



Hands-On

Hands-On ini digunakan pada kegiatan Microcredential Associate Data Scientist 2021

Tugas Mandiri Pertemuan 14

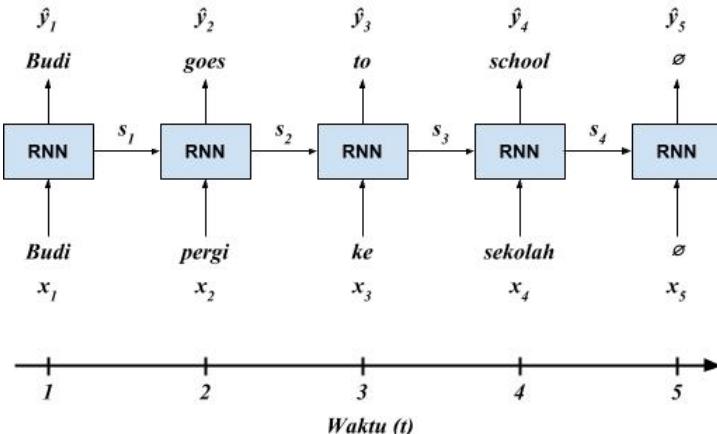
Pertemuan 14 (empatbelas) pada Microcredential Associate Data Scientist 2021 menyampaikan materi mengenai Membangun Model (RNN dan LSTM). silakan Anda kerjakan Latihan 1 s/d 5. Output yang anda lihat merupakan panduan yang dapat Anda ikuti dalam penulisan code :)

RNN

Jaringan saraf berulang atau recurrent neural network (RNN) adalah jenis arsitektur jaringan saraf tiruan yang pemrosesannya dipanggil berulang-ulang untuk memproses masukan yang biasanya adalah data sekuensial. RNN masuk dalam kategori deep learning karena data diproses melalui banyak lapis (layer). RNN telah mengalami kemajuan yang pesat dan telah merevolusi bidang-bidang seperti pemrosesan bahasa alami (NLP), pengenalan suara, sintesa musik, pemrosesan data finansial seri waktu, analisa deret DNA, analisa video, dan sebagainya.

RNN memproses input secara sekuensial, sampel per sampel. Dalam tiap pemrosesan, output yang dihasilkan tidak hanya merupakan fungsi dari sampel itu saja, tapi juga berdasarkan state internal yang merupakan hasil dari pemrosesan sampel-sampel sebelumnya (atau setelahnya, pada bidirectional RNN).

Berikut adalah ilustrasi bagaimana RNN bekerja. Misalkan kita membuat RNN untuk menerjemahkan bahasa Indonesia ke bahasa Inggris



Ilustrasi di atas kelihatan rumit, tapi sebenarnya cukup mudah dipahami.

- sumbu horizontal adalah waktu, direpresentasikan dengan simbol t . Dapat kita bayangkan pemrosesan berjalan dari kiri ke kanan. Selanjutnya kita sebut t adalah langkah waktu (time step).
- Keseluruhan input adalah kalimat, dalam hal ini:

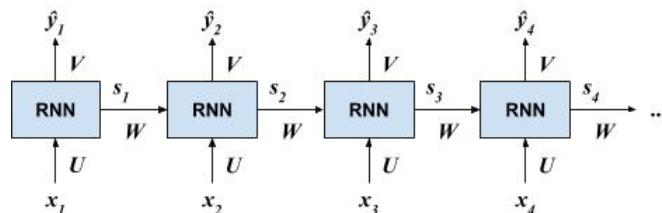
Budi pergi ke sekolah.

- Pemrosesan input oleh RNN adalah kata demi kata. Input kata-kata ini disimbolkan dengan x_1, x_2, \dots, x_5 , atau secara umum x_t .
- Output adalah kalimat, dalam hal ini:

Budi goes to school.

- RNN memberikan output kata demi kata, dan ini kita simbolkan dengan $\hat{y}_1, \hat{y}_2, \dots, \hat{y}_5$, atau secara umum \hat{y}_t .
- Dalam tiap pemrosesan, RNN akan menyimpan state internal yaitu s_t , yang diberikan dari satu langkah waktu ke langkah waktu berikutnya. Inilah "memori" dari RNN.

