DAFTAR PUSTAKA

Abraham Tara H. (2002). Asal usul intelektual jaringan saraf McCulloch – Pitts. Jurnal Sejarah Ilmu Perilaku 38.1 , 3-25.

Arfan Adhib, & ETP Lussiana. (2020). Perbandingan Algoritma Long Short-Term Memory dengan SVR pada Prediksi Harga Saham di Indonesia. 33-43.

Bakhri Syamsul IR. (2021). MINYAK BUMI DI INDONESIA.

Balaji SA, & Baskaran K. (2013). Perancangan dan pengembangan sistem jaringan syaraf tiruan (JST) menggunakan fungsi aktivasi sigmoid untuk memprediksi produksi padi tahunan di Tamilnadu.

Hadihardaja, I. K., & Sutikno, S. (2005). Pemodelan Curah Hujan-Limpasan Menggunakan Artificial Neural Network (ANN) dengan Metode Backpropagation. 249-258.

Indarwatin Siska Arin. (2019). Penyelesaian Persamaan Diferensial Parsial dengan Pendekatan Artificial Neural Network.

Ivanedra Kasyfi, & Mustikasari Metty. (2019). IMPLEMENTASI METODE RECURRENT NEURAL NETWORK PADA TEXT SUMMARIZATION DENGAN TEKNIK ABSTRAKTIF. 377-382.

Kumar Ayan, Md Quadir Abdul, JecksonChristyJ, & dkk. (2023). Predicting and Curing Depression Using Long Short Term Memory and Global Vector.

Novikaginanto. (2012). https://novikaginanto.wordpress.com/2012/11/14/backpropogation/.

Rizkilloh Farryz Moch, & Widiyanesti Sri. (2022). Prediksi Harga Cryptocurrency Menggunakan Algoritma Long Short Term Memory (LSTM), 25-31.

Song Xuanyi, Liu Yuetian, Xue Liang, & dkk. (2020). Time-series well performance prediction based on Long Short-Term Memory (LSTM) neural network model .

WatiFitria Andriati, Erwan Yulistia Elvina , & dkk. (2019). INDUSTRI PENGOLAHAN MINYAK BUMI DI INDONESIA.

Zahara Soffa , Sugianto, & Iimiddafid Bahril M. (2019). Prediksi Indeks Harga Konsumen Menggunakan Metode Long Short Term Memory (LSTM) Berbasis Cloud Computing. 357-363.

LAMPIRAN

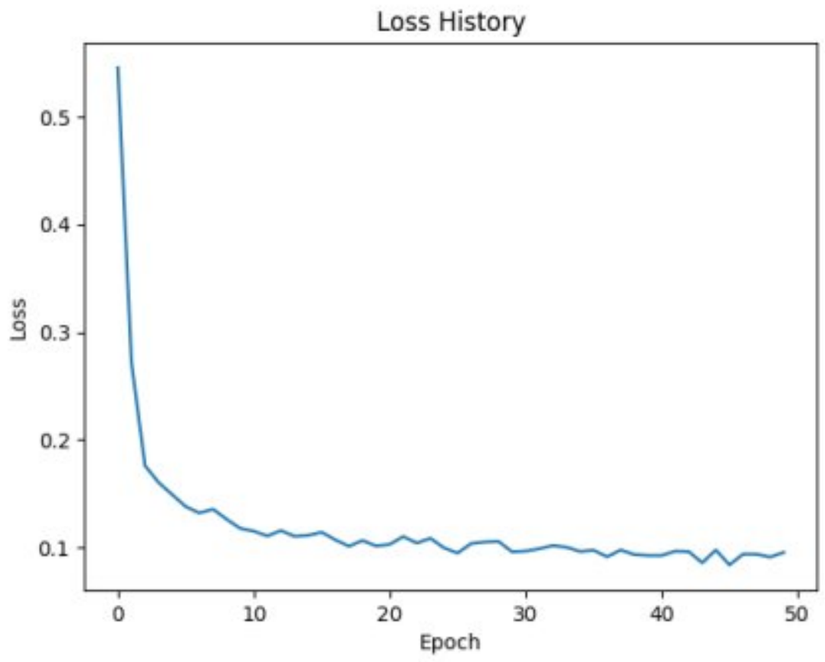
Lampiran 1. Data Produksi Minyak Bumi di Indonesia dari Tahun 1973 sampai 2022

TAHUN	Januari	Februari	Maret	April	Mei	Juni	Juli	Agustus	Septembe	Oktober	Novembe	Desember
1973	1157	1197	1216	1258	1364	1390	1373	1395	1390	1447	1432	1441
1974	1431	1416	1421	1480	1510	1469	1431	1332	1332	1234	1234	1214
1975	1333	1338	1193	1219	1100	1258	1326	1372	1376	1356	1410	1408
1976	1425	1456	1521	1528	1513	1478	1498	1511	1496	1539	1501	1581
1977	1629	1701	1701	1701	1696	1699	1698	1677	1670	1639	1708	1720
1978	1693	1698	1698	1677	1691	1642	1570	1608	1579	1603	1583	1582
1979	1593	1611	1622	1605	1593	1607	1601	1592	1571	1568	1568	1563
1980	1564	1553	1575	1553	1551	1575	1560	1569	1579	1586	1622	1636
1981	1632	1622	1634	1616	1614	1608	1604	1593	1589	1589	1574	1582
1982	1469	1444	1484	1239	1235	1301	1295	1235	1295	1357	1395	1315
1983	1188	984	1144	1358	1358	1358	1445	1445	1425	1474	1513	1396
1984	1415	1515	1505	1512	1415	1465	1340	1360	1350	1375	1300	1395
1985	1380	1401	1369	1369	1264	1106	1369	1369	1364	1327	1369	1317
1986	1458	1335	1335	1376	1463	1386	1381	1461	1345	1360	1306	1365
1987	1311	1281	1296	1311	1332	1332	1362	1485	1342	1352	1352	1352
1988	1278	1278	1329	1379	1379	1379	1379	1379	1278	1379	1278	1379
1989	1401	1401	1401	1401	1401	1401	1384	1434	1384	1434	1434	1434
1990	1306	1306	1411	1463	1411	1411	1442	1516	1536	1542	1568	1620
1991	1608	1608	1608	1608	1608	1608	1658	1608	1559	1510	1559	1559
1992	1580	1605	1630	1605	1530	1560	1550	1540	1550	1550	1550	1550
1993	1555	1535	1505	1485	1515	1515	1515	1515	1515	1485	1485	1515
1994	1506	1506	1506	1506	1506	1506	1506	1526	1506	1516	1516	1516
1995	1500	1480	1490	1490	1490	1490	1490	1490	1490	1540	1540	1540
1996	1540	1540	1540	1530	1530	1550	1520	1540	1560	1580	1570	1570
1997	1544	1564	1574	1534	1554	1504	1504	1504	1465	1465	1514	1514
1998	1520	1520	1520	1520	1520	1490	1490	1510	1510	1540	1540	1540
1999	1508	1488	1498	1498	1498	1478	1458	1448	1448	1448	1448	1448
2000	1422	1392	1392	1422	1452	1452	1452	1452	1452	1422	1412	1417
2001	1402	1407	1362	1325	1335	1354	1341	1333	1321	1311	1311	1282
2002	1292	1262	1262	1252	1252	1252	1247	1242	1242	1242	1232	1212
2003	1214	1209	1184	1164	1154	1149	1149	1134	1134	1129	1124	1124
2004	1109	1109	1099	1099	1094	1089	1089	1089	1089	1089	1089	1104
2005	1065	1074	1075	1066	1071	1062	1057	1062	1057	1038	1043	1044
2006	1011	1020	1021	1012	1017	1009	1004	1009	1004	986	991	988
2007	959	967	968	960	965	957	952	957	952	935	939	937
2008	982	990	991	983	988	980	975	980	975	957	962	959
2009	960	967	959	938	938	948	947	937	946	947	948	953
2010	944	957	961	966	970	960	943	961	946	897	922	913
2011	904	907	912	904	905	892	902	911	908	900	891	888
2012	884	884	882	870	879	860	854	853	845	837	833	835
2013	893	829	834	830	835	829	821	820	816	817	809	805
2014	789	789	789	789	789	789	789	789	789	789	789	789
2015	768	764	764	785	791	800	797	779	798	798	791	794
2016	818	840	846	821	842	840	831	832	833	834	833	810
2017	823	807	816	809	803	799	807	791	789	789	787	801
2018	768	787	780	776	779	782	733	752	805	747	762	770
2019	728	766	759	753	753	755	746	730	729	721	732	736
2020	727	729	729	718	708	709	705	708	657	707	705	697
2021	687	678	676	658	648	654	658	660.81	644	643.53	0	654.24
2022	616	670	602	628	607	640	613	591	593	606	620	610

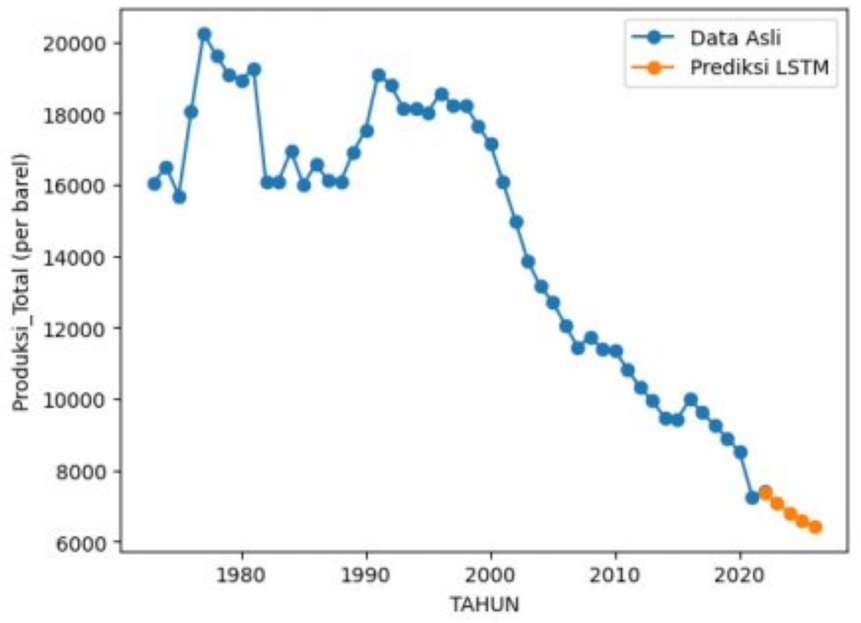
https://id.tradingeconomics.com/indonesia/crude-oil-production

