TIME SERIES DATA

In [1]:

```
!wget  https://s3.amazonaws.com/keras-datasets/jena_climate_2009_2016.csv.zip
!unzip jena_climate_2009_2016.csv.zip
```

```
--2024-04-08 21:33:51--  https://s3.amazonaws.com/keras-datasets/jena_climate_2009_20
16.csv.zip
Resolving s3.amazonaws.com (s3.amazonaws.com)... 52.216.208.8, 52.216.171.45, 52.216.
133.45, ...
Connecting to s3.amazonaws.com (s3.amazonaws.com)|52.216.208.8|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 13565642 (13M) [application/zip]
Saving to: ‘jena_climate_2009_2016.csv.zip’

jena_climate_2009_2 100%[===================>]  12.94M  57.3MB/s    in 0.2s

2024-04-08 21:33:51 (57.3 MB/s) - ‘jena_climate_2009_2016.csv.zip’ saved [13565642/13
565642]

Archive:  jena_climate_2009_2016.csv.zip
  inflating: jena_climate_2009_2016.csv
  inflating: __MACOSX/._jena_climate_2009_2016.csv
```

In [2]:

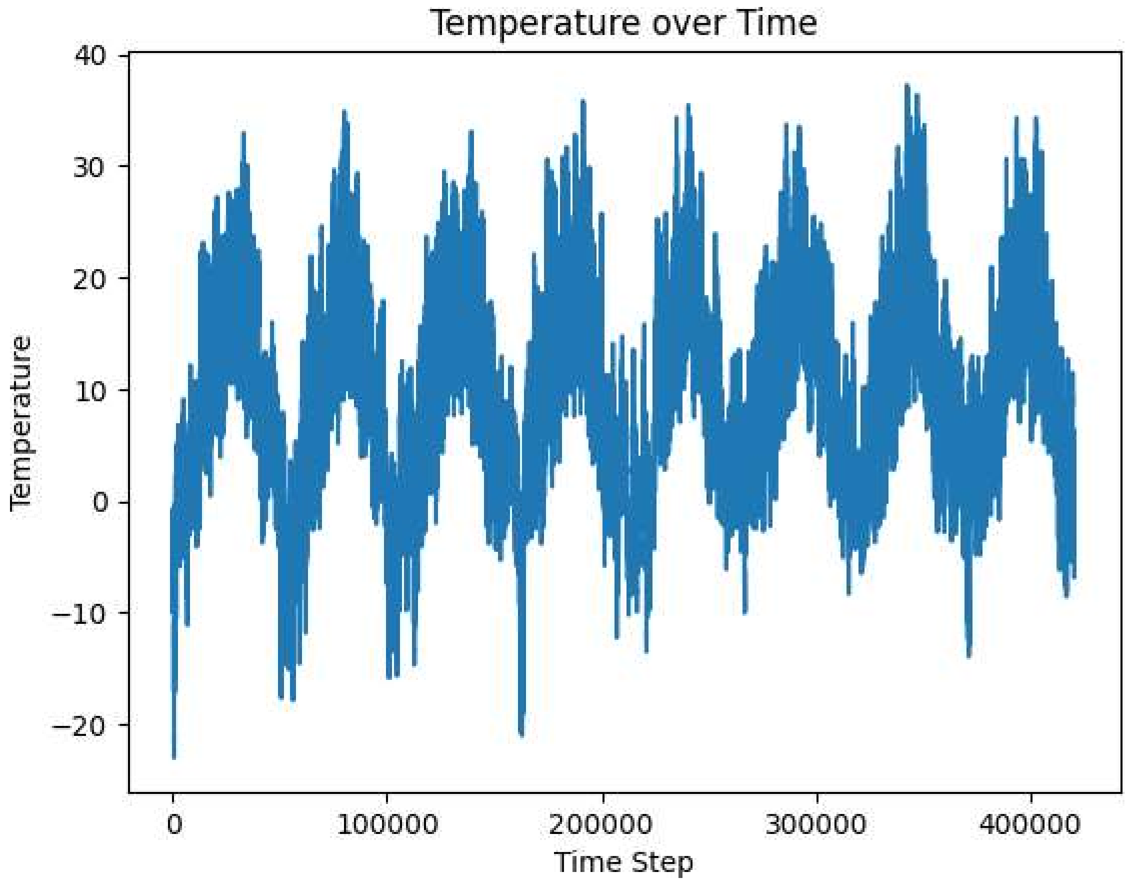
```
import os
# joining the file name with its path
fname = os.path.join("jena_climate_2009_2016.csv")
# reading the file
with open(fname) as f:
    # reading the entire content of the file
    data = f.read()
    lines = data.split("\n")
    header = lines[0].split(",")
    lines = lines[1:]
# printing the header and the number of lines
print(header)
print(len(lines))
```

```
['"Date Time"', '"p (mbar)"', '"T (degC)"', '"Tpot (K)"', '"Tdew (degC)"', '"rh
(%)"', '"VPmax (mbar)"', '"VPact (mbar)"', '"VPdef (mbar)"', '"sh (g/kg)"', '"H2OC (m
mol/mol)"', '"rho (g/m**3)"', '"wv (m/s)"', '"max. wv (m/s)"', '"wd (deg)"']
420451
```