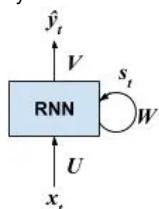
Dengan contoh di atas, kita bisa generalisasikan arsitektur RNN sebagai berikut:



Tambahan yang tidak terdapat di diagram sebelumnya adalah U , V , dan W , yang merupakan parameter-parameter yang dimiliki RNN. Kita akan bahas pemakaian parameter-parameter ini nanti.

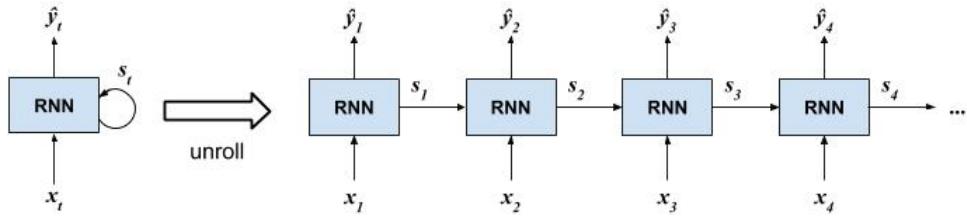
Penting untuk dipahami bahwa walaupun ada empat kotak RNN di gambar di atas, empat kotak itu mencerminkan satu modul RNN yang sama (satu instans model dengan parameter-parameter U , V , dan W yang sama). Penggambaran di atas hanya agar aspek sekuensialnya lebih tergambar.

Alternatif representasinya adalah seperti ini, agar lebih jelas bahwa hanya ada satu modul RNN:



Inilah sebabnya kenapa arsitektur ini disebut RNN. Kata recurrent (berulang) dalam RNN timbul karena RNN melakukan perhitungan yang sama secara berulang-ulang atas input yang kita berikan.

Sering juga kedua ilustrasi di atas digabungkan jadi satu sbb:



Latihan (1)

Melakukan import library yang dibutuhkan

```
In [42]: # import Library pandas
import pandas as pd

# Import Library numpy
import numpy as np

# Import Library matplotlib dan seaborn untuk visualisasi
import matplotlib.pyplot as plt
import seaborn as sns

# import Library for build model

from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
from keras.utils.vis_utils import plot_model


# import Library untuk data preprocessing
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler
#
#.....
```

Load Dataset

```
In [3]: #Panggil file (load file bernama Stock.csv) dan simpan dalam dataframe
dataset ="Stock.csv"
data = pd.read_csv(dataset)
```

```
In [4]: # tampilkan 5 baris data
data.head()
```

Out[4]:

	Date	Open	High	Low	Close	Volume	Name
0	2006-01-03	56.45	56.66	55.46	56.53	3716500	UTX
1	2006-01-04	56.80	56.80	55.84	56.19	3114500	UTX
2	2006-01-05	56.30	56.49	55.63	55.98	3118900	UTX
3	2006-01-06	56.45	56.67	56.10	56.16	2874300	UTX
4	2006-01-09	56.37	56.90	56.16	56.80	2467200	UTX