Lampiran 2. Grafik hasil training

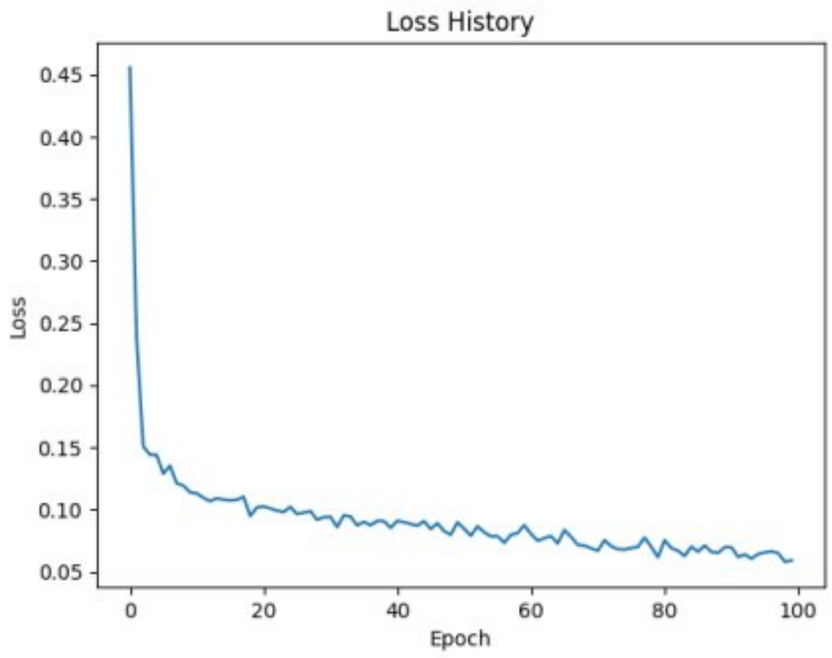
LSTM menggunakan Inisialisasi bobot



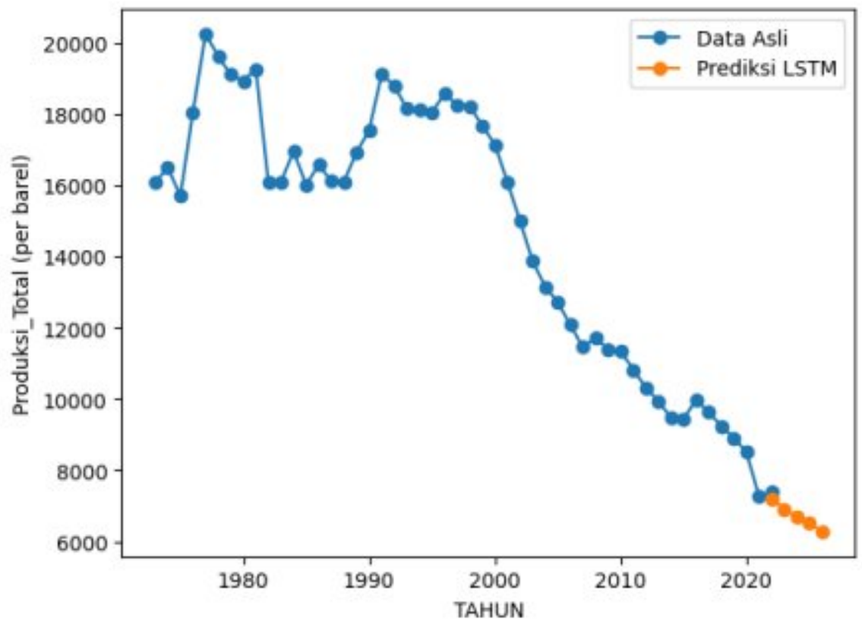
Gambar 1. Perubahan nilai RMSE pada epoch ke-50



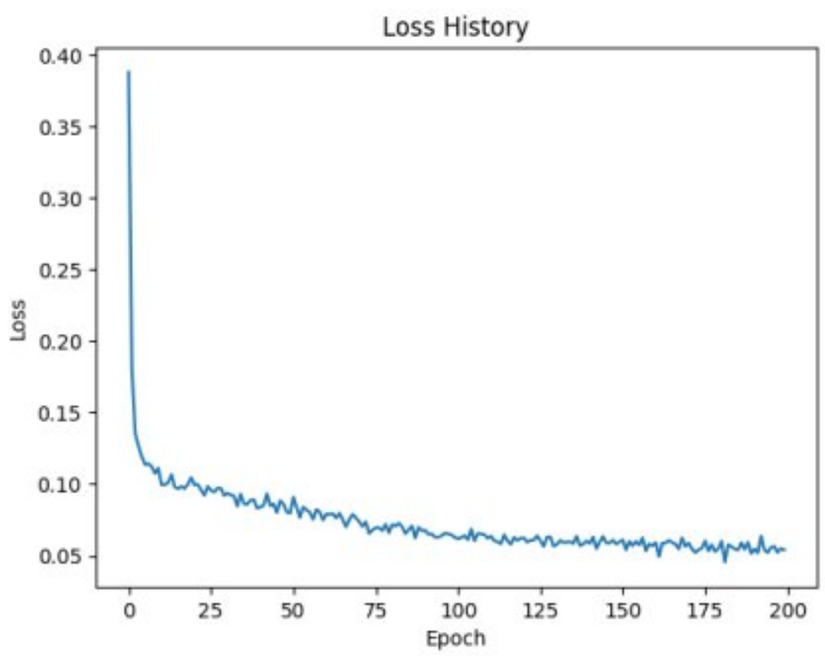
Gambar 2. Grafik Solusi LSTM menggunakan Inisialisasi Bobot pada epoch ke-50

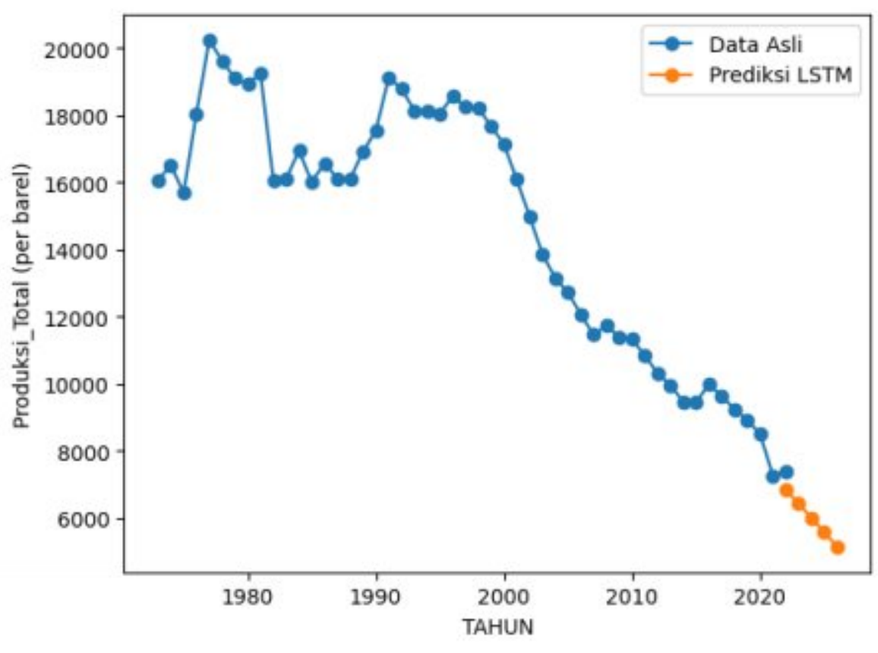


Gambar 3. Perubahan nilai RMSE pada epoch ke-100

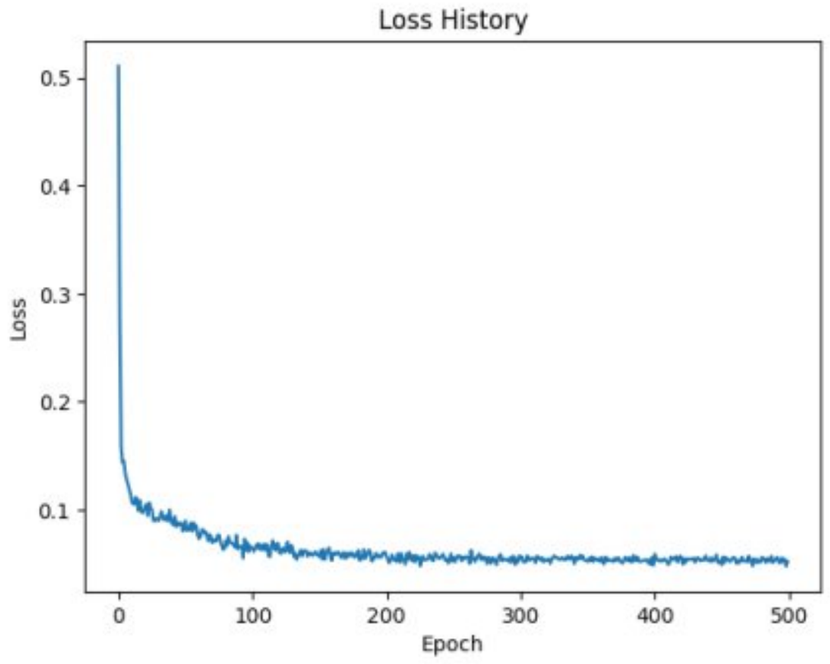


Gambar 4. Grafik Solusi LSTM menggunakan Inisialisasi Bobot pada epoch ke-100

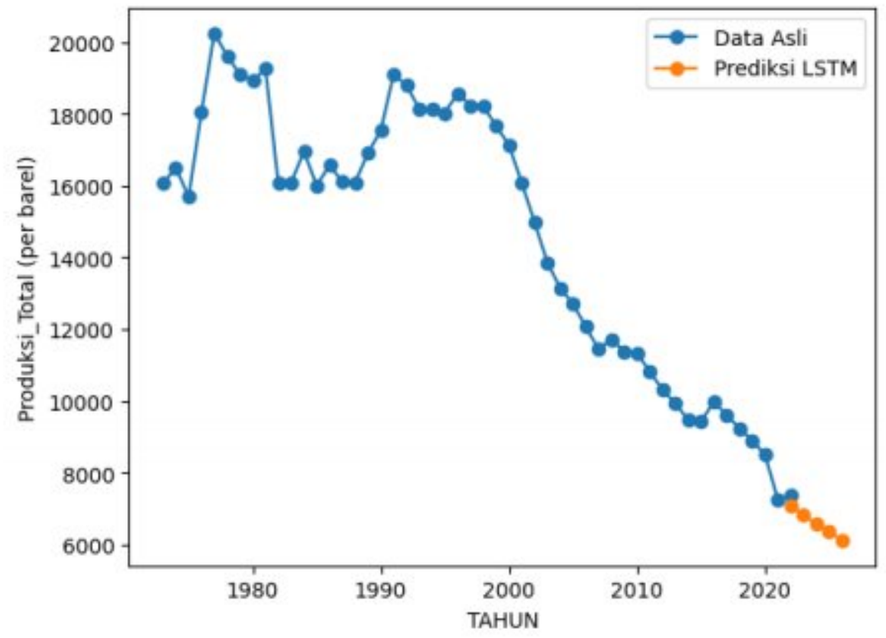




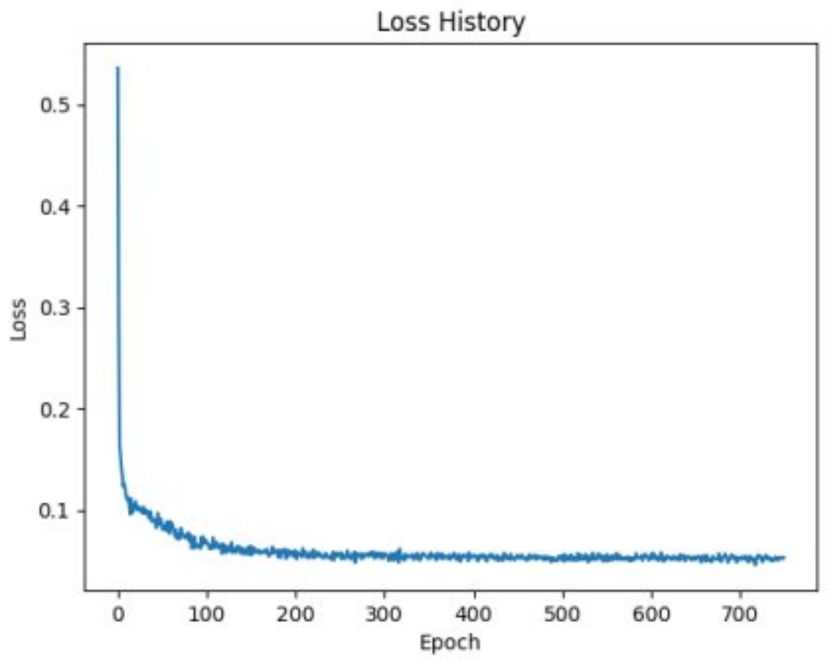
Gambar 5. Perubahan nilai RMSE pada epoch ke-200



