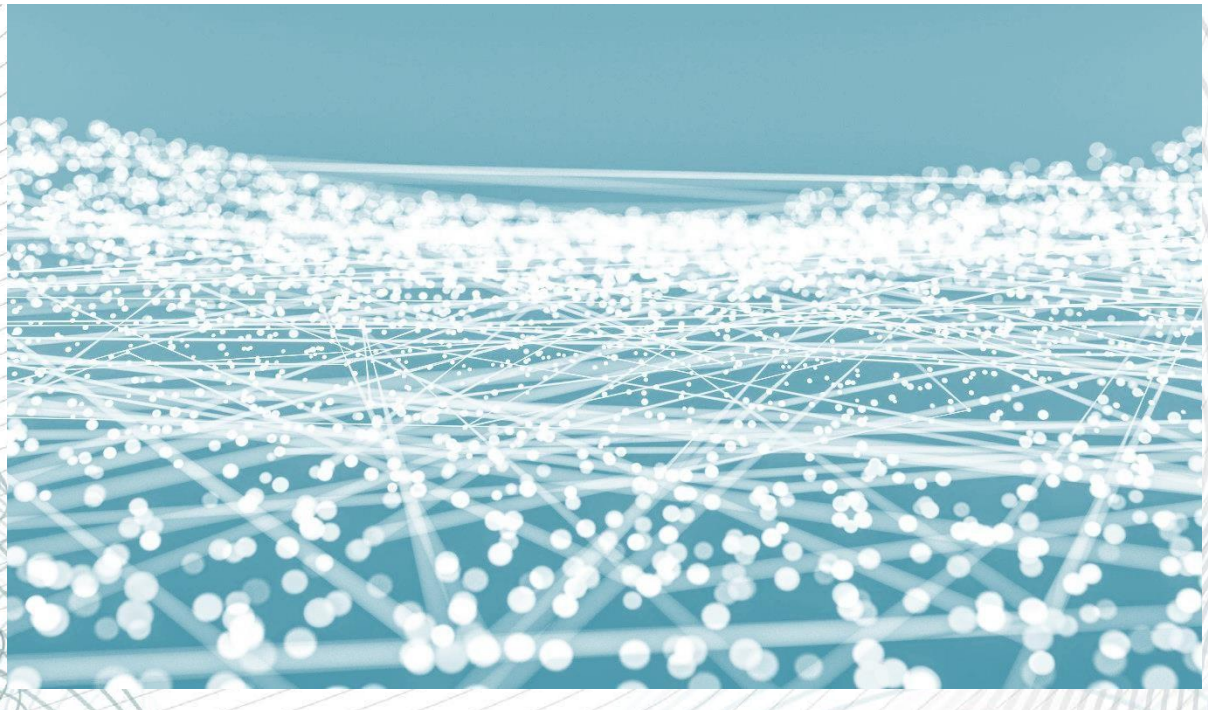
MODUL PRAKTIKUM

SD25-30003-Deep Learning



Program Studi Sains Data
Fakultas Sains
Institut Teknologi Sumatera

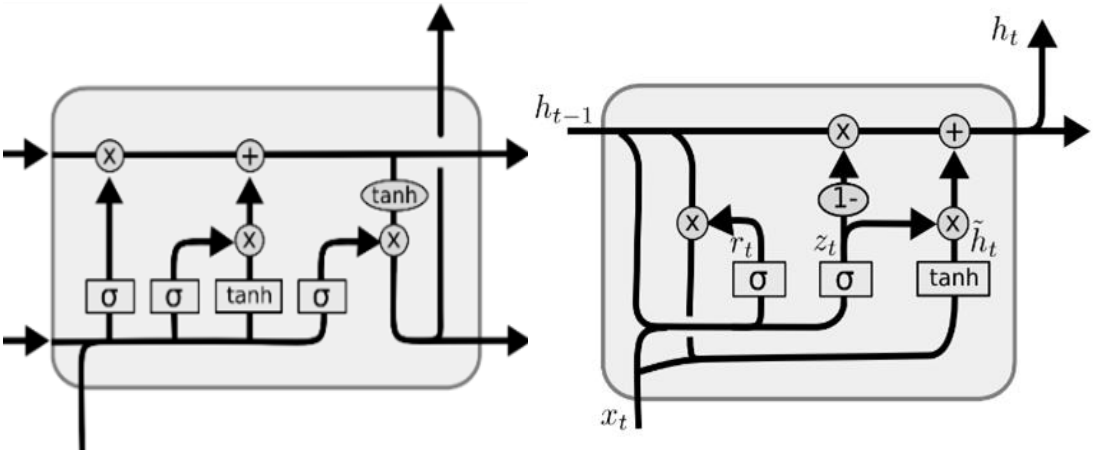
2025

MODUL 7

Long Short-Term Memory (LSTM) dan Gated Recurrent Unit (GRU)