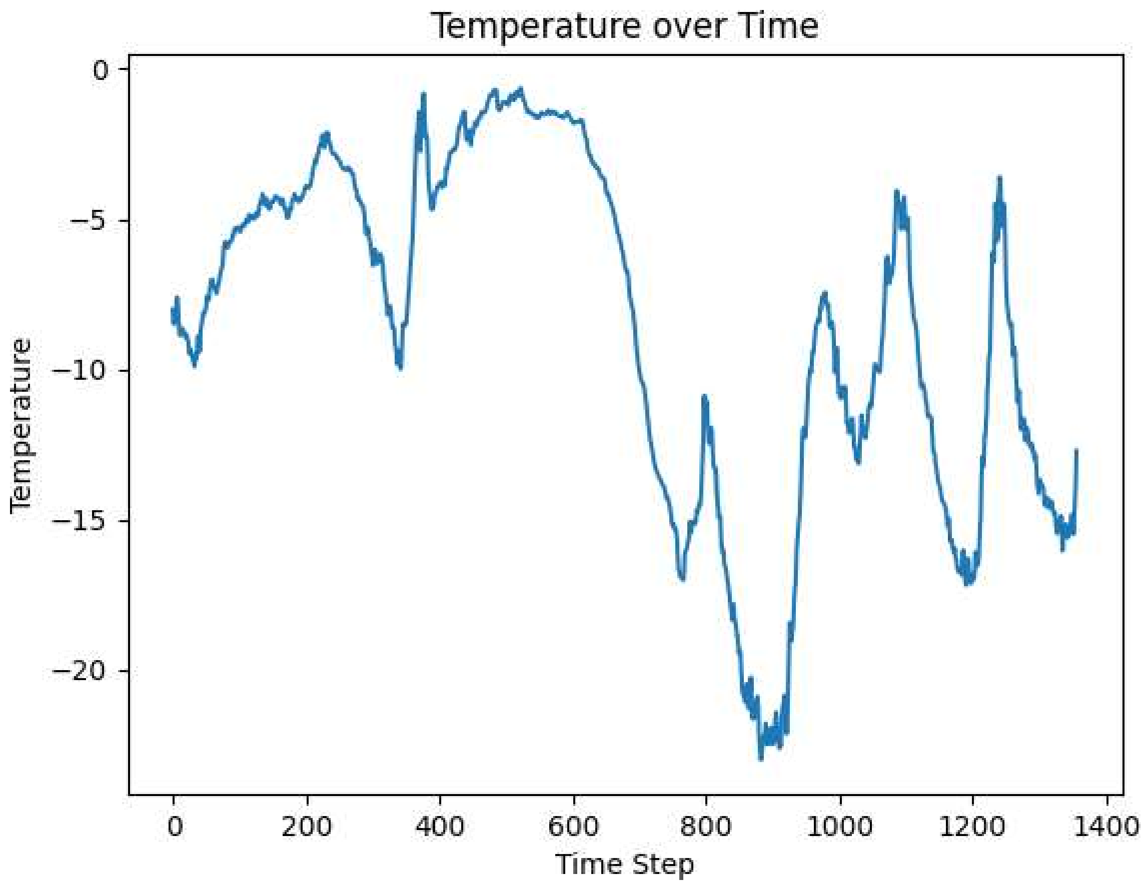
In [3]:

```
import numpy as np
# initialize arrays
temperature = np.zeros((len(lines),))
raw_data = np.zeros((len(lines), len(header) - 1))
# iterate over lines and parse values
for i, line in enumerate(lines):
    values = [float(x) for x in line.split(",")[1:]]
    temperature[i] = values[1]
    raw_data[i, :] = values[:]
```

In [10]:
```
# plot temperature data
from matplotlib import pyplot as plt
plt.plot(range(len(temperature)), temperature)
plt.xlabel('Time Step')
plt.ylabel('Temperature')
plt.title('Temperature over Time')
plt.show()
```



In [11]:
```
plt.plot(range(1356), temperature[:1356])
plt.xlabel('Time Step')
plt.ylabel('Temperature')
plt.title('Temperature over Time')
plt.show()
```



In [12]:

```
# calculating the number of samples for training, validation, and testing
num_train_samples = int(0.5 * len(raw_data))
num_val_samples = int(0.25 * len(raw_data))
num_test_samples = len(raw_data) - num_train_samples - num_val_samples
# print the number of samples for each set
print("Number of traning samples:", num_train_samples)
print("Number of validation samples:", num_val_samples)
print("Number of test samples:", num_test_samples)
```

```
Number of traning samples: 210225
Number of validation samples: 105112
Number of test samples: 105114
```

In [19]:

```
# normalize the data
mean = np.mean(raw_data[:num_train_samples], axis=0)
raw_data -= mean
std = np.std(raw_data[:num_train_samples], axis=0)
raw_data /= std
```