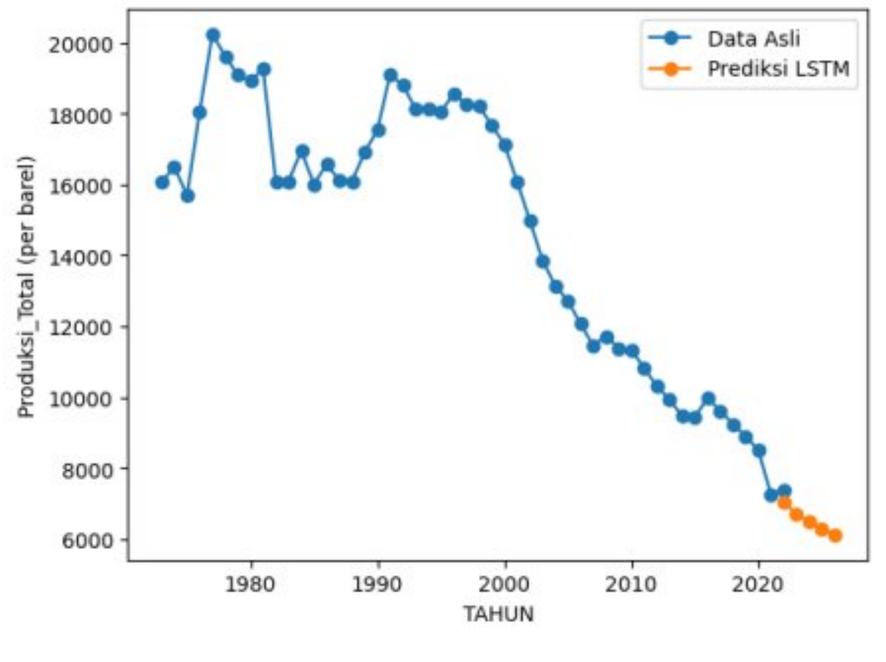
Gambar 6. Grafik Solusi LSTM menggunakan Inisialisasi Bobot pada epoch ke-200



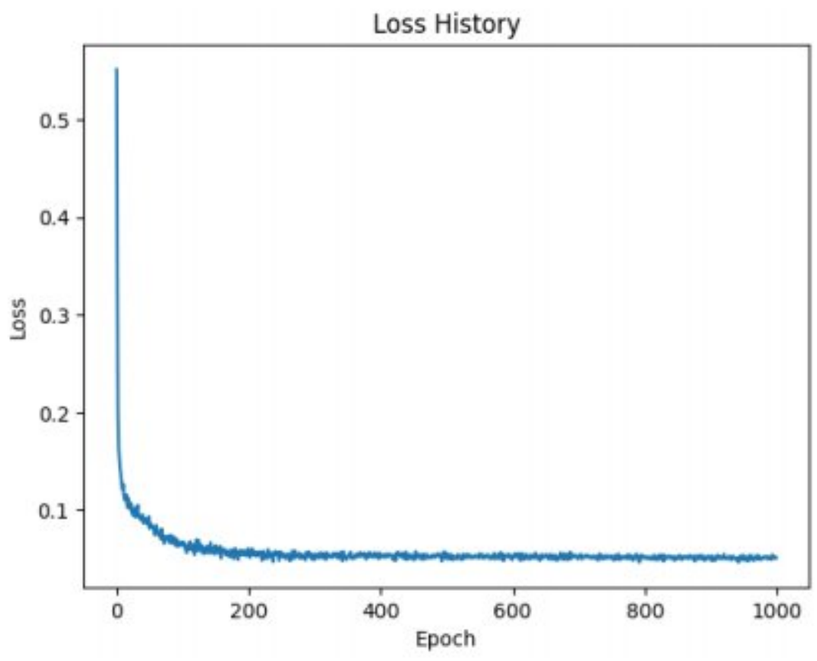
Gambar 7. Perubahan nilai RMSE pada epoch ke-500



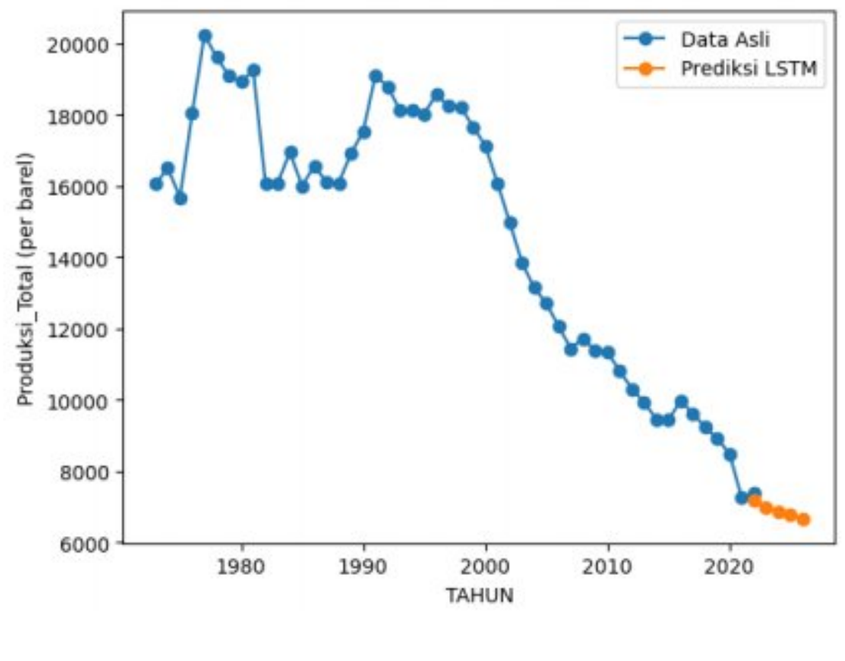
Gambar 8. Grafik Solusi LSTM mengunakan Inisialisasi Bobot pada epoch ke-500



Gambar 9. Perubahan nilai RMSE pada epoch ke-750



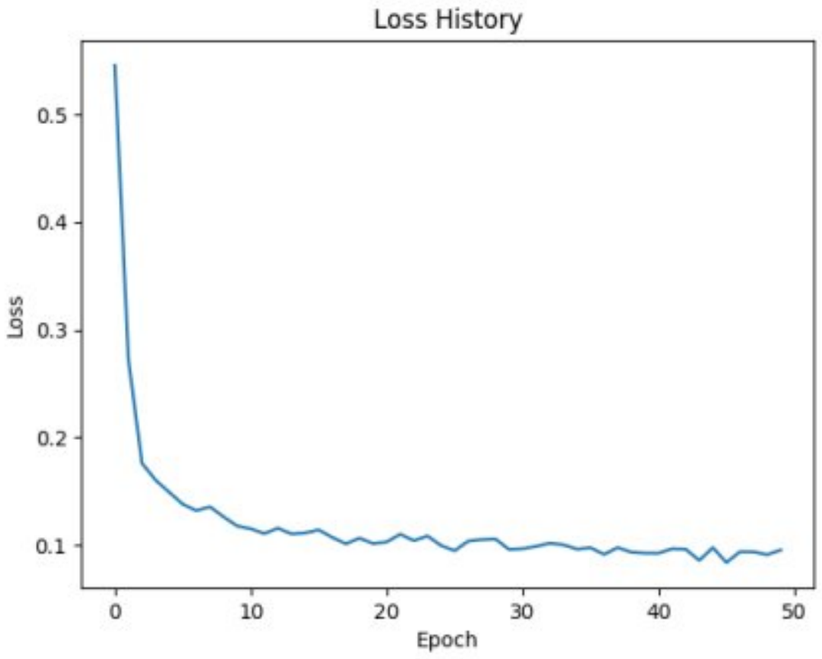
Gambar 10. Grafik Solusi LSTM mengunakan Inisialisasi Bobot pada epoch ke-750



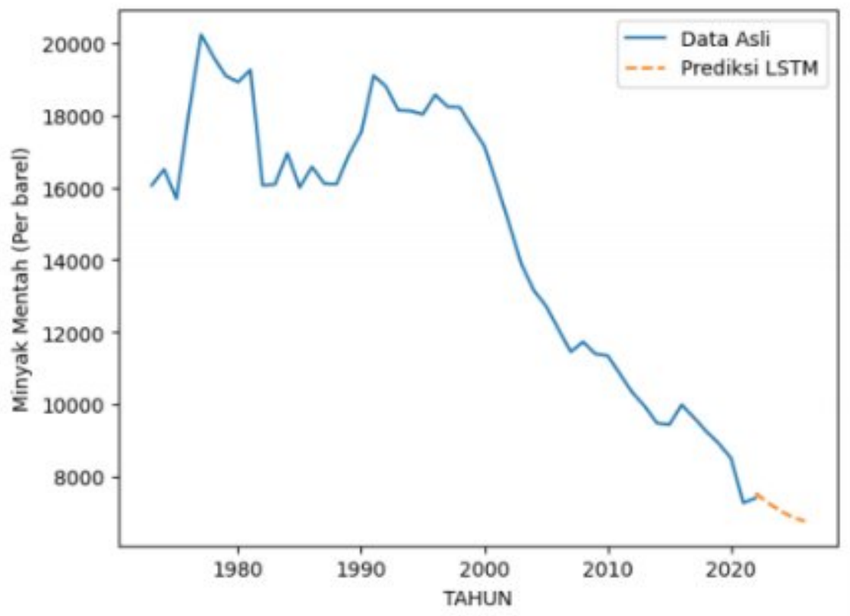
Gambar 11. Perubahan nilai RMSE pada epoch ke-1000

Gambar 12. Grafik Solusi LSTM mengunakan Inisialisasi Bobot pada epoch ke-1000

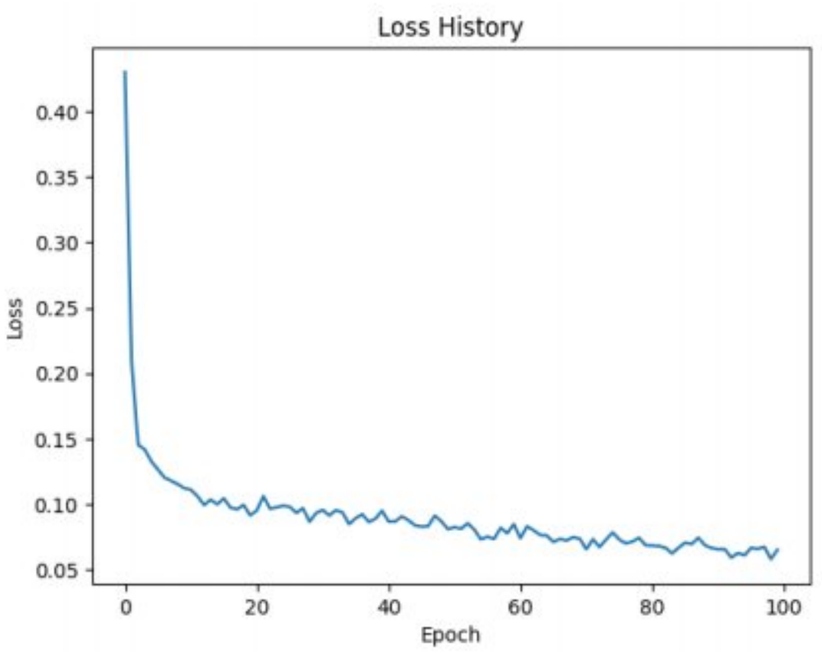
LSTM tanpa mengunakan Inisialisasi bobot