A. Konsep Dasar

Long Short-Term Memory (LSTM) adalah salah satu jenis *Recurrent Neural Network* (RNN) yang dikembangkan untuk mengatasi masalah *vanishing gradient* yang umum terjadi pada RNN standar. Masalah ini biasanya muncul saat memproses data berurutan panjang, di mana informasi penting dari waktu ke waktu dapat hilang atau menurun nilainya saat backpropagation dilakukan. LSTM memiliki struktur khusus yang mencakup unit memori atau "sel memori", yang mampu menyimpan informasi dalam jangka waktu yang lama. Dalam LSTM, setiap unit memori dilengkapi dengan tiga gerbang utama: *input gate, forget gate,* dan *output gate*. Gerbang-gerbang ini bertugas mengatur aliran informasi yang masuk, disimpan, dan dikeluarkan, sehingga memungkinkan model untuk menyimpan informasi penting dari urutan data dalam waktu yang lama sekaligus melupakan informasi yang tidak relevan. Berikut mekanisme kerja dari LSTM dan GRU.



Gambar 1. LSTM (kiri), GRU(kanan) (sumber)

Gated Recurrent Unit (GRU) merupakan salah satu jenis dari RNN yang diusulkan untuk mengatasi masalah vanishing gradient dalam jaringan berlapis-lapis. GRU lebih sederhana dibandingkan dengan LSTM karena hanya memiliki dua jenis gerbang (gate): reset gate dan update gate. Reset gate mengatur seberapa banyak informasi lama yang dilupakan. Sedangkan update gate menentukan seberapa banyak informasi lama dipertahankan dan informasi baru ditambahkan.

Tabel 1. Perbedaan utama RNN, LSTM dan GRU

Model	Karakteristik	(+)	(-)
RNN (Recurrent Neural Network)	Memproses data secara berurutan, membawa **hidden state** dari langkah sebelumnya	Sederhana, efisien untuk **urutan pendek**	Sulit menangani urutan panjang (vanishing / exploding gradient)
LSTM (Long Short-Term Memory)	Menambahkan **cell state** dan gates (**input, forget, output**) untuk menjaga memori jangka panjang	Cocok untuk data **urutan panjang**	Lebih kompleks dan lambat
GRU (Gated Recurrent Unit)	Versi sederhana LSTM dengan **dua gate (reset, update)**	Lebih cepat, efisien untuk **urutan menengah–panjang**	Sedikit kurang presisi dibanding LSTM pada urutan sangat panjang

Hyperparameter adalah variabel eksternal yang ditentukan sebelum pelatihan dimulai dan tidak dipelajari langsung dari data. Nilainya sangat memengaruhi kapasitas model, kecepatan konvergensi, dan akurasi hasil.

Tabel 2. *Hyperparameters* pada Model Data Sekuensial

Hyperparameter	Fungsi
Time step (sequence length)	Jumlah langkah waktu yang dilihat model sekaligus
Hidden units (neurons)	Kapasitas memori model
Number of layers	Kedalaman jaringan
Learning rate	Kecepatan update bobot
Batch size	Jumlah sampel per update
Epochs	Jumlah iterasi penuh pada dataset
Dropout rate	Regularisasi untuk mencegah overfitting
Activation function	Fungsi non-linear neuron
Optimizer	Algoritma pembaruan bobot

B. Tujuan Praktikum

I. Tujuan Instruksional Umum

Praktikum bertujuan untuk menerapkan model *Long Short-Term Memory (LSTM) dan Gated Recurrent Unit (GRU)* dalam pemahaman *Deep Learning.*

II. Tujuan Instruksional Khusus

1. Mahasiswa mampu menguasai konsep dasar model *Long Short-Term Memory (LSTM) dan Gated Recurrent Unit (GRU).*
2. Mahasiswa dapat membangun model LSTM dan GRU.
3. Mahasiswa mampu menganalisis hasil evaluasi *hyperparameter* model LSTM dan GRU.