Review Data

In [5]: # Melihat Informasi Lebih detail mengenai struktur DataFrame dapat dilihat menggunakan fungsi info()
data.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3020 entries, 0 to 3019
Data columns (total 7 columns):
 #   Column  Non-Null Count  Dtype  
--- 
 0   Date     3020 non-null   object  
 1   Open     3019 non-null   float64 
 2   High    3020 non-null   float64 
 3   Low     3020 non-null   float64 
 4   Close    3020 non-null   float64 
 5   Volume   3020 non-null   int64  
 6   Name     3020 non-null   object  
dtypes: float64(4), int64(1), object(2)
memory usage: 165.3+ KB
```

In [6]: # Kolom 'Low' yang akan kita gunakan dalam membangun model
Slice kolom 'Low'

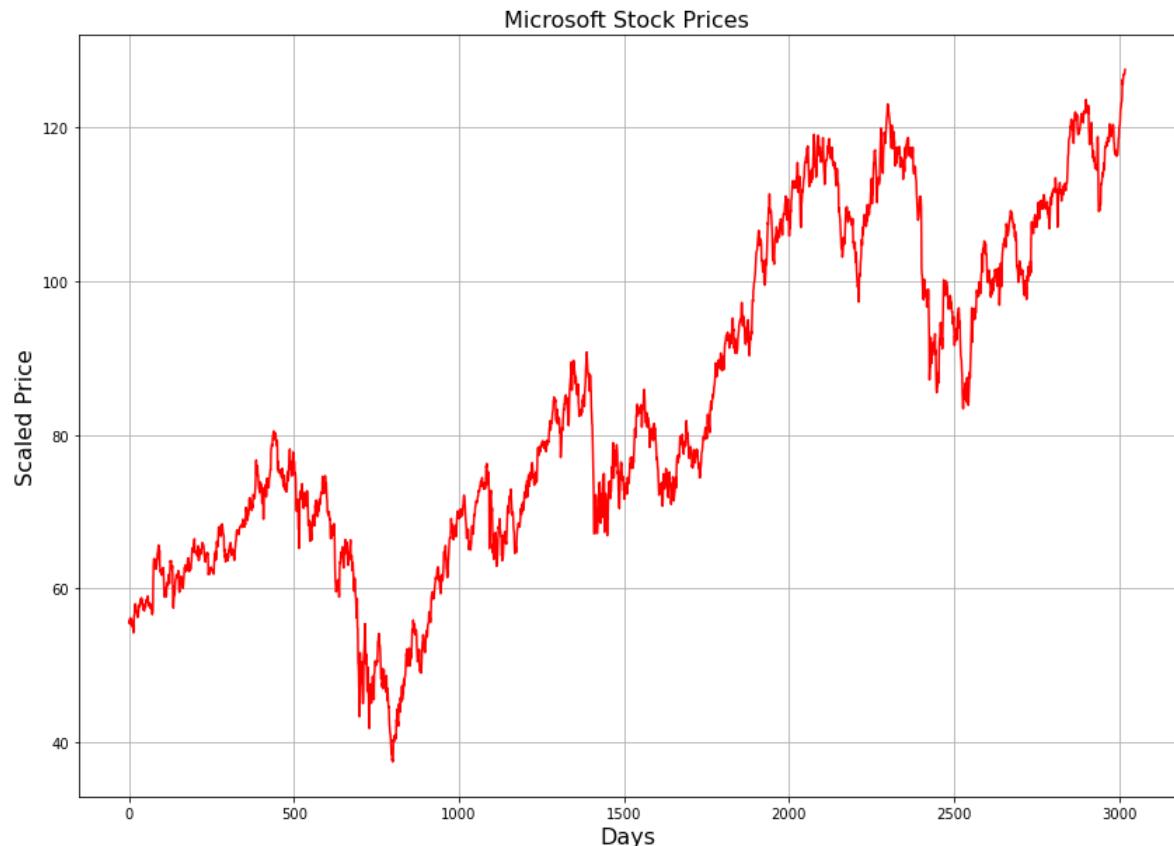
```
Low_data = data.iloc[:,3:4].values
```

In [7]: # cek output low_data
Low_data

```
Out[7]: array([[ 55.46],
       [ 55.84],
       [ 55.63],
       ...,
       [126.92],
       [127.29],
       [127.57]])
```

In [8]: # Visualizing Low_data

```
plt.figure(figsize=(14,10))
plt.plot(Low_data,c="red")
plt.title("Microsoft Stock Prices",fontsize=16)
plt.xlabel("Days",fontsize=16)
plt.ylabel("Scaled Price",fontsize=16)
plt.grid()
plt.show()
```



Latihan (2)

Data Preprocessing

```
In [11]: # Menskalakan data antara 1 dan 0 (scaling) pada Low data
```

```
sc = MinMaxScaler(feature_range = (0, 1))
training_set_scaled = sc.fit_transform(Low_data)
```

```
In [18]: # definisikan variabel step dan train
```

```
X_train = []
y_train = []
```

```
In [19]: # membuat fitur dan Lists label
```

```
for i in range(60, 2168): # upper bound is number of values
    X_train.append(training_set_scaled[i-60:i, 0]) #takes 60 previous stock prices from 60 past stock prices
    y_train.append(training_set_scaled[i, 0]) #contains stock price learned to predict
```

```
In [20]: # mengonversi List yang telah dibuat sebelumnya ke array
```

```
X_train, y_train = np.array(X_train), np.array(y_train) # make into numpy arrays
```

```
In [21]: # cek dimensi data dengan function .shape
```

```
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
```

```
In [19]: # 498 hari terakhir akan digunakan dalam pengujian
# 2500 hari pertama akan digunakan dalam pelatihan
```

```
.....  
.....  
.....  
.....
```

```
In [20]: # reshape data untuk dimasukkan kedalam Keras model
```

```
.....  
.....
```

```
In [21]: # cek kembali dimensi data yang telah di reshape dengan function .shape
```