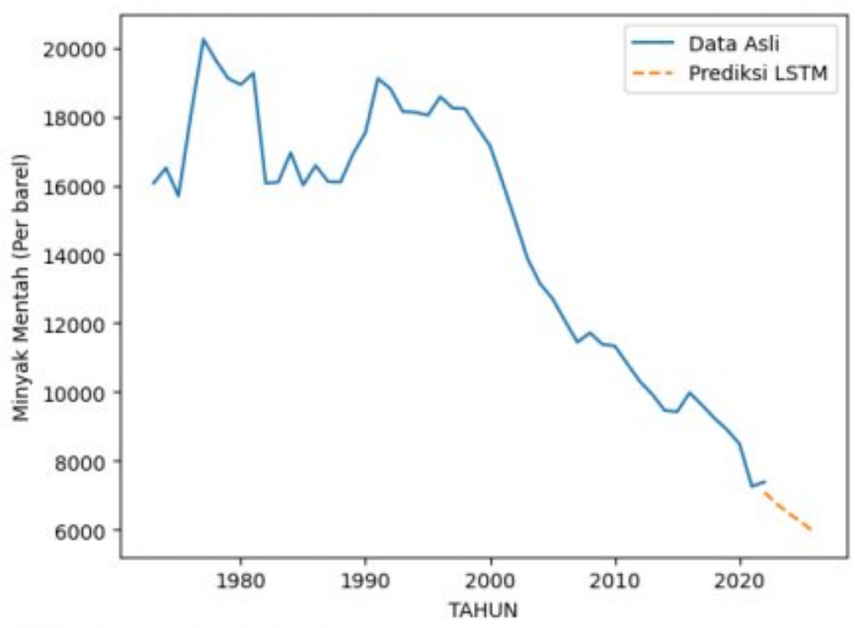
Gambar 1. Perubahan nilai RMSE pada epoch ke-50



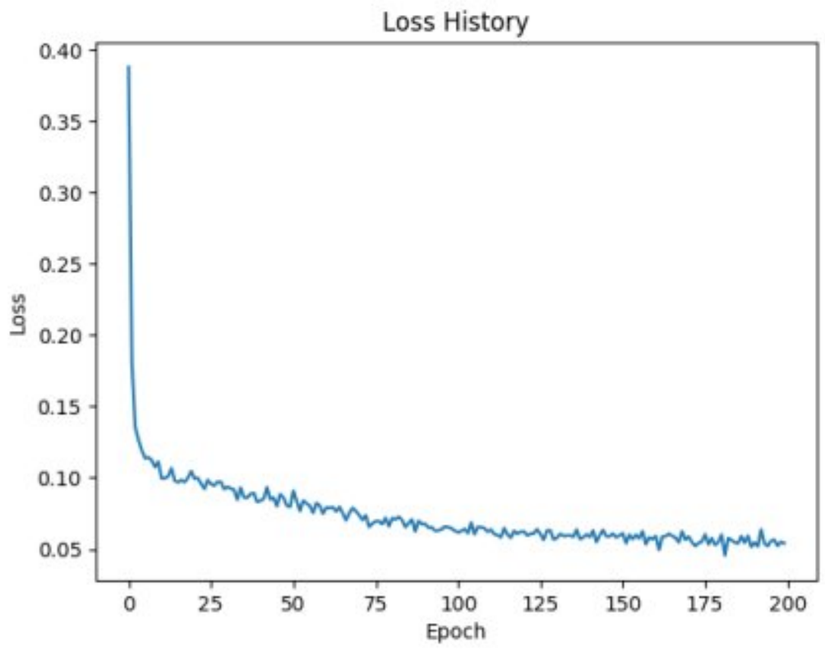
Gambar 2. Grafik Solusi LSTM tanpa mengunakan Inisialisasi Bobot pada epoch ke-50



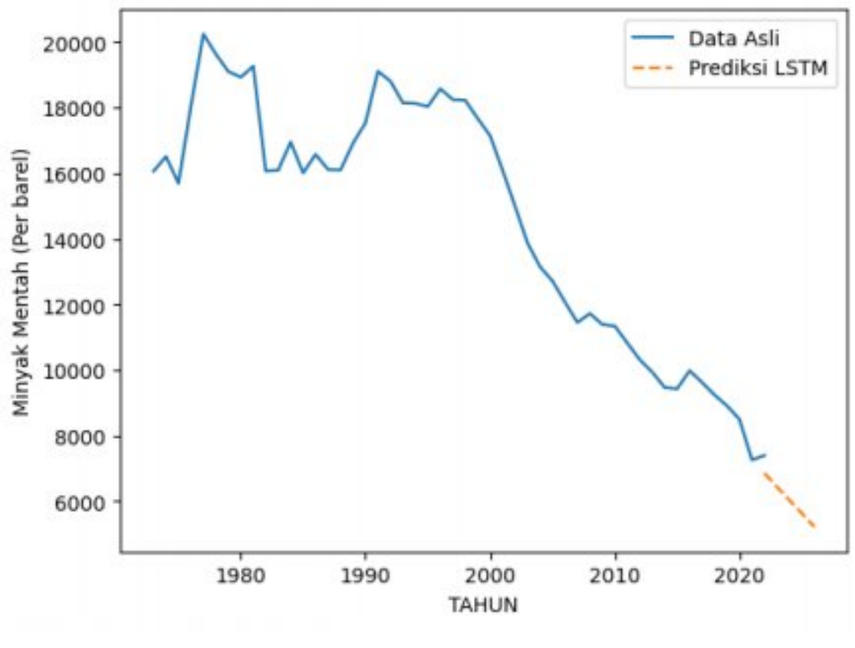
Gambar 3. Perubahan nilai RMSE pada epoch ke-100



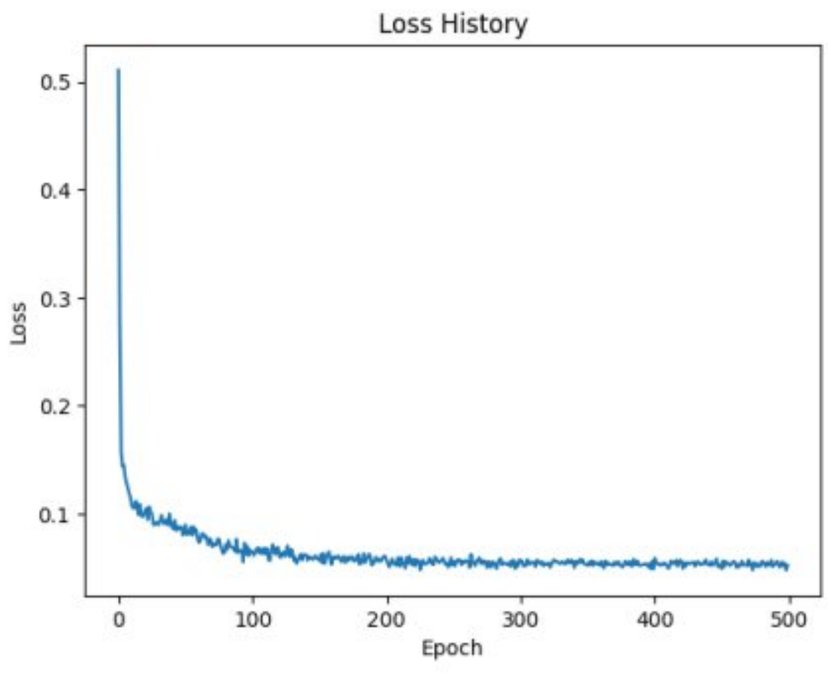
Gambar 4. Grafik Solusi LSTM tanpa mengunakan Inisialisasi Bobot pada epoch ke-100



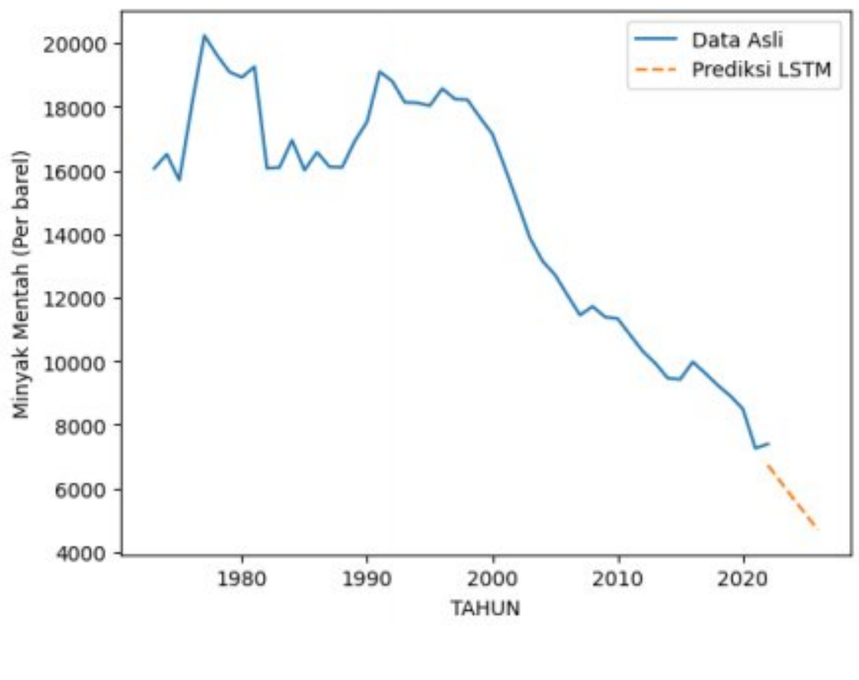
Gambar 5. Perubahan nilai RMSE pada epoch ke-200



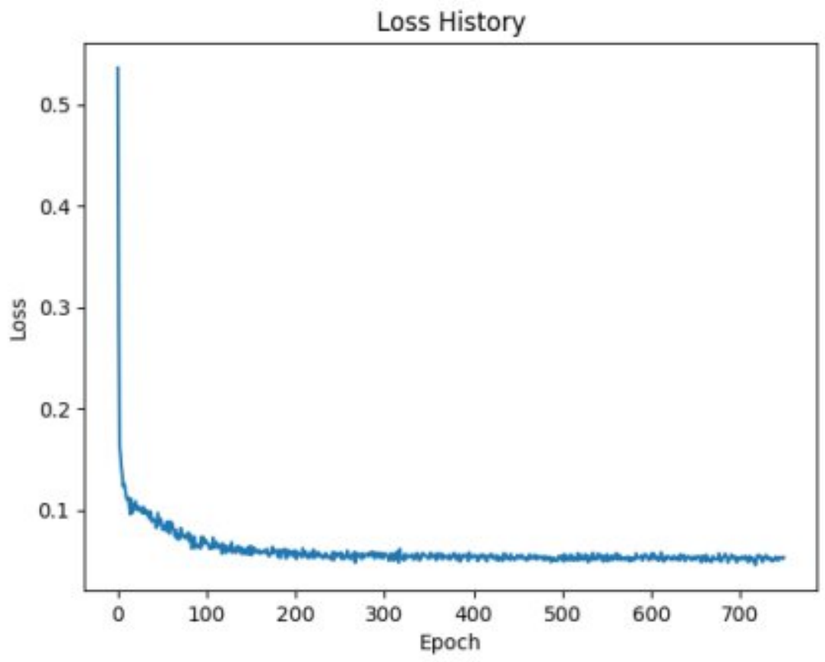
Gambar 6. Grafik Solusi LSTM tanpa mengunakan Inisialisasi Bobot pada epoch ke-200