In [22]:

```
import numpy as np
from tensorflow import keras
# generate time series dataset
int_sequence = np.arange(10)
sequence_length = 3
batch_size = 2
targets = int_sequence[3:]
dummy_dataset = keras.utils.timeseries_dataset_from_array(
    data = int_sequence[:-3],
    targets = targets,
    sequence_length = sequence_length,
    batch_size = batch_size,
```

```
)
# iterate over the dataset and print inputs and targets
for inputs, targets in dummy_dataset:
    for i in range(inputs.shape[0]):
        print([int(x) for x in inputs[i]], int(targets[i]))
```

```
[0, 1, 2] 3
[1, 2, 3] 4
[2, 3, 4] 5
[3, 4, 5] 6
[4, 5, 6] 7
```

In [26]:

```
ameters
sampling_rate = 6
sequence_length = 120
delay = sampling_rate * (sequence_length + 24 - 1)
batch_size = 256

# create training dataset
train_dataset = keras.utils.timeseries_dataset_from_array(
    raw_data[:-delay],
    targets=temperature[delay:],
    sampling_rate=sampling_rate,
    sequence_length=sequence_length,
    shuffle=True,
    batch_size=batch_size,
    start_index=0,
    end_index=num_train_samples)

# create validation dataset
val_dataset = keras.utils.timeseries_dataset_from_array(
    raw_data[:-delay],
    targets=temperature[delay:],
    sampling_rate=sampling_rate,
    sequence_length=sequence_length,
    shuffle=True,
    batch_size=batch_size,
    start_index=num_train_samples,
    end_index=num_train_samples + num_val_samples)

# create testing dataset
test_dataset = keras.utils.timeseries_dataset_from_array(
    raw_data[:-delay],
    targets=temperature[delay:],
    sampling_rate=sampling_rate,
    sequence_length=sequence_length,
    shuffle=True,
    batch_size=batch_size,
    start_index=num_train_samples + num_val_samples)
```

In [24]:

```
for samples, targets in train_dataset:
    print("Samples shape:", samples.shape)
    print("Targets shape:", targets.shape)
    break
```

```
Samples shape: (256, 120, 14)
Targets shape: (256,)
```

In [27]:

```
def evaluate_naive_method(dataset):
    total_abs_err = 0.
```

```
    samples_seen = 0
    for samples, targets in dataset:
        preds = samples[:, -1, 1] * std[1] + mean[1]
        total_abs_err += np.sum(np.abs(preds - targets))
        samples_seen += samples.shape[0]
    return total_abs_err / samples_seen

# evaluate on validation dataset
val_mae = evaluate_naive_method(val_dataset)
print(f"Validation MAE: {val_mae:.2f}")
#evaluate on testing dataset
test_mae = evaluate_naive_method(test_dataset)
print(f"Test MAE: {test_mae:.2f}")
```

```
Validation MAE: 10.28
Test MAE: 10.40
```

In [29]:

```
import keras
from keras import layers
```

In [ ]:

```
# define input layer
inputs  =  keras.Input(shape=(sequence_length,  raw_data.shape[-1]))
# define GRU layers
x  =  layers.GRU(32,  recurrent_dropout=0.5,  return_sequences=True)(inputs)
x = layers.GRU(32, recurrent_dropout=0.5)(x)
# add dropout layer
x = layers.Dropout(0.5)(x)
# output layer
outputs  =  layers.Dense(1)(x)
# define the model
model  =  keras.Model(inputs,  outputs)
# define callbacks
callbacks = [
    keras.callbacks.ModelCheckpoint("jena_stacked_gru_dropout.keras",
                                    save_best_only=True)
]
# compile the model
model.compile(optimizer="rmsprop",  loss="mse",  metrics=["mae"])
# train the model
history = model.fit(train_dataset,
                    epochs=15,
                    validation_data=val_dataset,
                    callbacks=callbacks)
```

```
Epoch 1/15
819/819 [==============================] - 424s 512ms/step - loss: 25.6053 - mae: 3.7
307 - val_loss: 9.9940 - val_mae: 2.4598
Epoch 2/15
819/819 [==============================] - 430s 525ms/step - loss: 14.0533 - mae: 2.9
035 - val_loss: 9.0268 - val_mae: 2.3269
Epoch 3/15
819/819 [==============================] - 350s 426ms/step - loss: 13.1620 - mae: 2.8
153 - val_loss: 8.8732 - val_mae: 2.3082
Epoch 4/15
819/819 [==============================] - 392s 479ms/step - loss: 12.6352 - mae: 2.7
536 - val_loss: 9.0377 - val_mae: 2.3357
Epoch 5/15
819/819 [==============================] - 389s 474ms/step - loss: 12.1224 - mae: 2.7
004 - val_loss: 9.1149 - val_mae: 2.3434
Epoch 6/15
819/819 [==============================] - 390s 476ms/step - loss: 11.6738 - mae: 2.6
508 - val_loss: 9.1215 - val_mae: 2.3449
Epoch 7/15
819/819 [==============================] - 391s 477ms/step - loss: 11.2826 - mae: 2.6
068 - val_loss: 9.3053 - val_mae: 2.3663
Epoch 8/15
819/819 [==============================] - 393s 479ms/step - loss: 10.9756 - mae: 2.5
679 - val_loss: 9.7665 - val_mae: 2.4301
Epoch 9/15
819/819 [==============================] - 388s 473ms/step - loss: 10.6350 - mae: 2.5
321 - val_loss: 9.1288 - val_mae: 2.3458
Epoch 10/15
819/819 [==============================] - 346s 422ms/step - loss: 10.3258 - mae: 2.4
934 - val_loss: 9.7865 - val_mae: 2.4293
Epoch 11/15
819/819 [==============================] - 389s 475ms/step - loss: 10.0524 - mae: 2.4
595 - val_loss: 9.0879 - val_mae: 2.3450
Epoch 12/15
819/819 [==============================] - 349s 426ms/step - loss: 9.8245 - mae: 2.43
50 - val_loss: 9.4766 - val_mae: 2.3838
Epoch 13/15
819/819 [==============================] - 343s 418ms/step - loss: 9.6894 - mae: 2.41
70 - val_loss: 9.5795 - val_mae: 2.4039
Epoch 14/15
819/819 [==============================] - 344s 420ms/step - loss: 9.4005 - mae: 2.38
03 - val_loss: 9.6661 - val_mae: 2.4057
Epoch 15/15
819/819 [==============================] - 348s 425ms/step - loss: 9.3069 - mae: 2.36
90 - val_loss: 9.6572 - val_mae: 2.4094
```