C. Dataset dan bahasa pemrograman Python

C.1 Dataset dalam praktikum

Praktikum pada modul ini menggunakan model LSTM dan GRU untuk memprediksi harga Bitcoin di masa mendatang. Model ini menggunakan data harga penutupan harian untuk Bitcoin (Download dataset). Berikut adalah isi dari dataset yang digunakan.

1	Date	Open	High	Low	Close	Adj Close	Volume
2	2014-09-17	465.864014	468.174011	452.421997	457.334015	457.334015	21056800
3	2014-09-18	456.859985	456.859985	413.104004	424.440002	424.440002	34483200
4	2014-09-19	424.102997	427.834991	384.532013	394.795990	394.795990	37919700
5	2014-09-20	394.673004	423.295990	389.882996	408.903992	408.903992	36863600
6	2014-09-21	408.084991	412.425995	393.181000	398.821014	398.821014	26580100
7	2014-09-22	399.100006	406.915985	397.130005	402.152008	402.152008	24127600
8	2014-09-23	402.092010	441.557007	396.196991	435.790985	435.790985	45099500
9	2014-09-24	435.751007	436.112000	421.131989	423.204987	423.204987	30627700
10	2014-09-25	423.156006	423.519989	409.467987	411.574005	411.574005	26814400

Data ini disimpan dalam berkas 'BTC-USD.csv'. Setiap baris dalam berkas mewakili satu hari, dan kolom 'Close' digunakan untuk pelatihan dan prediksi.

Berikut penjelasan untuk tiap kolom yang ada:

- Date: Tanggal pencatatan data pada masing-masing baris, dalam format YYYY-MM-DD.
- Open: Harga pembukaan Bitcoin pada awal hari.
- High: Harga tertinggi yang dicapai Bitcoin sepanjang hari.
- Low: Harga terendah yang dicapai Bitcoin pada hari itu.
- Close: Harga penutupan Bitcoin di akhir hari.
- Adj Close: Harga penutupan yang disesuaikan dengan faktor-faktor tertentu (seperti stock split atau dividen pada konteks lain; untuk Bitcoin, biasanya sama dengan Close karena tidak ada penyesuaian khusus)
- Volume: Jumlah total Bitcoin yang diperdagangkan sepanjang hari, menunjukkan tingkat aktivitas atau likuiditas pasar pada hari tersebut.

C.2 Bahasa Pemrograman Python

Pada praktikum kali ini, kita akan menggunakan IDE Python Service dari Google Colab, sebelumnya perlu menyiapkan akun google pribadi untuk mengakses servis ini di akun google masing-masing.

Library yang akan digunakan diantaranya Pandas, TensorFlow, dan Keras.

1) Pandas : untuk memanipulasi dan memproses data
2) NumPy : digunakan untuk operasi matematika berbasis array dan matriks. NumPy sangat penting dalam komputasi numerik dengan Python pada praktikum ini.
3) matplotlib.pyplot : untuk membuat visualisasi hasil model.
4) TensorFlow : framework yang kuat dan fleksibel untuk machine learning dan deep learning, cocok untuk proyek skala besar dan aplikasi industri.

5) Keras : menyediakan antarmuka yang lebih sederhana untuk membangun model, membuat pengembangan deep learning lebih cepat dan mudah dipahami.

Integrasi TensorFlow dan Keras

Sejak TensorFlow 2.0, Keras menjadi API default untuk membangun model di TensorFlow, membuat pengembangan model deep learning jauh lebih mudah dan efisien. Tanpa membangun code TensorFlow dari awal, Anda bisa menggunakan Keras untuk membangun arsitektur model dengan cara yang lebih sederhana dan cepat.

Praktikum deep learning pada modul ini dapat maksimal dipraktikan dengan pemahaman bidang atau mata kuliah terkait seperti algoritma pemrograman, struktur data, dan machine learning.

D. Implementasi LSTM

1) Import Library

```
# Import necessary libraries
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
import matplotlib.pyplot as plt
```

Pandas untuk memanipulasi dan memproses data. **MinMaxScaler** dari **sklearn.preprocessing** untuk menormalisasi data. **numpy** untuk manipulasi array. **tensorflow** untuk membangun dan melatih model neural network. Dan **matplotlib.pyplot** untuk membuat visualisasi hasil model.