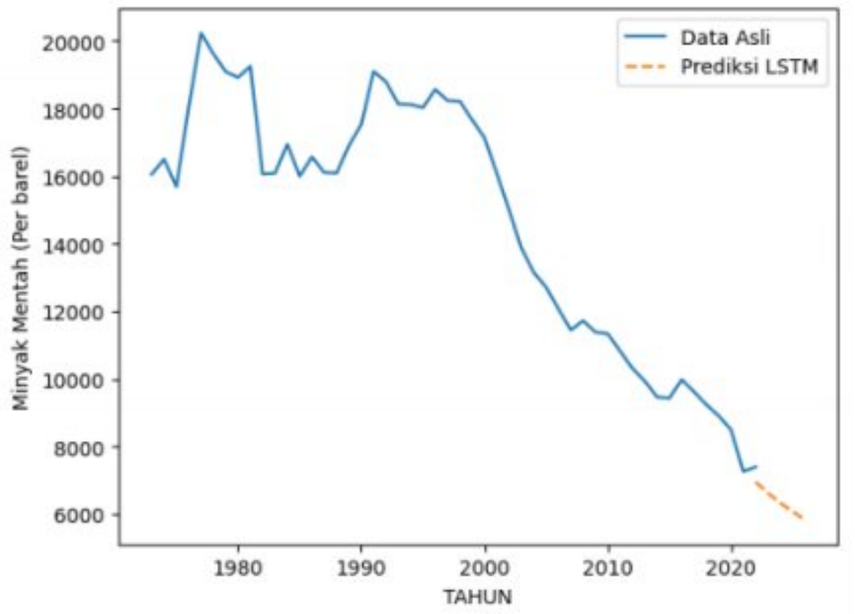
Gambar 7. Perubahan nilai RMSE pada epoch ke-500



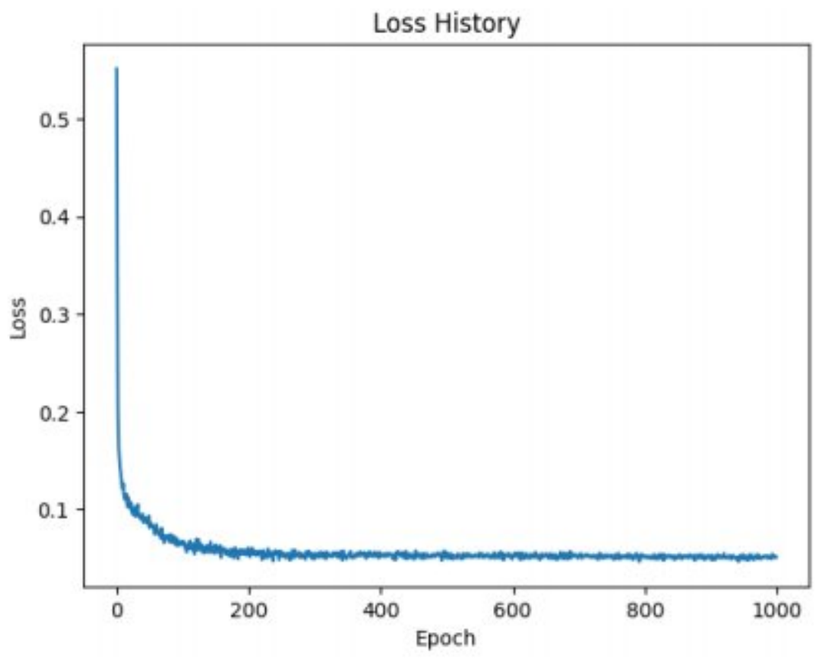
Gambar 8. Grafik Solusi LSTM tanpa mengunakan Inisialisasi Bobot pada epoch ke-500



Gambar 9. Perubahan nilai RMSE pada epoch ke-750

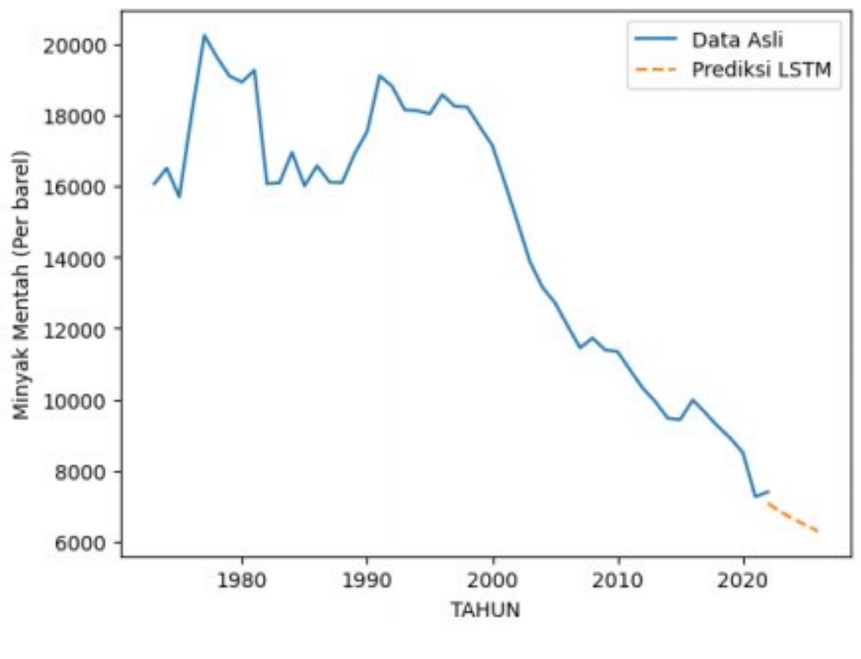


Gambar 10. Grafik Solusi LSTM tanpa mengunakan Inisialisasi Bobot pada



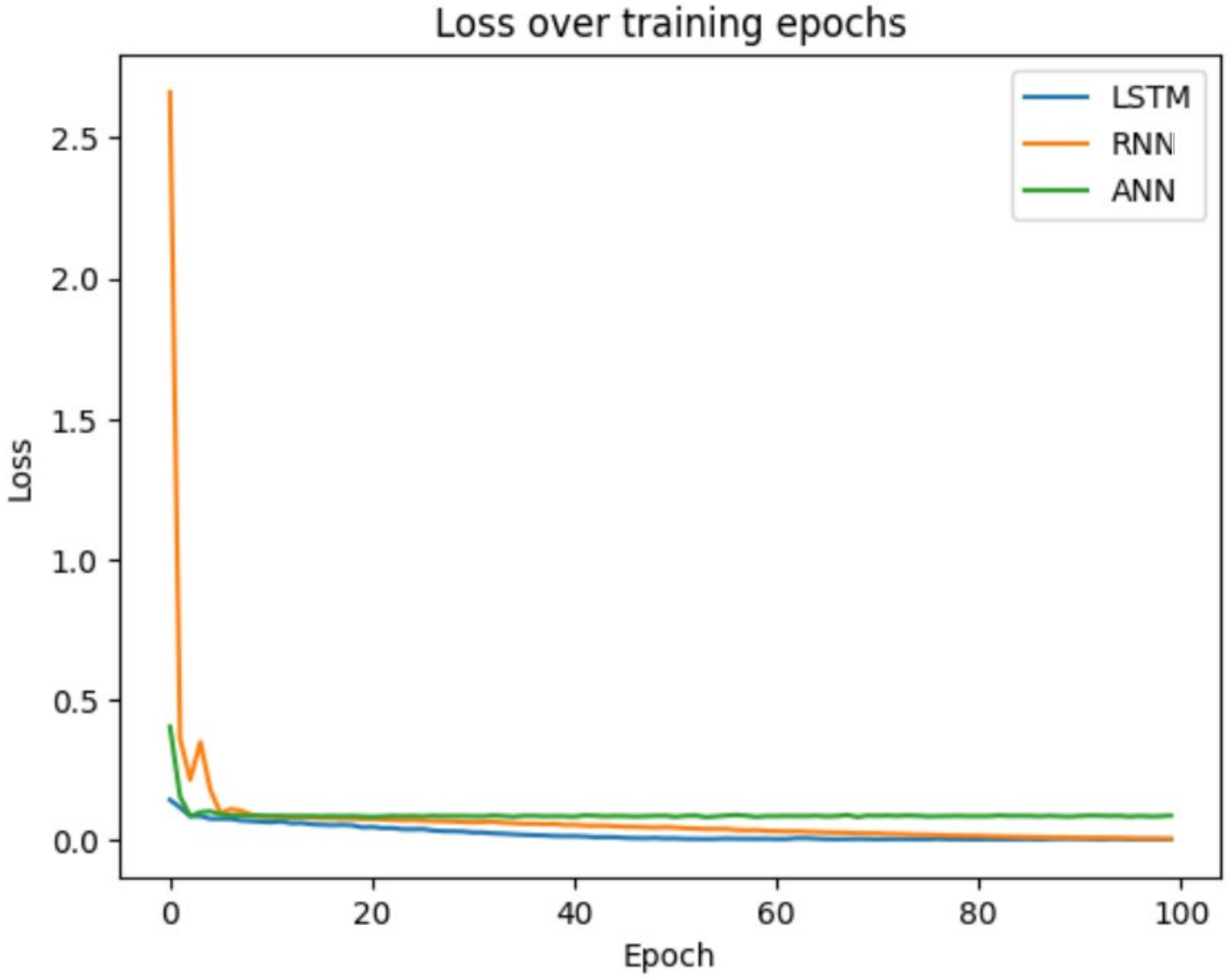
Gambar 11. Perubahan nilai RMSE pada epoch ke-1000