```
.....  
.....
```

```
(2500, 21, 1)
(498, 21, 1)
```

Sekarang kita bisa mulai membuat model kita, dimulai dengan RNN

Latihan (3)

Build Model - RNN

```
In [24]: # buat varibel penampung model RNN
```

```
regressor = Sequential()
```

In [25]: # Output dari SimpleRNN akan menjadi bentuk tensor 2D (batch_size, 40) dengan Dropout sebesar 0.15

```
regressor.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.shape[1], 1)))
regressor.add(Dropout(0.2))
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))

# Add a Dense Layer with 1 units.

regressor.add(LSTM(units = 50))
regressor.add(Dropout(0.2))
```

In [26]: # menambahkan Loss function kedalam model RNN dengan tipe MSE

```
regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')
```

In [27]: # fit the model RNN, dengan epoch 20 dan batch size 25

```
regressor.fit(X_train, y_train, epochs = 20, batch_size = 25)
```

```
Epoch 1/20
85/85 [=====] - 9s 53ms/step - loss: 0.0691
Epoch 2/20
85/85 [=====] - 5s 57ms/step - loss: 0.0453
Epoch 3/20
85/85 [=====] - 6s 67ms/step - loss: 0.0450
Epoch 4/20
85/85 [=====] - 5s 60ms/step - loss: 0.0445
Epoch 5/20
85/85 [=====] - 5s 56ms/step - loss: 0.0449
Epoch 6/20
85/85 [=====] - 5s 64ms/step - loss: 0.0449
Epoch 7/20
85/85 [=====] - 5s 58ms/step - loss: 0.0445
Epoch 8/20
85/85 [=====] - 5s 60ms/step - loss: 0.0445
Epoch 9/20
85/85 [=====] - 5s 61ms/step - loss: 0.0444
Epoch 10/20
85/85 [=====] - 5s 58ms/step - loss: 0.0447
Epoch 11/20
85/85 [=====] - 5s 57ms/step - loss: 0.0441: 0s - loss: 0.0
Epoch 12/20
85/85 [=====] - 5s 61ms/step - loss: 0.0440
Epoch 13/20
85/85 [=====] - 5s 64ms/step - loss: 0.0445
Epoch 14/20
85/85 [=====] - 5s 63ms/step - loss: 0.0434
Epoch 15/20
85/85 [=====] - 5s 62ms/step - loss: 0.0444:
Epoch 16/20
85/85 [=====] - 6s 68ms/step - loss: 0.0435
Epoch 17/20
85/85 [=====] - 7s 77ms/step - loss: 0.0437
Epoch 18/20
85/85 [=====] - 5s 59ms/step - loss: 0.0437
Epoch 19/20
85/85 [=====] - 5s 62ms/step - loss: 0.0434
Epoch 20/20
85/85 [=====] - 6s 66ms/step - loss: 0.0438
```

Out[27]: <keras.callbacks.History at 0x2233a5b6490>

In [32]: # Prediksi Model RNN

```
scores = regressor.evaluate(X_train,y_train, verbose=0)
```

```
#.....
```

In [34]: scores

Out[34]: 0.009450904093682766

Latihan (4)

Build Model - LSTM

```
In [35]: # buat variabel penampung model LSTM
lstm = Sequential()
```

```
In [48]: # Add a LSTM Layer with 40 internal units. dengan Dropout sebesar 0.15
```

```
lstm.add(LSTM(units = 40, return_sequences = True, input_shape = (X_train.shape[1], 1)))
lstm.add(Dropout(0.15))
lstm.add(LSTM(units = 40, return_sequences = True))
lstm.add(Dropout(0.15))
lstm.add(LSTM(units = 40, return_sequences = True))
lstm.add(Dropout(0.15))

# Add a Dense Layer with 1 units.
#lstm.add(Dense(1))
```

```
-----
ValueError                                     Traceback (most recent call last)
<ipython-input-48-36678d24bb99> in <module>
      3
      4
----> 5 lstm.add(LSTM(units = 40, return_sequences = True, input_shape = (X_train.shape[1], 1)))
      6 lstm.add(Dropout(0.15))
      7 lstm.add(LSTM(units = 40, return_sequences = True))

~\anaconda3\lib\site-packages\tensorflow\python\training\tracking\base.py in _method_wrapper(self, *args, **kwargs)
      528     self._self_setattr_tracking = False # pylint: disable=protected-access
      529     try:
--> 530         result = method(self, *args, **kwargs)
      531     finally:
      532         self._self_setattr_tracking = previous_value # pylint: disable=protected-access