2) Load dataset

Data CSV yang berisi harga Bitcoin diimpor dan disimpan dalam df.

```
# Load and preprocess data
# Load data
df = pd.read_csv('BTC-USD.csv')  # Replace with your filename
data = df.filter(['Close'])
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)
```

Hanya kolom "Close" (harga penutupan) yang dipilih untuk analisis. Kolom "Date" diatur sebagai indeks untuk memudahkan visualisasi data per tanggal.

3) Convert dataset

```
# Convert the dataframe to a numpy array
dataset = data.values

# Scale the data
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(dataset)
```

Data diubah menjadi numpy array, sehingga memudahkan proses dengan model LSTM. **MinMaxScaler** digunakan untuk menormalisasi data ke dalam rentang [0, 1], yang membantu menghindari vanishing gradient dan mempercepat konvergensi model.

4) Split dataset

```
# Split the data into training set and test set
train_ratio = 0.8
training_data_len = int(np.round(train_ratio * len(dataset)))
```

Data dipecah menjadi dua bagian: training set (80% dari data) dan test set (20% sisanya).

5) Buat Urutan titik data (Sequences of Data Points)

Lookback period ditentukan sebanyak 60, yang berarti model akan menggunakan 60 hari data sebelumnya untuk memprediksi harga pada hari ke-61. **x_train** berisi sekumpulan data dalam jendela waktu 60 hari, dan **y_train** berisi target prediksi pada hari ke-61. x_train direstrukturisasi ke dalam format 3D (**samples, timesteps, features**) yang dibutuhkan oleh LSTM.

```
# Define the lookback period and split into samples
lookback = 60
x_train, y_train = [], []

for i in range(lookback, len(scaled_data)):
    x_train.append(scaled_data[i-lookback:i, 0])
    y_train.append(scaled_data[i, 0])

# Convert x_train and y_train to numpy arrays
x_train, y_train = np.array(x_train), np.array(y_train)

# Reshape the data to 3-D so it's suitable for LSTM model
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
```

6) Membangun model LSTM

```
# Build / Define the modified LSTM model
# Build / Define the LSTM model
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(x_train.shape[1], 1)))
model.add(LSTM(50, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
```

Model LSTM dibuat dengan dua lapisan LSTM, lapisan pertama dengan 50 unit LSTM yang diatur **return_sequences=True**, untuk memastikan outputnya cocok dengan lapisan LSTM berikutnya. Lapisan kedua dengan 50 unit LSTM dan **return_sequences=False**, sehingga lapisan ini hanya menghasilkan satu output. Lapisan Dense dengan 25 neuron untuk pemetaan ke lapisan selanjutnya, dan lapisan Dense terakhir dengan 1 neuron untuk memprediksi harga.

7) Kompilasi dan lakukan pelatihan model

```
# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Train model with more epochs
history = model.fit(x_train, y_train, batch_size=32, epochs=1, validation_split=0.2)
```

Model dikompilasi menggunakan optimizer **adam** dan **mean squared error** sebagai fungsi **loss**. Model dilatih pada data latih dengan **batch_size=32** dan **epochs=1** (dapat ditingkatkan untuk hasil yang lebih akurat).

8) Persiapkan data uji (Testing data)

```
# Test model
# Create the testing data set
test_data = scaled_data[training_data_len - lookback:, :]

# Create the data sets x_test and y_test
x_test = []
y_test = dataset[training_data_len:, :]
for i in range(lookback, len(test_data)):
    x_test.append(test_data[i-lookback:i, 0])

# Convert the data to a numpy array
x_test = np.array(x_test)

# Reshape the data
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))
```

Data tes disiapkan dengan cara yang sama seperti data latih. Data tes terdiri dari data 60 hari sebelumnya hingga akhir dataset. x_test direstrukturisasi ke dalam format 3D untuk input LSTM. Pada tahap ini akan membuat prediksi masa depan untuk harga Bitcoin. Tanggal awal untuk prediksi masa depan ditetapkan pada 13 Juli 2023 dan tanggal akhir ditetapkan pada akhir tahun 2024.