epoch ke-750

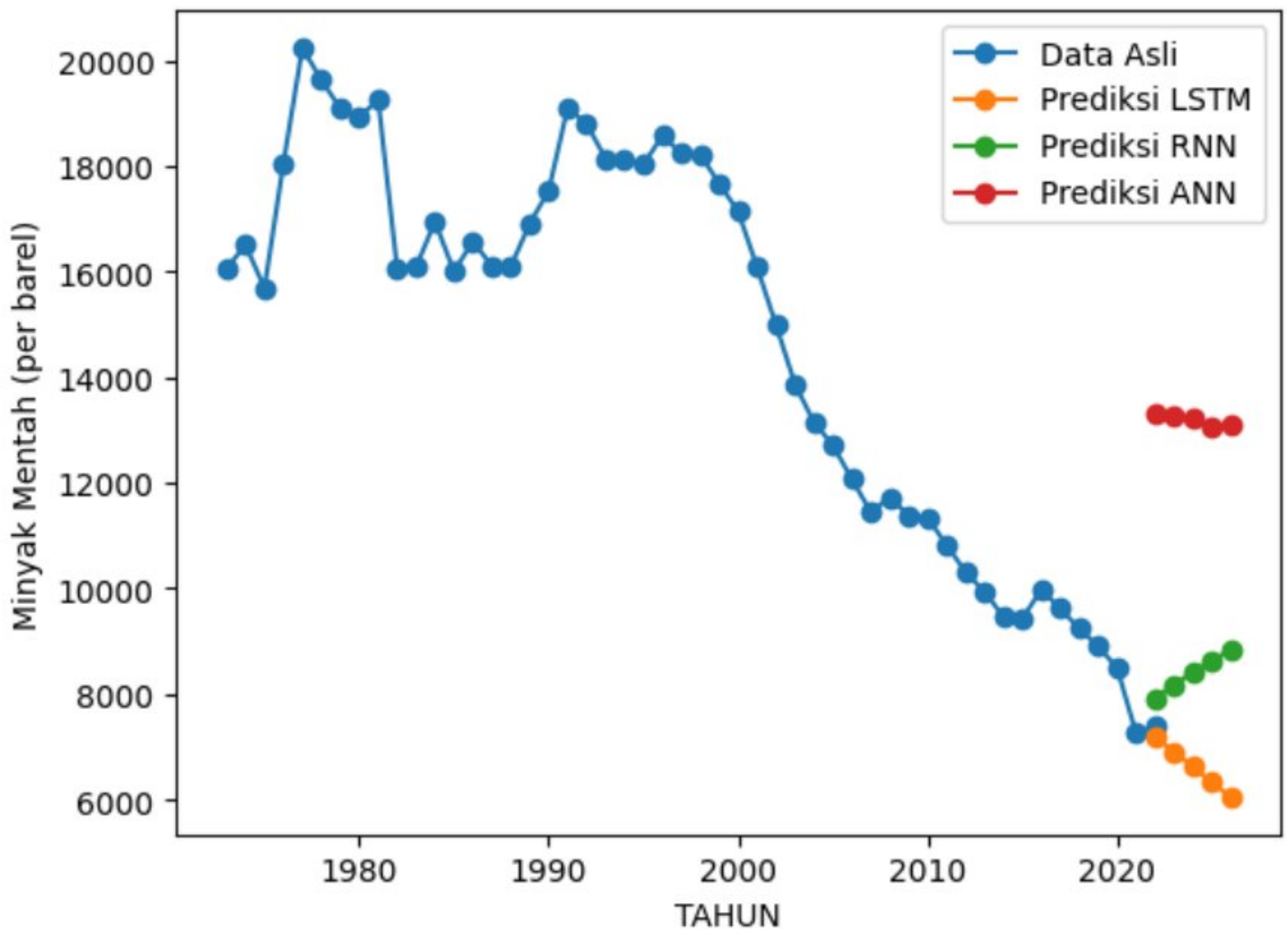


Gambar 12. Grafik Solusi LSTM tanpa mengunakan Inisialisasi Bobot pada epoch ke-1000

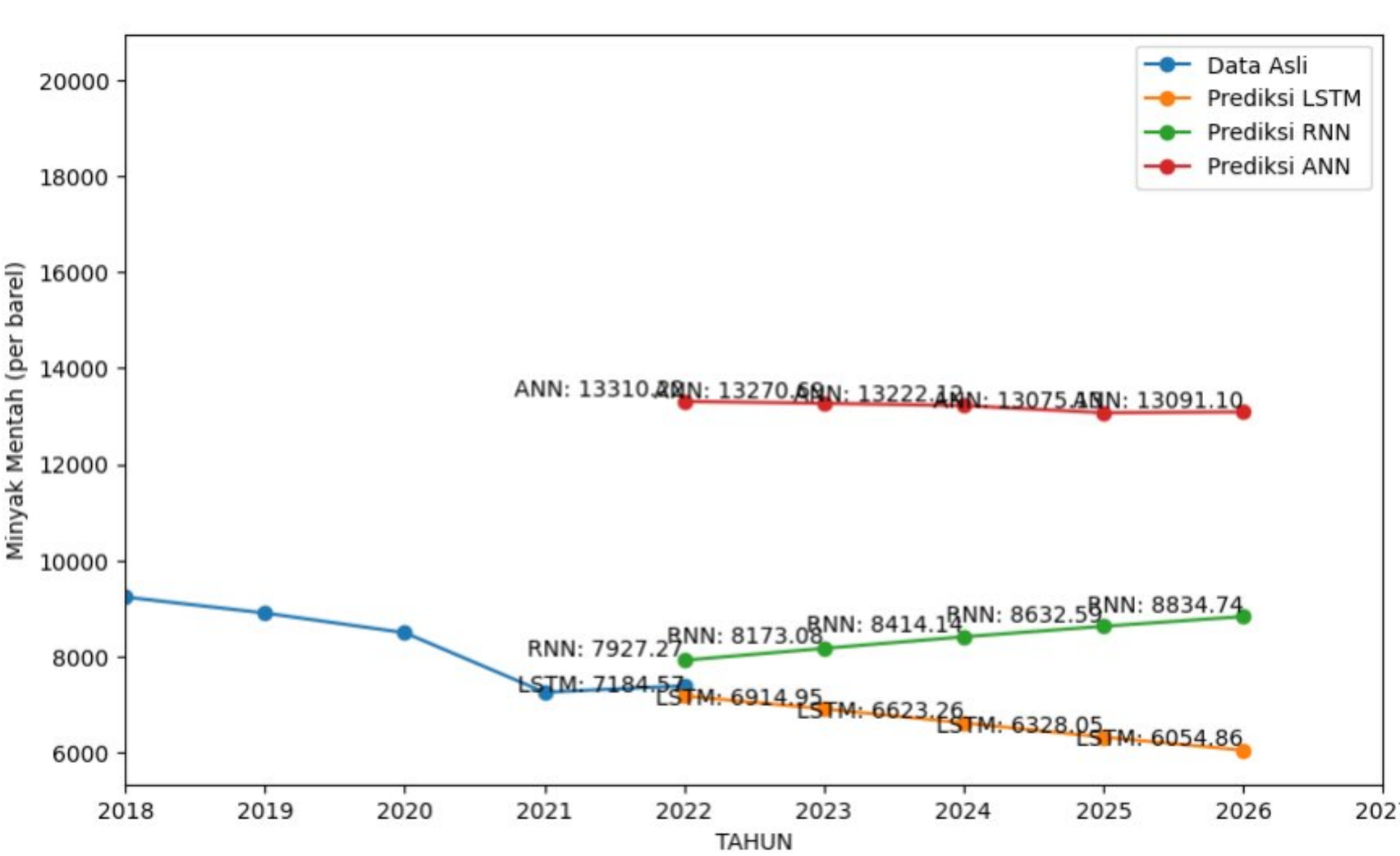
Perbandingan algoritma LSTM, RNN dengan algoritma ANN



Gambar 1. Perubahan nilai RMSE LSTM, RNN dan ANN pada epoch ke-100



Gambar 2. Grafik Solusi LSTM,RNN dan ANN pada epoch ke-100



Gambar 3. Grafik Solusi nilai LSTM,RNN dan ANN pada epoch ke-100

Lampiran 3. Kode Program Python

LSTM tanpa mengunakan Inisialisasi bobot

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, SimpleRNN, Dense
from tensorflow.keras import backend as K
from tensorflow.keras.initializers import VarianceScaling
from tensorflow.keras.callbacks import Callback


# Baca file CSV ke dalam DataFrame
df = pd.read_csv('DATA_PRODUKSI1.csv')

# Tampilkan DataFrame
print(df)

# Membaca data dari file CSV dengan pemisah titik koma
df = pd.read_csv('DATA_PRODUKSI1.csv', sep=';')

# Cetak info dataframe
print(df.info())

# Cetak beberapa baris pertama dataframe
print(df.head())

# Menggabungkan data dari kolom Januari hingga Desember ke dalam satu
kolom baru
bulan_columns = ['Januari', 'Februari', 'Maret', 'April', 'Mei', 'Juni', 'Juli', 'Agustus',
'September ', 'Oktober', 'November', 'Desember']
df['Produksi_Total'] = df[bulan_columns].astype(float).sum(axis=1)


# Membuat grafik garis

plt.plot(df['TAHUN'], df['Produksi_Total'])

# Menambahkan label dan judul
plt.xlabel('TAHUN')
plt.ylabel('Produksi Total (Januari-Desember)')
plt.title('Grafik Produksi Perbulan')

# Menampilkan grafik
plt.show()
# Mengubah tipe data kolom 'Minyak_Mentah' menjadi float
```

```
df['Produksi_Total'] = df['Produksi_Total'].replace(',', '', regex=True).astype(float)

# Menormalisasi data ke rentang 0-1
scaler = MinMaxScaler(feature_range=(0, 1))
df['Produksi_Total_Scaled'] =
scaler.fit_transform(df['Produksi_Total'].values.reshape(-1, 1))

# Fungsi untuk membuat dataset deret waktu
def create_time_series_data(data, time_steps):
    X, y = [], []
    for i in range(len(data) - time_steps):
        X.append(data[i:(i + time_steps)])
        y.append(data[i + time_steps])
    return np.array(X), np.array(y)

# Fungsi untuk membuat model LSTM
def rmse(y_true, y_pred):
    return K.sqrt(K.mean(K.square(y_pred - y_true)))

def create_lstm_model(time_steps, feature_dim):
    initializer = VarianceScaling(scale=0.7, mode="fan_avg",
distribution="uniform")

    model = Sequential()
    model.add(LSTM(units=50, activation='tanh', input_shape=(time_steps,
feature_dim), kernel_initializer=initializer))
    model.add(Dense(units=1))

    model.compile(loss=rmse)
# Menggunakan Root Mean Squared Error sebagai fungsi loss
    return model

# Membuat dataset untuk pelatihan
time_steps = 5  # Jumlah langkah waktu sebelumnya yang digunakan untuk
prediksi
X, y = create_time_series_data(df['Produksi_Total_Scaled'].values, time_steps)

# Reshape input untuk LSTM (jumlah sampel, langkah waktu, fitur)
X_lstm = X.reshape(X.shape[0], X.shape[1], 1)

# Membuat dan melatih model LSTM dengan loss RMSE
model_lstm = create_lstm_model(time_steps, 1)
model_lstm.compile(loss=rmse)  # Menggunakan Root Mean Squared Error
sebagai fungsi loss

# Callback untuk menyimpan nilai loss terkecil dan epochnya

class LossHistory(Callback):
```

```
    def __init__(self):
        super().__init__()
        self.losses = []  # Menyimpan nilai loss
        self.best_loss = float('inf')
# Inisialisasi dengan nilai tak terhingga
        self.best_epoch = 0

    def on_epoch_end(self, epoch, logs=None):
        loss_value = logs['loss']
        self.losses.append(loss_value)
        if loss_value < self.best_loss:
            self.best_loss = loss_value
            self.best_epoch = epoch + 1
# Menggunakan epoch dimulai dari 1
        print(f"Epoch {epoch + 1}/{epochs} - loss: {loss_value:.4f}")

history = LossHistory()

# Melatih model dengan callback untuk menyimpan loss history
epochs = 50
model_lstm.fit(X_lstm, y, epochs=epochs, batch_size=8, verbose=0,
callbacks=[history])

# Menampilkan hasil pelatihan
plt.plot(history.losses)
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Loss History')
plt.show()