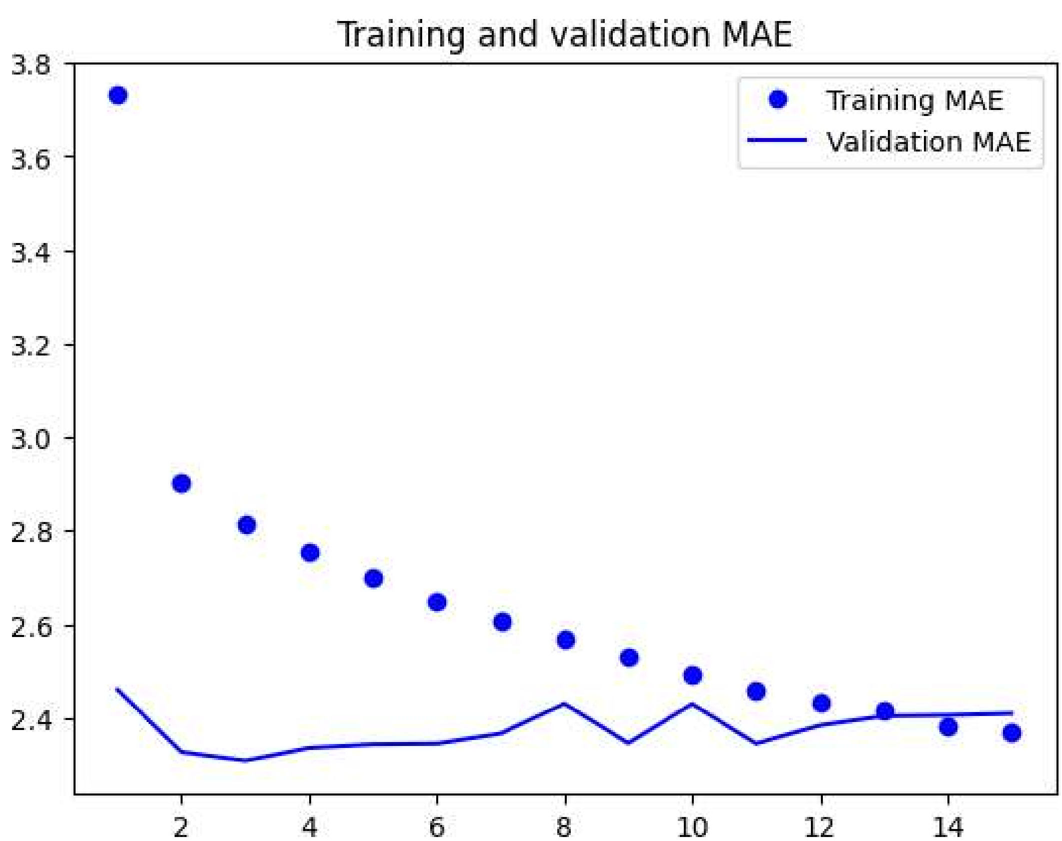
In [ ]:

```
# summary of the model
model.summary()
```

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 120, 14)]         0

 gru_2 (GRU)                 (None, 120, 32)           4608

 gru_3 (GRU)                 (None, 32)                6336

 dropout_1 (Dropout)         (None, 32)                0

 dense_1 (Dense)             (None, 1)                 33

=================================================================
Total params: 10977 (42.88 KB)
Trainable params: 10977 (42.88 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [ ]:

```
import matplotlib.pyplot as plt
# extract MAE values from the history object
loss = history.history["mae"]
val_loss = history.history["val_mae"]
epochs = range(1, len(loss) + 1)
# plot MAE
plt.figure()
plt.plot(epochs, loss, "bo", label="Training MAE")
plt.plot(epochs, val_loss, "b", label="Validation MAE")
plt.title("Training and validation MAE")
plt.legend()
plt.show()
```