~\anaconda3\lib\site-packages\keras\utils\traceback_utils.py in error_handler(*args, **kwargs)
      65     except Exception as e: # pylint: disable=broad-except
      66         filtered_tb = _process_traceback_frames(e.__traceback__)
---> 67         raise e.with_traceback(filtered_tb) from None
      68     finally:
      69         del filtered_tb

~\anaconda3\lib\site-packages\keras\engine\input_spec.py in assert_input_compatibility(input_spec, inputs, layer_name)
   211     ndim = shape.rank
   212     if ndim != spec.ndim:
--> 213         raise ValueError(f'Input {input_index} of layer "{layer_name}" '
   214                         'is incompatible with the layer: '
   215                         f'expected ndim={spec.ndim}, found ndim={ndim}.')
```

ValueError: Input 0 of layer "lstm_10" is incompatible with the layer: expected ndim=3, found ndim=2. Full shape received: (None, 1)

```
In [30]: # menambahkan Loss function kedalam model Lstm dengan tipe MSE
```

```
lstm.compile(loss='mean_squared_error', optimizer='adam')
plot_model(lstm, show_shapes=True, show_layer_names=True)
```

```
In [49]: # fit lstm model, dengan epoch 20 dan batch size 25
history=lstm.fit(X_train, y_train, epochs=20, batch_size=25, verbose=1, shuffle=False)

Epoch 1/20

-----
ValueError                                Traceback (most recent call last)
<ipython-input-49-af2d638c6964> in <module>
      1 # fit lstm model, dengan epoch 20 dan batch size 25
      2
----> 3 history=lstm.fit(X_train, y_train, epochs=20, batch_size=25, verbose=1, shuffle=False)

~/anaconda3/lib/site-packages/keras/utils traceback_utils.py in error_handler(*args, **kwargs)
    65     except Exception as e: # pylint:disable=broad-except
    66         filtered_tb = _process_traceback_frames(e.__traceback__)
---> 67         raise e.with_traceback(filtered_tb) from None
    68     finally:
    69         del filtered_tb

~/anaconda3/lib/site-packages/tensorflow/python/framework/func_graph.py in autograph_handler(*args, **kwargs)
   1127         except Exception as e: # pylint:disable=broad-except
   1128             if hasattr(e, "ag_error_metadata"):
-> 1129                 raise e.ag_error_metadata.to_exception(e)
   1130             else:
   1131                 raise

ValueError: in user code:

File "C:\Users\dwiah\anaconda3\lib\site-packages\keras\engine\training.py", line 878, in train_function *
    return step_function(self, iterator)
File "C:\Users\dwiah\anaconda3\lib\site-packages\keras\engine\training.py", line 867, in step_function **
    outputs = model.distribute_strategy.run(run_step, args=(data,))
File "C:\Users\dwiah\anaconda3\lib\site-packages\keras\engine\training.py", line 860, in run_step **
    outputs = model.train_step(data)
File "C:\Users\dwiah\anaconda3\lib\site-packages\keras\engine\training.py", line 808, in train_step
    y_pred = self(x, training=True)
File "C:\Users\dwiah\anaconda3\lib\site-packages\keras\utils\traceback_utils.py", line 67, in error_handler
    raise e.with_traceback(filtered_tb) from None
File "C:\Users\dwiah\anaconda3\lib\site-packages\keras\engine\input_spec.py", line 263, in assert_input_compatibility
    raise ValueError(f'Input {input_index} of layer "{layer_name}" is '
ValueError: Input 0 of layer "sequential_1" is incompatible with the layer: expected shape=(None, 1, 60), found
d shape=(None, 60, 1)
```