# Menampilkan nilai loss terkecil dan epochnya
print(f"Minimum Loss: {history.best_loss:.4f} at Epoch {history.best_epoch}")


# Melakukan prediksi menggunakan model yang telah dilatih (LSTM)
X_test_lstm = df['Produksi_Total_Scaled'].values[-time_steps:].reshape(1,
time_steps, 1)
predicted_values_scaled_lstm = []

for _ in range(5):  # Melakukan 5 langkah prediksi ke depan
    prediction_lstm = model_lstm.predict(X_test_lstm)
    predicted_values_scaled_lstm.append(prediction_lstm[0, 0])
    X_test_lstm = np.roll(X_test_lstm, -1)  # Bergeser ke kiri
    X_test_lstm[0, -1, 0] = prediction_lstm[0, 0]  # Mengganti nilai terakhir dengan
prediksi

# Mengembalikan data prediksi ke skala asli (LSTM)
predicted_values_lstm =
```

```
scaler.inverse_transform(np.array(predicted_values_scaled_lstm).reshape(-1, 1))

# Menampilkan hasil prediksi (LSTM dan RNN)
plt.plot(df['TAHUN'], df['Produksi_Total'], label='Data Asli')
plt.plot(np.arange(df['TAHUN'].max(), df['TAHUN'].max() +
len(predicted_values_lstm)), predicted_values_lstm, label='Prediksi LSTM',
linestyle='dashed')
plt.xlabel('TAHUN')
plt.ylabel('Minyak Mentah (Per barel)')
plt.legend()
plt.show()
```

LSTM tanpa mengunakan Inisialisasi bobot

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, SimpleRNN, Dense
from tensorflow.keras import backend as K
from tensorflow.keras.initializers import VarianceScaling
from tensorflow.keras.callbacks import Callback

# Baca file CSV ke dalam DataFrame
df = pd.read_csv('DATA_PRODUKSI1.csv')

# Tampilkan DataFrame
print(df)

# Membaca data dari file CSV dengan pemisah titik koma
df = pd.read_csv('DATA_PRODUKSI1.csv', sep=';')

# Cetak info dataframe
print(df.info())

# Cetak beberapa baris pertama dataframe
print(df.head())

# Menggabungkan data dari kolom Januari hingga Desember ke dalam satu
kolom baru
bulan_columns = ['Januari', 'Februari', 'Maret', 'April', 'Mei', 'Juni', 'Juli', 'Agustus',
'September ', 'Oktober', 'November', 'Desember']
df['Produksi_Total'] = df[bulan_columns].astype(float).sum(axis=1)
# Membuat grafik garis
plt.plot(df['TAHUN'], df['Produksi_Total'])

# Menambahkan label dan judul
```

```
plt.xlabel('TAHUN')
plt.ylabel('Produksi Total (Januari-Desember)')
plt.title('Grafik Produksi Perbulan')

# Menampilkan grafik
plt.show()

# Mengubah tipe data kolom 'Produksi_Total' menjadi float
df['Produksi_Total'] = df['Produksi_Total'].replace(',', '', regex=True).astype(float)

# Menormalisasi data ke rentang 0-1
scaler = MinMaxScaler(feature_range=(0, 1))
df['Produksi_Total_Scaled'] =
scaler.fit_transform(df['Produksi_Total'].values.reshape(-1, 1))

# Fungsi untuk membuat dataset deret waktu
def create_time_series_data(data, time_steps):
    X, y = [], []
    for i in range(len(data) - time_steps):
        X.append(data[i:(i + time_steps)])
        y.append(data[i + time_steps])
    return np.array(X), np.array(y)
# Fungsi RMSE
def rmse(y_true, y_pred):
    return K.sqrt(K.mean(K.square(y_pred - y_true)))

# Membuat dataset untuk pelatihan
time_steps = 5  # Jumlah langkah waktu sebelumnya yang digunakan untuk
prediksi
X, y = create_time_series_data(df['Produksi_Total_Scaled'].values, time_steps)

# Reshape input untuk LSTM (jumlah sampel, langkah waktu, fitur)
X_lstm = X.reshape(X.shape[0], X.shape[1], 1)

# Membuat model LSTM tanpa inisialisasi bobot eksplisit
model_lstm = Sequential()
model_lstm.add(LSTM(units=50, activation='tanh', input_shape=(time_steps, 1)))
model_lstm.add(Dense(units=1))

# Compile model dengan RMSE sebagai loss
model_lstm.compile(loss=rmse)

# Callback untuk menyimpan nilai loss terkecil dan epochnya

class LossHistory(Callback):
    def __init__(self):
        super().__init__()
        self.losses = []  # Menyimpan nilai loss
```

```
    self.best_loss = float('inf')  # Inisialisasi dengan nilai tak terhingga
    self.best_epoch = 0

def on_epoch_end(self, epoch, logs=None):
    loss_value = logs['loss']
    self.losses.append(loss_value)
    if loss_value < self.best_loss:
        self.best_loss = loss_value
        self.best_epoch = epoch + 1  # Menggunakan epoch dimulai dari 1
    print(f"Epoch {epoch + 1}/{epochs} - loss: {loss_value:.4f}")

history = LossHistory()
# Melatih model dengan callback untuk menyimpan loss history
epochs = 50
model_lstm.fit(X_lstm, y, epochs=epochs, batch_size=8, verbose=0,
callbacks=[history])

# Menampilkan hasil pelatihan
plt.plot(history.losses)
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Loss History')
plt.show()

# Menampilkan nilai loss terkecil dan epochnya
print(f"Minimum Loss: {history.best_loss:.4f} at Epoch {history.best_epoch}")

# Melakukan prediksi menggunakan model yang telah dilatih (LSTM)
X_test_lstm = df['Produksi_Total_Scaled'].values[-time_steps:].reshape(1,
time_steps, 1)
predicted_values_scaled_lstm = []

for _ in range(5):  # Melakukan 5 langkah prediksi ke depan
    prediction_lstm = model_lstm.predict(X_test_lstm)
    predicted_values_scaled_lstm.append(prediction_lstm[0, 0])
    X_test_lstm = np.roll(X_test_lstm, -1)  # Bergeser ke kiri
    X_test_lstm[0, -1, 0] = prediction_lstm[0, 0]  # Mengganti nilai terakhir dengan
prediksi

# Mengembalikan data prediksi ke skala asli (LSTM)
predicted_values_lstm =
scaler.inverse_transform(np.array(predicted_values_scaled_lstm).reshape(-1, 1))

# Menampilkan hasil prediksi (LSTM dan RNN)
plt.plot(df['TAHUN'], df['Produksi_Total'], label='Data Asli')
plt.plot(np.arange(df['TAHUN'].max(), df['TAHUN'].max() +
len(predicted_values_lstm)), predicted_values_lstm, label='Prediksi LSTM',
linestyle='dashed')
```

```
plt.xlabel('TAHUN')
plt.ylabel('Minyak Mentah (Per barel)')
plt.legend()
plt.show()
# Menampilkan hasil prediksi (LSTM,RNN dan ANN) dengan garis berupa titik-titik terhubung
plt.plot(df['TAHUN'], df['Produksi_Total'], 'o-', label='Data Asli')
plt.plot(np.arange(df['TAHUN'].max(), df['TAHUN'].max() + len(predicted_values_lstm)), predicted_values_lstm, 'o-', label='Prediksi LSTM')
plt.xlabel('TAHUN')
plt.ylabel('Produksi_Total (per barel)')
plt.legend()
plt.show()

# Melakukan prediksi menggunakan model yang telah dilatih (LSTM)
X_test_lstm = df['Produksi_Total_Scaled'].values[-time_steps:].reshape(1, time_steps, 1)
predicted_values_scaled_lstm = []

for _ in range(5):  # Melakukan 5 langkah prediksi ke depan
    prediction_lstm = model_lstm.predict(X_test_lstm)
    predicted_values_scaled_lstm.append(prediction_lstm[0, 0])
    X_test_lstm = np.roll(X_test_lstm, -1)  # Bergeser ke kiri
    X_test_lstm[0, -1, 0] = prediction_lstm[0, 0]  # Mengganti nilai terakhir dengan prediksi

# Mengembalikan data prediksi ke skala asli (LSTM)
predicted_values_lstm = scaler.inverse_transform(np.array(predicted_values_scaled_lstm).reshape(-1, 1))
# Misalnya, membuat gambar baru dengan ukuran lebar x tinggi = 10 x 6 inci
plt.figure(figsize=(10, 6))

# Menampilkan hasil prediksi (LSTM, RNN, dan ANN) dengan garis berupa titik-titik terhubung
tahun_max = df['TAHUN'].max()
tahun_prediksi = np.arange(tahun_max, tahun_max + len(predicted_values_lstm))

plt.plot(df['TAHUN'], df['Produksi_Total'], 'o-', label='Data Asli')
plt.plot(tahun_prediksi, predicted_values_lstm, 'o-', label='Prediksi LSTM')
# Tambahkan plot untuk prediksi RNN dan ANN sesuai kebutuhan

plt.xlabel('TAHUN')
plt.ylabel('Minyak Mentah (per barel)')
plt.legend()

# Menampilkan nilai data prediksi di grafik
for i, txt in enumerate(predicted_values_lstm):
```

```
    plt.text(tahun_prediksi[i], float(txt), f'{float(txt):.2f}', ha='right', va='bottom')

# Menampilkan grafik dengan ukuran yang diperbesar
plt.show()
```

Perbandingan algoritma LSTM, RNN dengan algoritma ANN

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, SimpleRNN, Dense
from math import sqrt
from prettytable import PrettyTable

# Baca file CSV ke dalam DataFrame
df = pd.read_csv('DATA_PRODUKSI1.csv')

# Tampilkan DataFrame
print(df)
# Membaca data dari file CSV dengan pemisah titik koma
df = pd.read_csv('DATA_PRODUKSI1.csv', sep=';')

# Cetak info dataframe
print(df.info())

# Cetak beberapa baris pertama dataframe
print(df.head())

# Menggabungkan data dari kolom Januari hingga Desember ke dalam satu
kolom baru
bulan_columns = ['Januari', 'Februari', 'Maret', 'April', 'Mei', 'Juni', 'Juli', 'Agustus',
'September ', 'Oktober', 'November', 'Desember']
df['Produksi_Total'] = df[bulan_columns].astype(float).sum(axis=1)


# Membuat grafik garis
plt.plot(df['TAHUN'], df['Produksi_Total'])

# Menambahkan label dan judul
plt.xlabel('TAHUN')
plt.ylabel('Produksi Total (Januari-Desember)')
plt.title('Grafik Produksi Perbulan')
```

```
# Menampilkan grafik
plt.show()
# Mengubah tipe data kolom 'Minyak_Mentah' menjadi float
df['Produksi_Total'] = df['Produksi_Total'].replace(',', '', regex=True).astype(float)

# Menormalisasi data ke rentang 0-1
scaler = MinMaxScaler(feature_range=(0, 1))
df['Produksi_Total_Scaled'] =
scaler.fit_transform(df['Produksi_Total'].values.reshape(-1, 1))
# Fungsi untuk membuat dataset deret waktu
def create_time_series_data(data, time_steps):
    X, y = [], []
    for i in range(len(data) - time_steps):
        X.append(data[i:(i + time_steps)])
        y.append(data[i + time_steps])
    return np.array(X), np.array(y)