9) Lakukan Prediksi dan Evaluasi model

```
# Get the model's predicted price values
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)

# Evaluate model
# Calculate RMSE, MAE, MAPE
rmse = np.sqrt(np.mean((predictions - y_test) ** 2))
mae = np.mean(np.abs(predictions - y_test))
mape = np.mean(np.abs((predictions - y_test) / y_test)) * 100

print(f'RMSE: {rmse}')
print(f'MAE: {mae}')
print(f'MAPE: {mape}%')
```

Prediksi dibuat pada data tes, dan hasilnya di-skala balik ke harga asli. Untuk mengevaluasi performa model perlu dihitung nilai dari:

- Root Mean Square Error (RMSE): Mengukur rata-rata kesalahan, memberi penalti besar untuk kesalahan besar.
- Mean Absolute Error (MAE) : Rata-rata kesalahan absolut, lebih mudah diinterpretasikan.

- Mean Absolute Percentage Error (MAPE): Kesalahan rata-rata dalam persentase, cocok untuk perbandingan antar model.

10) Visualisasi prediksi

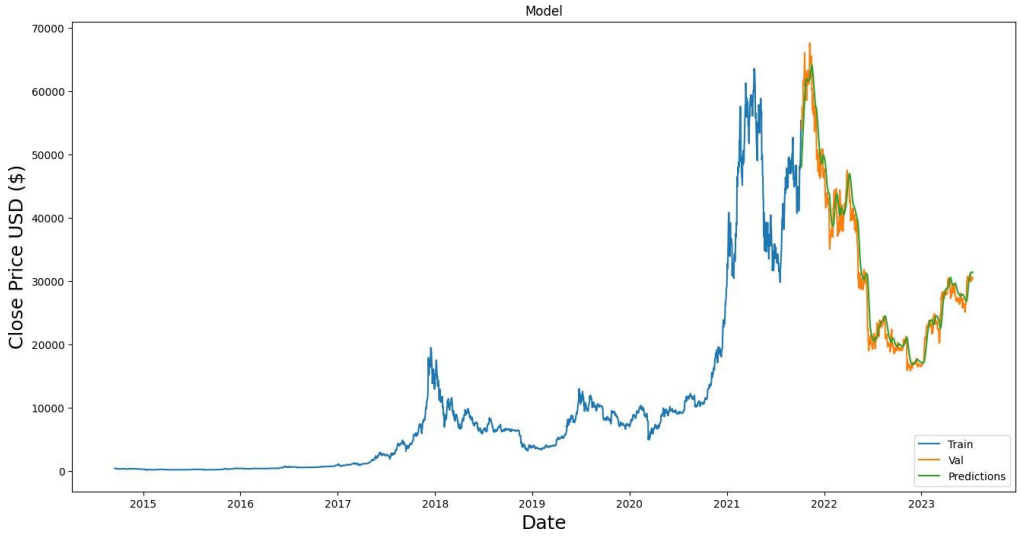
Data latih dan data tes ditampilkan pada grafik, bersama dengan prediksi model. Grafik menunjukkan performa model dalam memprediksi harga close di masa depan.

```
# Plot the data
train = df[:training_data_len]
valid = df[training_data_len:]
valid['Predictions'] = predictions

# Visualize the data
plt.figure(figsize=(16, 8))
plt.title('Model')
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.plot(train.index, train['Close'])
plt.plot(valid.index, valid[['Close', 'Predictions']])
plt.legend(['Train', 'Val', 'Predictions'], loc='lower right')
plt.show()
```

Output:

```
80/80 ━━━━━━━━━━━━━━━━━━━━ 9s 65ms/step - loss: 0.0176 - val_loss: 0.0013
21/21 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step
RMSE: 2524.444022084509
MAE: 1831.9705514545808
MAPE: 5.960884452212369%
```



TUGAS INDIVIDU

- Tugas individu dikerjakan saat praktikum berlangsung dengan waktu yang ditentukan.
- Implementasi model LSTM dan GRU dengan *hyperparameter* yang ada pada spreadsheets dari asisten praktikum.
- Satu mahasiswa menjalankan dua skenario (1 model LSTM dan 1 GRU)
- Setelah semua skenario selesai dijalankan semua praktikan, setiap mahasiswa melakukan analisis terkait *hyperparameter* yang berpengaruh terhadap hasil evaluasi (buat dalam PDF).