```
In [32]: # Prediksi Model LSTM
.....
.....
.....
In [39]: lstm_score
Out[39]: 0.8531204922506259
```

Latihan (5)

Evaluation

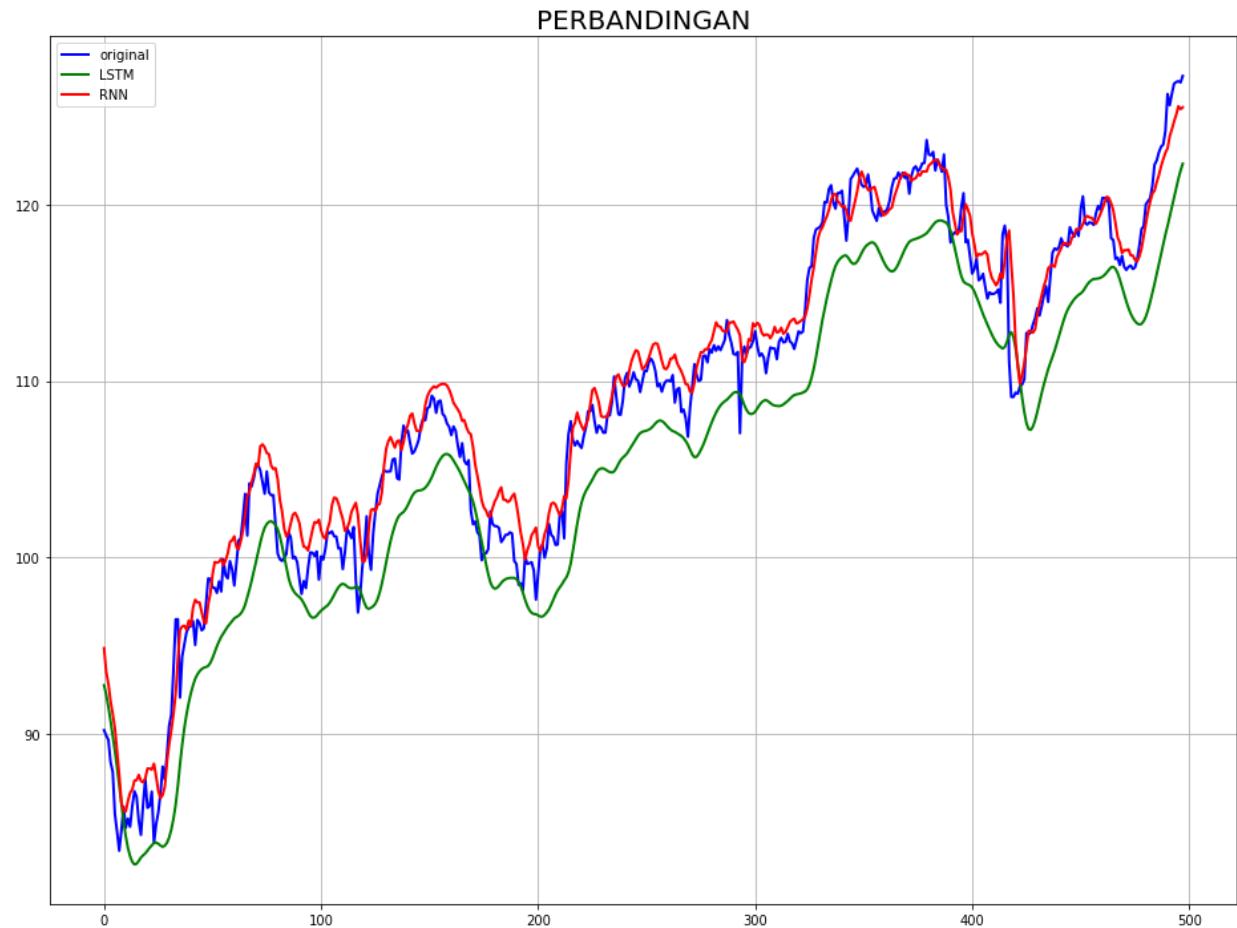
```
In [40]: # Cetak nilai prediksi masing-masing model dengan menggunakan r^2 square
.....
.....
R^2 Score dari model RNN 0.9695404783998754
R^2 Score dari model LSTM 0.8531204922506259
```

Visualisasi Perbandingan Hasil Model prediksi dengan data original

```
In [35]: lstm_predictions = scaler.inverse_transform(lstm_predictions)
rnn_predictions = scaler.inverse_transform(rnn_predictions)
test_y = scaler.inverse_transform(test_y.reshape(-1,1))
```

```
In [41]: plt.figure(figsize=(16,12))

plt.plot(test_y, c="blue", linewidth=2, label="original")
plt.plot(lstm_predictions, c="green", linewidth=2, label="LSTM")
plt.plot(rnn_predictions, c="red", linewidth=2, label="RNN")
plt.legend()
plt.title("PERBANDINGAN", fontsize=20)
plt.grid()
plt.show()
```



Berikan Kesimpulan Anda!

```
In [ ]: # write here
```