# Fungsi untuk membuat model LSTM
def create_lstm_model(time_steps, feature_dim):
    model = Sequential()
    model.add(LSTM(units=100, activation='sigmoid', input_shape=(time_steps,
feature_dim)))
    model.add(Dense(units=1))
    model.compile(optimizer='adam', loss='mse')  # Menggunakan RMSE sebagai
fungsi loss
    return model

# Fungsi untuk membuat model RNN
def create_rnn_model(time_steps, feature_dim):
    model = Sequential()
    model.add(SimpleRNN(units=100, activation='sigmoid',
input_shape=(time_steps, feature_dim)))
    model.add(Dense(units=1))
    model.compile(optimizer='adam', loss='mse')  # Menggunakan Mean Squared
Error sebagai fungsi loss
    return model

# Fungsi untuk membuat model ANN
def create_ann_model(time_steps, feature_dim):
    model = Sequential()
    model.add(Dense(units=100, activation='sigmoid', input_shape=(time_steps,
feature_dim)))
    model.add(Dense(units=1))
    model.compile(optimizer='adam', loss='mse')  # Menggunakan Mean Squared
Error sebagai fungsi loss
    return model
# Membuat dataset untuk pelatihan
time_steps = 5  # Jumlah langkah waktu sebelumnya yang digunakan untuk
```

```
prediksi
X, y = create_time_series_data(df['Produksi_Total_Scaled'].values, time_steps)

# Reshape input untuk LSTM (jumlah sampel, langkah waktu, fitur)
X_lstm = X.reshape(X.shape[0], X.shape[1], 1)
X_rnn = X.reshape(X.shape[0], X.shape[1], 1)
X_ann = X.reshape(X.shape[0], X.shape[1], 1)
# Membuat dan melatih model LSTM
model_lstm = create_lstm_model(time_steps, 1)
loss_history_lstm = []

for epoch in range(100):
    history_lstm = model_lstm.fit(X_lstm, y, epochs=1, batch_size=8, verbose=0)
 # Set verbose=1 to see training progress
    loss_lstm = history_lstm.history['loss'][0]
    loss_history_lstm.append(loss_lstm)
    print(f'LSTM Epoch {epoch + 1}/{100}, Loss: {loss_lstm}')
# Membuat dan melatih model RNN
model_rnn = create_rnn_model(time_steps, 1)
loss_history_rnn = []

for epoch in range(100):
    history_rnn = model_rnn.fit(X_rnn, y, epochs=1, batch_size=8, verbose=0)  #
Set verbose=1 to see training progress
    loss_rnn = history_rnn.history['loss'][0]
    loss_history_rnn.append(loss_rnn)
    print(f'RNN Epoch {epoch + 1}/{100}, Loss: {loss_rnn}')
# Membuat dan melatih model RNN
model_ann = create_ann_model(time_steps, 1)
loss_history_ann = []

for epoch in range(100):
    history_ann = model_ann.fit(X_ann, y, epochs=1, batch_size=8, verbose=0)  #
Set verbose=1 to see training progress
    loss_ann = history_ann.history['loss'][0]
    loss_history_ann.append(loss_ann)
    print(f'ANN Epoch {epoch + 1}/{100}, Loss: {loss_ann}')
# Plot nilai loss LSTM dan RNN dalam satu grafik
plt.plot(loss_history_lstm, label='LSTM')
plt.plot(loss_history_rnn, label='RNN')
plt.plot(loss_history_ann, label='ANN')
plt.title('Loss over training epochs')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
# Melakukan prediksi menggunakan model yang telah dilatih (LSTM)
X_test_lstm = df['Produksi_Total_Scaled'].values[-time_steps:].reshape(1,
```

```
time_steps, 1)
predicted_values_scaled_lstm = []

for _ in range(5):  # Melakukan 5 langkah prediksi ke depan
    prediction_lstm = model_lstm.predict(X_test_lstm)
    predicted_values_scaled_lstm.append(prediction_lstm[0, 0])
    X_test_lstm = np.roll(X_test_lstm, -1)  # Bergeser ke kiri
    X_test_lstm[0, -1, 0] = prediction_lstm[0, 0]  # Mengganti nilai terakhir dengan
prediksi

# Mengembalikan data prediksi ke skala asli (LSTM)
predicted_values_lstm =
scaler.inverse_transform(np.array(predicted_values_scaled_lstm).reshape(-1, 1))

# Melakukan prediksi menggunakan model yang telah dilatih (RNN)
X_test_rnn = df['Produksi_Total_Scaled'].values[-time_steps:].reshape(1,
time_steps, 1)
predicted_values_scaled_rnn = []

for _ in range(5):  # Melakukan 5 langkah prediksi ke depan
    prediction_rnn = model_rnn.predict(X_test_rnn)
    predicted_values_scaled_rnn.append(prediction_rnn[0, 0])
    X_test_rnn = np.roll(X_test_rnn, -1)  # Bergeser ke kiri
    X_test_rnn[0, -1, 0] = prediction_rnn[0, 0]  # Mengganti nilai terakhir dengan
prediksi

# Mengembalikan data prediksi ke skala asli (RNN)
predicted_values_rnn =
scaler.inverse_transform(np.array(predicted_values_scaled_rnn).reshape(-1, 1))
# Melakukan prediksi menggunakan model yang telah dilatih (ANN)
X_test_ann = df['Produksi_Total_Scaled'].values[-time_steps:].reshape(1,
time_steps, 1)
predicted_values_scaled_ann = []

for _ in range(5):  # Melakukan 5 langkah prediksi ke depan
    prediction_ann = model_ann.predict(X_test_ann)
    predicted_values_scaled_ann.append(prediction_ann[0, 0])
    X_test_ann = np.roll(X_test_ann, -1)  # Bergeser ke kiri
    X_test_ann[0, -1, 0] = prediction_ann[0, 0]  # Mengganti nilai terakhir dengan
prediksi

# Mengembalikan data prediksi ke skala asli (RNN)
predicted_values_ann =
scaler.inverse_transform(np.array(predicted_values_scaled_ann).reshape(-1, 1))
# Menampilkan hasil prediksi (LSTM,RNN dan ANN) dengan garis berupa titik-
titik terhubung
plt.plot(df['TAHUN'], df['Produksi_Total'], 'o-', label='Data Asli')
plt.plot(np.arange(df['TAHUN'].max(), df['TAHUN'].max() +
```

```
len(predicted_values_lstm)), predicted_values_lstm, 'o-', label='Prediksi LSTM')
plt.plot(np.arange(df['TAHUN'].max(), df['TAHUN'].max() +
len(predicted_values_rnn)), predicted_values_rnn, 'o-', label='Prediksi RNN')
plt.plot(np.arange(df['TAHUN'].max(), df['TAHUN'].max() +
len(predicted_values_ann)), predicted_values_ann, 'o-', label='Prediksi ANN')
plt.xlabel('TAHUN')
plt.ylabel('Minyak Mentah (per barel)')
plt.legend()
plt.show()
# Misalnya, membuat gambar baru dengan ukuran lebar x tinggi = 10 x 6 inci
plt.figure(figsize=(10, 6))

# Menampilkan hasil prediksi (LSTM, RNN, dan ANN) dengan garis berupa titik-
titik terhubung
tahun_max = df['TAHUN'].max()
tahun_prediksi = np.arange(tahun_max, tahun_max +
len(predicted_values_lstm))

plt.plot(df['TAHUN'], df['Produksi_Total'], 'o-', label='Data Asli')
plt.plot(tahun_prediksi, predicted_values_lstm, 'o-', label='Prediksi LSTM')
plt.plot(tahun_prediksi, predicted_values_rnn, 'o-', label='Prediksi RNN')
plt.plot(tahun_prediksi, predicted_values_ann, 'o-', label='Prediksi ANN')

plt.xlabel('TAHUN')
plt.ylabel('Minyak Mentah (per barel)')
plt.legend()

# Menampilkan nilai data prediksi di grafik (LSTM)
for i, txt in enumerate(predicted_values_lstm):
    plt.text(tahun_prediksi[i], float(txt), f'LSTM: {float(txt):.2f}', ha='right',
va='bottom')

# Menampilkan nilai data prediksi di grafik (RNN)
for i, txt in enumerate(predicted_values_rnn):
    plt.text(tahun_prediksi[i], float(txt), f'RNN: {float(txt):.2f}', ha='right',
va='bottom')

# Menampilkan nilai data prediksi di grafik (ANN)
for i, txt in enumerate(predicted_values_ann):
    plt.text(tahun_prediksi[i], float(txt), f'ANN: {float(txt):.2f}', ha='right',
va='bottom')
# Memperbesar ke sebelah kanan (misalnya, dari tahun 2020 hingga 2030)
plt.xlim(2018, 2027)
# Menampilkan grafik dengan ukuran yang diperbesar
plt.show()

# Simpan hasil prediksi dalam DataFrame
result_df = pd.DataFrame(list(zip(tahun_prediksi, predicted_values_lstm,
```

```
predicted_values_rnn, predicted_values_ann)),
                    columns=['TAHUN', 'Prediksi LSTM', 'Prediksi RNN', 'Prediksi
ANN'])

# Tampilkan DataFrame sebagai tabel dengan prettytable
result_table = PrettyTable()
result_table.field_names = result_df.columns

for row in result_df.itertuples(index=False):
    result_table.add_row(row)

print(result_table)
# Mengembalikan data prediksi ke skala asli (LSTM)
predicted_values_lstm =
scaler.inverse_transform(np.array(predicted_values_scaled_lstm).reshape(-1, 1))
# Mengembalikan data prediksi ke skala asli (RNN)
predicted_values_rnn =
scaler.inverse_transform(np.array(predicted_values_scaled_rnn).reshape(-1, 1))
# Mengembalikan data prediksi ke skala asli (ANN)
predicted_values_ann =
scaler.inverse_transform(np.array(predicted_values_scaled_ann).reshape(-1, 1))

# Melakukan prediksi menggunakan model yang telah dilatih (LSTM)
X_test_lstm = df['Produksi_Total_Scaled'].values[-time_steps:].reshape(1,
time_steps, 1)
predicted_values_scaled_lstm = []

for _ in range(5):  # Melakukan 5 langkah prediksi ke depan
    prediction_lstm = model_lstm.predict(X_test_lstm)
    predicted_values_scaled_lstm.append(prediction_lstm[0, 0])
    X_test_lstm = np.roll(X_test_lstm, -1)  # Bergeser ke kiri
    X_test_lstm[0, -1, 0] = prediction_lstm[0, 0]  # Mengganti nilai terakhir dengan
prediksi


# Menghitung kesalahan (error) untuk LSTM, RNN dan ANN
error_lstm = df['Produksi_Total'].iloc[-5:] - predicted_values_lstm.flatten()
error_rnn = df['Produksi_Total'].iloc[-5:] - predicted_values_rnn.flatten()
error_ann = df['Produksi_Total'].iloc[-5:] - predicted_values_ann.flatten()


# Plot distribusi kesalahan untuk LSTM
plt.figure(figsize=(20, 5))
plt.subplot(1, 3, 1)
sns.histplot(error_lstm, kde=True)
plt.title('Error Distribution - LSTM')

# Plot distribusi kesalahan untuk RNN
```

```
plt.subplot(1, 3, 2)
sns.histplot(error_rnn, kde=True)
plt.title('Error Distribution - RNN')

# Plot distribusi kesalahan untuk ANN
plt.subplot(1, 3, 3)
sns.histplot(error_ann, kde=True)
plt.title('Error Distribution - ANN')

plt.show()
```