In [ ]:
```
model = keras.models.load_model("jena_stacked_gru_dropout.keras")
# evaluate the model on the testing dataset
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

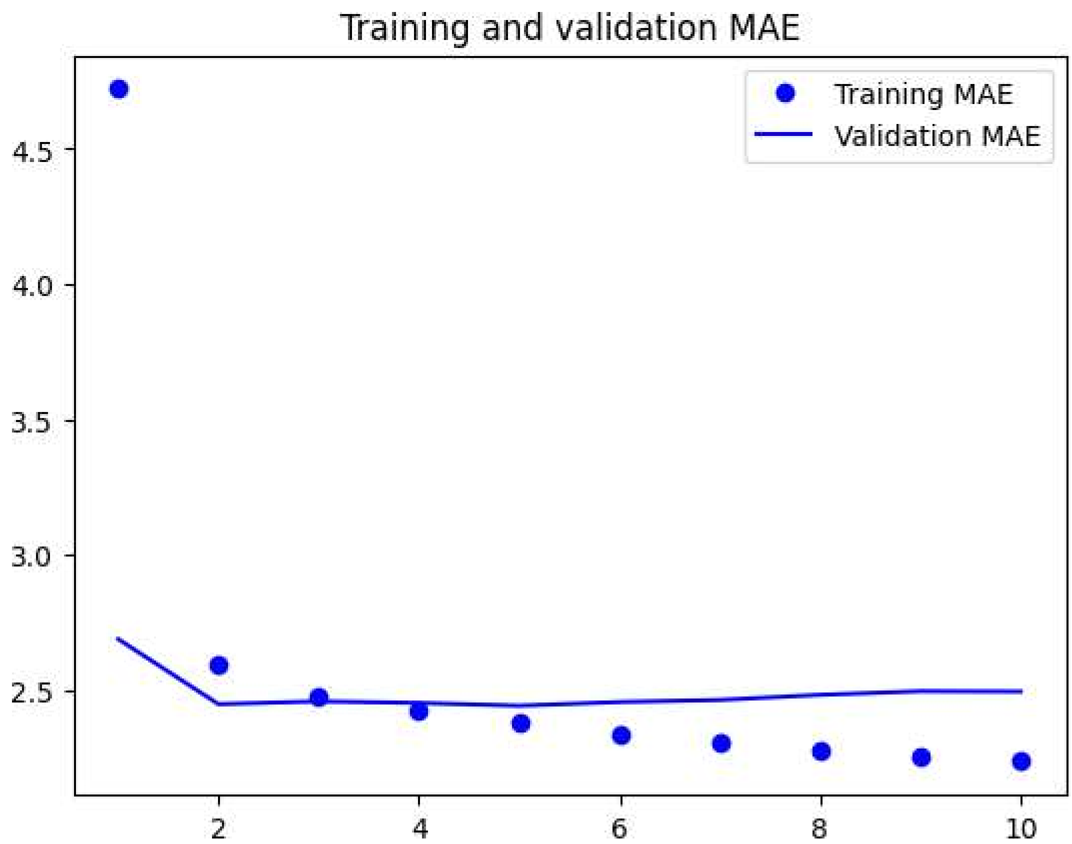
```
405/405 [==============================] - 58s 136ms/step - loss: 9.8269 - mae: 2.454
0
Test MAE: 2.45
```

In [ ]:
```
# define the model architecture
inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
x = layers.GRU(32, recurrent_dropout=0.5, return_sequences=True)(inputs)
x = layers.GRU(32, recurrent_dropout=0.5)(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1)(x)
model = keras.Model(inputs, outputs)

# define callbacks
callbacks = [
    keras.callbacks.ModelCheckpoint("jena_stacked_gru_dropout.keras",
                                    save_best_only=True)
]
# compile the model
model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])
# train the model
history = model.fit(train_dataset,
                    epochs=12,
                    validation_data=val_dataset,
                    callbacks=callbacks)
```

```
Epoch 1/12
819/819 [==============================]  - 408s 490ms/step - loss: 26.2144 - mae: 3.7
721 - val_loss: 9.3886 - val_mae: 2.3692
Epoch 2/12
819/819 [==============================]  - 390s 476ms/step - loss: 14.0768 - mae: 2.9
062 - val_loss: 9.9173 - val_mae: 2.4660
Epoch 3/12
819/819 [==============================]  - 394s 480ms/step - loss: 13.2664 - mae: 2.8
230 - val_loss: 9.1357 - val_mae: 2.3417
Epoch 4/12
819/819 [==============================]  - 371s 453ms/step - loss: 12.6449 - mae: 2.7
584 - val_loss: 8.6929 - val_mae: 2.2892
Epoch 5/12
819/819 [==============================]  - 368s 449ms/step - loss: 12.1608 - mae: 2.7
095 - val_loss: 8.8953 - val_mae: 2.3129
Epoch 6/12
819/819 [==============================]  - 397s 484ms/step - loss: 11.7411 - mae: 2.6
606 - val_loss: 8.7522 - val_mae: 2.2934
Epoch 7/12
819/819 [==============================]  - 391s 478ms/step - loss: 11.3610 - mae: 2.6
175 - val_loss: 8.9374 - val_mae: 2.3319
Epoch 8/12
819/819 [==============================]  - 392s 478ms/step - loss: 10.9618 - mae: 2.5
717 - val_loss: 9.1890 - val_mae: 2.3572
Epoch 9/12
819/819 [==============================]  - 392s 477ms/step - loss: 10.5974 - mae: 2.5
279 - val_loss: 9.0267 - val_mae: 2.3343
Epoch 10/12
819/819 [==============================]  - 369s 450ms/step - loss: 10.3152 - mae: 2.4
918 - val_loss: 9.5055 - val_mae: 2.4190
Epoch 11/12
819/819 [==============================]  - 350s 427ms/step - loss: 10.0797 - mae: 2.4
676 - val_loss: 9.5564 - val_mae: 2.4172
Epoch 12/12
819/819 [==============================]  - 392s 478ms/step - loss: 9.8528 - mae: 2.43
72 - val_loss: 9.4911 - val_mae: 2.4224
```

In [ ]:

```python
import matplotlib.pyplot as plt
# extract MAE values from the history object
loss = history.history["mae"]
val_loss = history.history["val_mae"]
epochs = range(1, len(loss) + 1)
# plot MAE
plt.figure()
plt.plot(epochs, loss, "bo", label="Training MAE")
plt.plot(epochs, val_loss, "b", label="Validation MAE")
plt.title("Training and validation MAE")
plt.legend()
plt.show()
```



In [ ]:
```
model = keras.models.load_model("jena_stacked_gru_dropout.keras")
# evaluate the model onthe testing dataset
print(f"Test MAE: {model.evaluate(test_dataset)[1]:.2f}")
```

```
405/405 [==============================] - 49s 117ms/step - loss: 9.6238 - mae: 2.426
7
Test MAE: 2.43
```

In [ ]:
```
# define the input layer
inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
# define the output layer
x = layers.LSTM(16)(inputs)
outputs = layers.Dense(1)(x)
# create the model
model = keras.Model(inputs, outputs)
# compile the model
model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])
# train the model
history = model.fit(train_dataset,
                    epochs=10,
                    validation_data=val_dataset)
```

```
Epoch 1/10
819/819 [==============================] - 135s 162ms/step - loss: 42.2236 - mae: 4.7
208 - val_loss: 12.6147 - val_mae: 2.6889
Epoch 2/10
819/819 [==============================] - 122s 149ms/step - loss: 11.1167 - mae: 2.5
969 - val_loss: 9.9500 - val_mae: 2.4484
Epoch 3/10
819/819 [==============================] - 125s 152ms/step - loss: 10.0431 - mae: 2.4
771 - val_loss: 10.0406 - val_mae: 2.4588
Epoch 4/10
819/819 [==============================] - 125s 152ms/step - loss: 9.6428 - mae: 2.42
39 - val_loss: 10.0231 - val_mae: 2.4529
Epoch 5/10
819/819 [==============================] - 106s 129ms/step - loss: 9.3506 - mae: 2.38
35 - val_loss: 9.9082 - val_mae: 2.4426
Epoch 6/10
819/819 [==============================] - 107s 130ms/step - loss: 8.9929 - mae: 2.33
94 - val_loss: 9.9654 - val_mae: 2.4564
Epoch 7/10
819/819 [==============================] - 120s 146ms/step - loss: 8.7444 - mae: 2.30
88 - val_loss: 10.0098 - val_mae: 2.4640
Epoch 8/10
819/819 [==============================] - 105s 128ms/step - loss: 8.5338 - mae: 2.27
97 - val_loss: 10.1657 - val_mae: 2.4828
Epoch 9/10
819/819 [==============================] - 124s 151ms/step - loss: 8.3408 - mae: 2.25
34 - val_loss: 10.2400 - val_mae: 2.4962
Epoch 10/10
819/819 [==============================] - 107s 130ms/step - loss: 8.2104 - mae: 2.23
66 - val_loss: 10.2903 - val_mae: 2.4951
```

In [ ]:

```python
# evaluate the model on testing dataset
model.evaluate(test_dataset)
```

```
405/405 [==============================] - 40s 98ms/step - loss: 10.9761 - mae: 2.601
5
```

Out[ ]:

```
[10.976113319396973, 2.6014609336853027]
```

In [ ]:

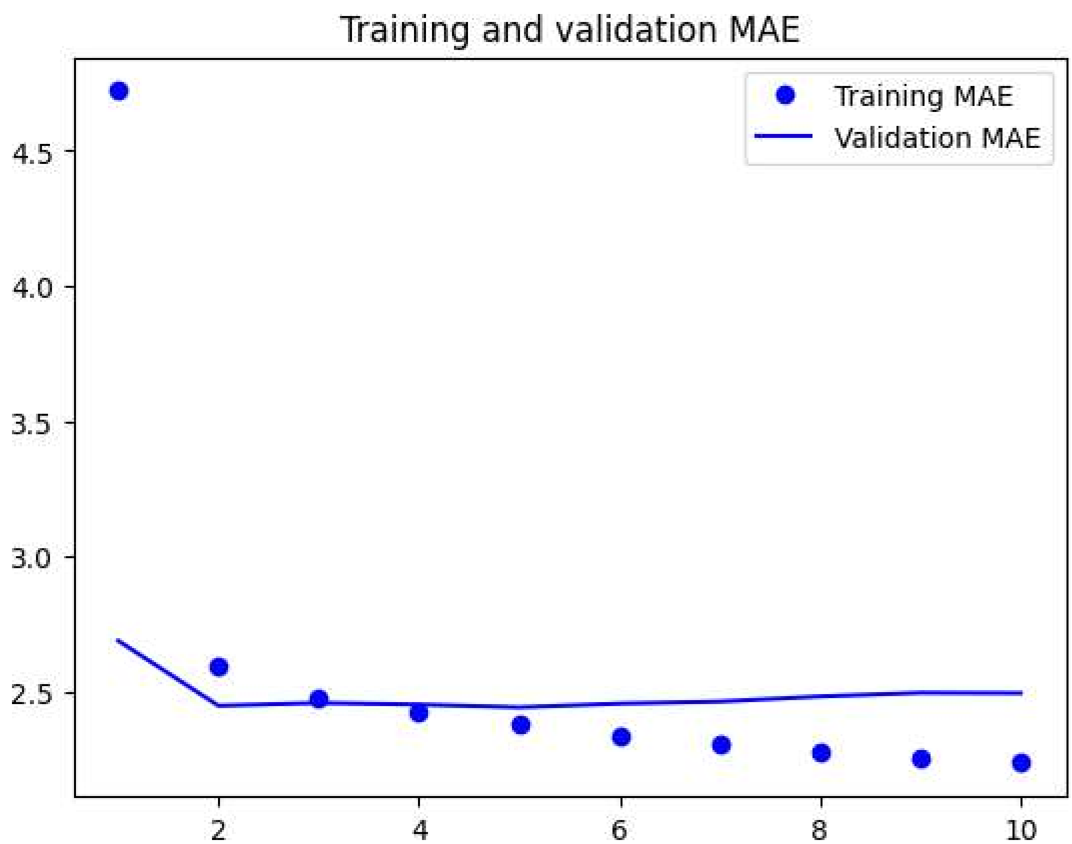
```python
# summary of the model
model.summary()
```

```
Model: "model_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_4 (InputLayer)        [(None, 120, 14)]         0

 lstm (LSTM)                 (None, 16)                1984

 dense_3 (Dense)             (None, 1)                 17

=================================================================
Total params: 2001 (7.82 KB)
Trainable params: 2001 (7.82 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [ ]:

```python
import matplotlib.pyplot as plt
# extract MAE values from the history object
loss = history.history["mae"]
```

```
val_loss = history.history["val_mae"]
epochs = range(1, len(loss) + 1)
# plot MAE
plt.figure()
plt.plot(epochs, loss, "bo", label="Training MAE")
plt.plot(epochs, val_loss, "b", label="Validation MAE")
plt.title("Training and validation MAE")
plt.legend()
plt.show()
```



In [ ]:

```
# define the input layer
inputs = keras.Input(shape=(sequence_length, raw_data.shape[-1]))
# define the convolutional neural network architecture
x = layers.Conv1D(8, 24, activation="relu")(inputs)
x = layers.MaxPooling1D(2)(x)
x = layers.Conv1D(8, 12, activation="relu")(x)
x = layers.MaxPooling1D(2)(x)
x = layers.Conv1D(8, 6, activation="relu")(x)
x = layers.GlobalAveragePooling1D()(x)
# define the output layer
outputs = layers.Dense(1)(x)
# create the model
model = keras.Model(inputs, outputs)
# define callbacks
callbacks = [
    keras.callbacks.ModelCheckpoint("jena_conv.keras",
                                    save_best_only=True)
]
# compile the model
model.compile(optimizer="rmsprop", loss="mse", metrics=["mae"])
# train the model
history = model.fit(train_dataset,
```

```
                    epochs=10,
                    validation_data=val_dataset)
# evaluate the model on te testing dataset
model.evaluate(test_dataset)
```

```
Epoch 1/10
819/819 [==============================]  - 89s 107ms/step - loss: 22.3020 - mae: 3.71
22 - val_loss: 15.7571 - val_mae: 3.1306
Epoch 2/10
819/819 [==============================]  - 84s 102ms/step - loss: 15.6251 - mae: 3.13
91 - val_loss: 17.2376 - val_mae: 3.2962
Epoch 3/10
819/819 [==============================]  - 84s 102ms/step - loss: 14.1910 - mae: 2.98
75 - val_loss: 14.0966 - val_mae: 2.9580
Epoch 4/10
819/819 [==============================]  - 79s 96ms/step - loss: 13.3656 - mae: 2.895
7 - val_loss: 14.4072 - val_mae: 2.9949
Epoch 5/10
819/819 [==============================]  - 79s 97ms/step - loss: 12.7696 - mae: 2.826
8 - val_loss: 14.5965 - val_mae: 3.0376
Epoch 6/10
819/819 [==============================]  - 82s 100ms/step - loss: 12.2956 - mae: 2.77
22 - val_loss: 14.4835 - val_mae: 3.0024
Epoch 7/10
819/819 [==============================]  - 84s 103ms/step - loss: 11.9059 - mae: 2.72
70 - val_loss: 13.6424 - val_mae: 2.9029
Epoch 8/10
819/819 [==============================]  - 83s 101ms/step - loss: 11.5457 - mae: 2.68
55 - val_loss: 13.9859 - val_mae: 2.9393
Epoch 9/10
 819/819 [==============================]  - 80s 97ms/step - loss: 11.2190 - mae: 2.649
9 - val_loss: 13.8619 - val_mae: 2.9112
Epoch 10/10
819/819 [==============================]  - 89s 108ms/step - loss: 10.9253 - mae: 2.61
38 - val_loss: 15.5370 - val_mae: 3.0851
405/405 [==============================]  - 22s 53ms/step - loss: 17.7821 - mae: 3.313
8
```

