29.524559 32.3227 43.777554 51.188503 57.03432 61.741848
57.025562 51.106476 50.24871 58.000607 64.55911 72.112335
61.50855 71.31736 79.54353 101.18656 107.90452 115.040794
108.574 90.5965 64.021126 98.0149 116.394714 101.63542
107.30144 97.21289 95.027245 87.01589 84.5397 85.66534
69.92878 95.28463 84.83131 96.3338 115.1716 96.16431
97.32599 84.10473 101.58494 66.59196 105.92151 108.65871
101.373314 97.82232 116.24684 110.46861 93.20166 103.72418
102.041855 99.70088 107.042816 125.60128 111.761765 109.58706
132.7364 104.63147 97.961006 80.62893 88.136314 98.253265
77.87444 77.54417 77.323425 88.7316 72.1582 65.22426
62.3559 58.03457 52.761154 49.708244 43.34345 50.544857
36.742672 46.267952 44.044575 33.271946 25.456364 27.537779
23.141676 22.526178 23.30675 21.257904 24.558527 21.752968
19.125467 19.41524 19.124939 19.812616 10.813507 9.524088
8.9117985 9.221161 11.106175 9.440088 10.182383 11.372046
12.518983 ]
```

Gambar 7.40. Kode Keseluruhan Bab 7

CATATAN

Berikut merupakan link untuk melihat notebook kode time series forecasting pada buku ini .

- Notebook Bab 3 Time Series Forecasting dengan Neural Network :
<https://bit.ly/NotebookBab3>
- Notebook Bab 4 Time Series Forecasting dengan Deep Neural Network :
<https://bit.ly/NotebookBab4>
- Notebook Bab 5 Time Series Forecasting dengan Recurrent Neural Network :
<https://bit.ly/NotebookBab5>
- Notebook Bab 6 Time Series Forecasting dengan LSTM :
<https://bit.ly/NotebookBab6>
- Notebook Bab 7 Time Series Forecasting dengan Real Dataset :
<https://bit.ly/NotebookBab7>
- Link Download Dataset :
<https://www.kaggle.com/robervalt/sunspots>

DAFTAR PUSTAKA

- [1] A. L. Samuel, “Some studies in machine learning using the game of checkers. II-Recent progress,” *Annu. Rev. Autom. Program.*, vol. 6, no. PART 1, pp. 1–36, 1969, doi: 10.1016/0066-4138(69)90004-4.
- [2] Mark Esposito, “What is machine learning?,” *theconversation.com*, 2017. <https://theconversation.com/what-is-machine-learning-76759> (accessed Jun. 10, 2021).
- [3] Dicoding, “Jenis-jenis Machine Learning,” *dicoding.com*, 2020. <https://www.dicoding.com/academies/184/tutorials/8326> (accessed Jun. 12, 2021).
- [4] Phil Winder, *Reinforcement Learning*, Chapter 1. O'Reilly Media, Inc, 2020.
- [5] T. K. Gautama, A. Hendrik, and R. Hendaya, “Pengenalan Objek pada Computer Vision dengan Pencocokan Fitur Menggunakan Algoritma SIFT Studi Kasus: Deteksi Penyakit Kulit Sederhana,” *J. Tek. Inform. dan Sist. Inf.*, vol. 2, no. 3, pp. 437–450, 2016, doi: 10.28932/jutisi.v2i3.554.
- [6] M. Bedy Purnama, S.Si., *Pengantar Machine Learning*. Bandung: Penerbit Informatika, 2019.
- [7] Reyvan Maulid, “Yuk Pahami Jenis-jenis Algoritma Deep Learning,” *dqlab.id*, 2021. <https://dqlab.id/yuk-pahami-jenis-jenis-algoritma-deep-learning> (accessed Jun. 14, 2021).
- [8] J. E. Hanke and D. W. Wichers, *Business Forecasting*, Eight Edit.

New Jersey: Pearson Prentice hall, 2005.

- [9] Y. Astuti, B. Novianti, T. Hidayat, and D. Maulina, “Penerapan Metode Single Moving Average Untuk Peramalan Penjualan Mainan Anak,” *Semin. Nas. Sist. Inf. dan Tek. Inform. Sensitif*, vol. 4, no. July, p. 255, 2019.
- [10] I. Firdaus, “Mengenal Time Series Forecasting,” *Medium.com*, 2019. <https://medium.com/ppl-d7-fasilkom-ui/mengenal-time-series-forecasting-647929e16b3f> (accessed Jun. 16, 2021).
- [11] L. Moroney, “Common patterns in time series,” *coursera.org*, 2020. <https://www.coursera.org/learn/tensorflow-sequences-time-series-and-prediction/lecture/dWnOy/common-patterns-in-time-series> (accessed Jun. 16, 2021).
- [12] Jason Brownlee, “What Is Time Series Forecasting?,” *machinelearningmastery.com*, 2016. <https://machinelearningmastery.com/time-series-forecasting/> (accessed Jun. 16, 2021).
- [13] L. Moroney, *AI and Machine Learning for Coders*, 1st Editio. O'Reilly Media, 2020.
- [14] C. R. Harris *et al.*, “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, 2020, doi: 10.1038/s41586-020-2649-2.
- [15] Fikisyi Habirawan, “Berkenalan dengan Matplotlib dan implementasinya,” *kotakode.com*, 2020. <https://kotakode.com/blogs/2665/Berkenalan-dengan-Matplotlib->

dan-implementasinya (accessed Jun. 17, 2021).

- [16] A. SkillPlus, “Pengenalan Features dan Labels Pada Machine Learning,” *skillplus.web.id*, 2020. <https://skillplus.web.id/pengenalan-features-dan-labels-pada-machine-learning/> (accessed Jun. 18, 2021).
- [17] C. J. Willmott and K. Matsuura, “Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance,” *Clim. Res.*, vol. 30, no. 1, pp. 79–82, 2005, doi: 10.3354/cr030079.
- [18] Dicoding, “Artificial Neural Network,” *dicoding.com*, 2020. <https://www.dicoding.com/academies/184/tutorials/8502?from=8497> (accessed Jun. 20, 2021).
- [19] L. Wiranda and M. Sadikin, “Penerapan Long Short Term Memory Pada Data Time Series Untuk Memprediksi Penjualan Produk Pt. Metiska Farma,” *J. Nas. Pendidik. Tek. Inform.*, vol. 8, no. 3, pp. 184–196, 2019.
- [20] C. Snijders, U. Matzat, and U. Reips, “‘Big Data’: Big Gaps of Knowledge in the Field of Internet Science,” *Int. J. Internet Sci.*, vol. 7, no. 1, pp. 1–5, 2013.
- [21] Especuloide, “Sunspots,” *kaggle.com*, 2021. <https://www.kaggle.com/robertvalt/sunspots> (accessed Jun. 26, 2021).

PROFIL PENULIS



Nama	: Mellania Permata Sylvie
Tempat, Tanggal Lahir	: Sidoarjo, 10 Mei 2000
Alamat	: Desa Wonokupang, RT.2 RW.1 Kec. Balongbendo Sidoarjo, Jawa Timur.
No. Telepon	: 085781761657
Email	: mellaniapermata@gmail.com
LinkedIn	: www.linkedin.com/in/mellania-sylvie
GitHub	: https://github.com/mellasyylvie

Penulis merupakan salah satu mahasiswa Universitas Trunojoyo Madura, program studi sistem informasi. Saat menyusun buku ini, penulis sedang berada di semester 6. Buku ini pun disusun untuk memenuhi syarat kelulusan kerja praktek prodi sistem informasi Universitas Trunojoyo Madura.