Out[ ]:

```
[17.782123565673828, 3.313836097717285]
```

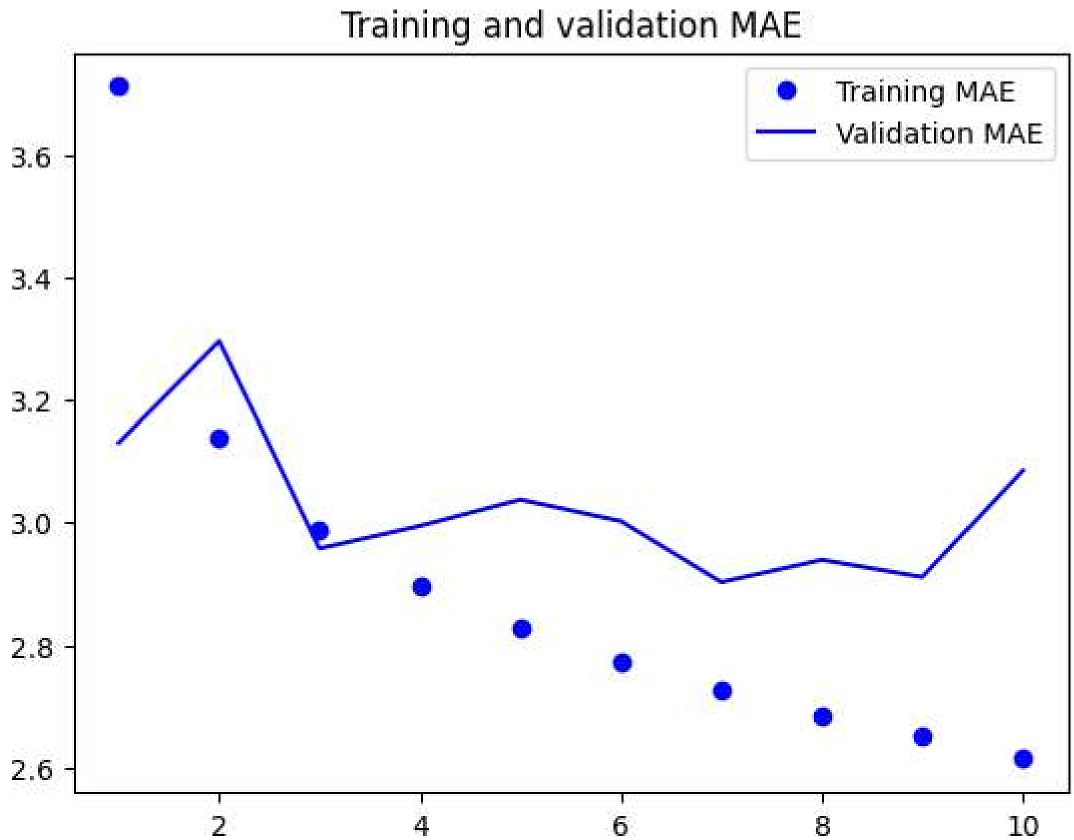
In [ ]:

```
# summary of the model
model.summary()
```

```
Model: "model_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_5 (InputLayer)        [(None, 120, 14)]         0         
                                                                 
 conv1d (Conv1D)             (None, 97, 8)             2696      
                                                                 
 max_pooling1d (MaxPooling1  (None, 48, 8)             0         
 D)                                                              
                                                                 
 conv1d_1 (Conv1D)           (None, 37, 8)             776       
                                                                 
 max_pooling1d_1 (MaxPoolin  (None, 18, 8)             0         
 g1D)                                                            
                                                                 
 conv1d_2 (Conv1D)           (None, 13, 8)             392       
                                                                 
 global_average_pooling1d (  (None, 8)                 0         
 GlobalAveragePooling1D)                                         
                                                                 
 dense_4 (Dense)             (None, 1)                 9         
                                                                 
=================================================================
Total params: 3873 (15.13 KB)
Trainable params: 3873 (15.13 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [ ]:

```python
import matplotlib.pyplot as plt
# extract MAE values from the history object
oss = history.history["mae"]
val_loss = history.history["val_mae"]
epochs = range(1, len(loss) + 1)
# plot MAE
plt.figure()
plt.plot(epochs, loss, "bo", label="Training MAE")
plt.plot(epochs, val_loss, "b", label="Validation MAE")
plt.title("Training and validation MAE")
plt.legend()
plt.show()
```



Training and validation MAE
Training MAE
Validation MAE
3.6
3.4
3.2
3.0
2.8
2.6
2
4
6
8
10